Title
A 4th-Order Particle-in-Cell Method with Phase-Space Remapping for the Vlasov--Poisson Equation

Journal
SIAM Journal on Scientific Computing, 39(3)

ISSN
1064-8275

Authors
Myers, A
Colella, P
Straalen, B Van

Publication Date
2017

DOI
10.1137/16m105962x

Peer reviewed

A 4TH-ORDER PARTICLE-IN-CELL METHOD WITH PHASE-SPACE REMAPPING FOR THE VLASOV-POISSON EQUATION *

A. MYERS [‡§†], P. COLELLA [‡], AND B. VAN STRAALEN [‡]

Abstract. Numerical solutions to the Vlasov-Poisson system of equations have important applications to both plasma physics and cosmology. In this paper, we present a new Particle-in-Cell (PIC) method for solving this system that is 4th-order accurate in both space and time. Our method is a high-order extension of one presented previously [B. Wang, G. Miller, and P. Colella, SIAM J. Sci. Comput., 33 (2011), pp. 3509–3537]. It treats all of the stages of the standard PIC update - charge deposition, force interpolation, the field solve, and the particle push - with 4th-order accuracy, and includes a 6th-order accurate phase-space remapping step for controlling particle noise. We demonstrate the convergence of our method on a series of one- and two- dimensional electrostatic plasma test problems, comparing its accuracy to that of a 2nd-order method. As expected, the 4th-order method can achieve comparable accuracy to the 2nd-order method with many fewer resolution elements.

Key words. Particle-in-Cell (PIC) methods, Higher Order, Phase-space remapping, Numerical noise, Vlasov–Poisson equation

AMS subject classifications. 35, 65, 76

1. Introduction.

In this paper, we present a method for solving the Vlasov-Poisson system of equations, which in non-dimensional form is given by:

$$\frac{\partial f}{\partial t} + \boldsymbol{v} \cdot \frac{\partial f}{\partial \boldsymbol{x}} - \boldsymbol{E} \cdot \frac{\partial f}{\partial \boldsymbol{v}} = 0 \tag{1.1}$$

and

$$\nabla^2 \phi = -\rho. \tag{1.2}$$

Here, $f(\boldsymbol{x}, \boldsymbol{v}, t)$ is the phase space distribution function defined on $(\boldsymbol{x}, \boldsymbol{v}) \in \mathbb{R}^D \times \mathbb{R}^D$, where $D = 1, 2$, or 3 is the number of spatial dimensions under consideration, $\boldsymbol{E}(\boldsymbol{x}, t) = -\nabla\phi$ is the electric field, $\phi(\boldsymbol{x}, t)$ is the potential, and $\rho(\boldsymbol{x}, t)$ is the charge density. For simplicity, we assume that the our system contains a single, negatively charged species, and that the ions form a fixed, neutralizing background, so that the charge density can be defined as:

$$\rho(\boldsymbol{x}, t) = 1 - \int_{\mathbb{R}^D} f(\boldsymbol{x}, \boldsymbol{v}, t) d\boldsymbol{v}. \tag{1.3}$$

Equations (1.1) and (1.2) describe the phase space evolution of a collisionless fluid under the influence of electrostatic forces. They are important for plasma physics,

*This material is based upon work supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research Program and performed under the auspices of the U.S. Department of Energy by Lawrence Berkeley National Laboratory under Contract DE-AC02-05CH11231. This work relied heavily on yt [1] and the core scientific Python packages, including SciPy [2] IPython [3], NumPy [4], and Matplotlib [5], for data analysis and plotting.

†AM wishes to thank Anshu Dubey, Daniel Graves, and Daniel Martin for sharing their time and expertise with Chombo.

‡Applied Numerical Algorithms Group, Lawrence Berkeley National Laboratory, MS 50A-1148, Berkeley, CA, 94720

§atmyers@lbl.gov

where they are used, for example, to model space plasmas, particle accelerators, and for controlled thermonuclear fusion. In a slightly different form, Equations (1.1) and (1.2) can also be applied to cosmology, where they are used to model the gravitational evolution of dark matter in an expanding universe. For simplicity, we have specialized to the plasma version of the Vlasov-Poisson system in this paper, but the methods presented here can be easily applied to the self-gravitating case as well.

The Vlasov Equation (1.1) is a nonlinear advection equation in phase space and can in principle be solved with a variety of grid-based methods [6, 7], including high-order methods [8, e.g.]. However, particle discretizations, which reduce the Vlasov-Poisson system to a set of coupled ordinary differential equations, have been more common in practice. The Particle-in-Cell (PIC) method [9], in which the forces are computed on an intermediate grid and then interpolated back to the particle positions, is a particularly simple approach that has been widely used in both cosmology [10, e.g.] and plasma physics [11, e.g.].

Traditional PIC methods, however, suffer from a few downsides. The first is that they are usually limited to 2nd-order accuracy. While 4th-order PIC methods have been developed in the context of fluid simulation [12], and high-order field solves and time integrators have been used in the plasma context [13, e.g.], to our knowledge, ours is the first PIC method for Vlasov-Poisson that treats *every* stage of the PIC update with fourth-order accuracy.

The second downside concerns the stability of PIC methods over long time evolutions. The convergence theory for electrostatic PIC [14] shows that the stability error for the electric field contains a term that grows exponentially with time. While the growth rate of this term is problem-dependent, given enough time it can eventually degrade the accuracy of the solution, a problem often described as "particle noise." The PIC method presented in [14], however, circumvents this problem by periodically restarting the calculation with a new set of particles that represent the same underlying distribution function. Such *remapping* or *remeshing* techniques have also been applied successfully in the context of fluid dynamics to vortex methods [15] and to smoothed particle hydrodynamics [16]. With remapping, PIC methods can obtain accurate numerical solutions to the Vlasov-Poisson problem for long time integrations in both the plasma [14, 17] and the cosmological [18] context.

However, the PIC method in [14] and [17] was only 2nd-order accurate. For obtaining accurate numerical solutions with a feasible number of resolution elements, higher-order methods are greatly desirable. This is particularly true given current trends in high-performance computing. As computer architectures evolve, the limiting factor affecting application performance is increasingly not the rate at which the processor can perform computations, but rather than rate at which data can be streamed to the processor from DRAM. In light of this, the Arithmetic Intensity (AI) - the number of arithmetic operations per byte of DRAM accessed - of a given numerical algorithm is considered to be a critical factor in achieving the theoretical maximum performance on current and next-generation supercomputing platforms [19]. 2nd-order PIC algorithms have peak arithmetic intensities of around 1 Flop/byte, putting them in the streaming-limited regime of current and planned supercomputing architectures and making it impossible to achieve maximum performance. High-order methods, however, which allow for more operations per byte, can potentially achieve peak performance on these machines.

In this paper, we extend the PIC algorithm of [14] and [17] to be 4th-order accurate in both space and time. The heart of our method is the use of the high-order

interpolating functions derived in [20] for charge deposition and force interpolation. We have implemented this algorithm in Chombo [21] and used it to solve a suite of test problems, demonstrating its fourth-order convergence rates. The outline of the paper is as follows. We begin by giving a review of PIC methods in Section 2.1. We then describe our 4th-order PIC method, as well as a 2nd-order PIC method that we use for comparison, in Sections 2.2 and 2.3. We present our numerical results on set of one- and two-dimensional test problems in Section 3. Finally, in Section 4, we present our conclusions and discuss future research directions.

