A Particle-in-cell Method with Adaptive Phase-space Remapping for Kinetic Plasmas

by

BEI WANG

B.S. (Southwest Jiaotong University, P.R.China) 2003
M.S. (University of California at Davis) 2005

DISSERTATION
Submitted in partial satisfaction of the requirements for the degree of
DOCTOR OF PHILOSOPHY

in

Applied Science

in the

OFFICE OF GRADUATE STUDIES
of the
UNIVERSITY OF CALIFORNIA
DAVIS

Approved:

Professor Gregory H. Miller, Chair

Dr. Phillip Colella

Professor Ann E. Orel

Committee in Charge

2011

Bei Wang
June 2011
Applied Science

A Particle-in-cell Method with Adaptive Phase-space Remapping for Kinetic Plasmas

Abstract

The numerical solution of the Vlasov-Poisson equation is usually performed by the particle-in-cell (PIC) method. However, it is well-known that, in some cases, the PIC method has difficulty in accurately modeling the phase space distribution function because of numerical noise, an inherent drawback of particle-based methods. In this thesis, a second-order PIC method is presented for computing the dynamics of kinetic plasmas. The method reduces the numerical noise by periodically remapping the distribution function on a hierarchy of locally refined grids in phase space.

The new remapping scheme has several features which result in an accurate and efficient method. First, a conservative high-order interpolation function, i.e., a modified B-spline, is chosen for remapping. This interpolation function approximates a quadratic function exactly without solving a matrix system. Second, a local mass redistribution algorithm is provided to enforce the positivity of the distribution function after high-order interpolation. Third, mesh refinement is introduced to reduce the number of small strength particles, mostly located at the tail of the distribution function. Remapping on a phase space grid also provides an opportunity to integrate a collisional model and an associated grid-based solver. A simplified Fokker-Planck equation is used as the collisional model and solved by a second-order finite volume discretization along with a second-order L_0 stable implicit time integrator. The collisional model is coupled to the Vlasov-Poisson equation with a second-order operator splitting.

A rigorous study of particle methods for the one-dimensional Vlasov-Poisson system has been performed by Cottet and Raviart [1]. We extend the approach to the PIC method for the one-dimensional Vlasov-Poisson equation based on the work in [1],

To my parents and Zhe

Contents

List of Figures

List of Tables

Acknowledgements

I would like to give my tremendous thanks and appreciation to my thesis advisors, Greg Miller and Phil Colella, for their wisdom, support and understanding. Greg has helped me to walk through every single phase of my graduate school. I do not think I could be able to finish this thesis without his encouragement and support. Special thanks goes to Phil, who has triggered my great interest in applied mathematics and computational science. I am alway impressed by Phil's physical intuition and immense knowledge of numerical mathematics. Much of the work in this thesis is built on the foundation of his ideas. I could not possibly have better advisors than them. I would like to thank Prof. Ann Orel for her carefully reading of the thesis draft and for her help during my time at Applied Science of UC Davis.

I also own a sincerely thanks to my colleagues in Applied Numerical Algorithm Group of Lawrence Berkeley National Laboratory and Academic Surge 1026 of UC Davis, for their helps in code development, supportive conservation and great time together. Jeff has helped me from Livermore to Berkeley and from the oral preparation to the draft of this thesis... I have asked tons of questions about Chombo code from Dan Graves, Dan Martin, Brian... They always answered my questions very patiently and quickly. Thanks for my officemates, Bakytzhan, Wan-Chi and Mehdi, in particular Bakytzhan, for our great time together. Thank you all!

Finally, I would like to thank my family for their unconditionally support throughout the years. Special thanks also goes to Prof. Shenggang Liu, who encouraged and recommended me to pursue my graduate study in United State.

The following funding support for the thesis work is acknowledged:

- the U.S. Department of Energy Office of Advanced Scientific Computing Research under contract number DE-AC02-05CH11231 at the Lawrence Berkeley National Laboratory.

Chapter 1

Introduction

1.1 Motivation

Computational plasma physics plays an important role in controlled thermonuclear fusion, accelerator modeling and astrophysics. Basically, there are two types of models for simulating plasmas. The simplest description is a fluid model, for example, magnetohydrodynamics or the two-fluid equations. In a fluid model, we assume that at each point in physical space, the velocity distribution is Maxwellian such that it can be specified by the mean velocity and the temperature. A plasma in this model is then described by averaged quantities over velocity space, such as charge density and velocity that are a function of physical space (x, y, z). However, a fluid model is not suitable for a high temperature plasma where deviations from thermal equilibrium can be maintained for long times. In this case, an accurate description is given by a kinetic model. In a kinetic model, the distribution function is obtained by solving the Boltzmann or Vlasov equations in phase space (x, y, z, v_x, v_y, v_z). The Boltzmann equation describes the evolution of the distribution function including binary collisions, whereas the Vlasov equation describes the distribution without collisions. In this thesis, we mainly consider plasmas in a kinetic model described by the Vlasov equation.

Generally, the Vlasov equation can be solved in two ways: particle methods and grid methods. Particle methods are the most commonly used method in kinetic plasma simulation. They were pioneered by

Buneman [2] and Dawson [3] in the late 1950s and early 1960s. The particles, a Lagrangian discretization of the distribution function, follow trajectories computed from the characteristic curves derived from the Vlasov equation. In the case of electrostatic forces, self-consistent fields are calculated via direct solution of Coulomb's law. The original method was improved by introducing a mesh for the solution of Coulomb's law. Particle-mesh methods, known as cloud-in-cell (CIC) or particle-in-cell (PIC), reduce the complexity of the original method from $O(N^2)$ to $O(MlogM+N)$, where N is the number of particles and M is the number of mesh points. Usually, the value of M is much smaller than N. Particle methods are naturally adaptive, since particles only occupy the domain where the distribution function is not zero. However, they suffer from numerical noise, for example, in simulating problems with large dynamic ranges, where the distribution function has large variations in the magnitude.

An alternative is to employ grid methods to solve the Vlasov equation on a grid in phase space. Grid methods provide a smooth representation of the distribution function, so they are noise free. In addition, methods of solution can be guided by the well-established numerical analysis of classical partial differential equations since the Vlasov equations are simply a nonlinear advection equation in a high-dimensional space. Early studies of grid methods dealt primarily with spectral methods, such as Fourier-Fourier transform by Knorr in 1963 [4], where the distribution function is Fourier transformed in physical and velocity space, and Fourier-Hermite transform by Armstrong and Montgomery in 1969 [5], where the distribution function is Fourier transformed in physical space and is Hermite represented in velocity space. These methods are based on expanding the solution in orthogonal polynomials: high order of accuracy can be achieved by increasing the number of terms. Spectral methods can be very expensive if a large number of terms are used to resolve the problem. The backward characteristics method was introduced by Cheng and Knorr [6] in 1976. In this method, the distribution function is computed by following each grid point along its characteristic curve backwards in time and then obtaining the value at the initial time through high-order interpolations. To compute the characteristic at the initial time, Sonnendrucker et al. [7] presented the cubic spline method and Nakamura and Yabe [8] presented the cubic interpolated propagation method. A forward characteristics method, also called the forward semi-Lagrangian method, has been proposed by Crouseilles et al. [9] that

advances each grid point along the characteristic curve forwards. Finite volume methods were developed by Boris and Book [10–12] in 1970s. In recent years, Fijalkow [13] presented the conservative flux balance method. A similar idea is used in a high-order finite volume method based on mapped coordinates by Colella et al. [14]. Filbet et al. [15] introduced the positive and flux conservative scheme by using limiters with the flux balance method. A finite element method [16] has also been proposed and applied to the solution of the Vlasov-Poisson equation. Although grid-based methods are accurate for problems with large dynamic ranges for which standard particle methods fail, they were not viewed as practical due to high dimensionality and high computational cost. However, there has been a tremendous development of grid-based methods in the past decade due to increases in processing power.

The purpose of this thesis is to investigate the incorporation of a remapping technique to the PIC method in order to reduce numerical noise. Remapping has been used in particle methods in fluid dynamics, i.e., vortex methods and smoothed particle hydrodynamics (SPH), to maintain regularity of the particle distribution and thereby improve accuracy [17, 18], but not widely been used for particle methods in plasma physics. We study the remapping algorithms in [17, 18] applied to the PIC method. The remapping scheme we have developed has several features which result in an accurate and efficient method. First, it is conservative: the total charge before and after remapping is identical. Second, it uses high-order interpolation, based on modified B-splines. Third, we provide two different algorithms to preserve the positivity of the distribution function. Both algorithms conserve the total charge exactly. Finally, we use mesh refinement to reduce the growth of the total number of particles due to remapping. Preliminary results for the algorithm have been obtained in up to four dimensions. Our remapped PIC method is similar to the forward semi-Lagrangian method of Crouseilles et al. [9] in that both methods advance particles along characteristics and periodically reconstruct the distribution in phase space. But, there are three major differences. First, we use a high-order modified B-spline function for remapping instead of standard B-splines. This avoids the solution of a matrix system in determining the new particle weights. Second, we provide a local mass redistribution algorithm to preserve the positivity of high-order interpolation functions. Third, we introduce mesh refinement to remapping for reducing the total number of small strength particles. A rigorous study of particle

methods for the one-dimensional Vlasov-Poisson system has been performed by Cottet and Raviart [1]. We extend the approach to the PIC method for the one-dimensional Vlasov-Poisson equation.

This chapter is organized as follows. First, we introduce the Vlasov equation. Then we review the literature on numerical solution of the Vlasov equations. This includes particle methods, semi-Lagrangian methods and finite volume methods. Discussion about the invariance of the system, e.g., total mass, by different methods is also given. The road map of this thesis will be presented at the end.

1.2 The Vlasov Equation

1.2.1 The Vlasov Equation

The Vlasov equation describing the dynamics of a species of charged particles under electric and magnetic fields is

$$\frac{\partial f_s}{\partial t} + v \cdot \nabla f_s + F \cdot \nabla f_s = 0, \tag{1.1}$$

$$F = \frac{q_s}{m_s}(E + E_{app} + v \times (B + B_{app})), \tag{1.2}$$

where $f_s(x,v,t)$ is the distribution function of species s on phase space $(x,v) \in \mathbb{R}^N \times \mathbb{R}^N$ of dimension $2N$. q_s is the charge and m_s is the mass of the species. E_{app} and B_{app} denote applied electric and magnetic field. The self-consistent fields, E and B, are described by Maxwell's equations:

$$\nabla \cdot E = \frac{\rho}{\varepsilon}, \quad \nabla \cdot B = 0, \quad \frac{dB}{dt} = -\nabla \times E, \quad \frac{dE}{dt} = \frac{1}{\varepsilon\mu}\nabla \times B - J.$$

The parameters ε and μ are the electric permitivity and magnetic permeability of the medium, respectively. ρ and J are the charge density and the current density created by the charge distribution function

$$\rho = q_s \sum_s \int_{\mathbb{R}^N} f_s dv_s, \quad J = q_s \sum_s \int_{\mathbb{R}^N} f_s v_s dv_s. \tag{1.3}$$

Here we consider the Vlasov equation under the self-consistent electrostatic field where the governing equations are reduced to

$$\frac{\partial f_s}{\partial t} + v \cdot \nabla f_s + \frac{q_s}{m_s} E \cdot \nabla f_s = 0, \tag{1.4}$$

$$\rho = q_s \sum_s \int_{\mathbb{R}^N} f_s dv_s, \quad \nabla \cdot \nabla \phi = \rho, \quad -\nabla \phi = \frac{E}{\varepsilon}. \tag{1.5}$$

For plasma problems in this thesis, we consider only two species, ions and electrons, of opposite charge and same magnitude. Since the mass ratio between the ion and the electron is very large, we can treat the electrons with the Vlasov equation and regard the ions as a fixed and uniform background. The equation can be further simplified by normalizing spatial length with the Debye length λ_D, velocity with thermal speed v_{th} and time with $\frac{\lambda_D}{v_{th}}$. The normalized Vlasov equation describing the dynamics of an electron distribution function reads

$$\frac{\partial f}{\partial t} + v \cdot \nabla f - E \cdot \nabla f = 0, \tag{1.6}$$

and

$$\nabla \cdot \nabla \phi = 1 - \int_{\mathbb{R}^N} f dv, \quad -\nabla \phi = E. \tag{1.7}$$

The subscript s is dropped as we only consider the distribution function for electrons. For a periodic system with period $L \in \mathbb{R}^N$, $t \geq 0$, f and E satisfy

$$f(0,v,t) = f(L,v,t), \tag{1.8}$$

$$E(0,t) = E(L,t), \tag{1.9}$$

with solvability condition

$$\int_{x \in [0,L]} \rho(x,t) dx = 0. \tag{1.10}$$

We complete this system by an initial distribution function

$$f(x,v,t=0) = f_0(x,v), \quad x \in [0,L], \quad v \in \mathbb{R}^N. \tag{1.11}$$

1.2.2 Characteristics of the Vlasov Equation

The characteristics of the Vlasov equation give the trajectories of information propagation in phase space. We write the normalized Vlasov equation (equation (1.6) and (1.7)) by introducing the characteristic curves

$$\frac{dZ(t)}{dt} = U(Z(t),t), \tag{1.12}$$

$$\frac{df(Z(t),t)}{dt} = \frac{\partial f}{\partial t} + \frac{dZ(t)}{dt} \cdot \nabla_Z f = \frac{\partial f}{\partial t} + U(Z(t),t) \cdot \nabla_Z f = 0. \tag{1.13}$$

where $Z = (x,v)$ and $U = (v,-E)$ are position and velocity in phase space, respectively, and $Z(t) = Z(t;x,0)$ is the position on the characteristic curve at time t beginning at x at time $t = 0$. This suggests using the method of characteristics for the solution of the system.

Let us denote Z_0 the position in phase space at time $t = t_0$ and $U(t)$ the the velocity in phase space in time interval $(t_0, t_0 + \Delta t)$, the solution of the equation of motion gives us $Z = Z_0 + \int_{t_0}^{t_0+\Delta t} U(t)dt$. Since the solution is constant along a characteristic curve, the distribution function at time $t_0 + \Delta t$ can be solved by

$$f(Z, t_0 + \Delta t) = f(Z_0 + \int_{t_0}^{t_0+\Delta t} U(t)dt, t_0 + \Delta t) \tag{1.14}$$

$$= f(Z_0, t_0) = f(Z - \int_{t_0}^{t_0+\Delta t} U dt, t_0). \tag{1.15}$$

One well-known property of this system is that the distribution function tends to develop steep gradients and small scale structures during the evolution, specifically in velocity space, even with smooth initial profiles. The phenomenon, called "filamentation", can be found in a similar system in fluid dynamics, the vorticity field in an incompressible flow. This requires progressively more computational resources to resolve the distribution function as the filaments develop a ever-diminishing scales.

1.2.3 The Invariants of The System

We recall some classical estimates of the Vlasov-Poisson equation [15]. The distribution function $f(x,v,t)$ is the number density of the charge distribution, so the initial data $f(x,v,0) \geq 0$. The distribution function remains positive for all time t. Since $\nabla_Z \cdot U = 0$, we can deduce that for all function $\beta \in C^1(\mathbb{R}^+, \mathbb{R}^+)$,

$$\frac{d}{dt} \int_{\mathbb{R}^N \times \mathbb{R}^N} \beta(f(x,v,t)dxv) = 0, \quad \forall t \in \mathbb{R}^+. \tag{1.16}$$

For example, all L^p norms, where $\beta(f) = f^p$ and $1 \leq p \leq +\infty$, are conserved. Moreover, the kinetic entropy is conserved where $\beta(f) = f\ln(f)$.

Numerical methods for the Vlasov equation should preserve the invariants of the system to the exact possible. We summarize the invariants below:

- Conservation of total charge (mass)

$$\int f(x,v,t)dxdv = constant \quad \forall t \geq 0, \tag{1.17}$$

- Minimum (Positivity) and maximum value of the distribution function

$$f_{min} \leq f(x,v,t) \leq f_{max} = 0 \quad \forall x,v,t, \tag{1.18}$$

- All L^p norms of the distribution function and kinetic entropy. L^1 is an indication of the positivity and L^2 is an indication of the numerical diffusion of a method.

- Conservation of total energy and momentum

In describing each numerical method, we will discuss their ability to preserve some of these invariants.

1.3 Review of Numerical Methods

Numerical solution of the Vlasov equation began in the 1960s. At that time, due to the limitation of computer power, investigations were only on one-dimensional simulations. The most widely used methods were particle-based methods, such as the PIC method. Some authors also used grid methods, like finite difference methods, finite element methods, characteristics methods and finite volume methods. Later in the 1970s and 1980s when two-dimensional simulations were possible for the PIC method, but not for direct methods, PIC dominated other methods, in particular for high-dimensional calculations. However, due to its inherent numerical noise, the PIC method has difficulty in solving some problems, for example, problem with large dynamic ranges, where large variation in magnitude happens over small distances. We witnessed an significant increase in computer power in the last decade. This makes 2D direct calculation for the Vlasov equation possible. Grid methods therefore have drawn much more attention in recent years. Much of recent literature concerns the direct solution of the Vlasov equation on phase space. There are two major discretization in 2D, 4D and even 5D phase spaces: semi-Lagrangian and finite volume methods. In semi-Lagrangian methods, Sonnendrucker et al. [7] proposed the cubic spline method and Nakamura and Yabe [8] proposed the cubic

interpolated propagation method. Crouseilles et al. [9] presented the forward semi-Lagrangian method. In finite volume methods, Fijalkow [13] presented the conservative flux balance method. Filbet et al. [15] introduced the positive and flux conservative scheme. Recently Filbet [19, 20] also gave a review about different Vlasov solvers.

In this section, we review major numerical methods for solving the Vlasov equation: particle methods, semi-Lagrangian methods and finite volume methods. These methods are all based on the methods of characteristics such that the time step is not constrained by the Courant, Friedrichs and Lewy (CFL) condition. The algorithms are presented for the solution of the one-dimensional Vlasov-Poisson equations and can be generalized to high dimensions naturally.

1.3.1 Particle Methods

In particle methods, the algorithm begins by sampling the initial distribution function on the whole problem domain in phase space. We first cover the problem domain with a rectangular grid with mesh spacing $h_x = L/N_x, h_v = v_{max}/N_v$, where N_x and N_v are the number of mesh point in physical space $x \in \mathbb{R}$ and velocity space $v \in \mathbb{R}$, respectively. This initialization is often called "quiet start". Then we approximate the initial distribution function $f(x,v,t=0)$ by a sum of delta function

$$\tilde{f}(x,v,t=0) = \sum_k q_k \delta(x-x_k)\delta(v-v_k), \tag{1.19}$$

where $q_k = f_0(x_k,v_k)h_x h_v$ denotes the weight of particle k. The particle position (x_k,v_k) is sampled at the cell center of the rectangular grid.

Each particle k then follows the trajectory of the flow starting at $(\tilde{X}_k(t=0) = x_k, \tilde{V}_k(t=0) = v_k)$ with

$$\frac{dq_k}{dt}(t) = 0, \tag{1.20}$$

$$\frac{d\tilde{X}_k}{dt}(t) = \tilde{V}_k(t), \tag{1.21}$$

$$\frac{d\tilde{V}_k}{dt}(t) = -\tilde{E}(\tilde{X}_k(t),t), \tag{1.22}$$

where the tilde symbol denotes that the value is approximated by a particle representation. For example, $(\tilde{X}_k(t), \tilde{V}_k(t))$ is the computed trajectory and $\tilde{E}_k(t)$ is the field induced by the computed particle trajectories. The above ordinary differential equations are then advanced in time using a numerical integration rule, such as the fourth-order Runge Kutta method.

For estimating the electric field, we can use convolution such that

$$\tilde{E}(x,t) = \int_{\mathbb{R}} K(x-y)\left[1-\int_{-\infty}^{\infty}\tilde{f}(y,v,t)dv\right]dy \tag{1.23}$$

$$= \int_{\mathbb{R}} K(x-y)dy - \sum_k q_k \int_{\mathbb{R}} K(x-y)\delta(y-\tilde{X}_k(t))dy, \tag{1.24}$$

where K is the corresponding Green's function for equation (1.7) with periodic boundary conditions. However, the Green's function K due to a point charge is discontinuous at $y = x$. To solve this problem, we replace the exact delta function δ with a smoothed one δ_ε where

$$\int_{-\infty}^{\infty}\delta_\varepsilon(x)dx = 1 \tag{1.25}$$

$$\delta_\varepsilon(x) = \frac{1}{\varepsilon}u(\frac{x}{\varepsilon}). \tag{1.26}$$

Figure (1.1) shows the zeroth- and first-order interpolation functions $u(y)$.

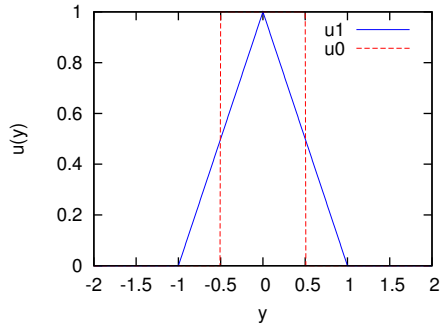


Figure 1.1: The zeroth-order ($u0$) and first-order ($u1$) interpolation functions.

Replacing the delta function in equation (1.24) with a smoothed delta function is equivalent to approximating the distribution function $f(x,v,t)$ with the smoothed delta function,

$$\tilde{f}(x,v,t) = \sum_k q_k \delta_\varepsilon(x-\tilde{X}_k(t))\delta(v-\tilde{V}_k(t)). \tag{1.27}$$

The approximated electric field then takes the form

$$\tilde{E}(x,t) = \int_{\mathbb{R}} K(x-y)\left[1-\int_{-\infty}^{\infty}\tilde{f}(y,v,t)dv\right]dy \tag{1.28}$$

$$= \int_{\mathbb{R}} K(x-y)dy - \sum_k q_k \int_{\mathbb{R}} K(x-y)\delta_\varepsilon(y-\tilde{X}_k(t))dy, \tag{1.29}$$

There are many ways to approximate self-consistent fields in particle methods, for example, fast multipole methods [21]. In this thesis, we use the PIC method where the electrostatic field is solved on a grid through the solution of the Poisson equation. The employment of a grid for the Poisson equation accelerates the method by fast Poisson solvers, such as fast Fourier transformations (FFTs) or multigrid methods. Compared with FFTs, multigrid methods are easier to use with mesh refinement. The algorithm will be given in detail in Chapter 2.

The PIC method conserves the total mass, L^p norms of the distribution function, as well as the kinetic entropy. However, it does not preserve the maximum value of the distribution function, although it preserves positivity.

1.3.2 Semi-Lagrangian Methods (SLM)

Semi-Lagrangian methods are based on the fact that the distribution function described by the Vlasov equation is constant along the characteristic curves. In semi-Lagrangian methods, the distribution function is approximated at each grid point of the problem domain. It is updated each time step using one of two different approaches. The forward semi-Lagrangian method [9] finds the position of each grid point along the characteristic curves by moving forwards. The updated grid point values are obtained through interpolating to the grid points using some kernel function. The picture of the one-dimensional algorithm is shown in Figure (1.2). The forward semi-Lagrangian method is very similar to our remapped PIC method. The major difference is it requires one to solve a matrix system to determine the weights of the distribution function on grid points. Another method is the backward semi-Lagrangian method. It is often referred as the semi-Lagrangian method. Instead of traveling along the characteristic curve forwards in time, the semi-Lagrangian method finds the origin of characteristic ending at grid points. The values at the origin are then

obtained by interpolation (see Figure (1.2)). The most frequently used interpolation methods are Lagrange interpolation and Hermite interpolation. There is a variant of the semi-Lagrangian method developed by Nakamura and Yabe [8]. This method, called the cubic interpolated propagation method, approximates not only the distribution function, but also the derivative of the distribution function. It has advantage of being local interpolation, so it is especially suitable for parallel computing. However, since this method requires one to store and approximate the derivative of the distribution in additional to the distribution function itself, it is more expensive.

Frequently, the semi-Lagrangian method for the Vlasov equation is used with operator splitting. By using operator splitting, the equation is simplified to a set of one-dimensional advection problems. For example, when using the second-order Strang's splitting (see 4.3.2), the algorithm in a single step is the following:

$$f^*(x,v) = f(x - v\Delta t/2, v, t^n), \tag{1.30}$$

$$f^{**}(x,v) = f^*(x, v - E^*(x)\Delta t), \tag{1.31}$$

$$f(x,v,t^{n+1}) = f^{**}(x - v\Delta t/2, v). \tag{1.32}$$

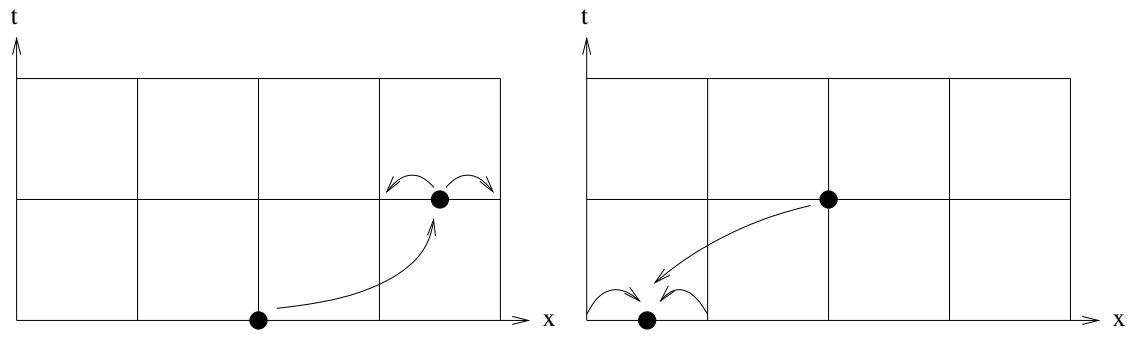


Figure 1.2: Update the distribution function using characteristics in one dimension. (Left) the forward semi-Lagrangian method. (Right) the backward semi-Lagrangian method.

In the general situation, the semi-Lagrangian method does not conserve the total mass. If the advection equation is linear with constant coefficient, like the electrostatic Vlasov equation, the total mass

will be conserved by using a centered approximation in interpolation [19]. The semi-Lagrangian method does not preserve the positivity of the distribution function.

1.3.3 Finite Volume Methods

Finite volume methods are based on the conservative form of the Vlasov equation. As in the semi-Lagrangian method, we use operator splitting and solve the system through a set of one-dimensional advection problems. We present the algorithm in one-dimensional under conservative form

$$\frac{\partial f(x,t)}{\partial t} + \frac{\partial}{\partial x}(vf(x,t)) = 0. \tag{1.33}$$

As we known from the characteristics of the equation,

$$f(x,t^{n+1}) = f(x - vdt, t^n), \tag{1.34}$$

where $dt = t^{n+1} - t^n$.

A finite volume discretization of this form on a Cartesian grid is

$$f_i^{n+1} = \frac{1}{\Delta x}\int_{x_{i-1/2}-vdt}^{x_{i+1/2}-vdt} f(x,t^n)dx, \tag{1.35}$$

$$= \frac{1}{\Delta x}\int_{x_{i-1/2}-vdt}^{x_{i-1/2}} f(x,t^n)dx + f_i^n + \frac{1}{\Delta x}\int_{x_{i+1/2}}^{x_{i+1/2}-vdt} f(x,t^n)dx. \tag{1.36}$$

where

$$f_i^n = \frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} f(x,t^n)dx. \tag{1.37}$$

We introduce the flux terms

$$\Phi_{i+1/2} = \int_{x_{i+1/2}}^{x_{i+1/2}-vdt} f(x,t^n)dx, \tag{1.38}$$

then the finite volume discretization can be represented as

$$f_i^{n+1} = f_i^n + \frac{\Phi_{i-1/2}(t^n) - \Phi_{i+1/2}(t^n)}{\Delta x}. \tag{1.39}$$

As the distribution function is known at each grid point at time t, the function can be reconstructed through interpolation. Figure (1.3) shows the updating of the distribution function using characteristics in the flux balance method in one dimension.

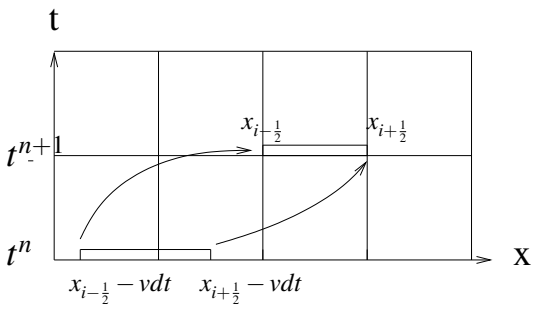


Figure 1.3: Update the distribution function using characteristics in finite volume methods in one dimension.

The flux balance method (FBM) was introduced by Fijalkow [13] in 1999. In this method, equation (1.38) is calculated by reconstructing the distribution function at t^n using first-order interpolation

$$f^n(x) = f_i^n + (x - x_i)\frac{f_{i+1}^n - f_{i-1}^n}{2\Delta x}. \tag{1.40}$$

The flux balance method conserves the total mass. However, the positivity of the distribution function is not preserved.

Based on the flux balance method, Filbet introduced the positive and flux conservative method (PFCM) by using limiters. For details of this method, interested reader can see [15]. Compared with the flux balance method, this scheme preserves both total mass and the positivity of the distribution function.

1.3.4 Summary of Different Methods

A numerical method for the Vlasov equation should preserve the conserved quantities of the system as much as possible. We summarize the invariance of the electrostatic Vlasov equation by different numerical methods in Table (1.4). For a more complete comparison between different grid methods, interested reader can find it in Filbet [20].

We also compare the advantages and disadvantages of particle methods and grid methods below:

Particle methods

Methods	total mass	L^1	maximum value	L^2	kinetic entropy	momentum	total energy
PIC:	✓	✓	–	✓	✓	✓	–
Backward SLM:	✓	–	–	–	–	–	–
Forward SLM:	✓	–	–	–	–	✓	–
FBM:	✓	–	–	–	–	–	–
PFCM:	✓	✓	✓	–	–	–	–

Figure 1.4: Summary of the invariants for the Vlasov-Poisson equation using different methods.

- Advantage:
  - Naturally adaptive since particles only occupy phase space locations where the distribution function is not zero
- Disadvantage:
  - Difficulties in dealing large variations in the magnitude of f and precisely resolving the low particle density region due to numerical noise

Grid methods

- Advantage: Smooth representation of f such that there is no "particle noise"
- Disadvantage: High computational cost in terms of memory usage for high dimensions up to six

1.4 Thesis Overview

In this thesis, we present an accurate and efficient PIC method with adaptive phase-space remapping for the solving of the Vlasov-Poisson equation. The following chapters will be organized as follows.

Chapter 2 reviews the algorithm of the standard PIC method including charge assignment, the solving of the Poisson equation, force interpolation and time integration. The error bound of the charge density and electric field for the PIC method will be given at the end.

Chapter 3 describes the phase-space remapping algorithm. This algorithm controls numerical noise of particle methods through periodically reconstructing the distribution function on a uniform grid using interpolation. We start this chapter by introducing the error in remapping. Then we discuss how to choose an interpolation function for remapping. The remapping scheme we use has several important features: conservation of total charge, high order of accuracy and positivity preservation. These result in a more accurate and stable method. We propose two approaches to enforce the positivity of the distribution function. One is local and another is global. Both algorithms conserve the total mass. In realistic simulations, the global algorithm is not very practical in high-dimensional Vlasov problems since it requires the solving of the Poisson equation in 4D or 6D. However, it will be very useful for preserving positivity in other particle methods, like vortex methods and smoothed particle hydrodynamics.

Remapping on phase space grids also provides us an opportunity to integrate a collision model into the Vlasov equation by solving it on phase space grids with a grid-based method. In Chapter 4, a simplified Fokker-Planck collision model and a second-order grid solver will be described. We couple these two systems through a second-order operator splitting. The coupled system describes the kinetic plasmas under the weakly collisional domain.

In Chapter 5, we introduce the algorithm on a hierarchy of locally refined grids. This requires us to modify each component of the remapping PIC method to work seamlessly on a nested hierarchy of grids. At the interfaces where coarse and fine grids meet, special modification is required for charge deposition and field interpolation, the remapping algorithm, the Poisson solver and the grid solver for the simplified Fokker-Planck equation.

Chapter 6 presents the error analysis of the PIC method for the one-dimensional Vlasov-Poisson equation. The accuracy of particle methods for the one-dimensional Vlasov-Poisson system has been investigated by Cottet and Raviart [1]. We start this chapter by giving a literature review for the convergence

theorem of particle methods in vortex methods. Then, following the work of Cottet and Raviart, we present our error analysis of the PIC method for the one-dimensional Vlasov-Poisson equation. The analysis in this chapter provides a good guidance for the design of an accurate particle method in solving the Vlasov-Poisson equation.

In Chapter 7, the method will be demonstrated by its application to a set of classical plasma problems, including linear Landau damping, nonlinear Landau damping and the two stream instability in both one and two dimensions. A space charged beam in paraxial model with a uniformly focused electric field in two dimensions will be shown at the end. Convergence results will be given for the test problems.

We have developed a multi-dimensional, parallel Vlasov-Poisson solver using the Chombo framework, a C++ and FORTRAN library for solving partial differential equations on block-structured grids with finite volume methods. Code implementation based on Chombo will be presented in Chapter 8. The issue of tuning the solver for scalability on massively parallel computers will be discussed.

Chapter 9 will conclude our current work with some discussion about future directions.

Chapter 2

The Particle-in-cell (PIC) Method on a Uniform Grid

2.1 Algorithms

Particle methods start by representing the distribution function as a set of finite-sized particles (1.27). The representation of the system using finite-sized particles instead of point-sized particles (1.19) has two effects: it reduces collisional effects by modifying the electrostatic potential when two particles are close to each other, and it allows us to have a continuous charge density.

Having a continuous charge density accelerates the solution of the Poisson equation by means of rapid Poisson solvers. In the PIC method, the charge density is defined on a regular grid. Generally, the grid size is chosen to be the same as the stencil size of the smoothed delta function ($\varepsilon = \Delta x$). Then the grid-based Poisson equation is solved by fast Poisson solvers, such as FFTs and multigrid methods. Finally, the grid-based field is interpolated back to particle positions. The flow of the PIC scheme for the solution of equation (1.6) and (1.7) is

- Interpolate particle charges on a Cartesian grid in physical space:

$$\rho(x_j,t) = 1 - \sum_k q_k \delta_\varepsilon(x_j - \tilde{X}_k(t)), \tag{2.1}$$

where $j \in \mathbb{Z}^N$ are the node index on physical space. The index k denotes all particles in phase space which have physical position within the stencil of the smoothed delta function centered at node j. The smoothed delta function has the form

$$\begin{aligned} \delta_\varepsilon(x) &= \frac{1}{\varepsilon} u(\frac{x}{\varepsilon}) & (2.2)\\ &= \prod_{d=0}^{N-1} \frac{1}{\varepsilon_d} u(\frac{x_d}{\varepsilon_d}) & (2.3) \end{aligned}$$

The typical kernels u for δ_ε are shown in Figure 1.1.

- Compute the grid-based approximation to the convolution and evaluate the electric field on the grid using a second-order finite difference

$$\phi_j^H = \sum_{j'} G_{j',j}^H, \quad G_{j',j}^H \approx G(|j'\Delta x - j\Delta x|), \tag{2.4}$$

and

$$E_{d,j}^H = \frac{\phi_{j-e^d}^H - \phi_{j+e^d}^H}{2\Delta x_d}. \tag{2.5}$$

where Δx_d is the Poisson solver mesh spacing in dimension d. With given boundary conditions, the approximation to the Green's function G^H is typically computed by a fast Poisson solver, i.e., FFTs or multigrid methods.

- Interpolate the calculated field back to particle locations:

$$\tilde{E}(\tilde{X}_k,t) = \sum_j E_j^H u(x_j - \tilde{X}_k(t)). \tag{2.6}$$

- Integrate the equation of motion (1.22) numerically, for example, using the second-order leapfrog scheme:

$$\begin{aligned} \tilde{X}_k^n &= \tilde{X}_k^{n-1} + \Delta t \tilde{V}_k^{n+1/2}, & (2.7)\\ \tilde{V}_k^{n+1/2} &= \tilde{V}_k^{n-1/2} - \Delta t \tilde{E}_k^n. & (2.8) \end{aligned}$$

In the following, we will explain each step in detail. The interested reader can also refer two classical books [22, 23]. The algorithm and analysis are based on the one-dimensional system. The algorithm can be extended to high dimension naturally.

2.1.1 Charge Assignment

Hockney and Eastwood [22] interpreted the charge assignment error as convolution and sampling error using Fourier analysis. Cottet and Raviart [1] interpreted this error in terms of moment and discretization errors using the approach in vortex methods. In this section, we review the work of Hockney and Eastwood. In Chapter 6, we will follow the approach in vortex methods as described by Cottet and Raviart.

Assuming that the point particle representation of the electron charge density in the one-dimensional infinite system is defined as

$$n_e(x) = -\sum_k q_k \delta(x - x_k), \tag{2.9}$$

where x_k is the position of particle k, then the continuous charge density represented by finite-sized particles is

$$\begin{aligned} \rho_e(x) &= \int n_e(x')\delta_\varepsilon(x-x')dx' && (2.10)\\ &= n_e(x) * \delta_\varepsilon(x). && (2.11) \end{aligned}$$

If the Fourier transform of $\rho_e(x)$ is denoted as

$$\begin{aligned} \hat{\rho}_e(k) &= \int_{-\infty}^{\infty} \rho_e(x) e^{-ikx} dx && (2.12)\\ &= \hat{n}_e(k)\hat{\delta}_\varepsilon(k) && (2.13) \end{aligned}$$

then the charge density $\rho_e(x)$ can be obtained from the inverse Fourier transform as

$$\rho_e(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \hat{\rho_e}(k) e^{ikx} dk. \tag{2.14}$$

Sampling $\rho_e(x)$ at mesh point j, we get

$$\rho_e(x_j) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \hat{\rho_e}(k) e^{ikx_j} dk. \tag{2.15}$$

From another point of view, if we first sample $\rho(x)$ in equation (2.11),

$$\rho_e(x_j) = \int n_e(x')\delta_\varepsilon(x_j - x')dx', \tag{2.16}$$

the Fourier transform of $\rho_e(x_j)$ will be

$$\hat{\rho_e}'(k) = \sum_j \rho_e(x_j)e^{-ikx_j}, \tag{2.17}$$

and the inverse transform will be

$$\rho_e(x_j) = \frac{1}{2\pi}\int_{k_g} \hat{\rho_e}'(k)e^{ikx_j}dk, \tag{2.18}$$

where $k_g = \frac{2\pi}{\Delta x}$ and Δx is the mesh spacing.

Since equations (2.15) and (2.18) are equivalent, this yields

$$\hat{\rho_e}'(k) = \sum_{n=-\infty}^{\infty} \hat{\rho_e}(k - nk_g) \tag{2.19}$$

$$= \sum_{n=-\infty}^{\infty} \hat{n}_e(k - nk_g)\hat{\delta}_\varepsilon(k - nk_g). \tag{2.20}$$

By comparing equations (2.13) and (2.20), we see the effect of assigning a continuous charge density on a regular grid. The mesh-defined value in the frequency domain ($\hat{\rho_e}'$) is given by the summation of the continuous function in frequency domain ($\hat{\rho_e}$) shifted by multiples of $k_g = \frac{2\pi}{\Delta x}$. This summation leads to a coupling of different frequencies and is often called "alias error". The alias error is closely related to the smoothness of the smoothed delta function since the smoothness of a function can be mapped to the asymptotic decay of harmonic amplitudes as k increase. For example, if the smoothed delta function δ_ε is continuous in all derivatives up to n^{th} derivative, the function in the frequency domain $\hat{\delta}_\varepsilon$ will decay asymptotically as $k^{-(n+1)}$. To reduce alias error, we wish that the delta function has a high degree of smoothness. However, high order of smoothness leads to a large number of stencil cells in the spatial domain. In the PIC method, usually a triangle function is used as the smoothed delta function, where

$$\delta_\varepsilon(x) = \begin{cases} \frac{1}{\varepsilon}\left(1 - \frac{|x|}{\varepsilon}\right) & 0 \le \frac{|x|}{\varepsilon} \le 1 \\ 0 & \text{otherwise,} \end{cases} \tag{2.21}$$

The Fourier transform of the above function is

$$\hat{\delta}_\varepsilon(k) = \frac{\sin(\frac{k}{2})}{\frac{k}{2}}. \tag{2.22}$$

2.1.2 Field Solver

Given boundary conditions, we can solve the Poisson equation on a regular grid with fast Poisson solvers. The most widely used fast Poisson solvers are FFT-based methods and multigrid methods. FFT-based methods are optimal in terms of the number of floating point operations, but they are limited to uniform grids. Multigrid methods are more effective when combined with mesh refinement. In this section, we give an algorithm for solving the Poisson equation using Fourier transforms. The multigrid method with mesh refinement (AMR multigrid) will be given in Chapter 5.

We consider the one-dimensional Poisson equation on domain $x \in [0, L]$ with periodic boundary conditions

$$\frac{d^2\phi(x)}{dx^2} = \rho(x). \tag{2.23}$$

We discretize the problem domain with a regular grid with mesh spacing $\Delta x = L/J$, where $J+1$ is the number of nodes in the domain. First, we apply the discrete Fourier transformation for $\rho(x)$,

$$\hat{\rho}_j = \frac{1}{J}\sum_{j'=0}^{J-1} \rho_{j'} e^{-ijj'2\pi/J}. \tag{2.24}$$

Then the discrete Fourier transformation of ϕ can be obtained as

$$\hat{\phi}_j = -\frac{\hat{\rho_j}}{j^2k^2} \tag{2.25}$$

for $j = 1$ to $J/2$, where $k = 2\pi/L$, and

$$\hat{\phi}_j = \hat{\phi}^*_{J-j} \tag{2.26}$$

for $j = J/2+1$ to $J-1$ which ensures that the potential ϕ remains real. Note that $\hat{\rho}_0 = 0$ since $\int \rho(x)dx = 0$, and $\hat{\phi}_0 = 0$. Finally, $\phi(x)$ can be determined by applying the inverse Fourier transforms to $\hat{\phi}$,

$$\phi_j = \sum_{j'=0}^{J-1} \hat{\phi}_{j'} e^{ijj'2\pi/J}. \tag{2.27}$$

2.1.3 Force Interpolation

As we obtain the grid-based electric field, we can interpolate it back to the particle location as in equation (2.6). The function used to interpolate the field u should be the same as the kernel we use for the smooth delta function. Otherwise, a single particle will experience a self-force [24]. This can be easily shown if we write the self-force induced by a single charge (x_0, q_0) through the algorithm as

$$E_0 = q_0 \sum_{i,j} \frac{1}{\varepsilon} u_a(i - \frac{x_0}{\Delta x}) u_b(\frac{x_0}{\Delta x} - j)(G^H_{i+1,j} - G^H_{i-1,j}) \tag{2.28}$$

$$= \frac{q_0}{\varepsilon} \sum_{i,j} \left(u_a(i - 1 - \frac{x_0}{\Delta x}) u_b(\frac{x_0}{\Delta x} - j) G^H_{i,j} - u_a(i - \frac{x_0}{\Delta x}) u_b(\frac{x_0}{\Delta x} - (j-1)) G^H_{i-1,j-1} \right). \tag{2.29}$$

Assuming that $G^H_{i,j} = G^H_{i-1,j-1}$, i.e., that the discrete Green's function is translation-invariant, the sum vanishes if u are even functions and are the same for both charge interpolation and field interpolation. Here we use subscript a and b to denote the interpolation function for charge deposition and field interpolation, respectively.

2.1.4 Equation of Motion

Finally, the particles are advanced by numerical integration. Usually, the standard leapfrog method or Runge-Kutta (RK) method are chosen as the approximation scheme. High-order RK methods, i.e., the 4^{th} order RK method, are expensive in terms of memory, in particular for high-dimensional calculations. In our numerical experiments, we choose the 2^{rd} order RK method.

2.2 Accuracy of the PIC Method

The accuracy of particle methods for the one-dimensional Vlasov-Poisson system has been investigated by Cottet and Raviart [1]. The convergence analysis shows that the accuracy depends on a number of factors such as the initial particle discretization of the system and the choice of the discrete delta function, as well as the specific physical problem. Our result is based on their work, but with some modification. Instead of estimating electric field error through convolution, we split the electric field error to two parts: consistency

error and stability error. The consistency error is estimated by first measuring the consistency error of the charge density. Then the consistency error of the electric field error is determined by noting that the discrete Laplacian operator is a stable operator. This modification enables us to estimate the consistency error of the charge density by using the smooth interpolation analysis of Schoenberg [25,26]. In addition, the error due to grid-based Poisson solver can be included easily. The stability error of the electric field is estimated using convolution as by Cottet and Raviart.

The derivation of the error estimates for the PIC method is presented in Chapter 6. We summarize the error bound for the charge density and the electric field in solving the one-dimensional Vlasov-Poisson equation with the PIC method below:

$$\begin{aligned} e^{\rho}(x,t) &= |\rho(x,t)-\tilde{\rho}(x,t)| \\ &\leq C_0(T)\left(\underbrace{\varepsilon^2+\varepsilon^2\left(\frac{h_x}{\varepsilon}\right)^2}_{\sim e_c^{\rho}(x,t)}+\underbrace{\left(\varepsilon+\varepsilon\left(\frac{h_x}{\varepsilon}\right)^2\right)(\exp(at)-1)}_{\sim e_s^{\rho}(x,t)}\right), \quad 0\leq t\leq T, \end{aligned} \tag{2.30}$$

and

$$\begin{aligned} e^{E}(x,t) &= |E(x,t)-\tilde{E}(x,t)| \\ &\leq C_1(T)\left(\underbrace{\varepsilon^2+\varepsilon^2\left(\frac{h_x}{\varepsilon}\right)^2}_{\sim e_c^{E}(x,t)}+\underbrace{\left(\varepsilon^2+\varepsilon^2\left(\frac{h_x}{\varepsilon}\right)^2\right)(\exp(at)-1)}_{\sim e_s^{E}(x,t)}\right), \quad 0\leq t\leq T, \end{aligned} \tag{2.31}$$

where $a(T)$ depends on $\left\|\frac{\partial E}{\partial x}(\cdot,t)\right\|_{L^\infty(\mathbb{R})}$ and parameters $a(T)$, $C_0(T)$ and $C_1(T)$ are independent of ε and h_x. The subscript c and s denote the consistency error and the stability error, respectively, and the superscript ρ and E denote the charge density and the electric field error, respectively. Compared with the results by Cottet and Raviart (equation 4.23 of [1]), our error estimate is one order higher than their estimate. For example, the consistency error of the electric field in their paper is $O(\varepsilon^2+h_x\left(\frac{h_x}{\varepsilon}\right)^2)$.

The consistency error for the PIC method depends on the smooth delta function, δ_ε, assuming that the charge density function is smooth. It is very important that the initial particle mesh spacing in physical space, h_x, is less than or equal to the stencil size of the smooth delta function, ε (or Δx). If the linear function

u_1 (Figure 1.1) is used as the kernel function, the consistency error is second order. The stability error is amplified by a time dependent term $\exp(at) - 1$.

Equation (2.31) and (2.32) provide guidance for the design of particle methods in solving the Vlasov-Poisson equation, and for the choice of optimal parameters. First, for convergence, the ratio of the initial particle mesh spacing in physical space and the Poisson solver mesh spacing needs to be bounded such that $\frac{h_x}{\varepsilon} \leq 1$. This is the so-called *overlapping* condition in particle methods. Second, we notice that the consistency error is amplified by a time dependent term (equation (6.49)). This implies that particle methods can lose accuracy for problems with large density change and for problems running over long times. It is very important that we have some mechanism in particle methods to control this factor. We provide such a mechanism in Chapter 3.

Chapter 3

Remapping on a Uniform Grid

The numerical errors in particle methods lead to the disorder of the particles, often referred to as *numerical noise*. Due to numerical noise, the PIC method has difficulty in solving problems with large dynamic ranges, where large variation of magnitude happens in a short distance or in precisely resolving the region where particle population is too low. In order to reduce the noise, we usually have several options. As we discussed in Chapter 1, grid methods are noise free and are of high order of accuracy. However, they are expensive for high-dimensional simulations since we need to store variables on a phase space up to six dimensions. Perturbative methods, e.g., the δf method, enable one to reduce considerably the noise for problems where the plasma is close to an equilibrium. In the δf method, we discretize only the perturbation δf of the distribution function with respect to an equilibrium f_0 with particle methods [27–29]. However, this algorithm does not work very well for problems which are far away from equilibrium.

Here we use a technique called remapping to reduce the numerical errors in particle methods. The idea of remapping is simple. Periodically, we restart the problem by reproducing the distribution function $f(x,v,t)$ on a uniform grid by interpolation. Then a new set of particles are created from the grid to replace the old particle distribution. This is equivalent to resetting the time t in the exponential terms in equations (2.31) and (2.32) to be zero.

People have been using remapping technique extensively in particle methods, such as vortex meth-

ods and smoothed particle hydrodynamics (SPH), for improving accuracy in fluid dynamics [17, 30, 18], but to a much more limited extent in the PIC method in plasma physics [31–33].

In 1985, Beale and Majda [34] first suggested remapping to improve the accuracy of vortex methods for long time calculations. In the beginning, a Gaussian function was used to interpolate the particle distribution to the regularized grid. The Gaussian function has infinite smoothness and can reconstruct a function with second-order accuracy. However, the Gaussian function has large number of supporting points and is not suitable for actual calculations. A certain type of interpolation functions, the so-called B-splines, were then suggested by Schoenberg [25]. B-splines can have any desired degree of smoothness with a compactly supported stencil. As the Gaussian function, the accuracy of B-splines is only limited to second order (see moment condition of Chapter 6 or p. 228 of [17]). Monaghan [35] presented a systematic way to increase the accuracy of a interpolation function based on B-splines while maintaining the smoothness of the function. Those high-order smooth functions are called modified B-splines. Modified B-splines are often used for interpolation in particle methods.

In SPH, remapping has developed as an extension to SPH to improve its overall accuracy without sacrificing the flexibility of a Lagrangian method. Two techniques are usually used to regularize the particle distribution. One method is high-order interpolation as in vortex methods. Another method, called regularized smoothed particle hydrodynamics (RSPH), is based on finding the exact volume occupied by the new and the old particles through Voronoi diagram techniques when their positions are known [36, 37]. Remapping based on high-order interpolation is easier to implement since we do not need to construct the Voronoi diagram to calculate particle volume. In addition, we can obtain higher order accuracy when using high-order interpolation functions.

In particle methods for plasma physics, Denavit's [31] developed a hybrid scheme by combining a particle method and a grid-based remapping. However, in their papers, remapping is used to introduce numerical diffusion to the system to suppress fine structures created in velocity space, so they use low-order interpolation functions. Recently, Vadlamani et. al [32] developed a similar scheme as an extension to Denavit's.

In this thesis, we apply the remapping technique to the PIC method for the solution of the Vlasov equation. Based on the standard remapping scheme, we introduce two new features which result in an accurate and efficient method. First, we use high-order interpolation stencils which improve accuracy but do not preserve positivity. We propose two approaches to enforce the positivity of the distribution function. Both of these methods conserve total charge, and have the property that a locally positive distribution remains unchanged. Second, we use remapping in concert with a hierarchy of locally refined grids. Remapping on a uniform grid tends to create a large number of particles with very small strength in the tail of the distribution function. The situation becomes worse with repeated remapping. We alleviate the problem by using locally refined grids. For example, a finer grid is used to reproduce the main body of the distribution function, while a coarser grid is used for the tail of the distribution. Remapping on a hierarchy of grids effectively reduces the number of small-strength particles. We describe a conservative and positive high-order remapping in this chapter. The remapping algorithm on a hierarchy of locally refined grids will be presented in Chapter 5.

3.1 Conservative High-Order Interpolation

3.1.1 Error in Remapping

Assuming that we reconstruct the particle distribution on a uniform grid through interpolation, the error in replacing the old particle distribution with a new set of particles on the cell center of a grid will depend on the accuracy of the interpolation function u. For simplicity, the analysis in this section is for remapping in one dimension. In real simulations, we apply remapping in all dimensions in phase space, i.e., physical space and velocity space, by using tensor product formulas. This error analysis is performed by Cottet and Koumoutsakos [17] as follows. If the new and old particle positions are denoted by x_i and x_k and their charges are denoted by q_i and q_k, respectively, the new particle charge value through interpolation is

$$q_i = \sum_k q_k u(\frac{x_i - x_k}{h_x}), \tag{3.1}$$

where u is a interpolation function, i is the index for the mesh with mesh spacing h_x and k is the index for particles in the neighborhood. The error in particle represented value, for example, the charge density, by replacing the old particle distribution with the new one is then

$$e(x) = \sum_k q_k \delta_\varepsilon(x_k - x) - \sum_i q_i \delta_\varepsilon(x_i - x). \tag{3.2}$$

Substituting equation (3.1) to the above equation, we get

$$\begin{aligned} e(x) &= \sum_k \left[q_k \delta_\varepsilon(x_k - x) - \left(\sum_i q_k u(\frac{x_i - x_k}{h_x}) \right) \delta_\varepsilon(x_i - x) \right] && (3.3) \\ &= \sum_k q_k \left[\delta_\varepsilon(x_k - x) - \sum_i u(\frac{x_i - x_k}{h_x}) \delta_\varepsilon(x_i - x) \right] && (3.4) \\ &= \sum_k q_k \sum_i [\delta_\varepsilon(x_k - x) - \delta_\varepsilon(x_i - x)] \, u(\frac{x_i - x_k}{h_x}) && (3.5) \\ &\leqq \sum_k q_k \sum_i \sum_\alpha ((x_i - x_k) \cdot \nabla \delta_\varepsilon)^\alpha u(\frac{x_i - x_k}{h_x}). && (3.6) \end{aligned}$$

We see that if

$$\sum_i (x_i - x_k)^\alpha u(\frac{x_i - x_k}{h_x}) = 0, \quad 1 \leqslant |\alpha| \leqslant m - 1, \tag{3.7}$$

then

$$e(x) = O(h_x^m). \tag{3.8}$$

This is the discrete analog of moment conditions in chapter 6 (equation (6.14)).

We notice that equation (3.7) is equivalent to

$$\sum_i x_i^\alpha u(\frac{x_i - x}{h_x}) = x^\alpha, \quad 0 \leqslant |\alpha| \leqslant m - 1, \tag{3.9}$$

which says the interpolation is exact for polynomials of order less than or equal to $m-1$. This brings us to the classical interpolation formula.

The fundamental interpolation analysis is due to Schoenberg [25]. He considers the interpolation function in the following form

$$\tilde{\psi}(x, h) = \sum_{i=-\infty}^{\infty} \psi(x_i) u(\frac{x_i - x}{h_x}), \tag{3.10}$$

where ψ is the interpolated function and u is the interpolating kernel. If $\tilde{\psi}(x_i) = \psi(x_i)$, u is called the ordinary interpolation formula, whereas if $\tilde{\psi}(x_i) \neq \psi(x_i)$, u is called the smooth interpolation formula. In

particle methods, the smooth interpolation formula, e.g., a B-spline type function, is often used, since ordinary interpolation functions usually do not have high order of smoothness and tend to introduce large errors in the interpolated value if the particle position has large fluctuation.

The error in interpolation can be described in terms of the Fourier transform of the interpolation kernel u according to Schoenberg [25] (see p.223 of [17]).

Theorem 3.1.1. *Consider the interpolation formula*

$$\tilde{\psi}(x,h) = \sum_{i=-\infty}^{\infty} \psi(x_i) u(\frac{x_i - x}{h_x}). \tag{3.11}$$

Let the interpolation function $u(y)$ decay fast enough to satisfy the condition

$$|u(y)| \leq Ae^{-B|y|}, \quad where \quad A > 0, B > 0. \tag{3.12}$$

The formula is of degree m if the following two conditions holds simultaneously:

1. *$g(k) - 1$ has a zero of order m at k=0,*
2. *$g(k)$ has zeros of order m at all k=$2\pi i$ ($i \neq 0$),*

where $g(k)$ is the Fourier transform of $u(y)$

$$g(k) = \int_{-\infty}^{\infty} u(y) e^{-iky} dy. \tag{3.13}$$

The first assumption corresponds to moment conditions in Chapter 6 (equation 6.14) when translated back to physical space (see also p. 224 of [17]).