2. Numerical Methods.

2.1. An overview of PIC methods. To solve equations (1.1) and (1.2) with PIC, the initial distribution function must be sampled by a set of Lagrangian particles, $\mathbb{P}$. In this paper, we generate this initial particle distribution using the approach outlined in [14]. The particles are initially laid out at the cell centers of a Cartesian grid in phase space with mesh spacings h_x and h_v in position and velocity space, respectively. For a given initial distribution function, $f(\boldsymbol{x}, \boldsymbol{v}, t = 0)$, the initial particle representation is computed by assigning each particle $p \in \mathbb{P}$ a charge q_p:

$$q_p = f(\boldsymbol{x}_p^i, \boldsymbol{v}_p^i, t = 0) h_x^D h_v^D. \tag{2.1}$$

where $\boldsymbol{x}_p^i$ and $\boldsymbol{v}_p^i$ are the initial position and velocity of particle p. For computational efficiency, we discard particles with charges less than some problem-dependent threshold value. The initial distribution function can then be approximated as:

$$f(\boldsymbol{x}, \boldsymbol{v}, t = 0) \approx \sum_{p \in \mathbb{P}} q_p \delta\left(\boldsymbol{x} - \boldsymbol{x}_p^i\right) \delta\left(\boldsymbol{v} - \boldsymbol{v}_p^i\right). \tag{2.2}$$

Once the particles have been generated, the equations of motion for their trajectories $(\boldsymbol{x}_p(t), \boldsymbol{v}_p(t))$ can be obtained by substituting Equation (2.2) into Equation (1.1). The result is the following system of ODEs:

$$\begin{aligned} \frac{dq_p}{dt} &= 0 \\ \frac{d\boldsymbol{x}_p}{dt} &= \boldsymbol{v}_p \\ \frac{d\boldsymbol{v}_p}{dt} &= -\boldsymbol{E}_p, \end{aligned} \tag{2.3}$$

where $\boldsymbol{E}_p$ is the acceleration on particle p induced by the surrounding charge distribution.

Equation 2.3, when coupled with a procedure for computing the electric field at the particle positions, can then be used to numerically advance Equations (1.1) and (1.2) in time. In PIC methods, this time advance is computed in a number of stages, as follows. First, the charge density is computed on a grid via a deposition step, in which each particle distributes its charge into a number of cells using some interpolating function. These functions are commonly taken to be one of the B-splines, e.g. the "Cloud-In-Cell" (CIC) and "Triangle-Shaped Cloud" (TSC) functions. However, while higher-order B-splines have increasing degrees of smoothness, they are limited to interpolating with at most 2nd-order accuracy [22]. Next, Poisson's equation for the potential is solved on the grid using a finite difference method along with some kind of fast Poisson solver, such as fast Fourier Transforms or multigrid methods.

Next, the electric field is computed on the grid using a finite difference approximation to the gradient. Once the field is known at the grid points, it can be interpolated back to the particle positions. It is important that this be done using the same interpolating function as in the deposition step, in order to avoid spurious self-forces. Finally, once the electric field at the particle positions is known, the particles can be advanced in time using some numerical ODE solver. If this solver contains multiple stages, such as with Runge-Kutta methods, this entire procedure will need to be completed several times per time step.

The final ingredient needed for accurate PIC calculations, as discussed in [14] and [17], is a particle remapping procedure. During the remap, the current set of particles is replaced by a new set that encodes the same underlying distribution function. This process is similar to the particle initialization procedure described above, except that instead of generating the particles from a given initial distribution function, we generate them by depositing the known particle distribution onto a high-dimensional, Cartesian grid in phase space and sampling the resulting distribution at each cell center. It is important to note that, unlike with pure grid methods, this process does not require storing the entire high-dimensional grid in phase space at once; rather, because each particle only contributes charge to nearby positions, it can be done on purely local chunks in phase space. It is also important to note that this process does not need to be done every time step. Once the new particles have been generated, the calculation can continue. This procedure prevents errors in the particle trajectories from compounding to the point that they significantly degrade the solution.

They key factors affecting the accuracy of PIC methods are thus 1) the accuracy of the interpolating function used for charge deposition and force interpolation, 2) the accuracy of the finite difference stencils used for the field solve, 3) the accuracy of the time integration scheme, and 4) the accuracy of the interpolating function used for the remap. In what follows, we first review a basic PIC method that is 2nd-order accurate (very similar to the one from [14]), and then we describe a new PIC method that extends the first scheme to 4th-order accuracy.

2.2. A 2nd-order PIC method. The stages of the 2nd-order PIC algorithm proceed as follows.

- **Particle Deposition** The charge density is defined on a Cartesian mesh $\boldsymbol{x_i} = (\boldsymbol{i} + 1/2)\,\Delta\boldsymbol{x}$, where $\boldsymbol{i} \in \mathbb{Z}^D$ are the cell indices and $\Delta\boldsymbol{x}$ is the cell spacing. At second order, we can use the following deposition step:

$$\rho_{\boldsymbol{i}} = \sum_p \left(\frac{q_p}{V_i}\right) \boldsymbol{W_2}\left(\frac{\boldsymbol{x}_i - \boldsymbol{x}_p}{\Delta x}\right). \tag{2.4}$$

  Here, $V_i = \Delta x^D$ is the volume of cell i and $\boldsymbol{W_2}(\boldsymbol{x})$ is a D-dimensional, "Cloud-in-Cell" interpolating function, given by:

$$\boldsymbol{W_2}(\boldsymbol{x}) = \prod_{d=1}^{D} W_2(x_d), \tag{2.5}$$

$$W_2(x) = \begin{cases} 1 - |x|, & 0 \le |x| \le 1, \\ 0 & \text{otherwise.} \end{cases} \tag{2.6}$$

  Note that in general, we do not use the same mesh spacing for the particle discretization and the Poisson mesh, i.e. $\Delta x \neq h_x$.

- **Field Solve** The next step is to solve the Poisson equation for the potential at the same grid points on which the density is defined. At 2nd order, we discretize the Laplacian operator using the standard $2D+1$ point centered difference approximation:

$$-\sum_{d=1}^{D} \frac{\phi_{\boldsymbol{i}+\boldsymbol{e}^d} - 2\phi_{\boldsymbol{i}} + \phi_{\boldsymbol{i}-\boldsymbol{e}^d}}{\Delta x^2} = -\rho_{\boldsymbol{i}}. \tag{2.7}$$

We solve the resulting linear system with geometric multigrid, using Gauss-Seidel with Red-Black ordering as the smoother. Through this paper, we use periodic boundary conditions and set the solver tolerance to 10^{-9}. After iterating to convergence, the electric field components can be computed on the mesh as

$$\boldsymbol{E}_{\boldsymbol{i}}^d = -\frac{\phi_{\boldsymbol{i}+\boldsymbol{e}^d} - \phi_{\boldsymbol{i}-\boldsymbol{e}^d}}{2\Delta x}. \tag{2.8}$$

- **Force Interpolation** Next, we interpolate the field back to the particle positions using the same interpolating function as in the deposition step:

$$\boldsymbol{E}_p = \sum_{\boldsymbol{i}} \boldsymbol{E}_{\boldsymbol{i}} V_i \, \boldsymbol{W}_{\boldsymbol{2}} \left(\frac{\boldsymbol{x}_i - \boldsymbol{x}_p}{\Delta x} \right). \tag{2.9}$$

- **Particle Push** The final stage is the update of the particle positions. To numerically integrate Equation (2.3), we use the following 2nd-order accurate Runge-Kutta (RK2) method:

$$\begin{aligned} \boldsymbol{x}^{n+1} &= \boldsymbol{x}^n + \frac{1}{2}\boldsymbol{k}_1 \Delta t^2 \\ \boldsymbol{v}^{n+1} &= \boldsymbol{v}^n + \frac{1}{2}\left(\boldsymbol{k}_1 + \boldsymbol{k}_2\right)\Delta t, \end{aligned} \tag{2.10}$$

where

$$\begin{aligned} \boldsymbol{k}_1 &= \boldsymbol{E}(\boldsymbol{x}^n) \\ \boldsymbol{k}_2 &= \boldsymbol{E}(\boldsymbol{x}^n + \boldsymbol{v}^n \Delta t). \end{aligned} \tag{2.11}$$

For each stage of the RK2 time step, we must generate a right-hand side for the Poisson equation corresponding to the appropriate particle positions by performing particle deposition, the field solve, and the force interpolation steps.