3.1.2 Smooth Interpolation Kernel

Schoenberg has introduced a certain type of interpolation kernels, called the central B-splines. The B-splines of order $n = 2, 3, 4$ are given below (see Figure (3.1)):

- $n = 2$

$$M_2(y) = \begin{cases} 1 - |y|, & \text{if } 0 \le |y| \le 1, \\ 0, & otherwise, \end{cases} \tag{3.14}$$

- $n = 3$

$$M_3(y) = \begin{cases} \frac{1}{2}(|y| + \frac{3}{2})^2 - \frac{3}{2}(|y| + \frac{1}{2})^2, & \text{if } 0 \le |y| \le \frac{1}{2}, \\ \frac{1}{2}(\frac{3}{2} - |y|)^2, & \text{if } \frac{1}{2} \le |y| \le \frac{3}{2}, \\ 0, & otherwise, \end{cases} \tag{3.15}$$

- $n = 4$

$$M_4(y) = \begin{cases} \frac{1}{6}(2 - |y|)^3 - \frac{4}{6}(1 - |y|)^3, & \text{if } 0 \le |y| \le 1, \\ \frac{1}{6}(2 - |y|)^3, & \text{if } 1 \le |y| \le 2, \\ 0, & otherwise. \end{cases} \tag{3.16}$$

The B-splines of order n have continuity of order $n-2$. However, since the Fourier transforms of B-splines have the form

$$g(k) = h_x \left[\frac{\sin(\pi k h_x)}{\pi k h_x} \right]^n, \tag{3.17}$$

they are limited to second order accuracy (condition 1 of theorem 3.1.1).

3.1.3 Ordinary Interpolation Kernel

Another type of interpolation is called ordinary interpolation. The starting point of this interpolation is to construct a function satisfying

$$\sum_i x_i^m u(\frac{x - x_i}{h_x}) = x^m, \tag{3.18}$$

with m the desired order of accuracy. The second-order ordinary interpolation coincides with the B-spline of order two, M_2. The third- and fourth-order interpolation functions are:

- third-order interpolation function($m = 3$)

$$\Lambda_2(y) = \begin{cases} 1-|y|^2, & \text{if } 0 \le |y| \le \frac{1}{2}, \\ \frac{1}{2}(1-|y|)(2-|y|), & \text{if } 1/2 \le |y| \le \frac{3}{2}, \\ 0 & \text{otherwise.} \end{cases} \tag{3.19}$$

- fourth-order interpolation function ($m = 4$)

$$\Lambda_3(y) = \begin{cases} \frac{1}{2}(1-|y|^2)(2-|y|), & \text{if } 0 \le |y| \le 1, \\ \frac{1}{6}(1-|y|)(2-|y|)(3-|y|), & \text{if } 1 \le |y| \le 2, \\ 0 & \text{otherwise.} \end{cases} \tag{3.20}$$

The drawback of the ordinary interpolation function is it is less smooth than the smooth interpolation function. If interpolated quantities have large fluctuations, a small error in particle position might introduce large error in the interpolated value.

3.1.4 Smooth High-Order Interpolation Kernel

Based on B-splines, Monaghan [35] presented a systematic way to increase the order the interpolation function while maintaining the smoothness of the function using the idea of extrapolation. He suggested a new kernel represented as

$$W(y) = \frac{1}{2}(3M + yM'), \tag{3.21}$$

where M is a B-spline and M' is its derivative.

For example, we can use the B-spline of order four to construct an interpolation function with third order accuracy. The modified B-spline W_4 is defined as (see Figure (3.1))

$$W_4(y) = \begin{cases} 1 - \frac{5|y|^2}{2} + \frac{3|y|^3}{2}, & \text{if } 0 \le |y| \le 1, \\ \frac{1}{2}(2-|y|)^2(1-|y|), & \text{if } 1 \le |y| \le 2, \\ 0 & \text{otherwise.} \end{cases} \tag{3.22}$$

Funcs.	Error Order of Interpolation	Order of Smoothness
M_2	2	0
M_3	2	1
M_4	2	2
W_4	3	1
Λ_2	3	-1
Λ_3	4	0

Table 3.1: Error order and smoothness order for interpolating kernels. W_4 (red row) is the interpolation function for our remapping.

The first and second order derivatives of W_4 are continuous. In addition, it has interpolation error of $O(h_x^3)$. W_4 has been used extensively in SPH for interpolation [26, 38]. We also see that it is extensively used in both vortex methods and SPH for remapping [17, 30, 18].

We summarize the interpolation order and the smoothness of different interpolation kernels in Table (3.1). W_4 is the best candidate for remapping with respect to both smoothness and interpolation order. The plots of different interpolation kernels are shown in Figure (3.1).

We would like to mention here that when we solve the Vlasov equation, the overall error introduced by remapping will be one order lower than the interpolation error since we lose one order of accuracy in the evolution step. For example, if the interpolation error is $O(h_x^2)$, the overall error from remapping for the Vlasov equation will be $O(h_x)$. In order to get a second-order method, the interpolation function should be of at least third order $O(h_x^3)$. We choose the smooth interpolation function W_4 with third order accuracy for remapping. The interpolation function for N dimension is the tensor product of the one-dimensional formula,

$$W_4(y) = \prod_{d=0}^{2N-1} W_4(y_d). \tag{3.23}$$

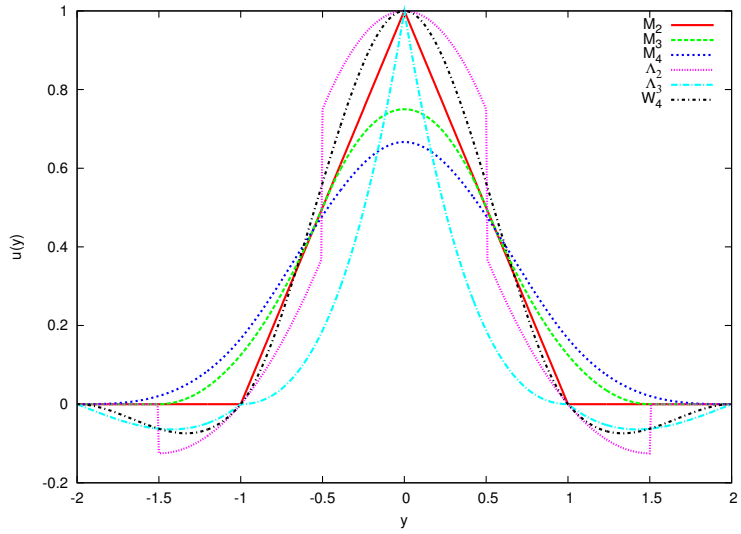


Figure 3.1: Plots for interpolating kernels.

3.2 Positivity

It is well known that high-order interpolation functions are not positivity preserving. This is easy to be proved by observing the moment conditions (equation (6.14)). From the physics point of view, an interpolation function without positivity might create nonphysical negative charge. This should be avoided in simulations. In this section, we propose two approaches to preserve the positivity of a high-order interpolation function.

3.2.1 Global Positivity Preserving Algorithm

Flux-corrected transport (FCT) is a technique used to preserve positivity and monotonicity in Godunov methods for hyperbolic conservation laws. To apply this method to the remapping algorithm, we first represent the remapping operation as a flux difference

$$f_i^{n+1} = f_i^n + \nabla \cdot F. \tag{3.24}$$

We can think of f_i^{n+1} as the distribution function after remapping and f_i^n as the distribution function before remapping. It is easy to formulate f by

$$f_i^{n+1} = \sum_k \frac{q_k}{h_x h_v} u(\frac{x_i - x_k}{h_x}) u(\frac{v_i - v_k}{h_v}). \tag{3.25}$$

and

$$f_i^n = \sum_k \frac{q_k}{h_x h_v} H(\frac{x_i - x_k}{h_x}) H(\frac{v_i - v_k}{h_v}). \tag{3.26}$$

where H is the hat function

$$H(y) = \begin{cases} 1, & \text{if } 0 \le |y| \le \frac{1}{2}, \\ 0 & \text{otherwise.} \end{cases} \tag{3.27}$$

The flux F can be calculated by solving the following well-defined Poisson system

$$\nabla \cdot F = f_i^n - f_i^{n+1} \tag{3.28}$$

$$= \sum_k \frac{q_k}{h_x h_v} H(\frac{x_i - x_k}{h_x}) H(\frac{v_i - v_k}{h_v}) - \sum_k \frac{q_k}{h_x h_v} u(\frac{x_i - x_k}{h_x}) u(\frac{v_i - v_k}{h_v}) \tag{3.29}$$

with boundary conditions $F_{x=0} = F_{x=L}$ and $\nabla \cdot F_{|v|=v_{\max}} = 0$.

We define a low-order flux F^{lo} and a high-order flux F^{hi} as using a low-order interpolation function u^{lo} (i.e.,M_2) and a high-order interpolation function u^{hi} (i.e.,W_4), respectively. The flux in each direction can be calculated by

$$F^{\text{lo}}_{i+\frac{e_d}{2}} = \frac{\varphi_i^{\text{lo}} - \varphi_i^{\text{lo}}}{h_d}, \quad F^{\text{hi}}_{i+\frac{e_d}{2}} = \frac{\varphi_i^{\text{hi}} - \varphi_i^{\text{hi}}}{h_d}, \tag{3.30}$$

where $d = 0, N-1$ is the dimension in phase space. φ^{lo} and φ^{hi} are the potentials resulting from solving the Poisson equation using a high-order and a low-order interpolating kernels, respectively. That is

$$\varphi_i^{\text{lo}} = \Delta_h^{-1} \left(\sum_k \frac{q_k}{h_x h_v} H(\frac{x_i - x_k}{h_x}) H(\frac{v_i - v_k}{h_v}) - \sum_k \frac{q_k}{h_x h_v} u^{\text{lo}}(\frac{x_i - x_k}{h_x}) u^{\text{lo}}(\frac{v_i - v_k}{h_v}) \right), \tag{3.31}$$

and

$$\varphi_i^{\text{hi}} = \Delta_h^{-1} \left(\sum_k \frac{q_k}{h_x h_v} H(\frac{x_i - x_k}{h_x}) H(\frac{v_i - v_k}{h_v}) - \sum_k \frac{q_k}{h_x h_v} u^{\text{hi}}(\frac{x_i - x_k}{h_x}) u^{\text{hi}}(\frac{v_i - v_k}{h_v}) \right). \tag{3.32}$$

We then use the multi-dimensional FCT algorithms by Zalesak [39] with a minor modification. First, we do not enforce the maximum limiting. Second, in estimating the extra capacity of a cell, Q_i (equation 3.37), we use use global minimum, 0, instead of local minimum. The algorithm proceeds as follows:

1. Compute $F^{\text{lo}}_{i+\frac{e_d}{2}}$ and $F^{\text{hi}}_{i+\frac{e_d}{2}}$ by a low-order and a-high order interpolation scheme, respectively.

2. Define the anti-diffusive flux as

$$A_{i+\frac{e_d}{2}} = F^{\text{lo}}_{i+\frac{e_d}{2}} - F^{\text{hi}}_{i+\frac{e_d}{2}}. \tag{3.33}$$

3. Limit the anti-diffusive flux by

$$A^C_{i+\frac{e_d}{2}} = C_{i+\frac{e_d}{2}} A_{i+\frac{e_d}{2}}, \tag{3.34}$$

where

$$C_{i+\frac{e_d}{2}} = \begin{cases} \min(1, R_i) & A_{i+\frac{e_d}{2}} \geq 0 \\ \min(1, R_{i+e_d}) & A_{i+\frac{e_d}{2}} < 0, \end{cases} \tag{3.35}$$

and

$$R_i = \begin{cases} \min(1, \frac{Q_i}{P_i}) & P_i > 0 \\ 0 & P_i = 0. \end{cases} \tag{3.36}$$

Q_i is the extra capacity of cell i

$$Q_i = \max(0, f^{\text{lo}}_i), \tag{3.37}$$

and P_i is the sum of all anti-diffusive fluxes away from the grid cell i defined as

$$P_i = \sum_{d=0}^{1} \left(\max(0, A_{i+\frac{e_d}{2}}) - \min(0, A_{i-\frac{e_d}{2}}) \right). \tag{3.38}$$

4. Compute the limited value as

$$f^{\text{hi}}_i = f^{\text{lo}}_i - \sum_{d=0}^{1} \left(A^C_{i+\frac{e_d}{2}} - A^C_{i-\frac{e_d}{2}} \right). \tag{3.39}$$

This FCT algorithm enforces positivity of the distribution function using a globally-determined function F in $\mathbb{R}^{2N}$. The Vlasov equation is in phase space, where $2N = 2, 4, 6$. For $N = 1$, this poses no particular difficulty. With $N = 2, 3$, this approach is unattractive since it requires us to solve the Poisson equation in $4D$ an $6D$. However, this algorithm will be very useful to enforce positivity in high-order remapping for other particle methods, such as vortex methods and SPH. Meanwhile, we develop a second approach which uses only local information.

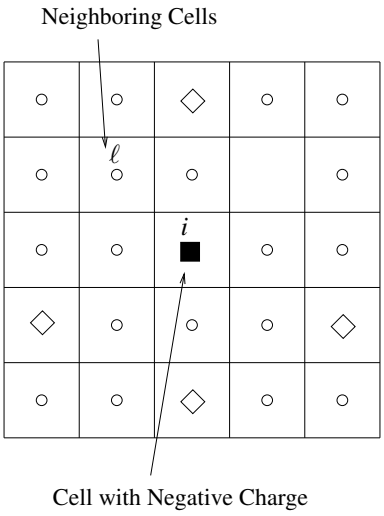


Figure 3.2: The plot shows local mass redistribution. The back square denotes the cell with negative value. The circles denote the positive neighboring cells. The diamonds denote the nonpositive neighboring cells.

3.2.2 Local Positivity Preserving Algorithm

The local algorithm is based on the mass redistribution idea of Chern and Colella [40], first applied to enforce positivity preservation by Hilditch and Colella [41]. In the algorithm, we redistribute the undershoot of cell i

$$\delta f_i = \min(0, f_i^n) \tag{3.40}$$

to its neighboring cells $i+\ell$ in proportion to their capacity ξ

$$\xi_{i+\ell} = \max(0, f_{i+\ell}^n). \tag{3.41}$$

The distribution function is conserved, which fixes the constant of proportionality

$$f_{i+\ell}^{n+1} = f_{i+\ell}^n + \frac{\xi_{i+\ell}}{\sum\limits_{k\neq 0}^{\text{neighbors}} \xi_{i+k}} \delta f_i \tag{3.42}$$

for $\ell \neq 0$ such that cell $i+\ell$ is a neighbor of cell i. Superscript n and $n+\ell$ denote the interpolated value before and after redistribution, respectively. The plot in phase space (x, v) is shown in Figure (3.2).

The drawback of this approach is that positivity is not guaranteed in a single pass. One might have

to apply the method iteratively. In practice, however, we find a few iterations to be sufficient.

3.3 Summary

In this chapter, we discuss the criteria for choosing an interpolation function for remapping. For a second-order method, a third-order interpolation function is required for remapping. However, high-order interpolation functions do not preserve positivity. We provide two techniques to preserve the positivity of a high-order interpolation function. The remapping function we constructed conserves the total mass, preserves positivity and has high order accuracy. As a conclusion, we would like to mention a few side effects of applying remapping. The first one is computational cost. Remapping introduces extra computational time and memory usage. The extra cost is usually not a concern for one-dimensional simulations. However, as the dimension increases, the extra cost could be significant. Fortunately, we can reduce the cost by using emerging GPU computing technology. The second side effect is the increase in the total number of particles. We reduce the influence of this side effect by using mesh refinement. Remapping with mesh refinement will be discussed in Chapter 5.

Chapter 4

Collisions

A more complete treatment of kinetic plasma includes a Fokker-Planck collision term to the Vlasov equation. This describes the plasma in the regime where instabilities are dominant but are modified by weak collisions. The phase space grid constructed at remapping time provides an opportunity to introduce a Fokker-Planck collisional term and solve the equation on a grid-based solver using finite difference methods. In the present work, we adopt a simplified Fokker-Planck collision model suggested by Rathmann and Denavit [42]. In this chapter, we first introduce a simplified Fokker-Planck collision model and its grid-based solver. Then we describe the procedure of coupling the collision model with the Vlasov equation using operator splitting.

4.1 The Collision Model

The simplified Fokker-Planck collision model suggested by Rathmann and Denavit is

$$\left(\frac{\partial f(v)}{\partial t}\right)_{col} = \nabla_v \cdot \left[\alpha v f + \beta \nabla_v (\alpha f)\right], \tag{4.1}$$

$$\alpha = C/\left(2\langle |v|^2\rangle + |v|^2\right)^{3/2}, \tag{4.2}$$

where C is a collision constant, and

$$\beta = \langle |v|^2 \alpha\rangle / \langle \alpha \rangle. \tag{4.3}$$

α is the velocity-dependent collision frequency parameter and β is a parameter specified by energy conservation [42]. The bracket $\langle \cdots \rangle$ denotes a weighted average quantity, $\langle g \rangle = \frac{\int g f dv}{\int f dv}$. The absolute value $|\cdots|$ denotes the magnitude of the value, $|v| = \sqrt{\sum_d v_d^2}$.

The first term on the right hand side of equation (4.1) represents a friction force due to collisions, which tends to slow the particles down and so reduces the kinetic energy, while the second term represents diffusion, which may lead to an increase in kinetic energy. The diffusion parameter β is chosen such that the total energy is conserved. The more complete version of the Fokker-Planck equation requires that α and β are vector and tensor in terms of velocity.

4.2 Discretization of the Collision Model

The above system is a nonlinear convection-diffusion equation. We solve it on a phase space grid at remapping time with a second-order finite difference method on a cell-centered Cartesian grid. The Fokker-Planck equation has variables in velocity space only. We can solve the equation on velocity space for each cell in physical dimension independently. As a diffusion-dominated problem, the convection term can be represented by a second-order central difference discretization. It is crucial to extend the domain in velocity space such that homogeneous Neumann boundary conditions can be applied. Here, we describe the spatial discretization on a uniform grid. The algorithm on a hierarchy of locally refined grids will be described in Chapter 5. The semi-discretized system is advanced using a second-order, L_0 stable implicit scheme such that the time step is not constrained by the CFL condition.

4.2.1 Spatial Discretization on a Uniform Grid

In two dimensions ($N = 2$), the problem domain is represented by a rectangular grid. The discretization is based on a control volume where the divergence operator is replaced as a flux difference around a cell. Let us denote the cell index as $i \in \mathbb{Z}^2$, then

$$\left(\frac{df}{dt}\right)_i = \sum_{d=0}^{1} \frac{F_{i+\frac{e^d}{2}} - F_{i-\frac{e^d}{2}}}{h_d}. \tag{4.4}$$

$F_{i\pm\frac{e^d}{2}}$ is the flux at the cell boundary where

$$F_{i+\frac{e^d}{2}} = v_{i+\frac{e^d}{2}} \frac{\alpha_{i+e^d} f_{i+e^d} + \alpha_i f_i}{2} + \beta_{i+\frac{e^d}{2}} \frac{\alpha_{i+e^d} f_{i+e^d} - \alpha_i f_i}{h_d} \tag{4.5}$$

Index $i(\pm e^d)$ and $i \pm \frac{e^d}{2}$ denote cell-centered and face-centered values, respectively.

Applying equation (4.5) to (4.4), the semi-discretized Fokker-Planck equation can be represented as

$$\begin{aligned}\left(\frac{df}{dt}\right)_i = \sum_{d=0}^{1} \Bigg(& v_{i+\frac{e^d}{2},d} \frac{\alpha_{i+e^d} f_{i+e^d} + \alpha_i f_i}{2h_d} - v_{i-\frac{e^d}{2},d} \frac{\alpha_i f_i + \alpha_{i-e^d} f_{i-e^d}}{2h_d} \\ & + \beta_{i+\frac{e^d}{2}} \frac{\alpha_{i+e^d} f_{i+e^d} - \alpha_i f_i}{h_d^2} - \beta_{i-\frac{e^d}{2}} \frac{\alpha_i f_i - \alpha_{i-e^d} f_{i-e^d}}{h_d^2} \Bigg).\end{aligned} \tag{4.6}$$

The diffusion coefficient β is a constant in the whole velocity space domain. We set the boundary far away from the distribution function such that the homogeneous Neumann boundary conditions can be applied. Under this boundary condition, the total charge is conserved inside the problem domain. The boundary conditions are enforced by using ghost cells.

4.2.2 Temporal Discretization

After spatial discretization, the partial differential equation reduces to a ordinary differential equation (ODE). If we denote the discretized collision operator as L, the ODE can be represented as

$$\left(\frac{\partial f}{\partial t}\right)_{col} = L(f). \tag{4.7}$$

As for the time integrator for equation (4.1), we use a second-order, L_0-stable implicit scheme by Twizell, Gumel and Arigu (TGA) [43]. In the TGA scheme, we approximate $f^{n+1} = f(t_n + \Delta t)$ as

$$(I - \mu_1 \Delta t L)^{-1} (I - \mu_2 \Delta t L)^{-1} f^{n+1} = (I + \mu_3 \Delta t L) f^n, \tag{4.8}$$

where

$$\begin{aligned}\mu_1 &= \frac{2a-1}{a+\sqrt{a^2-4a+2}},\\ \mu_2 &= \frac{2a-1}{a-\sqrt{a^2-4a+2}},\\ \mu_3 &= (1.0-a),\\ a &= 2.0-\sqrt{2}\end{aligned}$$

The integration can be solved in three steps:

1. : $e = (I+\mu_3\Delta tL)f^n$

2. : $(I-\mu_2\Delta tL)^{-1}v = e$

3. : $(I-\mu_1\Delta tL)^{-1}f^{n+1} = v$

The second and the third step involve the solving of a matrix system. We use a multigrid method with Red-Black-Gauss-Seidel as the smoother and BiCGSTAB as the bottom solver. Since the parameters α and β are nonlinear, we need to estimate their value at time $t^n, t^{n+\mu_2+\mu_3}, t^{n+1}$

α^n and β^n are easy to estimate by applying f^n to equation (4.2) and (4.3). For α^{n+1} and β^{n+1}, we approximate them through first predicting f^{n+1} with a backward Euler integrator:

$$\left(I+\Delta tL(\alpha(f^{k-1}),\beta(f^{k-1}))\right)^{-1}f^k = f^n, \tag{4.9}$$

$$f^0 = f^n. \tag{4.10}$$

This procedure is applied iteratively until α and β converge within some given threshold,

$$|\alpha^{k+1}-\alpha^k| \le \varepsilon, \quad |\beta^{k+1}-\beta^k| \le \varepsilon \tag{4.11}$$

where k denotes the iterative step and ε is the given threshold. Then the parameters at time $t^{n+\mu_2+\mu_3}$ can be approximated through linear interpolation,

$$\alpha^{n+\mu_2+\mu_3} = \alpha^n + (\mu_2+\mu_3)(\alpha^{n+1}-\alpha^n), \tag{4.12}$$

and

$$\beta^{n+\mu_2+\mu_3} = \beta^n + (\mu_2+\mu_3)(\beta^{n+1} - \beta^n). \tag{4.13}$$

4.3 Coupling of Collision with the Vlasov Equation

The coupling of the simplified Fokker-Planck term to the Vlasov equation is through operator splitting. Operator splitting was introduced by Cheng and Knorr [6] to solve the Vlasov equation with the semi-Lagrangian method in the 1970s. Here we use second-order Strang's splitting for the solution of the Vlasov-Fokker-Planck equation:

$$\frac{\partial f}{\partial t} + v \cdot \nabla_x f - E \cdot \nabla_v f = \nabla_v \cdot [\alpha v f + \beta \nabla_v (\alpha f)]\,. \tag{4.14}$$

4.3.1 Strang's Splitting

Assume that we have an ordinary differential equation

$$\frac{dy}{dt} = (A+B)y(t), \quad 0 < t \leq T < \infty \tag{4.15}$$

where A and B are some differential operators, the solution of the system can be written as

$$y = \exp^{(A+B)t}. \tag{4.16}$$

Instead of computing $\exp^{(A+B)t}$, we estimate $\exp^{At}$ and $\exp^{Bt}$ separately, and combine them in some way such that the solution approximates $\exp^{(A+B)t}$ with a desired accuracy. A very popular way of combing the two operators with second-order accuracy is Strang's splitting. In Strang's splitting, we approximate $\exp^{(A+B)t^{n+1}}$ in a sequence of three steps:

$$f^\star = \exp^{A\Delta t/2} f^n \tag{4.17}$$

$$f^{\star\star} = \exp^{B\Delta t} f^\star \tag{4.18}$$

$$f^{n+1} = \exp^{A\Delta t/2} f^{\star\star}. \tag{4.19}$$

The error of this approximation can be shown to be

$$\begin{aligned} e(\Delta t) &= |\exp^{(A+B)\Delta t} - \exp^{A\Delta t/2}\exp^{B\Delta t}\exp^{A\Delta t/2}| \cdot f^n \\ &= O(\Delta t^3). \end{aligned}$$

4.3.2 Operator Splitting for the Vlasov-Fokker-Planck Equation

If we apply Strang's splitting to the Vlasov-Fokker-Planck equation (4.14), the differential operator will be $A = -(v \cdot \nabla_x f - E \cdot \nabla_v f)$ and $B = \nabla_v \cdot [\alpha v f + \beta \nabla_v (\alpha f)]$. The procedure from time step t^N to t^{N+1} ($\Delta t = t^{N+1} - t^N$ is

- solve the Vlasov equation with step size $\Delta t/2$ by the PIC method:

$$\frac{\partial f}{\partial t} + v \cdot \nabla_x f - E \cdot \nabla_v f = 0. \tag{4.20}$$

  A half step size $\Delta t/2$ usually includes several PIC evolution steps.

- perform a phase-space remapping for the distribution function

- solve the simplified Fokker-Planck equation on the phase space grid from by the finite difference method with step size Δt

$$\left(\frac{\partial f}{\partial t}\right)_{col} = \nabla_v \cdot [\alpha v f + \beta \nabla_v (\alpha f)]. \tag{4.21}$$

- solve another Vlasov equation from with step size $\Delta t/2$ by the PIC method:

$$\frac{\partial f}{\partial t} + v \cdot \nabla_x f - E \cdot \nabla_v f = 0. \tag{4.22}$$

The procedure of solving the Vlasov-Poisson equation along with Fokker-Planck collision term in a single step is displayed in Figure (4.1).

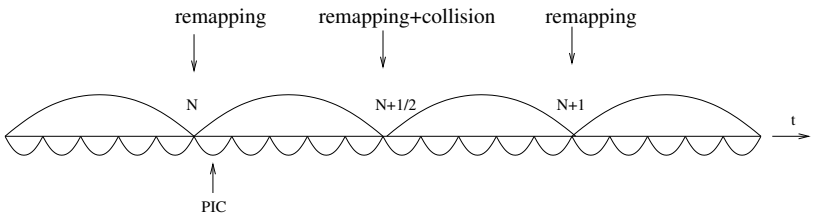


Figure 4.1: The flow chart of the algorithm

Chapter 5

Mesh Refinement

Plasma simulations are challenging because of a disparity of scales in time and in space [44]. For example, to model a beam in a heavy ion fusion accelerator from start to finish, one needs to resolve space scales spanning up to nine orders of magnitude, from microns (the Debye length) to a kilometer (the length of the accelerator). In addition, beam is often localized in a small subset of the physical domain. Mesh refinement, which has been successfully used to address the issue of wide range of space scales in other applications, i.e., fluid dynamics, is an attractive option. There has been much effort toward applying mesh refinement to plasma simulations using the PIC method [44, 45, 24]. In the domain of solving the electrostatic Vlasov-Poisson equation with the PIC method, the addition of mesh refinement complicates the algorithms of particle deposition and field interpolation, the discretization of the Laplacian operator, and the reconstruction of the distribution function on phase space used for remapping. In the domain of solving the simplified Fokker-Planck equation with finite volume discretization, special care needs to be taken for the differential operators at coarse-fine interfaces.

In this chapter, we will first describe the charge deposition algorithm on a hierarchy of node-centered grids on physical space. Then we give the node-centered local refinement algorithm for the Poisson equation by McCorquodale et al. [46]. The method uses a nodal-point discretization, adaptive mesh refinement (AMR) on Cartesian grids and the AMR multigrid solver of Almgren et al. [47]. Here the term

"adaptive"is used on locally refined grids, even when the grids are fixed in the evolution. In applications such as beam modeling, we need to solve the Poisson equation with infinite-domain boundary conditions. We give a new version of the James algorithm [48] for infinite-domain boundary conditions by McCorquodale et al. [46]. Finally, we describe the remapping algorithm on a hierarchy of cell-centered grids on phase space and the flux-matching condition for solving the simplified Fokker-Planck equation on a hierarchy of remapping grids.