- **Remapping** The remapping step involves replacing the current set of particles with a new set that is laid out on a Cartesian grid in phase space. As in particle initialization, the cell spacings of this mesh are h_x and h_v in position and velocity space, and we only keep particles with weights that exceed some problem-dependent threshold value. The charges of the new particles q_p^* are then calculated as

$$q_p^* = \sum_p q_p \, \boldsymbol{W}_3 \left(\frac{\boldsymbol{x}_{\boldsymbol{i}} - \boldsymbol{x}_p}{h_x} \right) \boldsymbol{W}_3 \left(\frac{\boldsymbol{v}_{\boldsymbol{j}} - \boldsymbol{v}_p}{h_v} \right), \tag{2.12}$$

where $\boldsymbol{W}_3(\boldsymbol{x})$ is the 3rd-order interpolating function from [14] and [17], given by

$$\boldsymbol{W}_3\left(\boldsymbol{x}\right) = \prod_{d=1}^{D} W_3\left(x_d\right), \tag{2.13}$$

and

$$W_3(x) = \begin{cases} 1 - \frac{5}{2}|x|^2 + \frac{3}{2}|x|^3, & 0 \le |x| \le 1, \\ \frac{1}{2}\left(2 - |x|\right)^2\left(1 - |x|\right), & 1 \le |x| \le 2, \\ 0 & \text{otherwise.} \end{cases} \tag{2.14}$$

A 3rd-order interpolant is necessary here in order for the overall scheme to be 2nd-order accurate, since one order of accuracy is lost in the remap step [14].

2.3. A 4th-order PIC method. We now describe our new, 4th-order method. This time, the PIC update proceeds as follows:

- **Particle Deposition** At fourth order, we can no longer use a B-spline to interpolate from the particle positions to the mesh cells. Instead, we use the following deposition step:

$$\rho_{\boldsymbol{i}} = \sum_p \left(\frac{q_p}{V_i}\right) \boldsymbol{W_4}\left(\frac{\boldsymbol{x}_i - \boldsymbol{x}_p}{\Delta x}\right), \tag{2.15}$$

where $\boldsymbol{W_4}(\boldsymbol{x})$ is the D-dimensional, 4th-order accurate interpolating function from [20], given by:

$$\boldsymbol{W_4}\left(\boldsymbol{x}\right) = \prod_{d=1}^{D} W_4\left(x_d\right), \tag{2.16}$$

$$W_4(x) = \begin{cases} 1 - \frac{|x|}{2} - |x|^2 + \frac{|x|^3}{2}, & 0 \le |x| \le 1, \\ 1 - \frac{11|x|}{6} + |x|^2 - \frac{|x|^3}{6}, & 1 \le |x| \le 2, \\ 0 & \text{otherwise.} \end{cases}$$

- **Field Solve** As before, we solve the Poisson equation for the electrostatic potential at the same grid points on which the density is defined. This time, we discretize the Laplacian operator using a 4th-order centered-difference approximation:

$$-\sum_{d=1}^{D} \frac{-\phi_{\boldsymbol{i}+2\boldsymbol{e}^d} + 16\phi_{\boldsymbol{i}+\boldsymbol{e}^d} - 30\phi_{\boldsymbol{i}} + 16\phi_{\boldsymbol{i}-\boldsymbol{e}^d} - \phi_{\boldsymbol{i}+2\boldsymbol{e}^d}}{12\Delta x^2} = \rho_{\boldsymbol{i}}, \tag{2.17}$$

which we solve using geometric multigrid, as before. The electric field can also be computed using 4th-order centered differences as:

$$\boldsymbol{E}_{\boldsymbol{i}}^d = -\frac{-\phi_{\boldsymbol{i}+2\boldsymbol{e}^d} + 8\phi_{\boldsymbol{i}+\boldsymbol{e}^d} - 8\phi_{\boldsymbol{i}-\boldsymbol{e}^d} + \phi_{\boldsymbol{i}+2\boldsymbol{e}^d}}{12\Delta x}. \tag{2.18}$$

- **Force Interpolation** The force is computed at the particle positions as:

$$\boldsymbol{E}_p = \sum_{\boldsymbol{i}} \boldsymbol{E_i} V_i \, \boldsymbol{W_4} \left(\frac{\boldsymbol{x}_i - \boldsymbol{x}_p}{\Delta x} \right). \tag{2.19}$$

- **Particle Push** Instead of RK2, we use a 4th-order accurate Runge-Kutta (RK4) method. This method assumes the special case of a velocity independent force, and thus has only 3 stages instead of the usual four [23]:

$$\begin{aligned} \boldsymbol{x}^{n+1} &= \boldsymbol{x}^n + \boldsymbol{v}^n \Delta t + \frac{1}{6} \left(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 \right) \Delta t^2 \\ \boldsymbol{v}^{n+1} &= \boldsymbol{v}^n + \frac{1}{6} \left(\boldsymbol{k}_1 + 4\boldsymbol{k}_2 + \boldsymbol{k}_3 \right) \Delta t, \end{aligned} \tag{2.20}$$

where

$$\begin{aligned} \boldsymbol{k}_1 &= \boldsymbol{E}(\boldsymbol{x}^n) \\ \boldsymbol{k}_2 &= \boldsymbol{E}(\boldsymbol{x}^n + \frac{1}{2}\boldsymbol{v}^n \Delta t + \frac{1}{8}\boldsymbol{k}_1 \Delta t^2) \\ \boldsymbol{k}_3 &= \boldsymbol{E}(\boldsymbol{x}^n + \boldsymbol{v}^n \Delta t + \frac{1}{2}\boldsymbol{k}_2 \Delta t^2). \end{aligned} \tag{2.21}$$

As before, for each of the three stages of the RK4 time step, we must generate a right-hand side for the Poisson equation corresponding to the appropriate particle positions by performing the particle deposition, field solve, and force interpolation steps.

- **Remapping**
  As before, in order to preserve the overall accuracy of the method, the remapping step must use a spatial interpolation method that is at least one order higher than the desired order of the method as a whole. Thus, to apply remapping in our 4th-order PIC method, we use the following 6th-order interpolation function from [20] to perform the remap:

$$\boldsymbol{W_6}\left(\boldsymbol{x}\right) = \prod_{d=1}^{D} W_6\left(x_d\right), \tag{2.22}$$

$$W_6(x) = \begin{cases} 1 - \frac{|x|}{3} - \frac{5|x|^2}{4} + \frac{5|x|^3}{12} + \frac{|x|^4}{4} - \frac{|x|^5}{12}, & 0 \le |x| \le 1, \\ 1 - \frac{13|x|}{12} - \frac{5|x|^2}{8} + \frac{25|x|^3}{24} - \frac{3|x|^4}{8} + \frac{|x|^5}{24}, & 1 \le |x| \le 2, \\ 1 - \frac{137|x|}{60} + \frac{15|x|^2}{8} - \frac{17|x|^3}{24} + \frac{|x|^4}{8} - \frac{|x|^5}{120}, & 2 \le |x| \le 3, \\ 0 & \text{otherwise.} \end{cases}$$

The new particle charges are then

$$q_p^* = \sum_p q_p \, \boldsymbol{W}_6 \left(\frac{\boldsymbol{x_i} - \boldsymbol{x}_p}{h_x} \right) \boldsymbol{W}_6 \left(\frac{\boldsymbol{v_j} - \boldsymbol{v}_p}{h_v} \right), \tag{2.23}$$