5.1 Mesh Refinement for the PIC Method

5.1.1 Grid Hierarchy

Our grid hierarchy is based on the block-structured approach developed by Berger and Oliger [49] for hyperbolic equations and extended to shock hydrodynamics by Berger and Colella [49]. In general, we have a number of different levels of refinement, from coarsest $\ell = 0$ to finest $\ell = \ell_{\max}$. Each level is composed of a union of rectangular grid patches. The mesh spacing ratio between levels ℓ and $\ell + 1$ is a scalar integer constant r in each coordinate direction. This is called isotropic refinement or uniform refinement. Usually we choose $r = 2$ or $r = 4$. The patches need to be "properly nested"such that any patch must bordered by either the physical boundary or another patch of the same level, or grid boundary of patches from the coarser level. If we denote $\Omega^\ell \in \mathbb{Z}^N$ as the union of grids at level ℓ, then the proper nested condition can be written as

$$\mathcal{R}_r(\mathcal{C}_r(\Omega^{\ell+1})) = \Omega^{\ell+1}, \quad \mathcal{C}_r(\Omega^{\ell+1}) \in \Omega^\ell, \tag{5.1}$$

where $\mathcal{C}_r$ is the coarsening operator

$$\mathcal{C}_r(i) = (\lfloor \frac{i_0}{r} \rfloor, ..., \lfloor \frac{i_{N-1}}{r} \rfloor). \tag{5.2}$$

and $\mathcal{R}_r$ is the refinement operator.

The grid can be defined as node-centered or cell-centered on the Cartesian cell. In a node-centered grid, the index is defined on the node of a cell where

$$x_i = iH. \tag{5.3}$$

In a cell-centered grid, the index is defined on the cell center of a cell where

$$x_i = (i+1/2)H. \tag{5.4}$$

In our implementation of the PIC method, the Poisson equation is solved on a node-centered grid. To easily understand the discretization on a hierarchy of node-centered grids, we first give the definition for the composite grid as following:

- The Interior Nodes of level ℓ, $\Omega^{\ell,int}$: For a union of node-centered rectangular at each level, the interior nodes $\Omega^{\ell,int}$ are those nodes for which all neighbors along coordinate axes are also in $\Omega^{\ell,int}$ (see Figure 5.1). That is,

$$\Omega^{\ell,int} = \Omega^{\ell} \bigcap_{d=0,\ldots,N-1} ((\Omega^{\ell}+e^d)\cap(\Omega^{\ell}-e^d)). \tag{5.5}$$

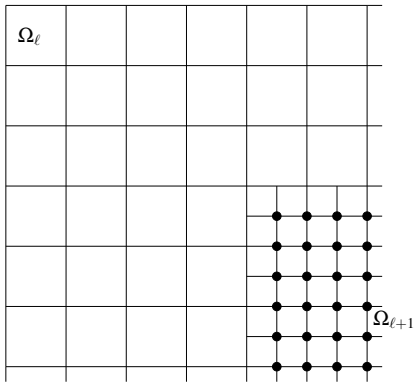


Figure 5.1: A grid hierarchy with two levels of refinement. The interior nodes of $\Omega^{\ell+1}$ at the finer level are indicated by black circles.

- The Valid Nodes of level ℓ, $\Omega^{\ell,valid}$: The valid nodes at each level are defined as those interior nodes which are not covered by the interior nodes at finer levels. That is

$$\Omega^{\ell,valid} = \Omega^{\ell,int} \setminus \mathcal{C}_r(\Omega^{\ell+1,int}). \tag{5.6}$$

- The Composite Grid, Ω^{comp}: The union of valid nodes from all levels consists of the composite grids

(see Figure (5.2)),

$$\Omega^{comp} = \bigcup_{\ell=0}^{\ell_{\max}} \Omega^{\ell,valid}. \tag{5.7}$$

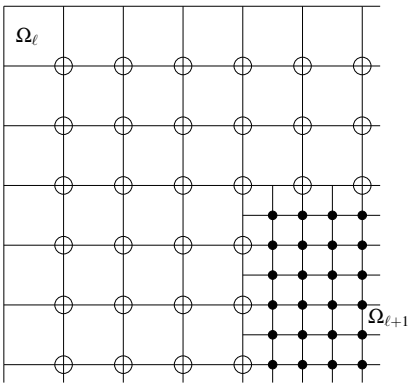


Figure 5.2: A grid hierarchy with two levels of refinement. The composite grid nodes are denoted by circles. The valid nodes of the finer level are denoted by black circles, and the valid nodes of the coarser level are denoted by open circles.

For each refinement level from $\ell = 0$ to $\ell = \ell_{\max}$, the discrete solution ϕ and charge density ρ for the right hand side of the Poisson equation are defined on the valid nodes. The solution and charge density on the composite grids are then

$$\phi^{comp} = \sum_{\ell=0}^{\ell_{\max}} \phi^{\ell,valid}, \quad \rho^{comp} = \sum_{\ell=0}^{\ell_{\max}} \rho^{\ell,valid}. \tag{5.8}$$

5.1.2 Charge Deposition and Field Interpolation

To solve the Poisson equation, we need to deposit the charges on the composite grid. We use a simple algorithm to interpolate charges from particle locations to the composite grid. For particles sufficiently far away from the coarse-fine interfaces, we interpolate the charges to the grid using linear interpolation (equation (2.3) with $u1$ in Figure 1.1) as before. The stencil size ε of the interpolation function is chosen to be the same as the grid mesh spacing H^ℓ, where level ℓ is the finest level at the charge location. For particles close to the coarse-fine interfaces, e.g., where the particles are located on the finer grid but bordered by a coarse-fine interface, it may be the case that a node to which one could normally deposit is not valid. In this

situation, we need to deposit the charges to the valid nodes of both the coarser level and the finer level. Figure 5.3 shows the interpolation nodes (circles) associated with charge locations (crosses) in different cases.

The interpolation of a node-based grid field to the particle location is a little different from the charge deposition algorithm. The fields on the particles inside those cells will be interpolated from the surrounding nodes of the cells. When some surrounding nodes are not valid, we first calculate the fields on those invalid nodes by quadratic interpolation from the valid nodes of the coarser level.

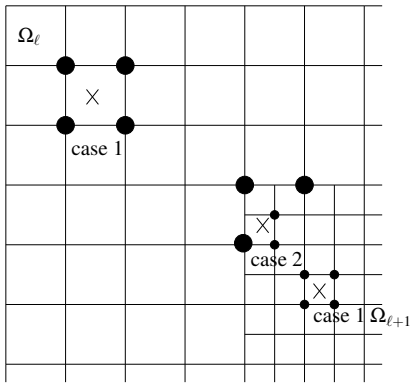


Figure 5.3: Charge deposition on the composite grids. Black circles denote the deposited nodes. The cross signs denote the particle locations. Case 1: For particles sufficiently far away from the coarse-fine interfaces, we interpolate the charges to the grid using linear interpolation. Case 2: For particles close to the coarse-fine interfaces, we interpolate the charges on the valid nodes of both the coarser level and the finer level.

5.1.3 Nodal Discretization and AMR Multigrid Solver

Multilevel Laplacian operator

It is well known that the second-order finite difference discretization for the Laplacian operator on a single level is

$$\rho_i^{\ell,valid} = \sum_{d=0}^{N-1} \left(\Delta^{H_d}\phi^\ell\right)_i = \sum_{d=0}^{N-1} \frac{\phi_{i-e^d}^\ell - 2\phi_i^\ell + \phi_{i+e^d}^\ell}{\left(H_d^\ell\right)^2}. \tag{5.9}$$

If the grid is uniform, then all points of the stencil are either valid nodes or nodes on the physical boundary. The multilevel Laplacian operator is based on the single-level discretization. However, for the composite

grid, there are three cases where i is a valid node, but $j = i \pm e^d$ is not:

- Case 1: Node j is on physical boundary. Then we use the boundary value directly (Figure 5.4).
- Case 2: Node j is covered by the valid node of a finer level. Then we use the value on the finer level node directly (Figure 5.5).
- Case 3: Node j is on the coarse-fine interface where no valid node is defined. Then we interpolate the value on j from the coarse node value (Figure 5.6).

The interested readers can find complete description for $3D$ case in [46].

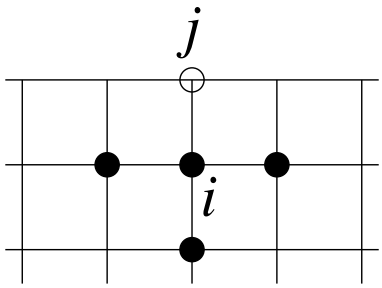


Figure 5.4: Case 1: Node j is on physical boundary. We use the boundary value directly.

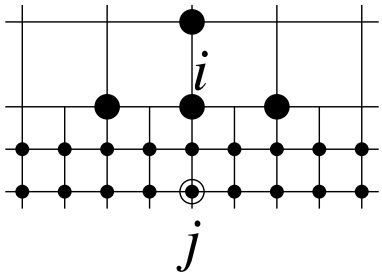


Figure 5.5: Case 2: Node j is covered by the valid node of a finer level. We use the value on the finer level node directly.

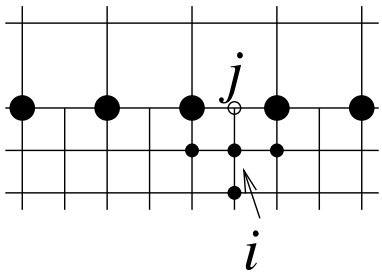


Figure 5.6: Case 3: Node j is on the coarse-fine interface where on valid node is defined. Big black circles are valid nodes of the coarser level from which we interpolate the value.

Truncation Error and Solution Error of the Multi-level Laplacian Operator

The truncation error ($\tau^H = \Delta\phi^e - \Delta^H\phi^e$) of the second-order Laplacian operator on a single level grid is

$$\tau^H = O(H^2). \tag{5.10}$$

The multilevel Laplacian operator modifies the truncation error when the stencil node is on the coarse-fine interface and its value ϕ_j is computed by interpolated from the valid nodes of the coarser level (Case 3). Because of the H^2 denominator in the Laplacian operator, the truncation error is two orders of accuracy lower than the interpolation order. To get second-order truncation error, we need fourth order accurate interpolation. Fortunately, the truncation error due to coarse-fine interface is in co-dimension one, so a third-order interpolation function is sufficient to get a second-order solution error.

AMR Multigrid for the Poisson equation

The solution of the discretized Poisson equation amounts to solve a linear system $Ax = b$. We use the multigrid method with Gauss-Seidel relaxation, red-black ordering, and BiCGSTAB bottom. We assume the reader is familiar with multigrid methods. For a complete introduction, the interested reader can see [50]. The extension of the multigrid algorithm to hierarchy of nested grids for the Poisson problem has been investigated by several authors [47,51,46]. We use the node-centered algorithm of Almgren [47].

We want to solve

$$L^{comp}(\phi) = \rho \quad on \quad \Omega^{comp}, \tag{5.11}$$

where L^{comp} is the second-order Laplacian operator on the composite grid. For all levels from $\ell = 0$ to $\ell = \ell_{\max}$, we need to store ϕ^ℓ and ρ^ℓ on $\Omega^\ell - \mathcal{C}_r(\Omega^{\ell+1})$. In addition, since the multigrid algorithm is processed by solving the residual equation, we need to store the residual R^ℓ and the correction e^ℓ on Ω^ℓ. To extend the multigrid algorithm to permit mesh refinement, the elements of the algorithm, such as the residual and correction need to be modified accordingly.

Before showing the algorithm, we define two operators we will use to generate the solution. The first is the multilevel Laplacian operator, $L^{\ell,comp}$, on a level $\Omega^\ell - \mathcal{C}_r(\Omega^{\ell+1})$. The right hand side of the Poisson equation is defined on the valid nodes of that level $\Omega^{\ell,valid}$. The solution on the stencil of a valid node is defined in (5.1.3). The second is the "no fine" Laplacian operator, $L^{\ell,nf}$, where the solution is defined on Ω^ℓ assuming that there is no finer grid. The boundary condition of $L^{\ell,nf}$ is either the physical boundary condition or the Dirichlet boundary condition interpolated from the coarser level.

We first set ρ and initialize ϕ on the composite grid. In AMR Multigrid, the multigrid V-cycle starts from the finest level. The residual is defined as

$$R^{\ell_{\max}} := \rho^{\ell_{\max}} - L^{\ell,nf}(\phi^{\ell_{\max}}, \phi^{\ell_{\max}-1}). \tag{5.12}$$

Then we save the solution on this level $\phi^{\ell,save} = \phi^\ell$ and relax the finest level residual equation by

$$e^\ell := e^\ell + \lambda\{L^{\ell,nf}(e^\ell, e^{\ell-1} = 0) - R^\ell\}, \tag{5.13}$$

with red-black ordering. λ is the relaxation parameter. It is chosen to be the inverse of the diagonal elements of the matrix. With the correction e^ℓ, we update ϕ^ℓ with

$$\phi^\ell := \phi^\ell + e^\ell. \tag{5.14}$$

We need the current version of ϕ to compute the residual of a coarser level. For calculating the residual of the coarser level, we modify the algorithm as follows. For the region covered by the finer level, the residual is obtained by average the current finer level residual. For the region not covered by the finer level, we obtain

the residual by applying the multilevel Laplacian operator. It is very important that we update the finer level solution first (equation 5.14) since the multilevel Laplacian operator needs to use the solution value on the finer nodes. The algorithm is

$$R^{\ell-1} := \begin{cases} Average(R^{\ell} - L^{\ell,nf}(e^{\ell}, e^{\ell}), & \text{on } C_r(\Omega^{\ell}) \\ \rho^{\ell-1} - L^{\ell-1,comp}(\phi), & \text{on } \Omega^{\ell-1} - C_r(\Omega^{\ell}) \end{cases} \tag{5.15}$$

The *Average* operator and the *Interpolate* operator later are the same as the standard multigrid algorithm.

We repeat this process till we get the coarsest level where the grid covers the whole problem domain. At this point we can use the standard multigrid algorithm and get the solution corrected by $\phi^0 = \phi^0 + e^0$. The correction up to the finest level $\ell_{\max}$ is a little different. As in the the standard multigrid algorithm, we update the finer grid correction as

$$e^{\ell} := e^{\ell} + Interpolate(e^{\ell-1}). \tag{5.16}$$

However, we can not relax the residual equation directly with RBGS_LEVEL since at this point we need boundary condition interpolated from the coarser level. We handle this problem by treating the Poisson equation with inhomogeneous Dirichlet boundary conditions with two problem: a boundary problem and a homogeneous problem. The correction algorithm is then modified to include two steps. First, we exclude the boundary condition

$$R^{\ell} := R^{\ell} - L^{\ell,nf}(e^{\ell}, e^{\ell-1}). \tag{5.17}$$

Then we perform Gauss Seidel iteration for the homogeneous boundary condition problem

$$\delta e^{\ell} := \delta e^{\ell} + \lambda\{L^{\ell,nf}(\delta e^{\ell}, \delta e^{\ell-1} = 0) - R^{\ell}\}. \tag{5.18}$$

Finally, we update the error and the solution as

$$e^{\ell} := e^{\ell} + \delta e^{\ell}, \tag{5.19}$$

$$\phi^{\ell} := \phi^{\ell,save} + e^{\ell}. \tag{5.20}$$

This whole algorithm is shown in Figure (5.7) assuming that $r = 2$:

$R := \rho - L(\phi)$

while ($||R|| > \varepsilon||\rho||$)

AMRVCycleMG(ℓ^{max})

$R := \rho - L(\phi)$

end while

procedure AMRVCycleMG(level ℓ): {

if ($\ell = \ell^{max}$) then $R^\ell := \rho^\ell - L^{\ell,nf}(\phi^\ell, \phi^{\ell-1})$

if ($\ell > 0$) then

$\phi^{\ell,save} := \phi^\ell$ on Ω^ℓ

$e^\ell := 0$ on Ω^ℓ

GSRB_LEVEL iteration $e^\ell := e^\ell + \lambda\{L^{\ell,nf}(e^\ell, e^{\ell-1} = 0) - R^\ell\}$

$\phi^\ell := \phi^\ell + e^\ell$

$e^{\ell-1} := 0$ on $\Omega^{\ell-1}$

$R^{\ell-1} := Average(R^{\ell-1} - L^{nf}(e^\ell, e^{\ell-1}))$ on $\mathcal{C}_r(\Omega^\ell)$

$R^{\ell-1} := \rho^{\ell-1} - L^{\ell-1,comp}(\phi)$ on $\Omega^{\ell-1} - \mathcal{C}_r(\Omega^\ell)$

AMRVCycleMG($\ell - 1$)

$e^\ell := e^\ell + Interpolate(e^{\ell-1})$

$R^\ell := R^\ell - L^{\ell,nf}(e^\ell, e^{\ell-1})$

$\delta e^\ell := 0$ on Ω^ℓ

GSRB_LEVEL iteration $\delta e^\ell := \delta e^\ell + \lambda\{L^{\ell,nf}(\delta e^\ell, \delta e^{\ell-1} = 0) - R^\ell\}$

$e^\ell := e^\ell + \delta e^\ell$

$\phi^\ell := \phi^{\ell,save} + e^\ell$

else

solve $L^{0,nf}(e^0, 0) = R^0$ on Ω^0, by one-levelmultigrid algorithm.

$\phi^0 := \phi^0 + e^0$

end if

}

Figure 5.7: Pseudo-code description of the AMR multigrid algorithm.

5.1.4 Infinite-domain (Free Space) Boundary Conditions

For modeling beam problems, we need to solve the Poisson equation with infinite domain (free space) boundary conditions. We use a new version of James' [48] algorithm by McCorquodale et al [52,53] that solves two Dirichlet boundary problems plus a simplified fast multipole method instead.

We briefly describe the algorithm below. Assume D_0 is the support domain of the right hand side, ρ, we can solve the Poisson equation with infinite domain boundary conditions on a slightly larger domain $D_1 > D$ with inhomogeneous Dirichlet boundary conditions. The boundary value can be calculated by Green's function convolution from the source ρ to the D_1 domain boundary, ∂D_1. The volume source ρ to boundary ∂D_1 convolution is relatively expensive, in particular for $3D$ problems. Instead of using a volume to boundary convolution, we can compute the boundary value by performing a boundary ∂D_1 to boundary ∂D_2 convolution and solving another Poisson equation on a domain $D_2 > D_1 > D$ with Dirichlet boundary condition. The algorithm is (Figure 5.8)

- Step 1: Solve the Poisson equation on domain D_1 with homogeneous Dirichlet boundary conditions

$$\Delta\phi_1 = \rho \quad on \quad D_1, \quad \phi_1 = 0 \quad on \quad \partial D_1. \tag{5.21}$$

- Step 2: Calculate the surface charge on ∂D_1

$$\partial\rho = \frac{\partial\phi_1}{\partial n} \quad on \quad \partial D_1. \tag{5.22}$$

- Step 3: Perform a boundary to boundary convolution from $\partial\rho$ to ∂D_2

$$\partial\phi = \int_{\partial D_1} G(x-y)\partial\rho(y)dA_y, \tag{5.23}$$

  using fast multiple methods [21].

- Step 4: Solve another Poisson equation on domain D_2 with inhomogeneous Dirichlet boundary conditions

$$\Delta\phi_2 = \rho \quad on \quad D_2, \quad \phi_2 = \partial\phi \quad on \quad \partial D_2. \tag{5.24}$$

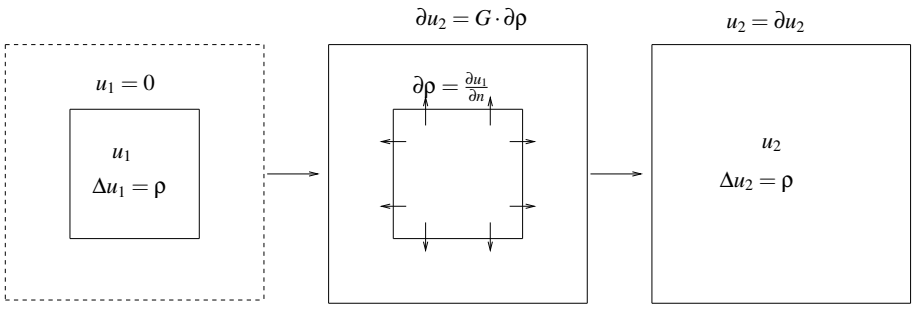


Figure 5.8: James Algorithm Left: Solve for u_1 on D_1 Middle: Calculate surface charge and perform convolution Right: Solve u_2 on D_2

5.2 Mesh Refinement for Remapping

Mesh refinement is an attractive option in improving the efficiency of phase-space remapping. The distribution function in phase space is inhomogeneous, in particular in velocity space (Maxwellian distribution). When we represent the system by particles, it is best that we can have each particle carries a similar amount of weights. Remapping on a hierarchy of locally refined grids is a good strategy for creating such a set of particles. From another point of view, remapping through interpolation is a numerically diffusive procedure. This results in a large number of small strength particles near the tail of the distribution function. The situation becomes worse with successive application of remapping. Remapping on a hierarchy of locally refined grids, with a coarser grid covering the tail of the distribution function, can significantly reduce the number of those small strength particles. In the following, we present an algorithm for remapping with mesh refinement. In designing the algorithm, we have two guiding principles: the total charge should be conserved; the overall accuracy on the field needs to be maintained.

We use a composite grid as in the previous section. But for remapping on phase space, we use a cell-centered composite grid instead of a node-centered one. The cell-centered discretization conserves the total charge on the composite grid. We define the valid cells of one level as the cells on this level that are not covered by a finer grid. That is,

$$\Omega_c^{\ell,valid} = \Omega_c^{\ell} \setminus \mathcal{C}_r(\Omega_c^{\ell+1}). \tag{5.25}$$

We use subscript c to denote that this grid is cell-centered. A composite grid with cell centered discretization is then

$$\Omega_c^{comp} = \bigcup_{\ell=0}^{\ell_{\max}} \Omega_c^{\ell,valid}. \tag{5.26}$$

In the remapping step, each particle first finds the valid cell in the composite grid it belongs to. One particle can only belong to a single valid cell. If the cell is far enough away from a coarse-fine interface, the charge can be interpolated to the grid as in equation (3.22). If the cell is near a coarse-fine interface such that the interpolation stencil intersects the interface, special care must be taken. First, we interpolate the charge to the surrounding cells as usual. After deposition, we find that not all deposited cells are valid cells. We need a further step to transfer the charge from an invalid cell to a valid cell. There are two cases depending on where the invalid cell is located. If the invalid cell is in a coarser level and it is covered by the valid cells of a finer level, we transfer the deposited charge from the coarser level to the finer level through interpolation. On the other hand, if the invalid cell is outside the grid of the current level, the charge is transfered by projection. Figure 5.9 shows the algorithm in two dimensions. The four-dimensional case can be generalized easily. It is worth mentioning that we lose one order of accuracy in interpolating the coarser level charge into the

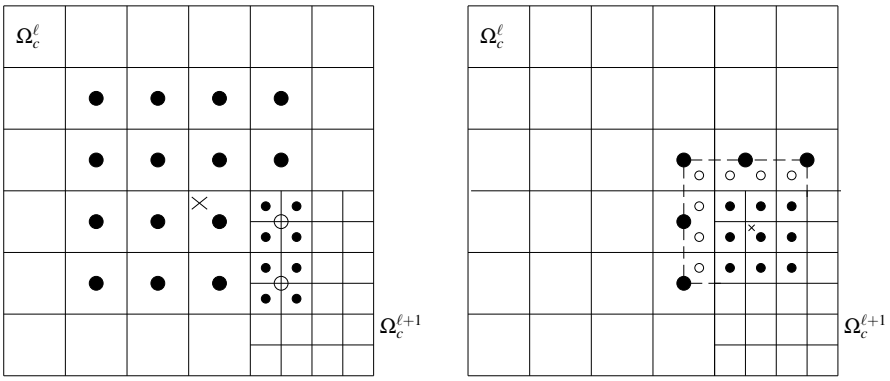


Figure 5.9: Cross signs denote the particle locations. The valid and the invalid deposited cells are denoted by filled circles and open circles, respectively. The refinement ratio is $r_0 = (2,2)$ in the plots. Left: Particle is at the coarser level side. Right: Particle is at the finer level side. Cross signs denote the particle locations.

finer level. However, since the coarse-fine interface is in co-dimension one, the expected accuracy in the

electric field, e.g., second order, will be preserved in the L_∞ error norm. Our current implementation does not have time-dependent adaptivity. This feature can be incorporated by selecting some refinement criterion, for example, each cell in phase space has similar number of particles.

5.3 Mesh Refinement for Collision

In the remapping step, the distribution function is interpolated on a hierarchy of locally refined grids. The simplified Fokker-Planck equation is actually solved on a hierarchy of grids instead of on a uniform grid. In discretizing the Fokker-Planck equation on a hierarchy of grids, we use the approach taken in solving Poisson equation on a hierarchy of cell-centered grids [51, 54].

Basically, the collisional operator is discretized on a cell-centered composite grid (5.26). For cells sufficiently far away from interfaces, we use the same discretization as for the uniform grid (see Chapter 4). For cells close to the interfaces, the algorithm needs to be modified so that both Dirichlet and Neumann boundary conditions are enforced at coarse-fine interfaces (flux matching condition).

Let us consider a finer grid to the left of a coarser grid in $2D$. The coarser cell adjacent to the interface is computed as

$$\left(\frac{df}{dt}\right)_{i^c} = \nabla \cdot F^c \tag{5.27}$$

$$= \frac{F_{i^c+\frac{e^0}{2}} - F^{ave}_{i^c-\frac{e^0}{2}}}{h^c_0} + \frac{F_{i^c+\frac{e^1}{2}} - F_{i^c-\frac{e^1}{2}}}{h^c_1}. \tag{5.28}$$

where the superscript c denotes the valid cell in coarser level (see Figure (5.10)). To enforce the flux matching condition, the flux at the coarser-fine interface $F_{i^c-\frac{e^0}{2}}$ needs to be modified so that the flux on the coarser side of the boundary matches the flux on the finer side of the boundary. That is,

$$F^{ave}_{i^c-\frac{e^0}{2}} = \frac{1}{2}(F^{top}_f + F^{bottom}_f). \tag{5.29}$$

$F^{top|bottom}_f$ are the flux calculated by the finer level discretization

$$F^{top}_f = v_{i^f+\frac{e^0}{2}} \frac{\alpha^{interp} f^{interp} + \alpha_{i^f} f_{i^f}}{2} + \beta_{i^f+\frac{e^0}{2}} \frac{\alpha^{interp} f^{interp} - \alpha_{i^f} f_{i^f}}{h^f_0} \tag{5.30}$$

To compute the value on the finer stencil where no valid cell is defined, f^{interp}, we need to interpolate the value from the coarser grid. The interpolation includes two passes. In the first pass, we get the intermediate value through coarser cell value interpolation. In the second pass, we get the interpolated value through finer cell values and the intermediate value we get from the first pass (see Figure (5.10)). It is worth mentioning here that quadratic interpolation is the minimum order necessary to maintain second-order accuracy overall. Assuming that the interpolation is of order p, the error will be $O(h^p)$. Since the Laplacian operator is a second-order derivative, the truncation error of the Laplacian due to interpolation will be $O(h^{p-2})$. In quadratic interpolation, the error will be $O(h)$. However, since the coarse-fine interface is a set of co-dimension one, we can gain one order of accuracy. This is easy to prove if we recall the solution error of a discretized Laplacian operator [55]. First, we can view the solution of a Poisson equation as being a continuous potential theory problem, in which the charge density is piecewise constant in a cell. If the interpolation is of order p, the charge density will be $O(h^{p-2})$. Since each cell has volume $O(h^N)$, where N is the dimension of the problem, the total charge will be $O(h^{p+N-2})$. There are $O(\frac{1}{h^{N-1}})$ number of charges at the coarse-fine interfaces. This results in a error of $O(h^{p+N-2-N+1}) = O(h^{p-1})$. When $p = 3$, we have second-order accuracy overall.

The finer cell adjacent to the coarse-fine interface is computed as

$$\left(\frac{df}{dt}\right)_{i^f} = \nabla \cdot F^f \tag{5.31}$$

$$= \frac{F_f^{top} - F_{i^f - \frac{e^0}{2}}}{h_0^f} + \frac{F_{i^f + \frac{e^1}{2}} - F_{i^f - \frac{e^1}{2}}}{h_1^f}, \tag{5.32}$$

where the superscript f denotes the finer grid value (see Figure (5.10)).