2.4. Positivity Preservation. We display all of the various interpolating functions we use in this work in Figure 2.1. It is important to note that, unlike 2nd-order interpolants, higher-order interpolating functions are not strictly positivity preserving, in that a single particle with negative charge will not produce a uniformly negative charge density when deposited onto a grid. This can happen during the remap stage of the 2nd-order method, and during both the charge deposition and remap stages of the 4th-order method. One way to account for this is to apply a mass-redistribution algorithm to the distribution function during the remap step, following [14]. That is, once we have generated the distribution function on a phase-space patch, we redistribute the undershoot in cell $\boldsymbol{i}$,

$$\delta f_{\boldsymbol{i}} = \min(0, f_{\boldsymbol{i}}^n) \tag{2.24}$$

to its neighbors $\boldsymbol{i} + \boldsymbol{l}$ in proportion to their capacity ξ,

$$\xi_{\boldsymbol{i}+\boldsymbol{l}} = \max(0, f_{\boldsymbol{i}+\boldsymbol{l}}^n) \tag{2.25}$$

so that

$$f_{\boldsymbol{i}+\boldsymbol{l}}^{n+1} = f_{\boldsymbol{i}+\boldsymbol{l}}^n + \frac{\xi_{\boldsymbol{i}+\boldsymbol{l}}}{\sum_{\boldsymbol{k}\neq 0}^{\text{neighbors}} \xi_{\boldsymbol{i}+\boldsymbol{k}}} \delta f_{\boldsymbol{i}}. \tag{2.26}$$

Here, n and $n+1$ refer to the value before and after redistribution. In general, this procedure might need to be repeated for several iterations before the distribution function is strictly positive. In the problems we consider in this paper, we find that 2 or 3 iterations is sufficient.

We perform this redistribution procedure during the remapping phase on all the runs presented in Section 3. However, we find that turning this off makes only a negligible difference on the test problems considered below. Because this may not be true for all problems, however, particularly those with large gradients in the distribution function, we include the positivity preservation algorithm here for completeness.

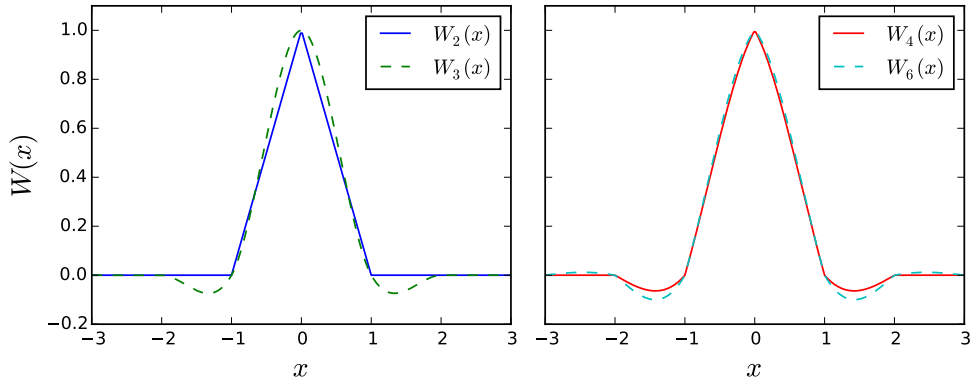

FIG. 2.1. *The interpolating kernels used in the various deposition / interpolation operations in this paper. Left panel - the kernels used by the 2nd-order method. Right panel - the kernels used by the 4th-order method.*

2.5. Implementation. We have implemented both of the above algorithms using the Chombo [21] software framework for solving partial differential equations, which includes tools for doing parallel particle simulations that were originally developed in [24]. The parallelization uses standard message passing with block-structured domain decomposition in physical space. This code, as well as our data analysis scripts, is available online [1]. Our one-dimensional results presented in this paper were generated on a Macbook Pro, while the two-dimensional results were run on NERSC's Edison machine using up to 256 MPI processes.

3. Numerical Results. In this section, we compare the performance of the 2nd- and 4th-order PIC methods on a set of standard test problems. We begin by considering two one-dimensional test problems, and then move on to perform similar tests in two spatial dimensions. In these numerical tests, we have fixed the ratio of $\Delta x/h_x$ to be 2. To compute convergence rates, we have used Richardson extrapolation to compute our error estimates. That is, if $\boldsymbol{E}^h$ is the electric field computed at a given spatial resolution and time step, and $\boldsymbol{E}^{2h}$ is the electric field computed with *all* the discrete elements of the problem (Δx, h_x, h_v, and Δt) coarsened by a factor of 2, then the solution error is defined as

$$e^h = |\boldsymbol{E}^h - \boldsymbol{E}^{2h}|, \tag{3.1}$$

and the order of the method q is computed as

$$q = \log_2 \left(\frac{||e^{2h}||}{||e^h||} \right). \tag{3.2}$$

When spatial interpolation is required to compare the electric field between runs with different resolutions, we have used cubic spline interpolation (as implemented in SciPy [2]), so as not to mask the 4th-order convergence results.

3.1. 1D Linear Landau Damping. The first test problem we consider is Linear Landau Damping - the damped propagation of a small-amplitude plasma wave. To begin, we perform the calculation in one spatial dimension. We take the initial distribution function to be

$$\begin{aligned} f(x, v, t = 0) &= \frac{1}{\sqrt{2\pi}} \exp\left(-v^2/2\right) (1 + \alpha \cos(kx)) \\ (x, v) &= [0, L = 2\pi/k] \times [-v_{\max}, v_{\max}], \end{aligned} \tag{3.3}$$

where the amplitude of the perturbation α is 0.01, its wavenumber k is 0.5, and $v_{\max} = 10$. This problem uses periodic boundary conditions on the physical space domain $x \in (0, L)$. During particle initialization and during each remap, we have discarded particles with weights less than 10^{-16}.

This problem has been used extensively as a test for plasma PIC codes. According to the analytic theory, the electric field is supposed to oscillate with a frequency $\omega = 1.416$, and the amplitude is supposed to be exponentially damped at a rate $\gamma = 0.1533$. We compare the results of our 2nd and 4th order PIC methods, both with and without remapping, to the analytic theory in Figure 3.1. The numerical solutions used $N_{\text{cells}} = 64$ cells for the Poisson solve, $N_x = 128$ and $N_v = 256$ for the initial particle grid, and a PIC time step of $dt = 1/32$. In the runs that used remapping, we

[1] https://bitbucket.org/atmyers/4thOrderPIC

applied the remap every 5 time steps. All of the runs track the expected damping rate well at early times. As in [14], however, the runs without remapping become noisy and fail to track the exact damping rate at late times. For both the 2nd-order and the 4th-order PIC methods, remapping appears to be necessary for long time evolutions.

Next, we compare the accuracy of the 4th-order and 2nd-order PIC methods on this problem. We conduct a resolution study starting at $N_{\text{cells}} = 32$, $N_x = 64$, $N_v = 128$ and $dt = 1/16$. We do four runs total, doubling the resolution and halving the time step with each successive calculation. We plot the max norm of the error in the electric field versus time in Figure 3.2, where we have used Richardson extrapolation to the estimate the error between each consecutive pair of resolutions. As expected, the 4th-order method is much more accurate than the 2nd-order method when run at the same resolution, by as much as two orders of magnitude for the highest resolution studied. Alternatively, the lowest resolution 4th-order runs are about as accurate as the highest resolution 2nd-order runs on this test problem.

The question of how often to apply the remapping procedure is an important consideration, and in general the answer will be problem dependent. In this test, we are concerned with demonstrating the 4th-order accuracy of our method. It is therefore crucial that the particle trajectory errors addressed by the remap be kept small compared to all the other sources of numerical error, so that 4th-order convergence can be observed. Experimentation suggests that, in order for this to happen, the remap needs to be applied every 5-10 time steps, which translates to remapping 100-200 times on the $N_{\text{cells}} = 256$ version of this test. However, for practical applications, it may not be necessary to keep the trajectory errors so small. Indeed, we find that remapping the particle distribution as few as 9 times over the course of the calculation greatly improves the situation, allowing us to track the expected damping rate all the way to $t = 30$ (see Figure 3.3).

We have enforced positivity preservation on all of our runs for this test problem; however, in this case, we find that turning it off makes no difference to the electric field to within machine precision.

3.2. 1D Two-Stream Instability. Next, we study the two-stream instability, again working in one spatial dimension. In this problem, there is a counter-streaming plasma flow in velocity space, along with a small initial density perturbation. The initial distribution function for this test is:

$$\begin{aligned} f(x, v, t = 0) &= \frac{1}{\sqrt{2\pi}} v^2 \exp\left(-v^2/2\right) (1 + \alpha \cos(kx)) \\ (x, v) &= [0, L = 2\pi/k] \times [-v_{\max}, v_{\max}], \end{aligned} \tag{3.4}$$

where we take $\alpha = 0.01$, $k = 0.5$. We have again adopted periodic boundary conditions in physical space and discarded particles with weights less than 10^{-16}.

To begin, we conduct a 4th-order run with $N_{\text{cells}} = 256$, $N_x = 512$, $N_v = 1024$, and $dt = 1/128$. The time evolution of the phase-space distribution function is shown in Figure 3.4. To construct the distribution function, we have used Equation (2.23) to deposit the particles onto a 512 by 1024 mesh in phase space. We again apply the particle remap every 5 PIC time steps; however, as in Section 3.1, we find that applying the remap as few as 9 times over the simulation greatly reduces the degree of particle noise visible in the solution (Figure 3.5).

Next, we conduct a resolution study as in Section 3.1. We start at $N_{\text{cells}} = 32$, $N_x = 64$, $N_v = 128$, and $dt = 1/16$, and once again conduct four runs, doubling

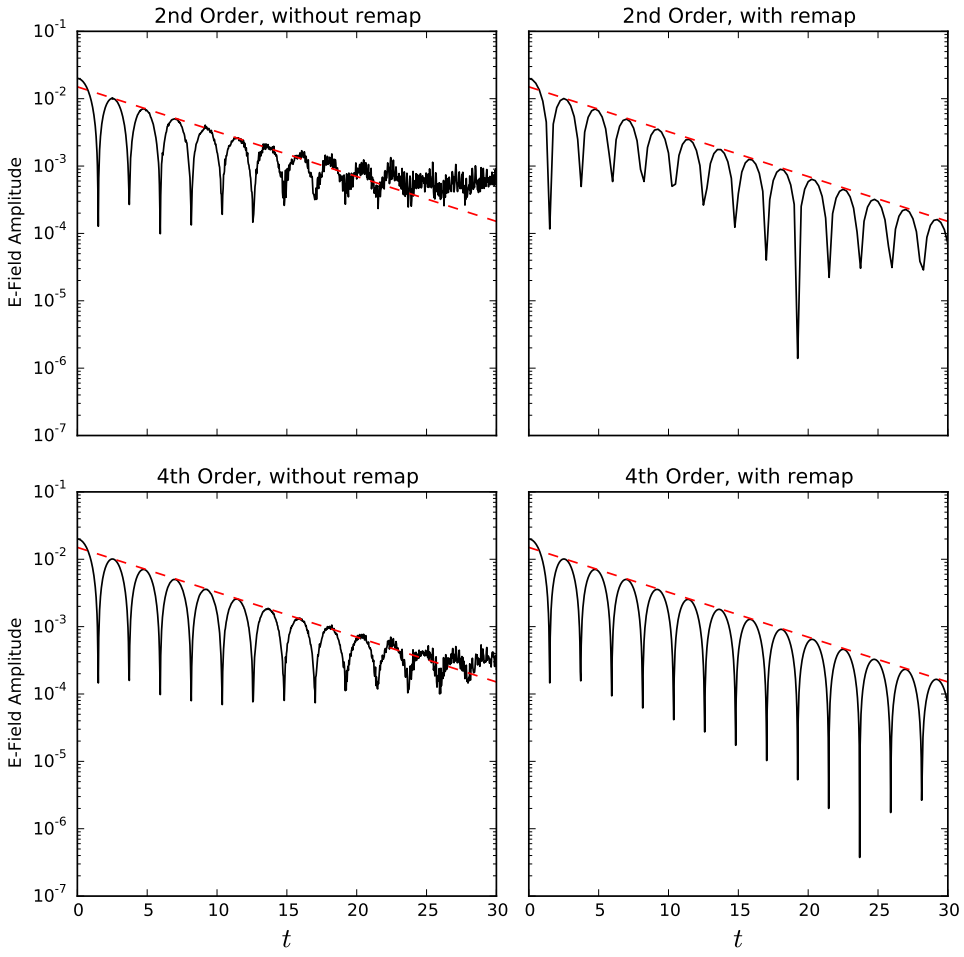

FIG. 3.1. *Amplitude of the electric field as a function of time for the one-dimensional linear Landau damping problem. The black solid line shows the numerical solution, while the red dotted line shows the theoretical damping rate. Top left - 2nd-order method, without remapping. Top Right - 2nd-order method, with remapping. Bottom Left - 4th-order method, without remapping. Bottom Right - 4th-order method, with remapping. All numerical calculations were performed with* $N_{cells} = 64$, $N_x = 128$, *and* $N_v = 256$, *and* $dt = 1/32$.

the resolution and decreasing the time step with each run. The resulting Richardson errors are shown in Figure 3.6. As in the Landau damping example, at early times the 4th-order method is much more accurate than the 2nd-order method, by as much as two orders of magnitude at the resolutions we investigate. In the two-stream test, however, the errors grow significantly with time in both the 2nd-order and the 4th-order cases. This is due to the formation of thin, filamentary structures in phase space, as visible in the bottom right panel of Figure 3.4. These features are not well-resolved at the resolutions we use in our convergence study, and hence reduce the accuracy of the simulation from the expected order, such that, at very late times, the errors made by the two methods are comparable. However, the 4th-order method delays this loss of accuracy due to filamentation considerably. As late as time 20, for instance, the

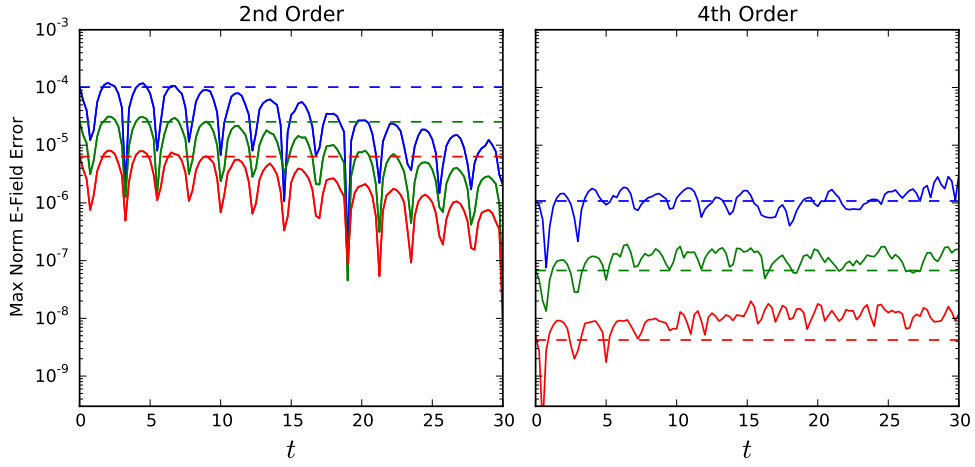

FIG. 3.2. *Max norm of the error in the electric field as a function of time for the one-dimensional linear Landau damping problem. Left - 2nd order method. Right - 4th order method. In both plots, the solid colored lines refer to the Richardson errors associated with consecutive pairs of resolutions, progressing from low (blue), to middle (green), to high (red). The dotted colored lines show how the initial errors should decrease if the methods performed at exactly 2nd (left) and 4th (right) orders.*