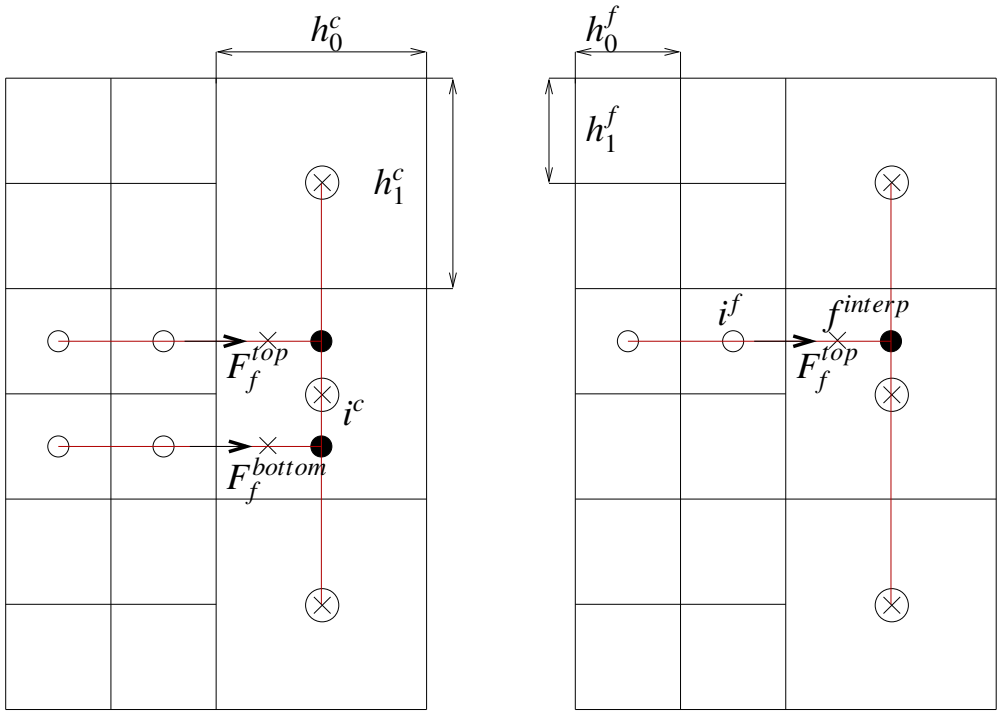


Figure 5.10: Interpolation on the coarse-fine boundary. The intermediate values are denoted by solid circles. They are obtained by quadratic interpolation from coarser cell values (denoted by circled crosses). The interpolated value (denoted by a cross) is obtained by quadratic interpolation from the finer cell values (denoted by circles) and the intermediate value.

Chapter 6

Error Analysis of the PIC Method

We analyze the error of the PIC method based on the convergence theory of vortex methods. We begin with a literature review of convergence theory for vortex methods.

In vortex methods, Euler's equations for incompressible flow are replaced by a system of ODEs through a particle approximation. The solution to the ODEs gives the approximate trajectories of a finite number of particles, called vortices. The convergence of vortex methods in two dimensions was first proved by Hald [56]. He showed that the approximate trajectories of the vortices converge to the exact particle trajectories as the number of vortices increases. However, his approach used stronger smoothing than is implicit in the PIC method. Hald's result was improved and generalized to three dimensions by Beale and Majda [57, 58]. By adapting the idea of Beale and Majda, Cottet and Raviart [1] gave a stronger consistency result, relying on the Bramble-Hilbert lemma in bounding the error in the quadrature rule. Anderson and Greengard [59] simplified the proof further using the Poisson summation formula [60] and incorporated time discretization in the analysis. In all these analyses, the convergence results are based on estimations of how close the approximated trajectories are to the exact trajectories.

A convergence theorem of particle methods for the one-dimensional Vlasov-Poisson equation was given by Cottet and Raviart [1]. It was generalized to the multi-dimensional Vlasov-Poisson equation by Victory and Allen [61]. Our analysis differs from Cottet and Raviart's the way we estimate the consistency error

of the electric field. Instead of estimating the consistency error of the electric field by convolution, we treat it as the field induced by the consistency error of the charge density. This gives us an opportunity to use the smooth interpolation analysis of by Schoenberg [25, 26] instead of quadrature rules on the discretization error (The consistency error includes the moment error and the discretization error). Compared with the quadrature rule, the smooth interpolation analysis results in a higher order error bound for the same interpolation function, e.g., the B-spline of order two (M_2).

The sections below will be organized as follows. In §6.1.1, we analyze the consistency error of the charge density through particle approximation. The discretization error is estimated by the smoothed interpolation formula. §6.1.2 estimates the consistency error of the electric field from the consistency error of the charge density by noting that the Laplacian operator is a stable operator. §6.1.3 describes the stability error in both charge density and electric field by following the same approach as in [1]. In the end, the convergence of both charge density and electric field error is given in §6.1.4.

6.1 Error Analysis of the PIC Method for the 1D Vlasov-Poisson Equation

For simplicity, the analysis in this section considers the PIC method for the Vlasov-Poisson equation in one dimension with periodic boundary conditions. The result also applies to problems with bounded domains provided that the initial charge density vanishes at the boundary such as occurs in beam focusing problem.

The PIC method begins from approximating the initial distribution function (equation (1.11)) with particle representation. First, we cover the phase space $\mathbb{R}^2$ with a uniform rectangular mesh with mesh spacing (h_x, h_v). We denote the center of a rectangular to be $(x_i, v_i) = (h_x(1/2 + i_0), h_v(1/2 + i_1))$, where $i = (i_0, i_1) \in \mathbb{Z}^2$ is the cell index. Then the initial distribution function is represented as

$$\tilde{f}_0(x,v) = \sum_{i \in \mathbb{Z}^2} q_i \delta(x, x_i) \delta(v, v_i), \tag{6.1}$$

where

$$q_i = h_x h_v f_0(x_i, v_i). \tag{6.2}$$

Each particle is transported along the trajectory by

$$\begin{aligned} \frac{dq_k}{dt}(t) &= 0, & (6.3)\\ \frac{d\tilde{X}_k}{dt}(t) &= \tilde{V}_k(t), & (6.4)\\ \frac{d\tilde{V}_k}{dt}(t) &= -\tilde{E}(\tilde{X}_k(t),t), & (6.5) \end{aligned}$$

with initial conditions $q_k(t=0) = q_i$ and $(\tilde{X}_k(t=0) = x_i, \tilde{V}_k(t=0) = v_i)$.

Under particle representation, the distribution function is approximated as

$$\tilde{f}(x,v,t) = \sum_k q_k \delta_\varepsilon(x,\tilde{X}_k)\delta(v,\tilde{V}_k), \tag{6.6}$$

and the electric field is approximated by using finite-size particles,

$$\begin{aligned} \tilde{E}(x,t) &= \int_0^L K(x,y)(1-\int_{-\infty}^{\infty}\tilde{f}(x,v,t)dv)dy & (6.7)\\ &= \int_0^L K(x,y)dy - \sum_k q_k K_\varepsilon(x-\tilde{X}_k(t)). & (6.8) \end{aligned}$$

where $K(x,y) = \frac{\partial G(x,y)}{\partial y}$ and $K_\varepsilon = K * \delta_\varepsilon$. G is the Green's function for the Poisson equation with periodic boundary conditions. This is equivalent to approximating the charge density as

$$\tilde{\rho}(x,t) = 1 - \sum_k q_k \delta_\varepsilon(x-\tilde{X}_k(t)). \tag{6.9}$$

At this point, we can represent the error in charge density as

$$\begin{aligned} e^\rho(x,t) &= |\rho(x,t) - \tilde{\rho}(x,t)| \\ &= |\rho(x,t) - (1 - \sum_k q_k \delta_\varepsilon(x-\tilde{X}_k(t)))| \\ &\le \underbrace{\left|\rho(x,t) - (1 - \sum_k q_k \delta_\varepsilon(x-X_k(t)))\right|}_{e_c^\rho(x,t)} \\ &\quad + \underbrace{\left|\sum_k q_k \delta_\varepsilon(x-X_k(t)) - \sum_k q_k \delta_\varepsilon(x-\tilde{X}_k(t))\right|}_{e_s^\rho(x,t)}. \end{aligned} \tag{6.10}$$

Similarly, the error in electric field can be written as

$$\begin{aligned} e^E(x,t) &= |E(x,t) - \tilde{E}(x,t)| \\ &= |E(x,t) - (\int_0^L K(x,y)dy - \sum_k q_k K_\varepsilon(x - \tilde{X}_k(t)))| \\ &\leq \underbrace{\left|E(x,t) - (\int_0^L K(x,y)dy - \sum_k q_k K_\varepsilon(x - X_k(t)))\right|}_{e_c^E(x,t)} \\ &\quad + \underbrace{\left|\sum_k q_k K_\varepsilon(x - X_k(t)) - \sum_k q_k K_\varepsilon(x - \tilde{X}_k(t))\right|}_{e_s^E(x,t)}, \end{aligned} \tag{6.11}$$

The term e_c is the consistency error caused by representing the continuous system with discretized particles, while the term e_s is the stability error introduced by approximating the system with the computed trajectories $(\tilde{X}_k(t), \tilde{V}_k(t))$ instead of the exact ones $(X_k(t), V_k(t))$.

6.1.1 Consistency Error for Charge Density

The consistency error measures the error is associated with representing a smooth charge density by arbitrarily distributed particles. Since the error in replacing arbitrarily distributed particles with a set of uniformly distributed particles is of the same order as the consistency error (see equation 3.8 in Chapter 3), we can start our analysis by measuring the error associated with representing a smooth charge density on a rectangular grid. In addition, since there is no error in representing ion charge density, the charge density error is reduced to the particle representation of electron charge density. That is,

$$\begin{aligned} e_c^\rho(x,t) &= \left|\rho_e(x,t) - \sum_i q_i \delta_\varepsilon(x - x_i)\right| \\ &\leq \underbrace{\left|\rho_e(x,t) - \int_{\mathbb{R}} \rho_e(y,t)\delta_\varepsilon(x-y)dy\right|}_{e_m(x,t)} \\ &\quad + \underbrace{\left|\int_{\mathbb{R}} \rho_e(y,t)\delta_\varepsilon(x-y)dy - \sum_i q_i \delta_\varepsilon(x - x_i)\right|}_{e_d(x,t)}, \end{aligned} \tag{6.12}$$

where ρ_e denotes the exact electron charge density, and $e_m(x,t)$ is the moment error and $e_d(x,t)$ is the discretization error. Particles are located at the cell center of a rectangular grid. This replacement enables the

analysis to be based on the well-developed interpolation theorem for data on a grid.

Moment Error

Moment error $e_m(x,t)$ is a consequence of approximating the exact delta function δ with a discrete delta function δ_ε. It can be estimated as follows using Taylor expansion,

$$\begin{aligned} e_m(x,t) &= \left| \rho_e(x) - \int_{\mathbb{R}} \rho_e(y)\delta_\varepsilon(x-y)dy \right| \\ &= \left| \rho_e(x) - \int_{\mathbb{R}} \sum_{\ell=0}^{\infty} \frac{(-1)^\ell}{\ell!} \frac{\partial \rho_e(x)^\ell}{\partial x^\ell} (x-y)^\ell \delta_\varepsilon(x-y)dy \right| \\ &= \left| \rho_e(x) - \sum_{\ell=0}^{\infty} \frac{(-1)^\ell}{\ell!} \frac{\partial \rho_e(x)^\ell}{\partial x^\ell} \int_{\mathbb{R}} (x-y)^\ell \delta_\varepsilon(x-y)dy \right|. \end{aligned} \tag{6.13}$$

If the moment condition satisfies

$$\int_{\mathbb{R}} (x-y)^\ell \delta_\varepsilon(x-y)dy = \begin{cases} 1 & \ell = 0 \\ 0 & \ell < m \le p, \end{cases} \tag{6.14}$$

where p is the order of differentiability of the charge density $\rho_e(x)$, and m is the moment with the largest contribution to e_m, then assuming that $\int_{\mathbb{R}} |\delta_\varepsilon(x-y)dy|$ is bounded, the moment error is estimated as

$$e_m = O(\varepsilon^m). \tag{6.15}$$

This implies that a positive function has $m \le 2$.

Discretization Error

The discretization error is usually estimated by numerical quadrature rules. But in some cases, particularly when B-splines (M_n) are used as the discrete delta function, the smooth interpolation formula of Schoenberg [25] gives a more accurate estimate. In the following, we first give Monaghan's derivation for the smooth interpolation formula [38]. A discretization error estimate based on the smoothed interpolation formula is presented next.

The error associated with approximating the convolution integral by a sum can be represented as

$$\sum_{p=-\infty}^{\infty} \rho_e(ph_x)\delta_\varepsilon(x-ph_x) - \int_{-\infty}^{\infty} \rho_e(uh_x)\delta_\varepsilon(x-uh_x)du = 2\sum_{r=1}^{\infty} \int_{-\infty}^{\infty} \rho_e(uh_x)\delta_\varepsilon(x-uh_x)\cos(2\pi ru)du, \tag{6.16}$$

where $p \in \mathbb{Z}$, $u \in \mathbb{R}$, and δ_ε is a discrete delta function. This formula is usually known as the Poisson summation formula [60]. Now let us expand the right-hand-side of the function around $uh_x = x$, and take the dominant contribution term $r = 1$. It becomes

$$\begin{aligned} e_{r=1} &= \left| 2\int_{-\infty}^{\infty} \rho_e(uh_x)\delta_\varepsilon(x-uh_x)\cos(2\pi u)du \right| \\ &= \left| \frac{2}{h_x}\sum_{a=0}^{\infty}\frac{\rho_e^a(x)}{a!}\int_{-\infty}^{\infty} w^a\delta_\varepsilon(w)\cos(\frac{2\pi(x+w)}{h})dw \right|, \end{aligned} \tag{6.17}$$

where $w = uh_x - x$. The integral in equation (6.17) has the form

$$\begin{aligned} I_a(x,\varepsilon) &= \int_{-\infty}^{\infty} w^a\delta_\varepsilon(w)\cos\left(\frac{2\pi(x+w)}{h_x}\right)dw \\ &= \cos\left(\frac{2\pi x}{h_x}\right)\int_{-\infty}^{\infty} w^a\delta_\varepsilon(w)\cos\left(\frac{2\pi w}{h_x}\right)dw \\ &\quad + \sin\left(\frac{2\pi x}{h_x}\right)\int_{-\infty}^{\infty} w^a\delta_\varepsilon(w)\sin\left(\frac{2\pi w}{h_x}\right)dw. \end{aligned} \tag{6.18}$$

It is easy to observe that

$$I_a(x,\varepsilon) = \begin{cases} \cos\left(\frac{2\pi x}{h_x}\right)\frac{(-1)^{a/2}}{(2\pi^a)}\left[\frac{d^a}{dt^a}g(z)\right]_{z=1/h_x} & a \text{ is even} \\ \sin\left(\frac{2\pi x}{h_x}\right)\frac{(-1)^{(a-1)/2}}{(2\pi^a)}\left[\frac{d^a}{dt^a}g(z)\right]_{z=1/h_x} & a \text{ is odd,} \end{cases} \tag{6.19}$$

where

$$g(z) = \int_{-\infty}^{\infty}\delta_\varepsilon(w)\cos(2\pi zw)\,dw \tag{6.20}$$

is the Fourier cosine transformation of the discrete delta function.

With the above smooth interpolation formula, the discretization error in equation (6.12) is

$$\begin{aligned} e_d(x,t) &= \left| \int_{\mathbb{R}} \rho_e(y,t)\delta_\varepsilon(x-y)dy - \sum_i q_i\delta_\varepsilon(x-x_i) \right| \\ &\le c_1(t)\left[\frac{d^n}{dt^n}g(z)\right]_{z=1/h_x}, \end{aligned} \tag{6.21}$$

where e_d is bounded by the n^{th} order derivative of the Fourier transformation of the discrete delta functions at points $th = \pm 1, \pm 2, \cdots$ with

$$\left[\frac{d^a}{dt^a}g(z)\right]_{z=1/h_x} = 0, \quad \text{when } a < n\ . \tag{6.22}$$

For example, the Fourier cosine transformation of a B-spline (M_n) of order n (Schoenberg [25]) has the form

$$\left[\frac{\sin(\pi z\varepsilon)}{\pi z\varepsilon}\right]^n. \tag{6.23}$$

Since

$$\left[\frac{d^a}{dt^a}\hat{M}_n(z)\right]_{z=1/h_x} = \begin{cases} 0 & a < n \\ \varepsilon^n \left(\frac{h_x}{\varepsilon}\right)^n & a = n, \end{cases} \tag{6.24}$$

where $\hat{M}_n(z) = \int_{-\infty}^{\infty} M_n(w,\varepsilon)\cos(2\pi z w)dw$, when M_n is taken as the discrete delta function, the discretization error is bounded by

$$e_d(x,t) = O\left(\varepsilon^n \left(\frac{h_x}{\varepsilon}\right)^n\right). \tag{6.25}$$

The ratio $\frac{h_x}{\varepsilon}$ will be shown to be important for convergence. Compared with the trapezoidal sum rule approach, the smooth interpolation formula provides us a higher order error bound for the same discrete delta function, i.e., the B-spline of order two.

Combining the moment error and the discretization error, we get the consistency estimate using B-spline type discrete delta functions as

$$e_c^{\rho_e}(x,t) = O\left(\varepsilon^m + \varepsilon^n \left(\frac{h_x}{\varepsilon}\right)^n\right). \tag{6.26}$$

For a nonnegative function, the moment order m is less than or equal to 2, although the discretization order n can be increased arbitrarily. B-spline type discrete delta functions only lead to a second-order method since all B-splines functions are nonnegative (see p. 228 of [17]).

6.1.2 Consistency Error for Electric Field

The consistency error of the electric field can be estimated as in §6.1.1. However, this limits the discretization error order to that obtained from by using the quadrature rule. Fortunately, we can get a better estimate by regarding it as the field induced by the consistency error of the charge density. Since the discrete Laplacian operator is a stable operator, by using the same analysis as in [55,46], we obtain

$$e_c^E(x,t) = O\left(\varepsilon^m + \varepsilon^n \left(\frac{h_x}{\varepsilon}\right)^n + \varepsilon^2\right). \tag{6.27}$$

The last term ε^2 is the error associated with using a second-order Poisson solver.

6.1.3 Stability Error

The stability error of the charge density is written as

$$\begin{aligned} e_s^{\rho}(x,t) &= \left| \sum_k q_k \delta_\varepsilon(x - X_k(t)) - \sum_k q_k \delta_\varepsilon(x - \tilde{X}_k(t)) \right| \\ &= \left| \sum_k q_k \int_{\tilde{X}_k(t)}^{X_k(t)} \frac{\partial \delta_\varepsilon}{\partial y}(x,y)dy \right|. \end{aligned} \tag{6.28}$$

The sum includes all particles whose have intervals with endpoints $X_k(t)$ and $\tilde{X}_k(t)$ intersect the support of the discrete delta function $[x-\varepsilon, x+\varepsilon]$. That is

$$|X_k(t) - x| \leqq \varepsilon + |\tilde{X}_k(t) - X_k(t)|. \tag{6.29}$$

We estimate the bound of e_s^{ρ} by following Cottet and Raviart's (Lemma 10 in [1]) as below.

Lemma 6.1.1. *Assuming that the distribution function satisfies the initial condition*

$$|f(x,v,0)| \leq c(1+|v|)^{-\lambda}, \quad \lambda > 1, \quad x, v \in \mathbb{R}, \tag{6.30}$$

and

$$\max_k |\tilde{X}_k(t) - X_k(t)| \leq L - 2\varepsilon,$$

then there exists a constant $C(T) > 0$ independent of h such that for $0 \leq t \leq T$

$$\left| \sum_k q_k \int_{\tilde{X}_k(t)}^{X_k(t)} \frac{\partial \delta_\varepsilon}{\partial y}(x,y)dy \right| \leq C(T)\frac{1}{\varepsilon}\left(1 + \frac{h}{\varepsilon}\right) \max_k |\tilde{X}_k(t) - X_k(t)|, \tag{6.31}$$

where $h = \sqrt{h_x^2 + h_v^2}$.

Proof. Considering particles that satisfy equation (6.29), their initial velocity position can be represented as

$$m \leqq v_k \leqq m+1, \quad m \in \mathbb{Z}. \tag{6.32}$$

Since

$$V_k(t) = v_k - \int_0^t E(X_k(s), s)ds, \tag{6.33}$$

there exists a constant $d = d(T)$ such that

$$m - d \leqq V_k(t) \leqq m + 1 + d, \quad 0 \leqq t \leqq T. \tag{6.34}$$

Let $B_k(t)$ be the image of the mapping $(x_k, v_k) \longrightarrow (X_k(t), V_k(t))$ and $J(m,t)$ the set of k which satisfy equation (6.29) and (6.34), then it is easy to see that $B_k(t)$ intersects the rectangle

$$[x - \varepsilon - \max_k |\tilde{X}_k(t) - X_k(t)|, x + \varepsilon + \max_k |\tilde{X}_k(t) - X_k(t)|] \times [m - d, m + 1 + d]. \tag{6.35}$$

We can predict the number of the particles which intersect the rectangle if we notice that $\frac{\partial(X,V)}{\partial(x,v)} = 1$, and

$$meas(B_k(t)) = h_x h_v \qquad diam(B_k(t)) < c_0(T)h. \tag{6.36}$$

That is

$$card(J(m,t)) \leqq \frac{c_1(T)}{h^2}(\varepsilon + h + \max_k |\tilde{X}_k(t) - X_k(t)|), \quad 0 \leqq t \leqq T. \tag{6.37}$$

At this point, we are able to estimate Equation (6.28),

$$\begin{aligned} e_s^{\rho}(x,t) &= \left| \sum_k q_k \int_{\tilde{X}_k(t)}^{X_k(t)} \frac{\partial \delta_\varepsilon}{\partial y}(x,y)dy \right| \\ &\leqq \sum_{m \in \mathbb{Z}} \sum_{k \in J(m,t)} |q_k| \left| \int_{\tilde{X}_k(t)}^{X_k(t)} \frac{\partial \delta_\varepsilon}{\partial y}(x,y)dy \right| \\ &\leqq c_2(T) \left(\varepsilon + h + \max_{k \in \mathbb{Z}} |\tilde{X}_k(t) - X_k(t)| \right) \sum_{m \in \mathbb{Z}} (1 + |m|)^{-\lambda} \max_{k \in J(m,t)} \left| \int_{\tilde{X}_k(t)}^{X_k(t)} \frac{\partial \delta_\varepsilon}{\partial y}(x,y)dy \right| \\ &\leqq c_3(T) \left((\varepsilon + h) \max_{k \in J(m,t)} \left| \int_{\tilde{X}_k(t)}^{X_k(t)} \frac{\partial \delta_\varepsilon}{\partial y}(x,y)dy \right| + \max_{k \in \mathbb{Z}} |\tilde{X}_k(t) - X_k(t)| \int_{-\infty}^{\infty} |\frac{\partial \delta_\varepsilon}{\partial y}(x,y)|dy \right). \end{aligned} \tag{6.38}$$

Since

$$\left| \int_{\tilde{X}_k(t)}^{X_k(t)} \frac{\partial \delta_\varepsilon}{\partial y}(x,y)dy \right| \leqq c_4(T) \frac{1}{\varepsilon^2} \max_k |\tilde{X}_k(t) - X_k(t)|, \tag{6.39}$$

and

$$\int_{-\infty}^{\infty} |\frac{\partial \delta_\varepsilon}{\partial y}(x,y)|dy \leqq c_5(T) \frac{1}{\varepsilon}, \tag{6.40}$$

we can prove that

$$\left| \sum_k q_k \int_{\tilde{X}_k(t)}^{X_k(t)} \frac{\partial \delta_\varepsilon}{\partial y}(x,y)dy \right| \leq C(T) \frac{1}{\varepsilon} \left(1 + \frac{h}{\varepsilon} \right) \max_k |\tilde{X}_k(t) - X_k(t)|. \tag{6.41}$$

Similarly, we can prove the stability error of the electric field (see [1]) that

$$\begin{aligned} e_s^E(x,t) &= \left| \sum_k q_k K_\varepsilon(x - X_k(t)) - \sum_k q_k K_\varepsilon(x - \tilde{X}_k(t)) \right| \\ &= \left| \sum_k q_k \int_{\tilde{X}_k(t)}^{X_k(t)} \frac{\partial K_\varepsilon}{\partial y}(x,y)dy \right| \qquad (6.42) \\ &\leq C'(T)\left(1 + \frac{h}{\varepsilon}\right) \max_k |\tilde{X}_k(t) - X_k(t)|. \qquad (6.43) \end{aligned}$$

6.1.4 Convergence

The remaining step is to estimate $\max_k |\tilde{X}_k(t) - X_k(t)|$ to determine the convergence of the PIC method.

By using equation (1.22) and the exact trajectory function we have

$$(\tilde{X}_k - X_k)(t) = \int_0^t (\tilde{V}_k - V_k)(t')dt', \qquad (6.44)$$

and

$$\begin{aligned} (\tilde{V}_k - V_k)(t) &= -\int_0^t (\tilde{E}(\tilde{X}_k(t'),t') - E(X_k(t'),t'))dt' \\ &= -\int_0^t ((\tilde{E} - E)(\tilde{X}_k(t'),t') + (E(\tilde{X}_k(t'),t') - E(X_k(t'),t')))dt'. \end{aligned} \qquad (6.45)$$

Summing the above two equations, we obtain

$$\begin{aligned} |(\tilde{X}_k - X_k)(t)| + |(\tilde{V}_k - V_k)(t)| \leq & \int_0^t ((\tilde{E} - E)(\tilde{X}_k(t'),t')dt' \\ & + \int_0^t \max\left(1, \left\| \frac{\partial E}{\partial x}(\cdot,t') \right\|_{L^\infty(\mathbb{R})}\right) \left(\left| \left(\tilde{X}_k - X_k\right)(t') \right| + \left| \left(\tilde{V}_k - V_k\right)(t') \right| \right) dt'. \end{aligned} \qquad (6.46)$$

By combining equation (6.27) and (6.43), we know that

$$|E(x,t) - \tilde{E}(x,t)| \leq c_6(T)\left(\varepsilon^m + \varepsilon^n \left(\frac{h_x}{\varepsilon}\right)^n + \varepsilon^2 + \left(1 + \frac{h}{\varepsilon}\right) \max_k |\tilde{X}_k(t) - X_k(t)|\right). \qquad (6.47)$$

If we set $e(t) = \max_k(|(\tilde{X}_k - X_k)(t)| + |(\tilde{V}_k - V_k)(t)|)$, then

$$e(t) \leq c_7(T)\left(\varepsilon^m + \varepsilon^n \left(\frac{h_x}{\varepsilon}\right)^n + \varepsilon^2\right) t + a(T) \int_0^t e(t')dt', \quad 0 \leq t \leq T. \qquad (6.48)$$

where $a(T)$ depends on $\left\| \frac{\partial E}{\partial x}(\cdot,t) \right\|_{L^\infty(\mathbb{R})}$. This is a variation on Gronwall's inequality.

Solving equation (6.48), we obtain the trajectory error bounded as

$$e(t) \leq \frac{c_7}{a}\left(\varepsilon^m + \varepsilon^n\left(\frac{h_x}{\varepsilon}\right)^n + \varepsilon^2\right)(\exp(at)-1), \quad 0 \leq t \leq T, \tag{6.49}$$

The above error bound is the basic convergence result for the PIC method.

By substituting equation (6.49) into (6.31) and combining consistency error (6.26), we can obtain the error bound of the charge density as

$$\begin{aligned} & |\rho(x,t) - \tilde{\rho}(x,t)| \\ = \; & O\left(\varepsilon^m + \varepsilon^n\left(\frac{h_x}{\varepsilon}\right)^n + \left(\varepsilon^{m-1} + \varepsilon^{n-1}\left(\frac{h_x}{\varepsilon}\right)^n + \varepsilon\right)(\exp(at)-1)\right). \end{aligned} \tag{6.50}$$

Similarly, by substituting equation (6.49) into (6.43) and combining consistency error (6.27), the electric field error is bounded as

$$\begin{aligned} & |E(x,t) - \tilde{E}(x,t)| \\ = \; & O\left(\varepsilon^m + \varepsilon^n\left(\frac{h_x}{\varepsilon}\right)^n + \varepsilon^2 + \left(\varepsilon^m + \varepsilon^n\left(\frac{h_x}{\varepsilon}\right)^n + \varepsilon^2\right)(\exp(at)-1)\right). \end{aligned} \tag{6.51}$$

6.2 Summary

Equation (6.50) and (6.51) provides guidance for the design of particle methods in solving the Vlasov-Poisson equation, and for the choice of optimal parameters. First, for convergence, the ratio of the initial particle mesh spacing in physical space and the Poisson solver mesh spacing needs to be bounded such that $\frac{h_x}{\varepsilon} \leq 1$. This is the so-called *overlapping* condition in particle methods. Second, we notice that the consistency error is amplified by a time dependent exponential term. This implies that particle methods can lose accuracy for problems with large density change and for problems running over long times. The remapping algorithm we provide controls this factor by eliminating the exponential term periodically.

Chapter 7

Numerical Results

We demonstrate the PIC method with adaptive phase-space remapping on a set of classical plasma problems in both one and two dimensions. This includes linear and nonlinear Laudau damping, the two stream instability and beam focusing under uniform external electric field. We also demonstrate the collisional model on the two stream instability problem in one dimension. In implementing remapping on a hierarchy of locally refined grids for plasma problems, we refine the grid in velocity space only, since small structures occur in velocity space, but not in physical space in those tests.