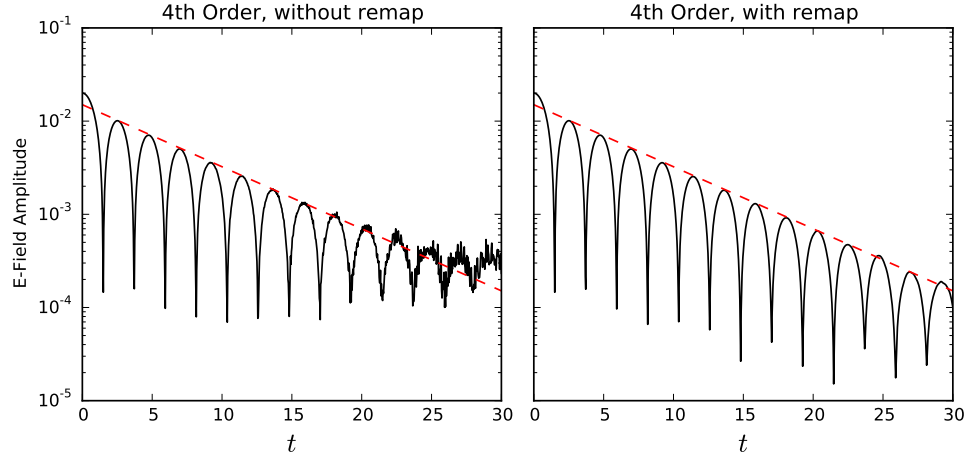

FIG. 3.3. *Amplitude of the electric field as a function of time for the one-dimensional linear Landau damping problem. The black solid line shows the numerical solution, while the red dotted line shows the theoretical damping rate. Left - 4th-order method, without remapping. Right - 4th-order method, remapping 9 times over the course of the simulation. All numerical calculations were performed with* $N_{cells} = 64$, $N_x = 128$, *and* $N_v = 256$, *and* $dt = 1/32$.

4th order method is still around 2 orders more accurate than the 2nd-order method.

As before, we have included positivity preservation during the remap step on this test problem. Unlike with the linear Landau damping test, however, the change in the electric field is significantly higher than machine precision (as high as $\sim 10^{-6}$ at late times) on this problem, particularly at late times when filamentation gives rise

to large gradients in the distribution function. This difference is still small, however, compared to the other numerical errors (Figure 3.6), and therefore running without positivity preservation does not affect the 4th-order convergence rates on this problem.

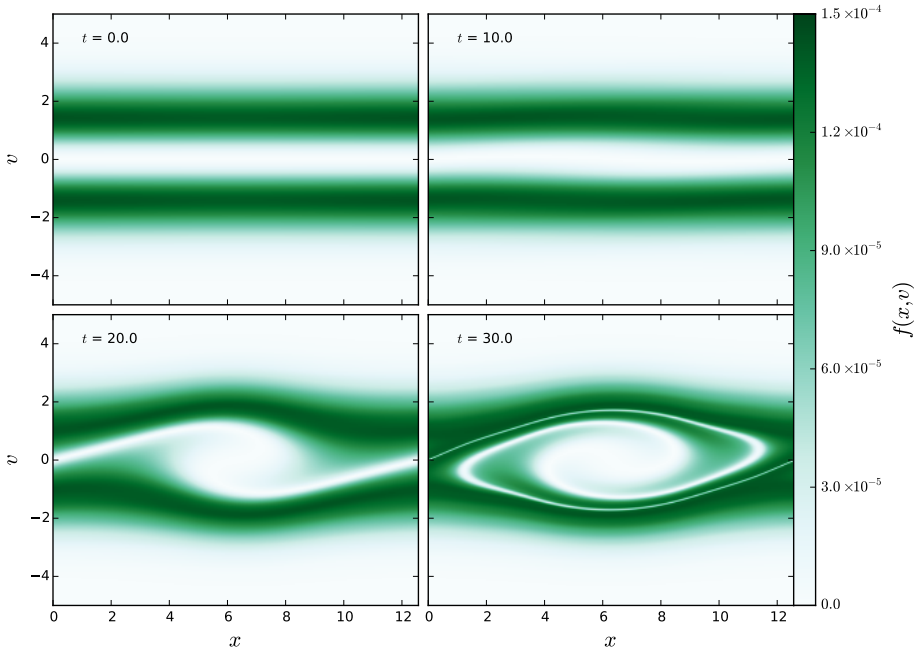

FIG. 3.4. *Phase-space distribution function at four selected times for the one-dimensional, two-stream instability problem. The data for this figure comes from the $N_{cells} = 256$, 4th-order run. For details of how this figure was generated, see Section 3.2*

3.3. 2D Linear Landau Damping. The previous test problems both used one spatial dimension and one velocity dimension. Since certain numerical instabilities may only occur in higher-dimensional problems, it is also important to also check our algorithm on two-dimensional problems. To do so, we first repeat the linear Landau damping problem, this time performing the calculation in two-dimensional space. We

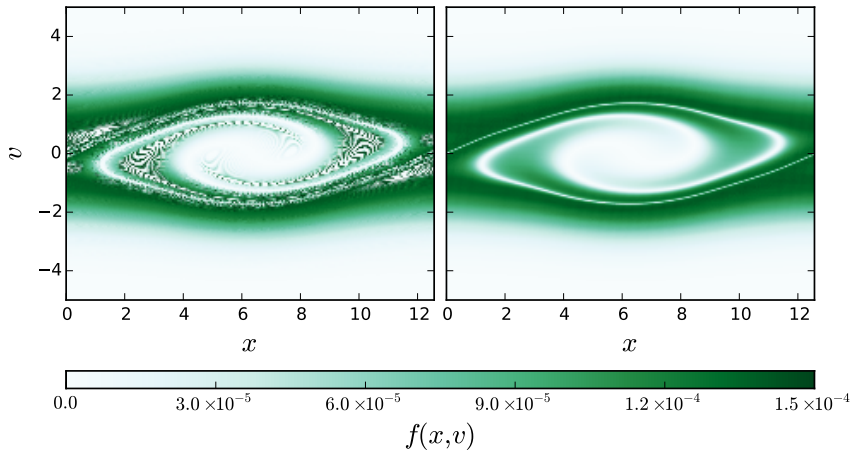

FIG. 3.5. *Phase-space distribution functions for the one-dimensional, two-stream instability problem. Left - 4th-order method, without remapping. Right - 4th-order method, remapping 9 times over the course of the simulation.*

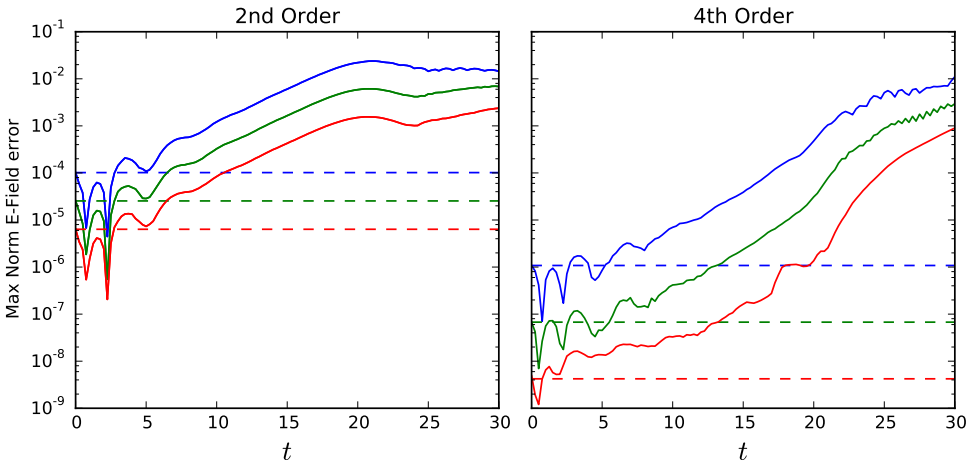

FIG. 3.6. *Max norm of the error in the electric field as a function of time for the one-dimensional, two-stream instability problem. Left - 2nd order method. Right - 4th order method. The line styles and colors have the same meaning as in Figure 3.2.*

take the initial distribution function to be

$$\begin{aligned} f(x, y, v_x, v_y, t = 0) &= \frac{1}{2\pi} \exp\left(-(v_x^2 + v_y^2)/2\right) \left(1 + \alpha \cos\left(k_x x\right) \cos\left(k_y y\right)\right) \\ (x, y) &= [0, L = 2\pi/k_x] \times [0, L = 2\pi/k_y] \\ (v_x, v_y) &= [-v_{\max}, v_{\max}] \times [-v_{\max}, v_{\max}], \end{aligned} \tag{3.5}$$

where we set $\alpha = 0.05$, $k_x = k_y = 0.5$, $v_{\max} = 6.0$, and use periodic boundary conditions in physical space.

As before, this problem has an analytic solution for the damping rate of the electric field amplitude, which for these parameter choices should be $\gamma = -0.394$.

To compare against this theory, we set $N_{\text{cells}} = 32$, $N_x = 64$, and $N_v = 128$, and $dt = 1/16$, and evolve the initial distribution out to time $t = 30.0$. We again apply remapping every 5 PIC time steps and discard particles with weights less than 10^{-12}. The simulation electric field amplitude is compared against the theoretical expectation in Figure 3.7. Even at this relatively low resolution, we find that the numerical result track the expected damping rate well out to time $t \approx 28$.

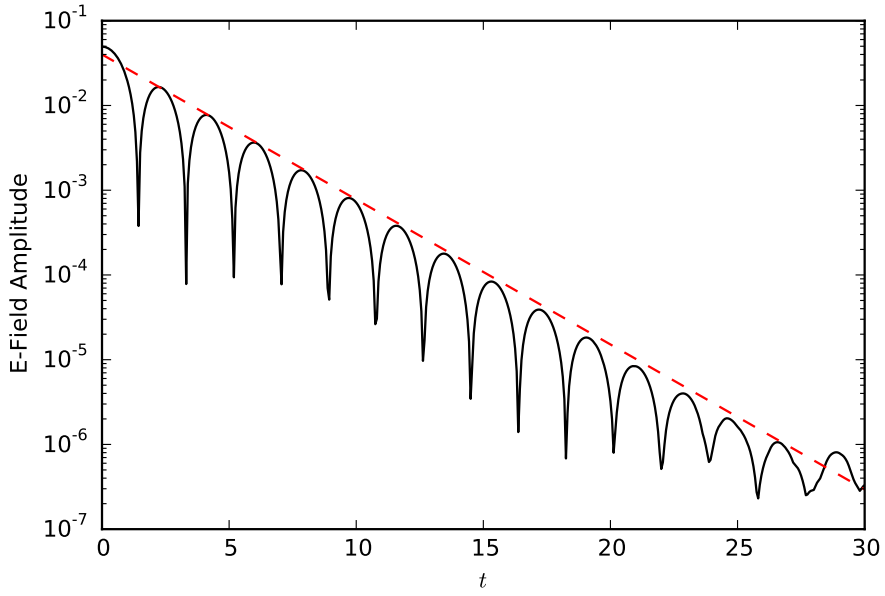

FIG. 3.7. *The amplitude of the electric field versus time for the 2D linear Landau damping problem. The black solid line shows the numerical result for the electric field, while the red-dotted line shows the expected damping rate of* $\gamma = -0.394$.

3.4. 2D Two-Stream Instability. Finally, we perform the two-dimensional version of the two-stream instability problem.The initial distribution function is

$$\begin{aligned} f(x, y, v_x, v_y, t = 0) &= \frac{1}{12\pi} \exp\left(-(v_x^2 + v_y^2)/2\right) (1 + \alpha \cos(k_x x)) \left(1 + 5v_x^2\right) \\ (x, y) &= [0, L = 2\pi/k_x] \times [0, L = 2\pi/k_y] \\ (v_x, v_y) &= [-v_{\max}, v_{\max}] \times [-v_{\max}, v_{\max}], \end{aligned} \tag{3.6}$$

where $\alpha = 0.05$, $k_x = k_y = 0.5$, $v_{\max} = 9.0$, and we again use periodic boundary conditions in physical space.

We set $N_{\text{cells}} = 64$, $N_x = 128$, and $N_v = 256$, and $dt = 1/32$, and evolve the initial conditions out to time $t = 30.0$. As before, we apply remapping every 5 PIC time steps and discard particles with weights less than 10^{-12}. The time evolution of a two-dimensional slice through the phase-space distribution function is shown in Figure 3.8. The generate the $2D$ version of the distribution in (x, v_x) space, we deposit the particles using Equation (2.23) onto a $2D$ mesh with 128 by 256 grid points. That is,

we compute

$$f_{i,j} = \sum_p \left(\frac{q_p}{\Delta x \Delta v}\right) W_6\left(\frac{x_i - x_p}{\Delta x}\right) W_6\left(\frac{v_{x,i} - v_{x,p}}{\Delta v}\right). \tag{3.7}$$

This is equivalent to depositing the particles onto a $4D$ mesh and integrating over y and v_y. As before, when remapping is employed, our 4th-order method tracks the evolution of the distribution function without visible particle noise.

Finally, we perform another convergence study, this time on the 2-dimensional setup. As before, we conduct 4 runs in total, starting at $N_{\text{cells}} = 8$, $N_x = 16$, and $N_v = 32$, and $dt = 1/4$, and increasing the resolution by a factor of 2 with each run. The resulting errors are shown as functions of time in Figure 3.9. As in the one-dimensional version of this problem, our method achieves 4th-order accuracy until late times, when under-resolved filaments reduce the accuracy of the solution to 2nd-order. Figure 3.9 also compares these errors to those made by the 2nd-order method on the same problem setup. As in the one-dimensional case, the 4th-order method is significantly more accurate, particularly at early times. Overall, our method does not appear to suffer from unexpected numerical instabilities when operating in greater than one spatial dimension.

4. Conclusions and Future Research. We have presented a 4th-order accurate Particle-in-Cell algorithm for solving the Vlasov-Poisson equation in the context of electrostatic plasmas, including a remapping step that controls particle noise by periodically re-initializing the particle distribution on a Cartesian grid in phase space. We have demonstrated the accuracy of our method by comparing its performance to that of a 2nd-order method on a set of one-dimensional test problems. We have also gauged our method's performance on a set of two-dimensional test problems, finding that the 4th-order convergence of our method is maintained.

In future work, we will explore extending the current method to 3 spatial and 3 velocity dimensions. While algorithmically straightforward, working in six total dimensions is computationally challenging, particularly during the particle remapping step, which presently involves working with high-dimensional grids. We will address this issue in two ways. First, we will explore phase-space aware parallelization strategies that more naturally map to the work distribution of warm plasma calculations. In contrast, our current parallelization strategy is based on domain decomposition in physical space only, such that the work per process grows more rapidly than the number of parallel domains as the problem size increases. This clearly limits the degree to which our implementation can scale to the high process counts that will be critical in six phase-space dimensions. Second, we will explore grid-free methods of performing the remap step. This will prevent us from needing to allocate temporary six-dimensional grids, which will greatly reduce the memory requirements and communication costs associated with the remap step.

REFERENCES

[1] M. J. Turk, B. D. Smith, J. S. Oishi, S. Skory, S. W. Skillman, T. Abel, and M. L. Norman. yt: A Multi-code Analysis Toolkit for Astrophysical Simulation Data. *The Astrophysical Journal Supplement Series*, 192:9, January 2011.

[2] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed 2015-11-11].

[3] Fernando Pérez and Brian E. Granger. IPython: a system for interactive scientific computing. *Computing in Science and Engineering*, 9(3):21–29, May 2007.