7.1 Notation and Assumptions

The following notation is used to denote the numerical parameters in this chapter. h is the phase space mesh spacing, where $h = (h_x, h_{v_x})$ in one dimension and $h = (h_x, h_y, h_{v_x}, h_{v_y})$ in two dimensions. In two dimensions, the mesh spacing in physical space and in velocity space are uniform, respectively, where $h_x = h_y$ and $h_{v_x} = h_{v_y}$. $\varepsilon = (\varepsilon_x, \varepsilon_y)$ denotes the mesh spacing of the Poisson solver in two dimensions, where $\varepsilon_x = \varepsilon_y$.

The ratio of the initial particle mesh spacing in physical space and the solver mesh spacing is $r_h = \frac{h_x}{\varepsilon_x}$. Remapping is applied periodically with parameter $K = \frac{\Delta t_r}{\Delta t_p}$, where Δt_r and Δt_p are the remapping

step size and PIC integration step size, respectively. We set $K = 5$ for all tests below. That is, we apply remapping every 5 PIC steps.

We use Richardson extrapolation for the error estimation. If $\tilde{E}^h$ is the electric field computed with the initial phase space discretization h, Poisson solver mesh spacing ε, and PIC integration step Δt_p, and $\tilde{E}^{2h}$ computed with $2h$, 2ε and $2\Delta t_p$, the relative solution error in direction d is defined as

$$e_d^h = |\tilde{E}_d^h - \tilde{E}_d^{2h}|. \tag{7.1}$$

q is the order of the method and is calculated by

$$q = \min_d \log_2 \left(\frac{||e_d^{2h}||}{||e_d^h||} \right). \tag{7.2}$$

For adaptive remapping, we refine the grid in velocity space only, since small structures occur in velocity space, but not in physical space in the plasma tests below.

7.2 1D Vlasov-Poisson System for Plasma Problems

7.2.1 Linear Landau Damping

The initial distribution for the linear Landau damping problem is

$$\begin{aligned} f(x,v,t=0) &= \frac{1}{\sqrt{2\pi}} \exp(-v^2/2)(1+\alpha\cos(kx)) \\ (x,v) &\in [0, L = 2\pi/k] \times [-v_{\max}, v_{\max}], \end{aligned} \tag{7.3}$$

where $\alpha = 0.01$ is the intensity of the perturbation, $k = 0.5$ is the first mode of the electric field, and $v_{\max} = 10$. In initializing particles, we set a threshold for the particle charge intensity. We ignore particles which have weight less than 1.0×10^{-9}.

In the test, we are interested in the evolution of the electric energy, approximated by

$$\xi_e(t) = \frac{1}{L} \sum_j E(x_j,t)^2 \varepsilon, \tag{7.4}$$

where j is the Poisson solver mesh index and ε is the solver mesh spacing.

This problem has been studied extensively [6,62,8,15,63]. According to Landau's theory, the electric energy $\xi_e(t)$ is expected to decrease exponentially with damping rate $\gamma = 0.153$ and frequency $\omega = 1.416$. We initialize the problem first on a uniform grid in phase space, $h_x = L/32, h_v = v_{\max}/32$, without remapping. The PIC time step is $\Delta t_p = 1/8$. We set the ratio of the particle mesh spacing to the solver mesh spacing to be $r_h = 1/2$. Figure (7.1a) shows that the standard PIC method (without remapping) fails to track the exponential damping. We then solve the same problem by increasing the total number of particles through initializing on two levels of grids, with another finer level covering sub-domain $v \in (-5,5)$ with mesh spacing $h_v = v_{\max}/64$. The mesh spacing in physical space h_x is the same since we apply refinement in velocity space only. Figure (7.1b) shows that for the standard PIC method, increasing the total number of particles improves the result. The exponential decay is successfully captured during the early time of the simulation. However, it fails to track the damping rate after $t = 12$.

We then solve the problem with uniform initialization as in Figure (7.1a), but apply remapping with parameter $K = 5$. Figure (7.1c) shows that PIC with remapping gives a very good approximation (Figure (7.1c)). The computed damping rate $\gamma = 0.140$ and frequency $\omega = 1.395$, agree well with the theoretical values. We only show the results up to $t = 30$ since the simulation will be influenced by the recurrence effect [64] at later time. Figure (7.1d) shows the result obtained by initializing and remapping particles on two levels of grids. Refinement is applied on velocity space on sub-domain $v \in [-5,5]$. Compared with Figure (7.1c), the base grid is coarser in Figure (7.1d). We see that the exponential decay is successfully captured with fewer particles than in Figure (7.1c). Table (7.2) shows the maximum number of particles and the CPU time used during the simulation in each case above. Comparing case (b) and case (d), we see that the new method gives much better results without significantly increasing the computational time. To further demonstrate the new method, we also show the evolution of the electric energy using the standard PIC method with a large number of particles in Figure (7.3). We hope that the standard PIC method can give us similar energy plot as in the remapped PIC method (Figure (7.1d)) by increasing the total number of particles. The total number of particles and the CPU time in Figure (7.3) show that the standard PIC method is much more expensive than the remapped PIC method.

Finally, we compare the convergence rates of the electric field errors in the cases without and with remapping. The particles in the simulation are initialized on two levels of grids with the finer grid covering velocity space sub-domain $v \in (-5,5)$. The particle and the solver mesh spacing ratio is set to be $r_h = 1/2$ as before. Figure (7.4) shows the L_∞ norm of the errors at the case without remapping in three different resolutions. The corresponding convergence rates are shown on the right of the error plots. Comparing Figure (7.5) with (7.4), we see that remapping largely reduces L_∞ error of the electric field. Second-order convergence rates are obtained after applying remapping.

7.2.2 Nonlinear Landau Damping

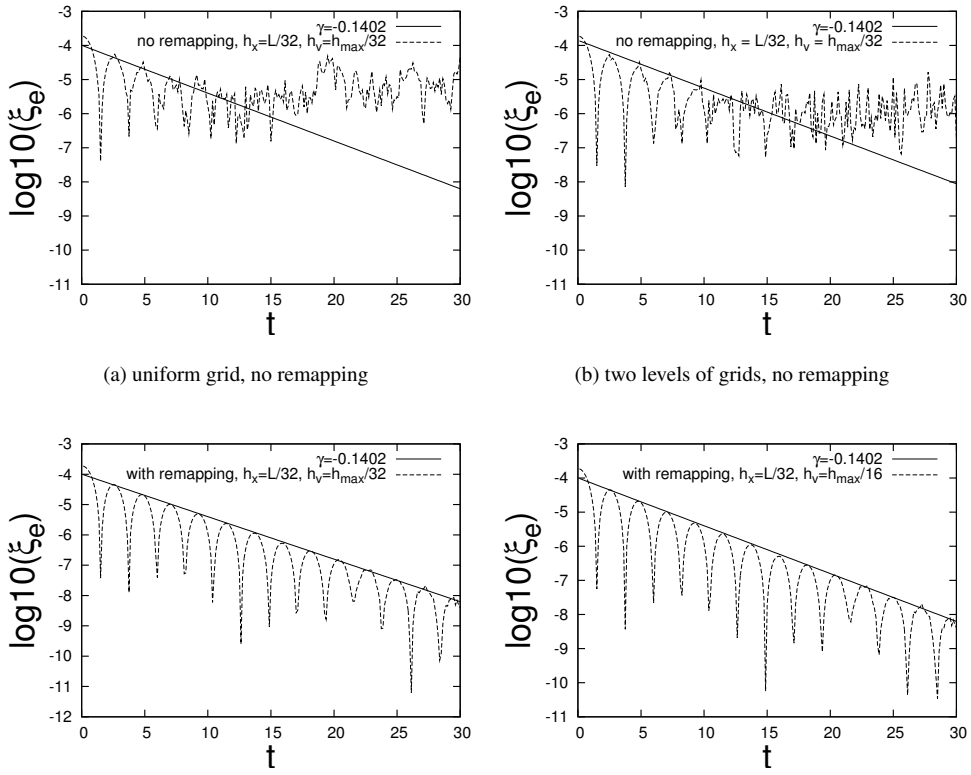
The initial distribution for nonlinear Landau damping is

$$\begin{aligned} f(x,v,t=0) &= \frac{1}{\sqrt{2\pi}} \exp(-v^2/2)(1+\alpha\cos(kx)) \\ (x,v) &\in [0, L = 2\pi/k] \times [-v_{\max}, v_{\max}], \end{aligned} \tag{7.5}$$

where $\alpha = 0.5$, $k = 0.5$ and $v_{\max} = 10$. As in the linear Landau damping problem, particle charges with strength less than 1.0×10^{-9} are ignored.

We initialize and remap the problem on two levels of grids starting with $h_x = L/32$, $h_v = v_{\max}/64$. We refine velocity space on sub-domain $v \in [-5,5]$ as before. The PIC integration step size is $\Delta t_p = 1/8$. The particle and the solver mesh spacing ratio is $r_h = 1/2$. The evolution of the distribution function is shown in Figure (7.6). We see the development of filaments starting around $t = 25$. The filaments are smoothed as they are too small to be resolved around $t = 55$.

We compare the convergence rates of the electric field errors without and with remapping. The particles in the simulation are initialized on two levels of grids with the finer grid covering velocity space sub-domain $v \in (-5,5)$. Figure (7.7) shows the L_∞ norm of the errors in the case without remapping in three different resolutions. The corresponding convergence rates are shown on the right of the error plots. We see that second-order convergence rates are lost early in the simulation, around $t = 10$. Figure (7.8) is the results with remapping. Second-order convergence rates are observed over a longer time, until around $t = 30$.



(a) uniform grid, no remapping

(b) two levels of grids, no remapping

(c) uniform grid, with remapping

(d) two levels of grids, with remapping

Figure 7.1: Comparison of the electric field for linear Landau damping solved by the PIC method with different remapping grids. Scales (h_x, h_v) above denote the particle grid mesh spacing at the base level. In the upper two plots, we start by putting particles on a uniform grid and two levels of grids respectively, and we do not apply remapping during the whole evolution. In the bottom two plots, we start by putting particles on a uniform grid and two levels of grids respectively, but apply periodic remapping. The remapping is done on a uniform grid and two levels of grids respectively, as in the starting layout.

case	total number of particles	CPU time (s)
case (a)	1600	3.44
case (b)	2624	5.10
case (c)	2048	5.90
case (d)	1539	5.21

Figure 7.2: The maximum number of particles and CPU running time used during the simulation in different cases in Figure 7.1. (a): the standard PIC method fails to solve the Landau damping problem, (b): increasing the total number of particles in the standard PIC method improves the results, (c): the PIC method with remapping succeeds in tracking the exponential decay of the electric field, (d): mesh refinement reduces the total number of particles without destroying the simulation.

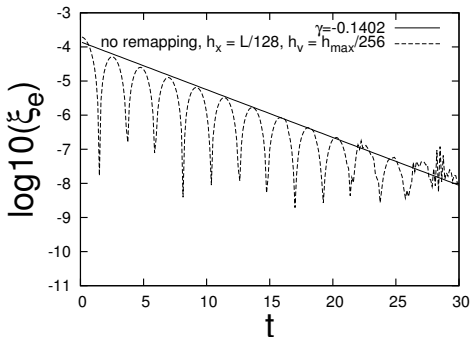
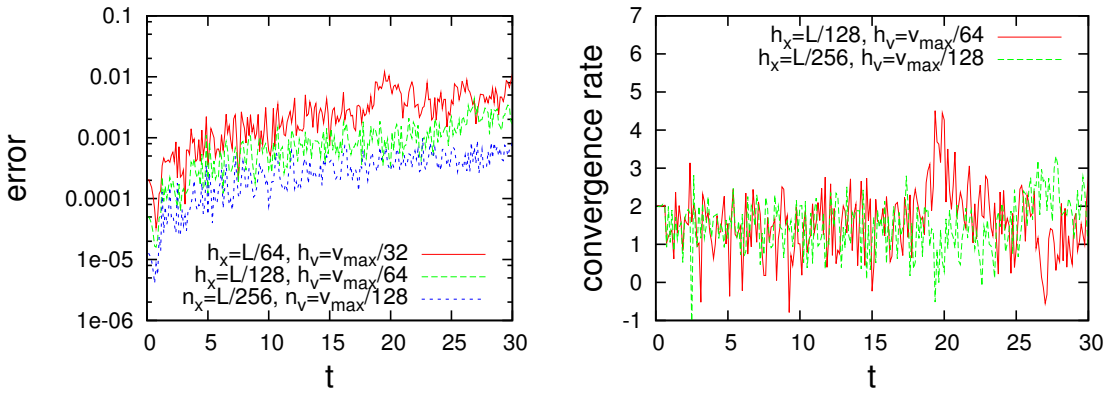


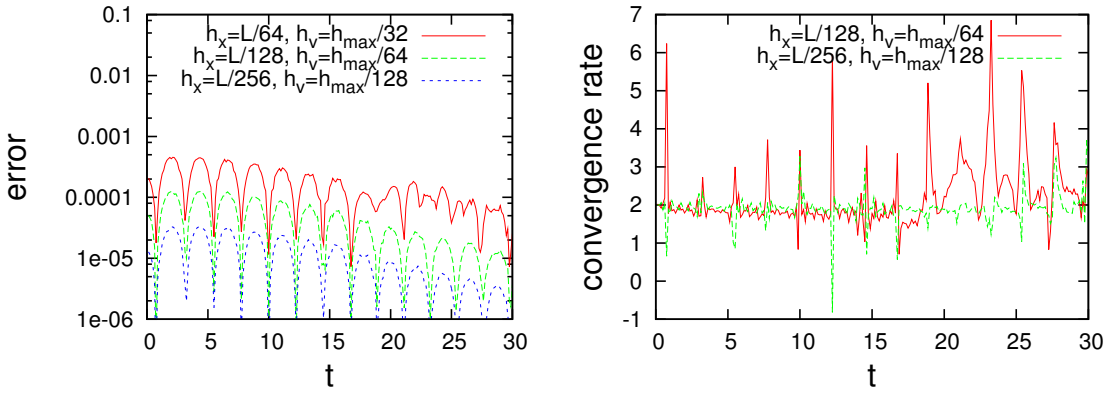
Figure 7.3: The evolution of the electric energy for linear Landau damping problem solved by the standard PIC method. Scales (h_x, h_v) above denote the particle grid mesh spacing at the base level. We have two levels of grids in the plot. The total number of particles is 60416. The CPU running time is 141.6 seconds. Comparing with Figure (7.1d), the standard PIC method is much more expensive to get the similar energy plot.



(a) L_∞ errors of the electric field

(b) Convergence rates

Figure 7.4: Error and convergence rate plots for linear Landau damping problem without remapping. Scales (h_x, h_v) above denote the particle grid mesh spacing at the base level. (a): the L_∞ norm of the electric field errors on three different resolutions. (b): the convergence rates for the errors on the left plot.



(a) L_∞ errors of the electric field

(b) Convergence rates

Figure 7.5: Error and convergence rate plots for linear Landau damping problem with remapping. Scales (h_x, h_v) above denote the particle grid mesh spacing at the base level. (a): the L_∞ norm of the electric field errors on three different resolutions. Comparing with the case without remapping in Figure (7.4), we see the errors are largely reduced. (b): the convergence rates for the errors on the left plot.

The loss of accuracy after $t = 30$ is caused by the development of filaments as in the two stream instability problem.

We also show the amplitude of the first three modes of the electric field in Figure (7.9). The first three modes, $k = 0.5, 1.0, 1.5$, of the electric field E_k are computed as

$$E_k = \frac{1}{2\pi}(E_k^{c2} + E_k^{s2})^{1/2}, \tag{7.6}$$

where

$$E_k^c = \sum_j E_j cos(kx_j)\Delta x, \quad E_k^s = \sum_j E_j sin(kx_j)\Delta x. \tag{7.7}$$

The observed damping and oscillatory behavior agrees very well with other calculations, e.g., grid methods [6, 62, 8, 15]. In Figure (7.10) we show the amplitudes of the first three Fourier modes of the electric field, from a computation without using positivity enforcement. The two cases are very similar. It has been suggested that a grid-based method without positivity preservation will become numerically unstable [63]. Our results do not support this hypothesis for the current problem. We also compare several invariants of the system (see Section 1.2.3): L_p norms and kinetic entropy in the simulations, with and without positivity. Figure (7.11) shows these invariants, the L_1 norm, the L_2 norm and kinetic entropy S, calculated as

$$L_1 = \frac{1}{L}\sum_k |f_k| h_x h_v, \tag{7.8}$$

$$L_2 = \frac{1}{L}\sum_k |f_k|^2 h_x h_v, \tag{7.9}$$

and

$$S = -\frac{1}{L}\sum_k |f_k| \ln |f_k| h_x h_v. \tag{7.10}$$

The results are similar except for L_1, where the absence of positivity is clearly seen. We see a nonphysical increase in entropy caused by remapping in both cases. The entropy is slightly more diffusive in the positivity preserving case because redistributing the undershoot to neighboring cells is dissipative. We compare the distribution function at the same instant in time, $t = 35$, without and with positivity enforcement in Figure (7.12). The comparison of these two plots shows that the high-order interpolation function creates a very

small portion of negative charge and it is safe to be used even without enforcing positivity, at least for the current problem.

To better understand the nonlinear phenomenon, we show the evolution of the spatial integration of the distribution function in Figure (7.13),

$$F(v,t) = \int_0^L f(x,v,t)dx. \tag{7.11}$$

At $t = 15$, a plateau and a bump appear around the phase velocity. This represents particles trapped by the electrostatic field. Later, the bump develops into a wave-like structure spreading across the whole domain around $t = 25$. The wave-like structure is the physical instability, *filaments*, developed in velocity space. This can be verified by comparing $F(v,t)$ running under two different resolutions at the same instant time $t = 35$ in Figure (7.14). By $t \approx 55$, the wave-like structure is smoothed out when the filaments become too small to be resolved.

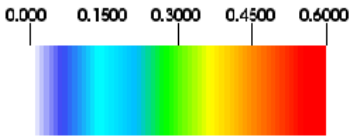
7.2.3 The Two Stream Instability

We consider the two stream instability with the initial distribution

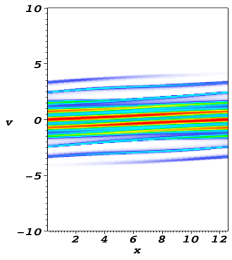
$$\begin{aligned} f(0,x,v) &= \frac{1}{\sqrt{2\pi}} v^2 \exp(-v^2/2)(1+\alpha\cos(kx)), \\ (x,v) &\in [0, L = 2\pi/k] \times [-v_{\max}, v_{\max}], \end{aligned} \tag{7.12}$$

where $\alpha = 0.01$, $k = 0.5$ and $v_{\max} = 10$. We ignore particles which have strength less than 1.0×10^{-9}.

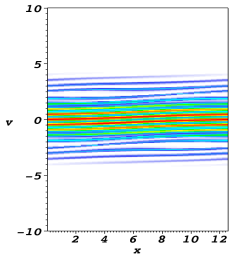
Figure (7.15) shows the evolution of the distribution function in phase space. The distribution function plots are obtained by reproducing the particle-based distribution function through a second-order interpolation on a phase space grid. In the simulation, the distribution function is initialized and remapped on two levels of grids starting with $h_x = L/128$, $h_v = v_{\max}/256$. We refine velocity space on sub-domain $v \in [-5,5]$. We choose $\Delta t_p = 1/32$ for the PIC integration step size and $r_h = 1/2$ for the particle and solver mesh spacing ratio. At $t = 10$, a vortex forms. A hole structure associated with particle trapping is seen around $t = 20$. After $t = 20$, the vortex rotates periodically with $T \approx 17$. Comparing it with the other calculations [6, 62, 8, 15, 63], we see very good agreement.



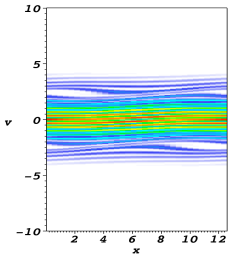
(a) color map



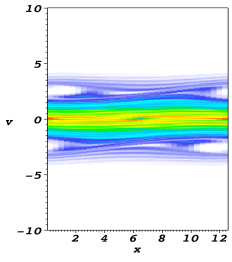
(b) $t = 15$



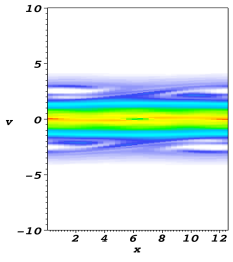
(c) $t = 25$



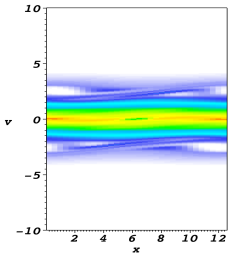
(d) $t = 35$



(e) $t = 45$

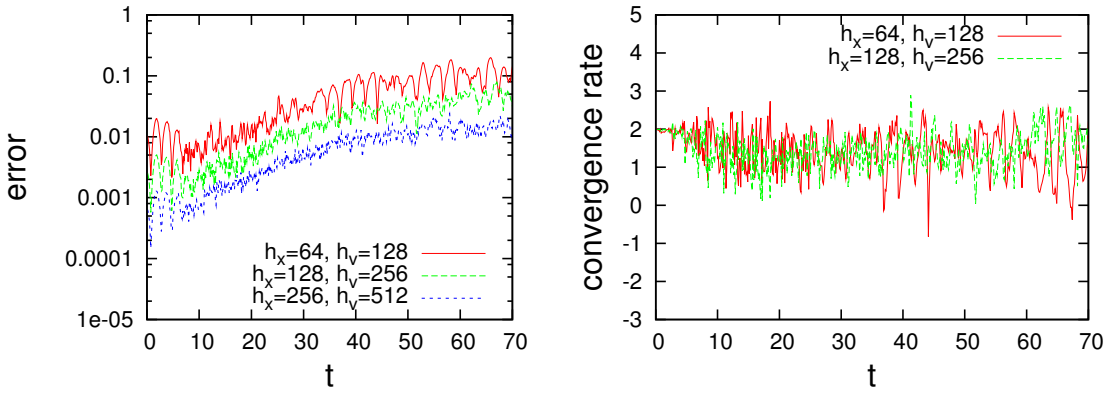


(f) $t = 55$



(g) $t = 65$

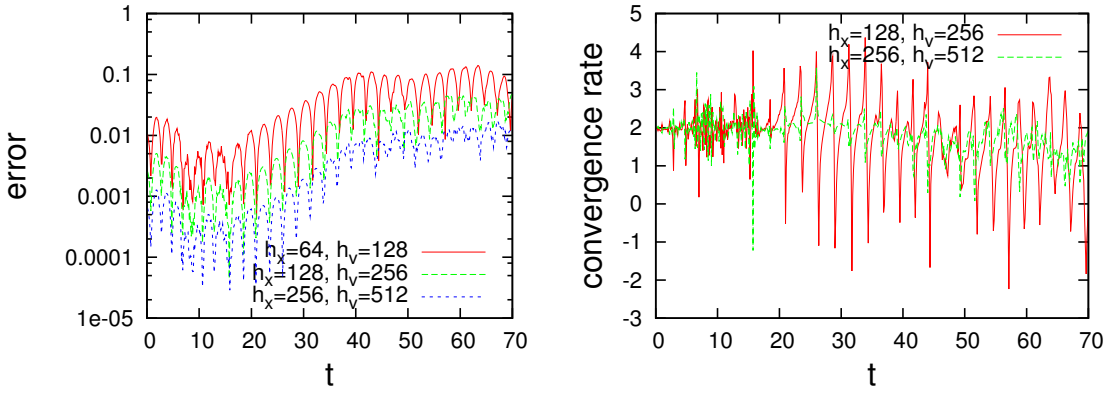
Figure 7.6: The distribution function $f(x,v,t)$ for nonlinear Landau damping at time $t = 15, 25, 35, 45, 55, 65$, respectively. We initialize and remap the distribution function on two levels of grids, with base grid at $h_x = L/32$, $h_v = v_{\max}/64$. The grid is refined by factor of 2 in v space on sub-domain $v \in [-5, 5]$. Filaments start to develop at $t = 25$. At $t = 55$, filaments are smoothed out as they are too small to be resolved by the grid.



(a) L_∞ errors of the electric field

(b) Convergence rates of the errors on the left

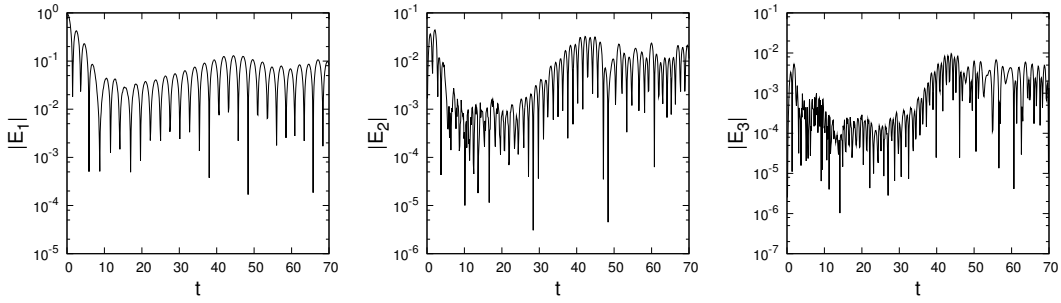
Figure 7.7: Error and convergence rate plots for nonlinear Landau damping running without remapping. (h_x, h_v) above denote the particle grid mesh spacing at the base level. Left: the L_∞ norm of the electric field errors for three different resolutions. Right: the convergence rates for the errors on the left plot. Second-order convergence rates are lost quickly at the early of the simulation, around $t = 10$.



(a) L_∞ errors of the electric field

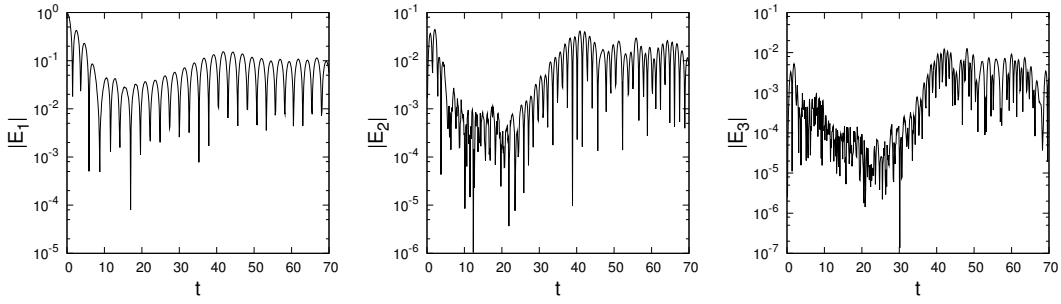
(b) Convergence rates of the errors on the left

Figure 7.8: Error and convergence rate plots for nonlinear Landau damping running with remapping. (h_x, h_v) above denote the particle grid mesh spacing at the base level. Left: the L_∞ norm of the electric field errors for three different resolutions. Right: the convergence rates for the errors on the left plot. Second-order convergence rates are observed till $t = 30$.



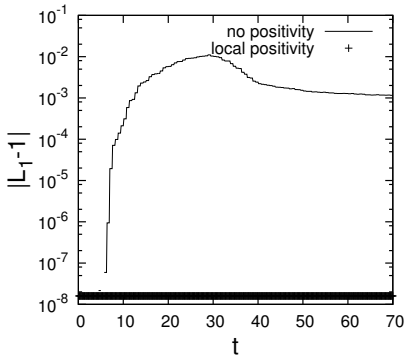
(a) E_1 (b) E_2 (c) E_3

Figure 7.9: The amplitude of the first three Fourier modes of the electric field for nonlinear Landau damping using local positivity algorithm. We solve the problem on two levels of grids, with base grid at $h_x = L/32$, $h_v = v_{\max}/64$. The grid is refined by factor of 2 in v space on sub-domain $v \in [-5,5]$.

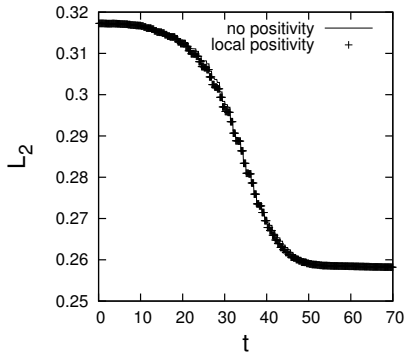


(a) E_1 (b) E_2 (c) E_3

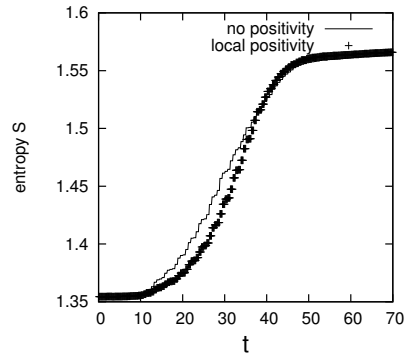
Figure 7.10: Same as Figure 7.9, but without positivity preservation. The plot is very similar to the case with positivity preservation in Figure 7.9. It does not show numerical instability due to unphysical negative charges.