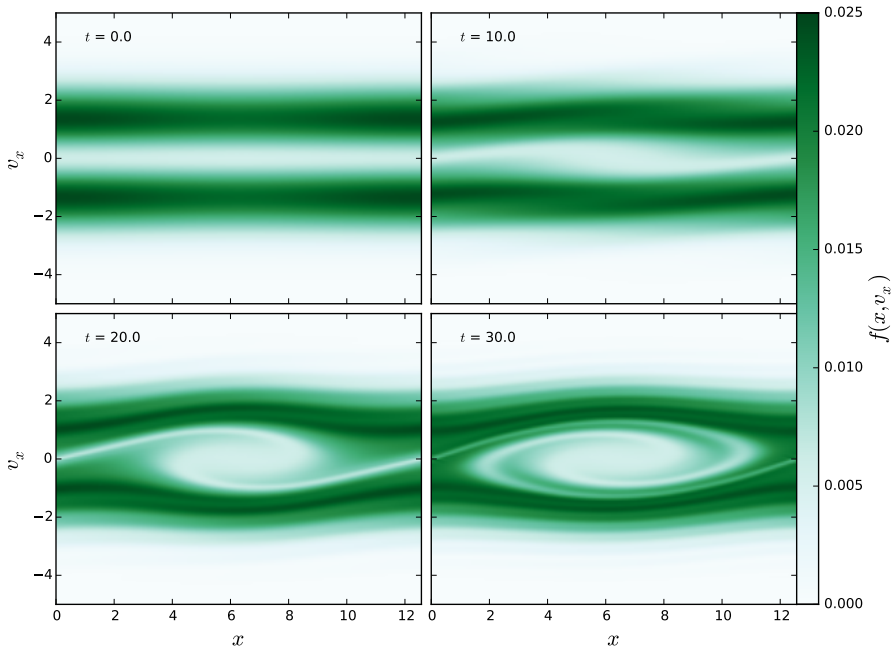

FIG. 3.8. *A 2D representation of the 4D phase space distribution function at four select times for the two-dimensional two-stream instability problem.*

[4] S. van der Walt, S.C. Colbert, and G. Varoquaux. The numpy array: A structure for efficient numerical computation. *Computing in Science and Engineering*, 13(2):22–30, March 2011.

[5] J.D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science and Engineering*, 9(3):90–95, June 2007.

[6] F. Filbet and E. Sonnendrücker. Comparison of Eulerian Vlasov solvers. *Computer Physics Communications*, 150:247–266, February 2003.

[7] J. W. Banks and J. A. F. Hittinger. A New Class of Nonlinear Finite-Volume Methods for Vlasov Simulation. *IEEE Transactions on Plasma Science*, 38:2198–2207, September 2010.

[8] G. V. Vogman, P. Colella, and U. Shumlak. Dory-Guest-Harris instability as a benchmark for continuum kinetic Vlasov-Poisson simulations of magnetized plasmas. *Journal of Computational Physics*, 277:101–120, November 2014.

[9] R. W. Hockney and J. W. Eastwood. *Computer Simulation Using Particles*. 1981.

[10] Katrin Heitmann, Paul M. Ricker, Michael S. Warren, and Salman Habib. Robustness of cosmological simulations. i. large-scale structure. *The Astrophysical Journal Supplement Series*, 160:28–58, September 2005.

[11] D. P. Grote, A. Friedman, J.-L. Vay, and I. Haber. The WARP Code: Modeling High Intensity Ion Beams. In M. Leitner, editor, *Electron Cyclotron Resonance Ion Sources*, volume 749 of *American Institute of Physics Conference Series*, pages 55–58, March 2005.

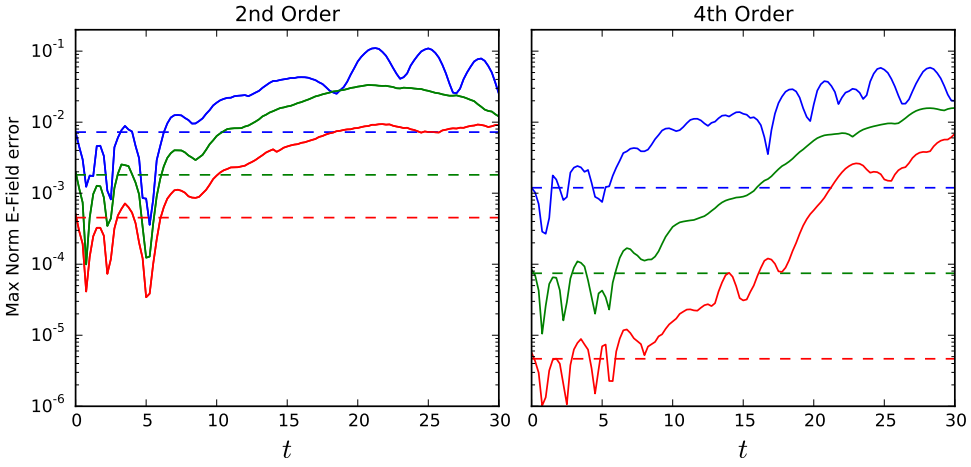

FIG. 3.9. *Max norm of the error in the electric field as a function of time for the 4th-order method on the two-dimensional, two-stream instability problem. Left - the 2nd-order method. Right - the 4th-order method. The line styles and colors have the same meaning as in Figure 3.2.*

[12] E. Edwards and R. Bridson. A high-order accurate particle-in-cell method. *International Journal for Numerical Methods in Engineering*, 90:1073–1088, June 2012.

[13] G. B. Jacobs and J. S. Hesthaven. High-order nodal discontinuous Galerkin particle-in-cell method on unstructured grids. *Journal of Computational Physics*, 214:96–121, May 2006.

[14] B. Wang, G. Miller, and P. Colella. A particle-in-cell method with adaptive phase-space remapping for kinetic plasmas. *SIAM Journal on Scientific Computing*, 33(6):3509–3537, January 2011.

[15] Georges-Henri Cottet and Petros D. Koumoutsakos. *Vortex Methods: Theory and Practice*. Cambridge University Press, March 2000.

[16] A. K. Chaniotis, D. Poulikakos, and P. Koumoutsakos. Remeshed smoothed particle hydrodynamics for the simulation of viscous and heat conducting flows. *Journal of Computational Physics*, 182:67–90, October 2002.

[17] B. Wang, G. Miller, and P. Colella. An adaptive, high-order phase-space remapping for the two dimensional vlasov–poisson equations. *SIAM Journal on Scientific Computing*, 34(6):B909–B924, January 2012.

[18] A. Myers, P. Colella, and B. Van Straalen. The Convergence of Particle-in-Cell Schemes for Cosmological Dark Matter Simulations. *ArXiv e-prints*, March 2015.

[19] Samuel Williams, Andrew Waterman, and David Patterson. Roofline: An insightful visual performance model for multicore architectures. *Commun. ACM*, 52(4):65–76, April 2009.

[20] B. Lo, V. Minden, and P. Colella. A Real-Space Green's Function Method for Numerical Solution of Maxwell's Equations. *in preparation*, 2015.

[21] M. Adams, P. Colella, D. T. Graves, J.N. Johnson, N.D. Keen, T. J. Ligocki, D. F. Martin, P.W. McCorquodale, D. Modiano, P.O. Schwartz, T.D. Sternberg, and B. Van Straalen. *Chombo Software Package for AMR Applications - Design Document*, Lawrence Berkeley National Laboratory Technical Report LBNL-6616E.

[22] J. J. Monaghan. Particle methods for hydrodynamics. *Computer Physics Reports*, 3:71–124, October 1985.

[23] M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions*. 1972.

[24] F. Miniati and P. Colella. Block structured adaptive mesh and time refinement for hybrid, hyperbolic + N-body systems. *Journal of Computational Physics*, 227:400–430, November 2007.