(a) L_1

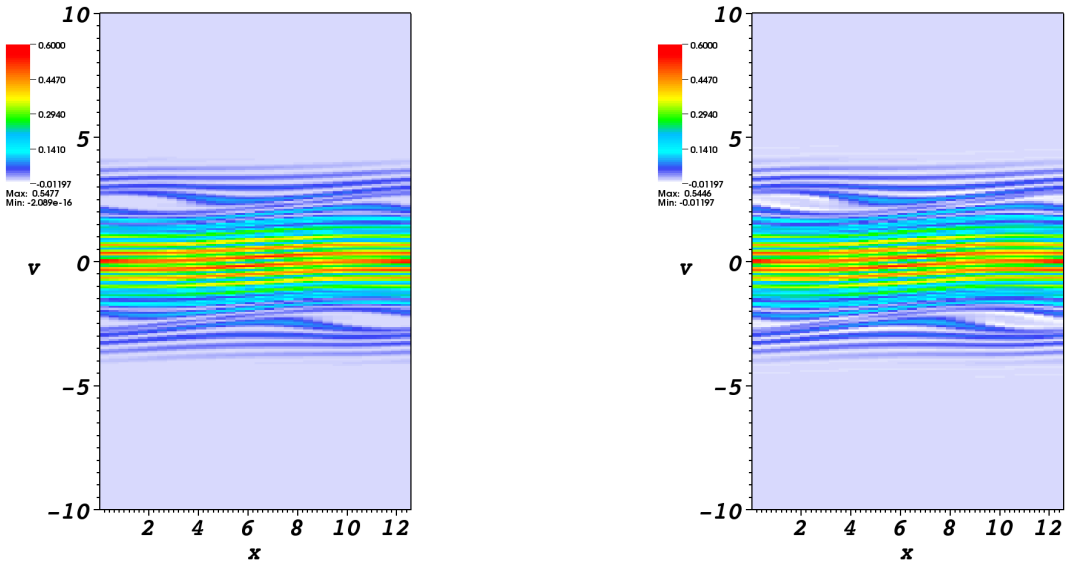


(b) L_2



(c) entropy

Figure 7.11: Comparison of L_1 (Equation (7.8)), L_2 (Equation (7.9)) and entropy S (Equation (7.10)) with and without positivity preservation. The results are similar except for in L_1 where the absence of positivity is clearly seen. The entropy is slightly more diffusive in the positivity preserving case because redistributing the undershoot to neighboring cells is dissipative.



(a) with positivity at t=35

(b) without positivity at t=35

Figure 7.12: Comparision of the distribution function $f(x,v,t)$ for nonlinear Landau damping at the same instant time $t = 35$ with positivity (Left) and without positivity (Right). The maximal and minimal value in each case show that the high-order interpolation function only creates a very small portion of negative charges and it is safe to be used even without enforcing positivity, at least for the current problem.

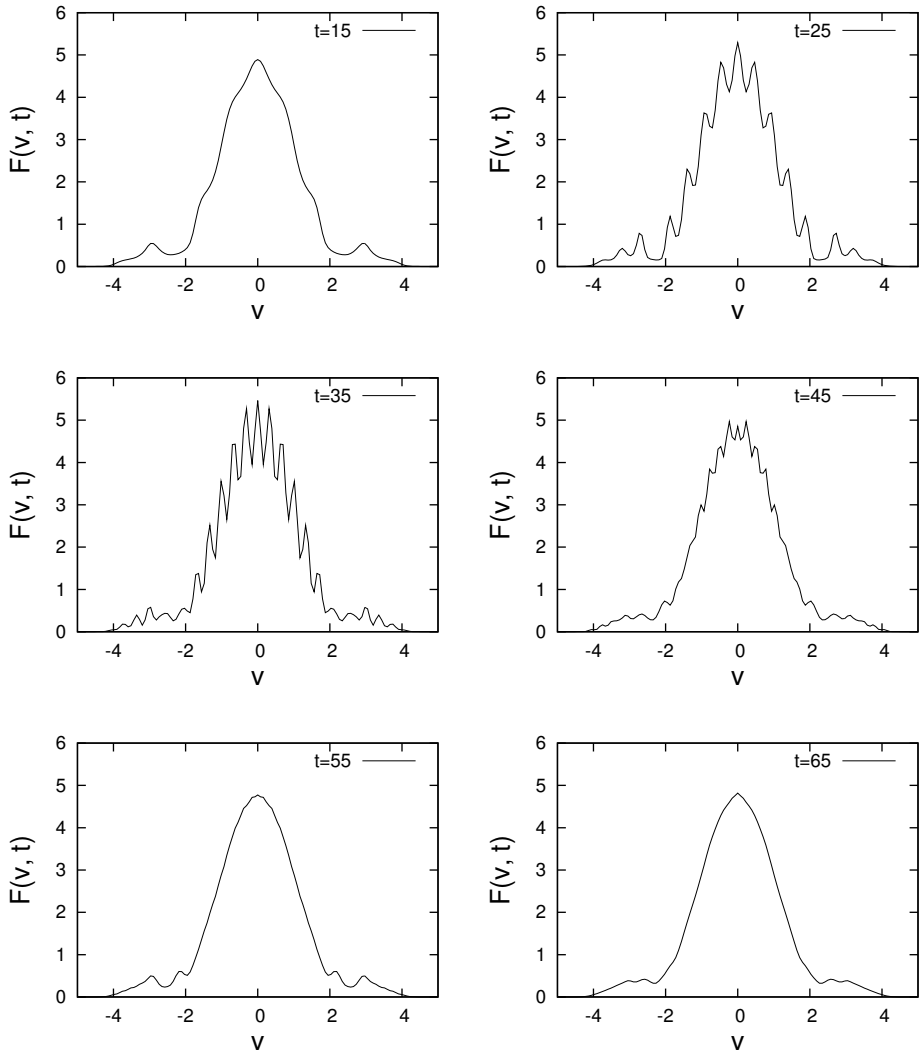


Figure 7.13: The spatial integration of the distribution function $F(v,t)$ for nonlinear Landau damping described in Figure (7.6) at time $t = 15, 25, 35, 45, 55, 65$, respectively. At $t = 15$, a plateau and a bump appear around the phase velocity. At $t = 25$, a wave-like structure begins to develop across the whole domain. The wave-like structure represents filaments. At $t = 55$, the filaments are smoothed out.

We investigate the convergence of the method in two series of tests. In the first one, we compare the convergence rates of the electric field errors with and without remapping. The ratio of the particle and the solver mesh spacing is set to be $r_h = 1/2$. The particles in the simulation are initialized on two levels of grids with the finer grid covering velocity space sub-domain $v \in (-5,5)$. Figure (7.16) shows the L_∞ norm of the errors in the case without remapping in three different resolutions. The corresponding convergence rates are shown on the right of the error plots. Second-order convergence rates are lost early in the simulation, around $t = 20$. Comparing with the results with remapping in Figure (7.17), we see that remapping extends the second-order convergence rates to longer times, until $t = 28$. The loss of accuracy after $t = 28$ is caused by the development of filaments. In the second series of tests, we set $r_h = 1$. According to Cottet's convergence formula [1], the simulation will only result in a first-order method even with remapping. However, Figure (7.18) and (7.19) shows that the PIC method can be second order even with $r_h = 1$. This further demonstrates our error formula (equation 2.32) which says the consistency error is second order as long as $r_h \leq 1$.

We also compare the distribution function at the same instant at time, $t = 20$, by both methods in Figure (7.20). For visualization purpose, in the case of the PIC method without remapping, we interpolate the particle-based distribution on a phase space grid. We see that the classical PIC method results in a very noisy solution in Figure (7.20a). In addition, the maximum of the approximated distribution function has large error comparing with the analytic value, $f_{\max} = 0.3$. Figure (7.20b) shows the distribution function solved by the PIC method with remapping. Compared with the case without remapping, remapping obviously controls numerical noise and reduces the maximum error. We can preserve the maximum of the distribution function by applying the mass redistribution algorithm as in positivity preservation.

Finally, we compare the evolution of the total number of particles in three cases. In the first case, we initialize and remap the problem on two levels of grids, with refinement in velocity space by a factor of 2 on sub-domain $v \in (-5,5)$. The base grid mesh spacing is $h_x = L/64$, $h_v = v_{\max}/128$. In the second case, we initialize and remap the problem on a uniform grid with mesh spacing $h_x = L/64$, $h_v = v_{\max}/256$. In the last one, we initialize the problem on two levels of grids as in the first case, but with grid size double in velocity space in all levels. We do not apply remapping in this case. Figure (7.21) shows that remapping

with mesh refinement reduces the total number of particles dramatically without losing of accuracy. In the remapping cases, the total number of particles increases linearly at the early state of the evolution. The number is saturated as the evolution becomes stable since we ignore the particles when the strength is smaller than the machine accuracy, e.g., 10^{-16}.

7.3 1D Vlasov-Poisson-Fokker-Planck System for Plasma Problems

The initial distribution function is defined in (7.12) as in the collisionless case. We solve the problem on two levels of grids starting at $h_x = L/128$, $h_v = v_{\max}/256$. Velocity space is refined by a factor of 2 on sub-domain $v \in [-5,5]$. We choose $\Delta t_p = 1/32$ for the PIC integration step size. Figure (7.22) shows the errors and the corresponding convergence rates for the simulation for the coupled system. Second-order convergence rates are observed over the whole simulation times. Figure (7.23) shows that filaments are smoothed by the dissipative term which results in a well-resolved system.

7.4 2D Vlasov-Poisson System for Plasma Problems

7.4.1 Linear Landau Damping

The initial distribution for linear Landau damping in 2D is

$$f_0(x,y,v_x,v_y) = \frac{1}{2\pi}\exp(-(v_x^2+v_y^2)/2)(1+\alpha\cos(k_x x)\cos(k_y y)) \tag{7.13}$$

where $\alpha = 0.05$, $k_x = k_y = 0.5$ and $v_{\max} = 6.0$. The physical domain is $(x,y) \in [0, L = 2\pi/k_x] \times [0, L = 2\pi/k_y]$ with periodic boundary conditions. Particle charges with strength less than 1.0×10^{-9} are ignored.

In the first test, we are interested in the evolution of the electric energy as in the 1D case, approximated by

$$\xi_e(t) = \frac{1}{L^2}\sum_i\sum_j E^2(x_i,x_j,t)H_xH_y, \tag{7.14}$$

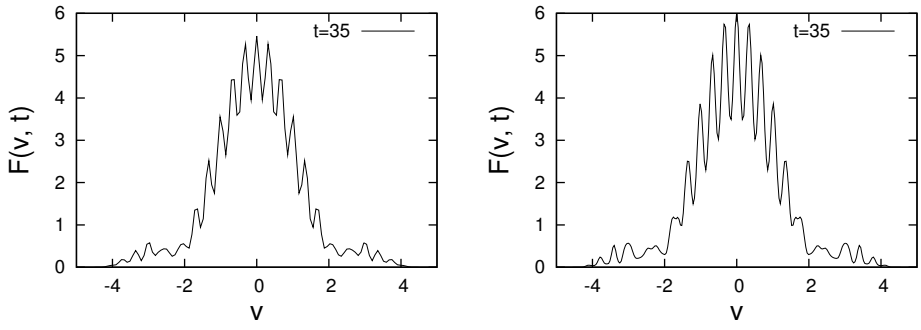
where (i,j) is the 2D Poisson solver mesh index. According to Landau's theory, the electric energy $\xi_e(t)$

is expected to decrease exponentially with damping rate $\gamma = -0.394$. Exponential decay behavior has been observed by many other authors, mostly using grid methods [8, 15, 65, 66].

We initialize the problem on two levels of grids started with $h_x = h_y = L/16, h_{v_x} = h_{v_y} = v_{\max}/8$. Velocity space is refined on sub-domain $v \in [-3,3] \times [-3,3]$ with a refinement ratio 2. The PIC step is $\Delta t_p = 1/4$. We compare the simulation with remapping and without remapping in Figure (7.24). In the case with remapping, the computed damping $\gamma = -0.365$ is very close to the theoretical value. The simulation without remapping fails to track the exponential decay.

In the second test, we compare the electric field errors and corresponding convergence rates with and without remapping in Figure (7.25) and (7.26). We see that remapping largely reduces the electric field errors and improve its corresponding convergence rate.

In the third test, we measure the parallel speedup of the algorithm. Two different approaches are used to measure the speedup. In the first group of tests, we compute the solution for the same problem with different number of processors (weak scaling). We hope that the computational time is inversely proportional to the number of processors used. The results are shown in Table (7.1). The scaling factor is about 3.3, which is near what we would expect, 4.0. The computational overhead is mostly introduced by particle communication between processors in PIC integration step. In the second group of tests, the number of processors is scaled in proportion to the size of the problem in physical domain (strong scaling). Due to the increase of the size in velocity domain, the time and memory usage are expected to be proportional to the size of the problem in velocity space. The results are shown in Table (7.2). The memory usage is very close to what we expect. The time is better than we expect at small processor numbers and close to the expected factor at large processor numbers. The current parallel algorithm based on physical space domain decomposition is in limited usage. A better scaling strategy should be based on phase space domain decomposition, where the number of processors is scaled in proportion to the size of the problem in phase space. In that case, the size of the problem in each processor would remain constant as the problem size increases. The proposed phase space domain decomposition is more suitable for the Vlasov problem for which the data are too numerous to fit into a single processor.



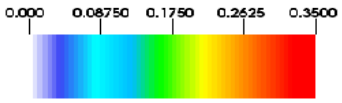
(a) level 0: $h_x = L/32, h_v = v_{\max}/64$

(b) level 0: $h_x = L/64, h_v = v_{\max}/128$

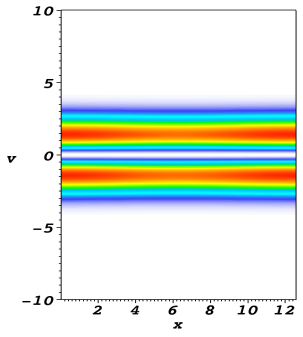
Figure 7.14: Comparsion of $F(v,t)$ at the same instant time $t = 35$ under two different resolutions. Left: the same resolution as in Figure (7.9). Right: increase the resolution by a factor of 2. The wave-like structure does not disappear as we increase the resolution. On the contrary, it is better in resolving the fine scale structure.

processor numbers	4	16	64
patch size in physical space	16×16	8×8	4×4
running time (s)	133.99	40.10	12.03

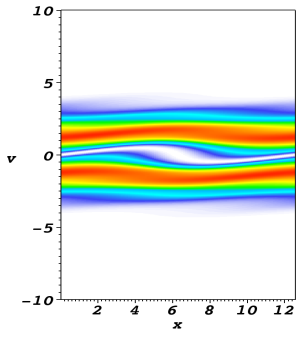
Table 7.1: Comparision of running time scaled by the number of processors on a uniform grid with resolution $32^2 \times 64^2$. The simulation is running on one remapping circle including 10 PIC integration steps with a single positive remapping at step 5. The scaling factor is about 3.3, which is near what we would expect 4.0.



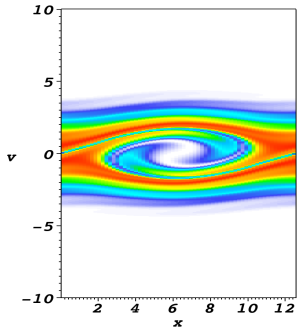
(a) colar map



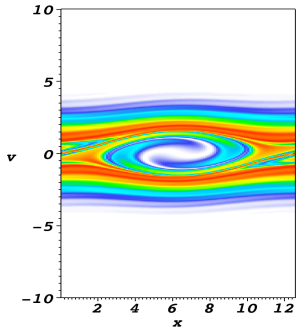
(b) $t = 0$



(c) $t = 10$

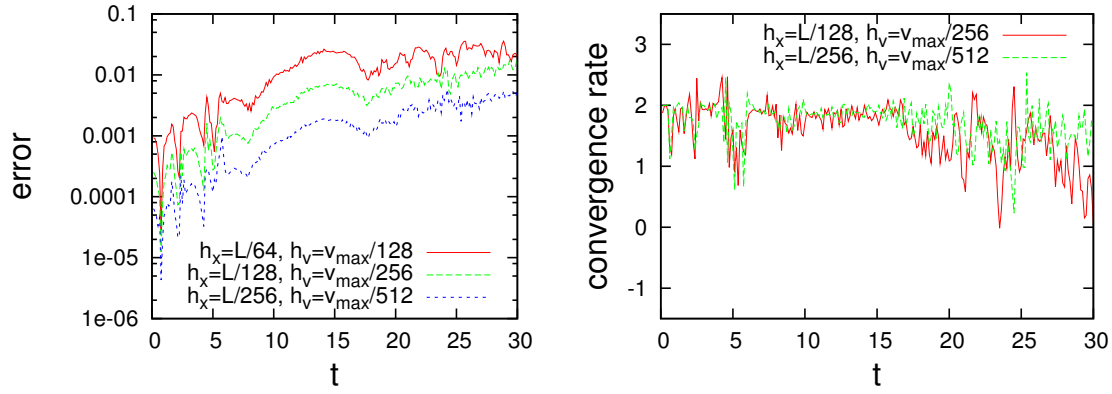


(d) $t = 20$



(e) $t = 30$

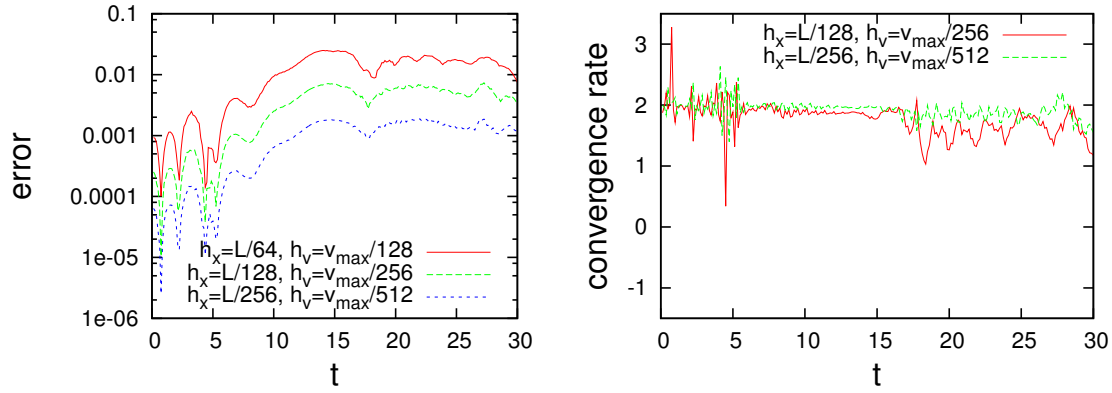
Figure 7.15: The distribution function $f(x,v,t)$ for the two stream instability at time $t = 0, 10, 20, 30$, respectively. We initialize and remap the distribution function on two levels of grids, with base grid at $h_x = L/128$, $h_v = v_{\max}/256$. The grid is refined by factor of 2 in v space on sub-domain $v \in [-5,5]$. We see filamentation phenomena around $t = 30$.



(a) L_∞ errors of the electric field

(b) Convergence rates of the errors on the left

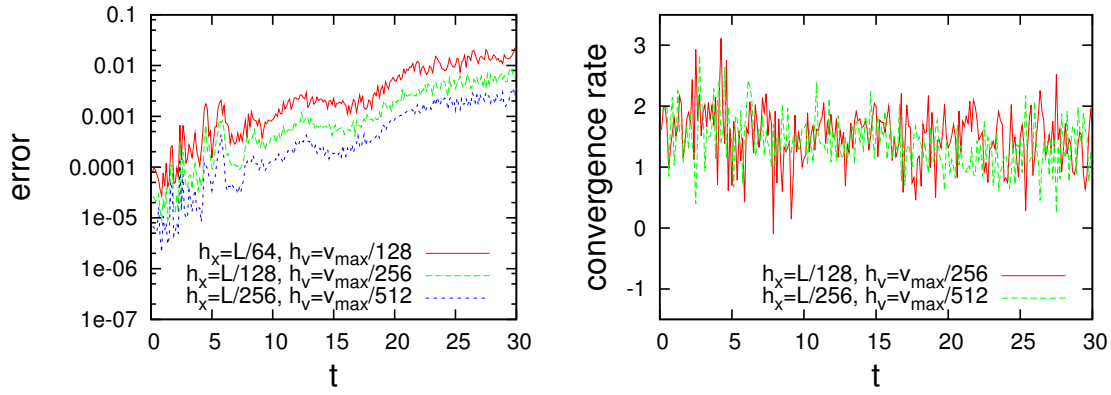
Figure 7.16: Error and convergence rate plots for the two stream instability without remapping. We set $r_h = 1/2$. Scales (h_x, h_v) above denote the particle grid mesh spacing at the base level. Left: the L_∞ norm of the electric field errors for three different resolutions. Right: the convergence rates for the errors on the left plot. Second-order convergence rates are lost around $t = 20$.



(a) L_∞ errors of the electric field

(b) Convergence rates of the errors on the left

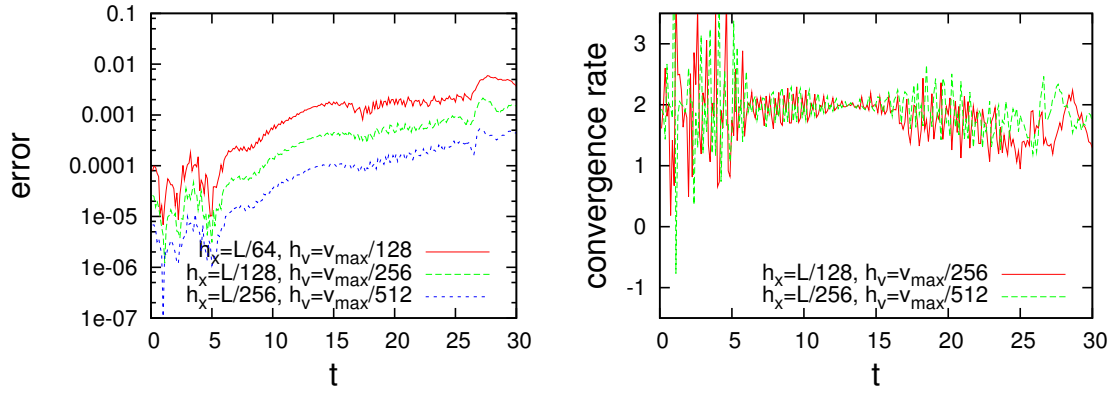
Figure 7.17: Error and convergence rate plots for the two stream instability with remapping. We set $r_h = 1/2$. Scales (h_x, h_v) above denote the particle grid mesh spacing at the base level. Left: the L_∞ norm of the electric field errors for three different resolutions. Right: the convergence rates for the errors on the left plot. Second-order convergence rates are observed till $t = 28$. The lost of accuracy after $t = 28$ is due to filamentation.



(a) L_∞ errors of the electric field

(b) Convergence rates of the errors on the left

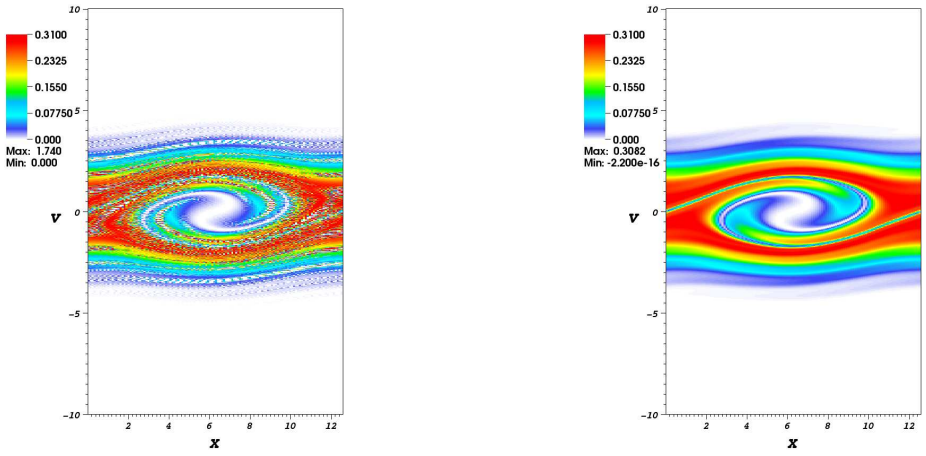
Figure 7.18: Error and convergence rate plots for the two stream instability without remapping. We set $r_h = 1$. Scales (h_x, h_v) above denote the particle grid mesh spacing at the base level. Left: the L_∞ norm of the electric field errors for three different resolutions. Right: the convergence rates for the errors on the left plot.



(a) L_∞ errors of the electric field

(b) Convergence rates of the errors on the left

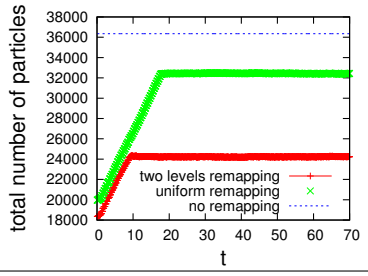
Figure 7.19: Error and convergence rate plots for the two stream instability with remapping. We set $r_h = 1$. Scales (h_x, h_v) above denote the particle grid mesh spacing at the base level. Left: the L_∞ norm of the electric field errors for three different resolutions. Right: the convergence rates for the errors on the left plot.



(a) without remapping at time, $t = 20$ $\tilde{f}_{\max} = 1.591, \tilde{f}_{\min} = 0$

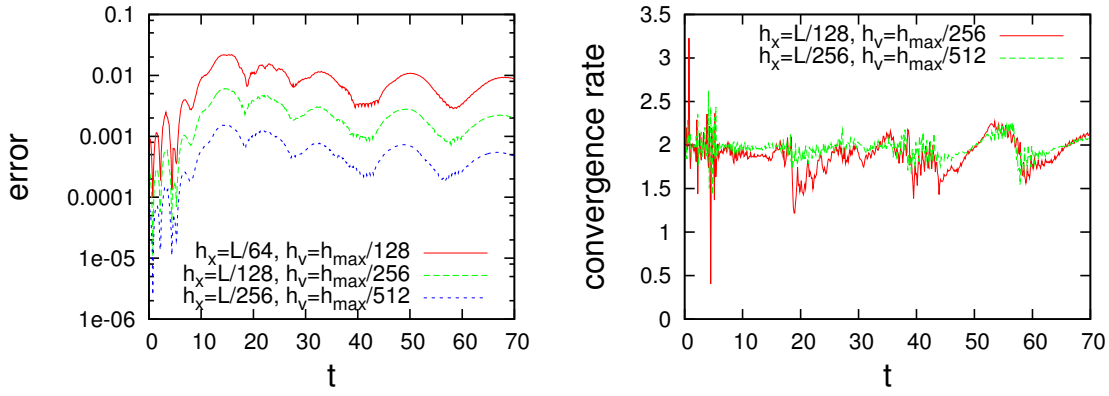
(b) with remapping at time, $t = 20$ $\tilde{f}_{\max} = 0.306, \tilde{f}_{\min} = 0$

Figure 7.20: Comparision of the distribution function at the same instant of time $t = 20$ without (Left) and with remapping (Right). The grid-based distribution function is obtained by reproducing the particle-based distribution function through a second-order interpolation. We initialize the distribution function on two levels of grids, with base grid at $h_x = L/128$, $h_v = v_{\max}/256$. The grid is refined by factor of 2 in v space on sub-domain $v \in [-5, 5]$. The classical PIC method results in a very noisy solution with large maximum error. If we apply remapping to the PIC method, both numerical noise and maximum error are largely reduced. The analytic maximum of the distribution function in the two stream instability is about $f_{\max} = 0.3$.



case	error
case 1: two-levels remapping	0.02292
case 2: uniform remapping	0.02292
case 3: no remapping	0.03546

Figure 7.21: Comparsion of the total number of particles in three different remapping grids. Plus line (case 1): initialize and remap on two levels of grids. The base grid mesh spacing is $h_x = L/64$, $h_v = v_{\max}/128$. We refine the mesh in v space by a factor of 2 on sub-domain $v \in [-5,5]$. Cross line (case 2) : initialize and remap on a uniform grid with mesh spacing $h_x = L/64$, $h_v = v_{\max}/256$. The mesh spacing in case 2 is the same as the finer level mesh spacing in case 1. Dash line (case 3): initialize on two levels of grids, without remapping. The base grid mesh spacing is $h_x = L/64$, $h_v = v_{\max}/256$. We refine the mesh in v space by a factor of 2 on sub-domain $v \in [-5,5]$ as in case 1. Compare with case 1, in case 3, we double the total number of initial particles. The table shows the L_∞ norm error of the electric field in the above simulations at $t = 70$. We see that refinement on velocity space reduces the total number of particles dramatically without losing accuracy.



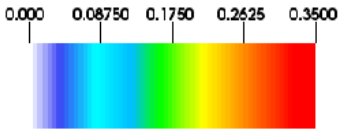
(a) L_∞ error of the electric field

(b) Convergence rate of the error on the left

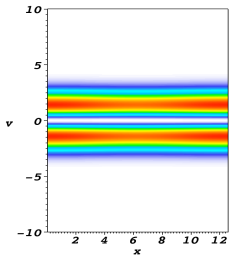
Figure 7.22: Error and convergence plots for the two stream instability including collision on Figure (7.15). Scales (h_x, h_v) above denote the particle grid mesh spacing at the base level. Left: the L_∞ norm of the electric field errors for three different resolutions. Right: the convergence rates for the errors on the left plot. The second-order convergence rate is maintained.

processor numbers	16	64	256
problem size in physical space	16×16	32×32	64×64
problem size in velocity space	32×32	64×64	128×128
patch size in physical space	4×4	4×4	4×4
peak memory usage in each processor (M)	14	51	192
running time (s)	5.52	12.03	46.10

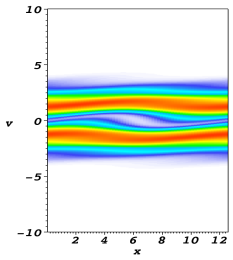
Table 7.2: Comparision of running time scaled by the number of processors in proportion to the problem size in physical domain at different resolutions. The simulation is running on one remapping circle including 10 PIC integration step with a single positive remapping at step 5. The memory usage and running time increase at most by the same factor as the problem size in velocity domain.



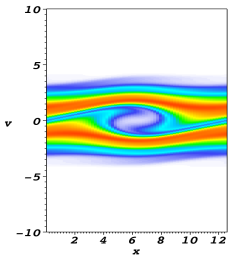
(a) color map



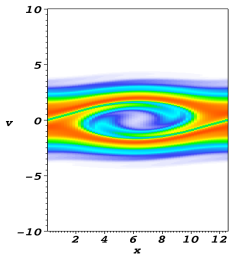
(b) $t = 0$



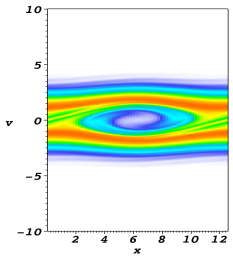
(c) $t = 10$



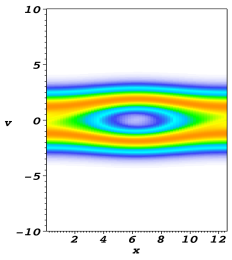
(d) $t = 20$



(e) $t = 30$

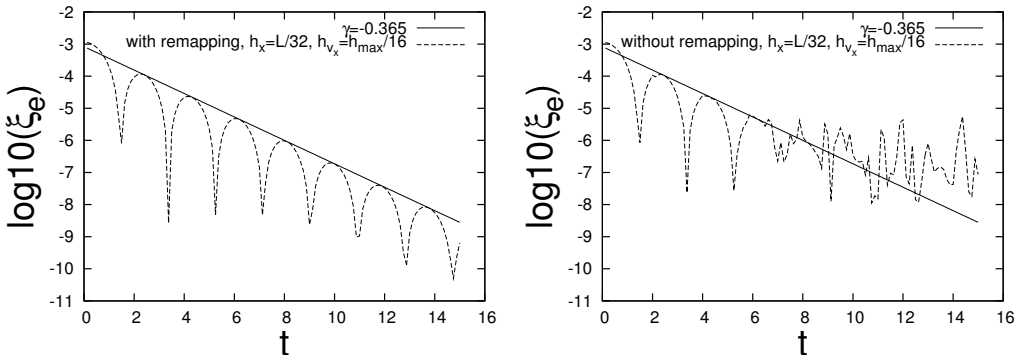


(f) $t = 50$



(g) $t = 70$

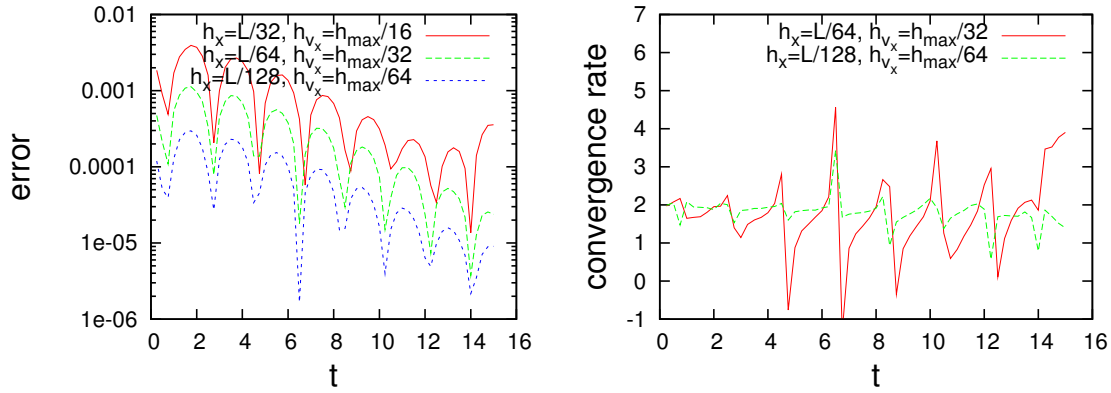
Figure 7.23: The distribution function $f(x,v,t)$ for the two stream instability including collision at time $t = 0, 10, 15, 20, 30, 60$, respectively. Compare with the collisionless case (7.15). Filaments are smoothed by the collision term.



(a) two levels of grids, with remapping

(b) two levels of grids, without remapping

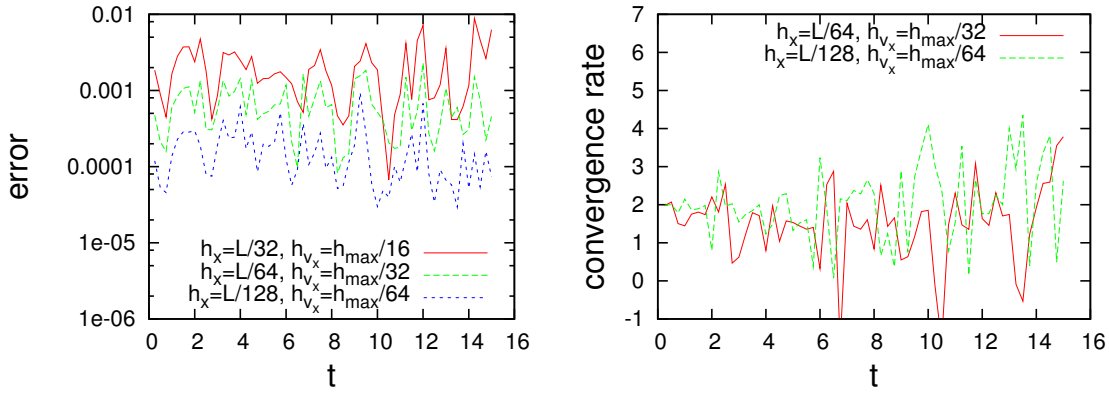
Figure 7.24: The electric field energy for 2D linear Landau damping problem. Scales (h_x, h_{v_x}) above denote the particle grid mesh spacing at the base level. We have $h_x = h_y$ and $h_{v_x} = h_{v_y}$. With remapping, the computed damping rate is very close to the theoretical value. Without remapping, the simulation fails to track the exponential decay after a few damping circles.



(a) L_∞ errors of $\tilde{E}_x$

(b) Convergence rates

Figure 7.25: Error and convergence rate plots for 2D linear Landau damping problem with remapping. Scales (h_x, h_{v_x}) above denote the particle grid mesh spacing at the base level. We have $h_x = h_y$ and $h_{v_x} = h_{v_y}$. Second-order convergence rates are obtained. The oscillatory behavior of the convergence rates is due to the phase error. (a): the L_∞ norm of the electric field errors for three different resolutions. (b): the convergence rates for the errors on the left plot.



(a) L_∞ errors of $\tilde{E}_x$

(b) Convergence rates

Figure 7.26: Error and convergence rate plots for 2D linear Landau damping problem without remapping. Scales (h_x, h_v) above denote the particle grid mesh spacing at the base level. We have $h_x = h_y$ and $h_{v_x} = h_{v_y}$. The errors without remapping are much larger than the case with remapping in Figure (7.25), the errors are much larger. (a): the L_∞ norm of the electric field errors for three different resolutions. (b): the convergence rate for the errors on the left plot.

7.4.2 The Two Stream Instability

The initial distribution for the two stream instability in 2D is

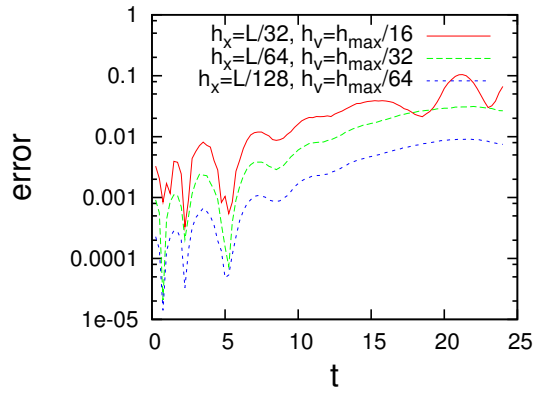
$$f_0(x,y,v_x,v_y) = \frac{1}{12\pi}\exp(-(v_x^2+v_y^2)/2)(1+\alpha\cos(k_x x))(1+5v_x^2) \tag{7.15}$$

where $\alpha = 0.05$, $k_x = 0.5$ and $v_{\max} = 9.0$. The physical domain is $(x,y) \in [0, L = 2\pi/k_x] \times [0, L = 2\pi/k_y]$ with periodic boundary conditions. Particle charges with strength less than 1.0×10^{-9} are ignored.

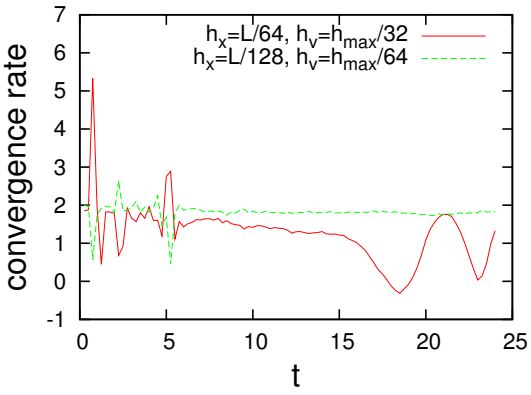
We begin by initializing the problem on two levels of grids started with $h_x = h_y = L/16, h_{v_x} = h_{v_y} = v_{\max}/8$ as in the Landau damping problem. Velocity space is refined on sub-domain $v \in [-4.5, 4.5] \times [-4.5, 4.5]$ with a refinement ratio 2. The PIC time step is $\Delta t_p = 1/4$.

We investigate the method in two series of tests. In the first one, we compare the electric field errors and their convergence rates with and without remapping. Figure (7.28) shows the L_∞ norm of the errors at the case without remapping in two different resolutions. The corresponding convergence rates are shown on the right of the error plots. Second-order convergence rates are lost early in the simulation. Compared with the results with remapping in Figure (7.27), we see that remapping extends the second-order convergence rates to longer times.

We also compare the projected distribution function on space (x, v_x) at the same time instant, $t = 20$, by both methods in Figure (7.29). For visualization purposes, in the case of the PIC method without remapping, we interpolate the particle-based distribution on a phase space grid. We see that the classical PIC method results in a very noisy solution in Figure (7.29b). Figure (7.29a) shows the distribution function obtained by the PIC method with remapping. Compared to the case without remapping, remapping obviously controls numerical noise. The plots also show the maximum and the minimal values of the distribution function. It is obvious that the simulation without remapping introduces a large amount of overshoot and undershoot.

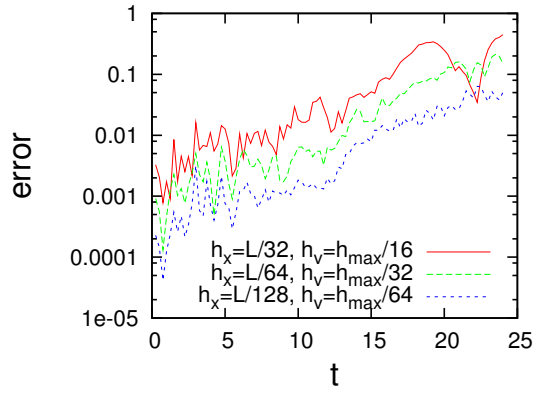


(a) L_∞ errors of $\tilde{E}_x$

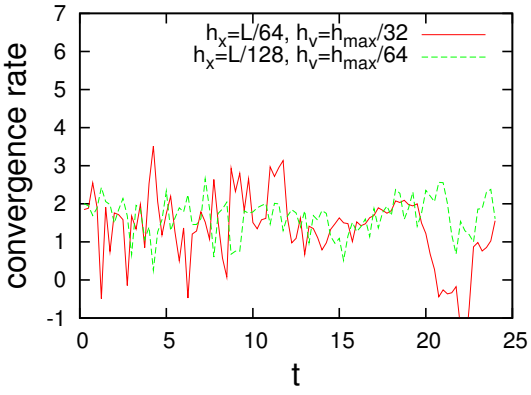


(b) Convergence rates of the errors on the left

Figure 7.27: Error and convergence rate plots for the two stream instability with remapping. Scales (h_x, h_v) above denote the particle grid mesh spacing at the base level. Left: the L_∞ norm of the electric field errors on three different resolutions. Right: the convergence rate for the errors on the left plot.

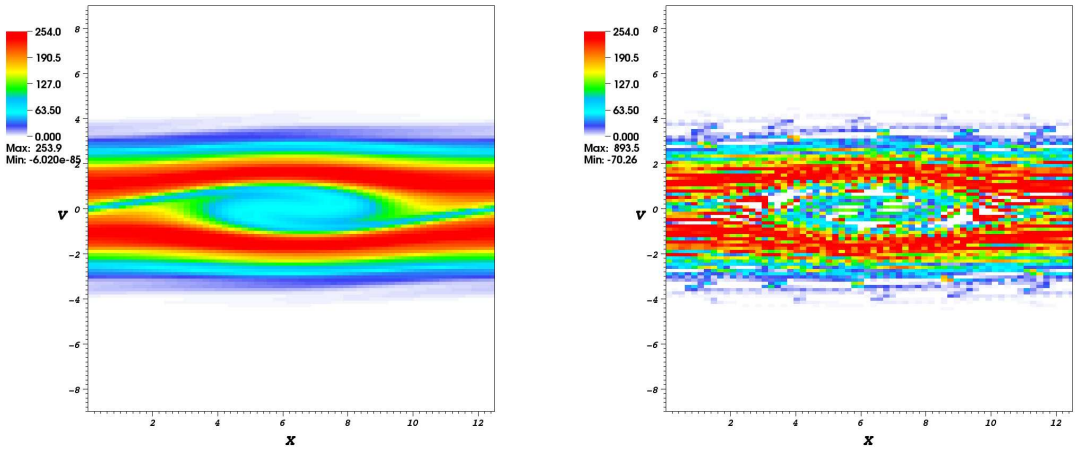


(a) L_∞ errors of $\tilde{E}_x$



(b) Convergence rates of the errors on the left

Figure 7.28: Error and convergence rate plots for the two stream instability without remapping. Scales (h_x, h_v) above denote the particle grid mesh spacing at the base level. Left: the L_∞ norm of the electric field errors on three different resolutions. Right: the convergence rate for the errors on the left plot.



(a) with remapping at time $t = 20$

(b) without remapping at time $t = 20$

Figure 7.29: Comparision of $F(x, v_x)$, the distribution function projected on space (x, v_x), at the same instant of time $t = 20$ with (Left) and without remapping (Right) for the 2D two stream instability problem. The projected value is $F(x, v_x) = \int_0^L \int_{-\infty}^{\infty} f(x, y, v_x, v_y) dy dv_y$. The grid-based distribution function is obtained by reproducing the particle-based distribution function through a second-order interpolation. We initialize the distribution function on two levels of grids, with base grid at $h_x = h_y = L/64$, $h_{v_x} = h_{v_y} = v_{\max}/32$. The grid is refined by factor of 2 in velocity space on sub-domain $v \in [-4.5, 4.5] \times [-4.5, 4.5]$. The classical PIC method results in a very noisy solution with large maximum error. If we apply remapping to the PIC method, both numerical noise and maximum error are largely reduced. The maximum value in the simulation also show that the PIC method without remapping introduces a large amount of overshoot. The undershoot in the case without remapping is a superficial effect due to the projection of 4D data to 2D using high-order interpolation for visualization purpose.

7.5 2D Vlasov-Poisson System for Beam Problems

7.5.1 Paraxial Model

The paraxial model is often used to study the propagation of beams possessing an optical axis in accelerator physics. In the paraxial model, the particles of the beam remain close to the optical axis, such that the transverse width l of the beam is much smaller than the characteristic length scale L. It is an approximation to the steady-state Vlasov-Maxwell equations in three dimensions.

The Vlasov equation in paraxial model without an external magnetic field can be written as

$$\frac{\partial f}{\partial z}+\frac{v}{v_b}\nabla_x f+\frac{q}{\gamma_b m v_b}\left(-\frac{1}{\gamma_b^2}\nabla_x\phi^s+E^e\right)\cdot\nabla_v f=0, \tag{7.16}$$

coupled with the Poisson equation

$$\Delta_x\phi^s=\frac{q}{\varepsilon_0}\int_{\mathbb{R}^2} f(z,x,v)dv, \tag{7.17}$$

where $f=f(z,x,v)$, $x=(x,y)$, $v=(v_x,v_y)$ and $\phi=\phi(x,z)$. v_b is the beam velocity in z direction. q is the charge and m is the mass of one particle. $\gamma_b=(1-\beta^2)^{-1/2}$ where $\beta=\frac{|v|}{c}$ and c is the velocity of light in free space. The boundary conditions for the Poisson equation are infinite domain boundary conditions (see Chapter 5 for James' method of solving the infinite domain Poisson equation). For a complete mathematically analysis of this model, the interested reader can see [67].

7.5.2 The Kapchinchinky-Vladimirsky (K-V) Distribution

The K-V distribution is a measure solution of the paraxial model. It is defined as

$$f(z,x,y,P_x,P_y)=\frac{N_0}{\pi^2\varepsilon_x\varepsilon_y}\delta_0\left(\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{(aP_x-a'x)^2}{\varepsilon_x^2}+\frac{(bP_y-b'y)^2}{\varepsilon_y^2}-1\right), \tag{7.18}$$

where a and b are the solution of the following equations, the so called "envelope" equations

$$a''+k_x(z)a+\frac{2K}{a+b}-\frac{\varepsilon_x^2}{a^3}=0,\quad b''+k_y(z)b+\frac{2K}{a+b}-\frac{\varepsilon_y^2}{b^3}=0, \tag{7.19}$$

where a' denotes the derivative respect to z. $k_x(x)$ and $k_y(z)$ are given external focusing field. ε_x, ε_y are called the emittance. The perveance K is defined as

$$K = \frac{q^2 N_t}{2\pi\varepsilon_0 \gamma_b^3 m v_b^2}. \tag{7.20}$$

With the K-V distribution, the self-consistent electric field described by the Poisson equation satisfies

$$E^s(x,y) = \begin{cases} \left(\frac{qN_0 x}{\pi\varepsilon_0(a+b)a}, \frac{qN_0 y}{\pi\varepsilon_0(a+b)b}\right), & \text{if } \frac{x^2}{a^2} + \frac{y^2}{b^2} \le 1, \\ 0, & \text{as } x, y \text{ goes to } \infty \end{cases} \tag{7.21}$$

The proof can be found in [67].

Assuming that the beam is focused by a uniform electric field given as

$$E^e(x,y) = -\frac{\gamma_b m}{q}\omega_0^2(xe_x + ye_y), \tag{7.22}$$

we get $k_x(z) = k_y(z) = \frac{w_0^2}{v_b^2}$. The the envelope equations are then

$$a'' + \frac{w_0^2}{v_b^2}a + \frac{2K}{a+b} - \frac{\varepsilon_x^2}{a^3} = 0, \quad b'' + \frac{w_0^2}{v_b^2}b + \frac{2K}{a+b} - \frac{\varepsilon_y^2}{b^3} = 0. \tag{7.23}$$

Since k_x and k_y are independent of z, $a' = a'' = b' = b'' = 0$. If $\varepsilon_x = \varepsilon_y$, then $a = b$. The solution of the envelope equation is given as

$$a = \sqrt{\frac{K + \sqrt{K^2 + 4k_x^2\varepsilon_x^2}}{2k_x^2}}. \tag{7.24}$$

Similarly, assuming that we are given a and tune depression $\eta = \frac{\varepsilon_x}{a^2\omega_0}$, the external electric field can be determined as

$$E^e(x,y) = -\frac{\gamma_b m K v_b^2}{qa^2(1-\eta^2)}(xe_x + ye_y), \tag{7.25}$$

7.5.3 Focusing a Beam

Filbet and Sonnendrucker [67] gave a complete derivation and analysis about focusing an arbitrary beam according to its equivalent K-V beam. We summarize their result here. Two beams are called equivalent if they satisfy the following conditions

- they consist of the same number of identical particles N_t

- the particles have the same energy v_b
- they have the same root mean square value of x, y, v_x, v_y, where the root mean square quantities is defined as

$$\chi^{rms}(f) = \sqrt{\frac{\int \chi^2 f dx dy dv_x dv_y}{\int f dx dy dv_x dv_y}}. \tag{7.26}$$

Given a set of physical parameters such as N_t, a and v_b, and an arbitrary initial distribution function $f_0(x, y, v_x, v_y)$, we can focus the beam with the concept of matched K-V beam. First, a unique K-V distribution is determined by the set of physical parameters. If we are also given tune depression η, the external focusing field is then determined. Second, given an arbitrary distribution function, we can scale the function such that the scaled function is equivalent to the given K-V distribution. The scaled system can then be focused by the same external focusing field as we use for the K-V beam.

We scale an arbitrary distribution function f_0 as below. Let us set the scaled function as

$$f(x, y, v_x, v_y) = N_t f_0(\frac{x}{a_0}, \frac{y}{b_0}, \frac{v_x}{c_0}, \frac{v_y}{d_0}). \tag{7.27}$$

Then our purpose is to find the scaling parameters (a_0, b_0, c_0, d_0) such that the scaled system is equivalent to the K-V beam. This is easy if we observe that

$$x^{rms}(f) = a_0 x^{rms}(f_0), \quad y^{rms}(f) = b_0 y^{rms}(f_0), \quad v_x^{rms}(f) = c_0 v_x^{rms}(f_0), \quad v_y^{rms}(f) = d_0 v_y^{rms}(f_0). \tag{7.28}$$

As we know that a K-V distribution has root mean square quantities

$$x^{rms}(f_{KV}) = \frac{a}{2}, \quad y^{rms}(f_{KV}) = \frac{b}{2}, \quad v_x^{rms}(f_{KV}) = \frac{\varepsilon_x}{2a}, \quad v_y^{rms}(f_{KV}) = \frac{\varepsilon_y}{2b}, \tag{7.29}$$

then we can get the scaling parameters as follows

$$a_0 = \frac{a}{2x^{rms}(f_0)}, \quad b_0 = \frac{b}{2y^{rms}(f_0)}, \quad c_0 = \frac{\varepsilon}{2a_0 v_x^{rms}(f_0)}, \quad d_0 = \frac{\varepsilon}{2b_0 v_y^{rms}(f_0)}. \tag{7.30}$$

7.5.4 Parameter Normalization

The scaled distribution function can not be directly used in the numerical simulation. We usually normalize the system, i.e., the governing equation and the initial condition, such that all numerical values are

in a reasonable range that a computer can represent. Assume that (x, v_x) are normalized by (x_0, v_0), then we define the normalization parameters for z, E, N_t and the external electric field $E(x,y) = -k(x,y)$ to be

$$z_0 = \frac{x_0 v_b}{v_0}, \quad E_0 = \frac{m v_0^2}{q x_0} \quad , N_0 = \frac{\varepsilon_0 m v_0^2}{q^2 x_0^2}, \quad k_0 = \frac{\gamma_b m v_0^2}{q x_0^2}. \tag{7.31}$$

7.5.5 Semi-Gaussian Beam

We consider an initial semi-Gaussian beam in an uniform focusing channel. The semi-Gaussian beam is focused by an uniform external electric field such that the self-consistent and the external forces are well-balanced. The magnitude of the external force is estimated by using the concept of matched K-V beam discussed above. Although, the model has been considered by many authors [68, 65, 66], very few of them describes the details of scaling and normalization. We give this detail below.

Given a semi-Gaussian function

$$f_0(x,y,v_x,v_y) = \begin{cases} \frac{1}{2\pi^2}\exp(-(v_x^2+v_y^2)/2), & x^2+y^2 \le 1 \\ 0, & \text{otherwise} \end{cases} \tag{7.32}$$

the root mean square quantities of the semi-Gaussian function is

$$x^{rms}(f_0) = y^{rms}(f_0) = 1/2 \quad v_x^{rms}(f_0) = v_y^{rms}(f_0) = 1. \tag{7.33}$$

Following Section 7.5.3, we can compute the scaling parameters as

$$a_0 = b_0 = a, \quad c_0 = d_0 = \frac{\varepsilon_x}{2a}. \tag{7.34}$$

Then the scaled semi-Gaussian beam reads

$$f(x,y,v_x,v_y) = \begin{cases} \frac{N_t}{2\pi^2 a_0 b_0 c_0 d_0}\exp(-\frac{v_x^2}{2c_0^2} - \frac{v_y^2}{2d_0^2}), & x^2+y^2 \le 1 \\ 0, & \text{otherwise} \end{cases} \tag{7.35}$$

Assuming that the tune depression η is given, the normalized semi-Gaussian beam can be expressed as

$$f_{normalized}(x,y,v_x,v_y) = \begin{cases} \frac{4(1-\eta^2)}{\pi\eta^2}\exp(-(v_x^2+v_y^2)/2), & x^2+y^2 \le 1 \\ 0, & \text{otherwise} \end{cases} \tag{7.36}$$

and the external electric field is

$$E^e(x,y) = -\frac{4}{\eta^2}(x,y). \tag{7.37}$$

This is the initial distribution function and external field we will apply to the normalized Vlasov equation for beam problems

$$\frac{\partial f}{\partial t} + v \cdot \nabla f + (E^s + E^e) \cdot \nabla f = 0, \tag{7.38}$$

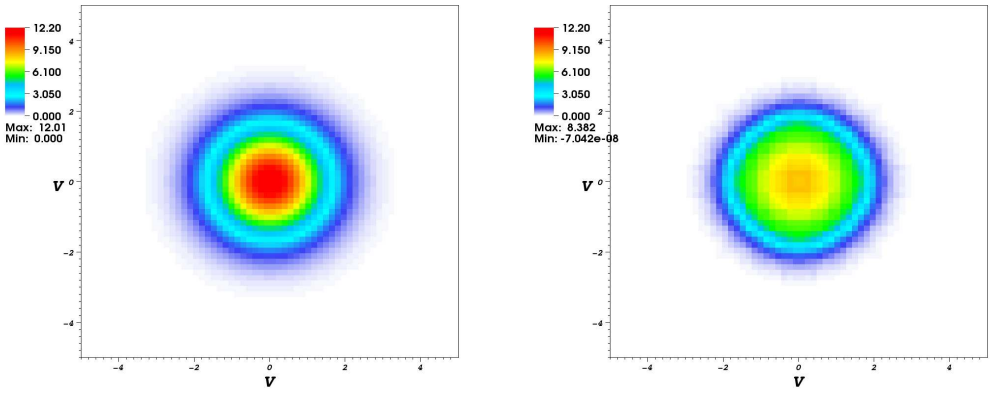
coupled with the Poisson equation

$$\nabla \cdot \nabla \phi = \int_{\mathbb{R}^N} f dv, \quad -\nabla \phi = E^s, \tag{7.39}$$

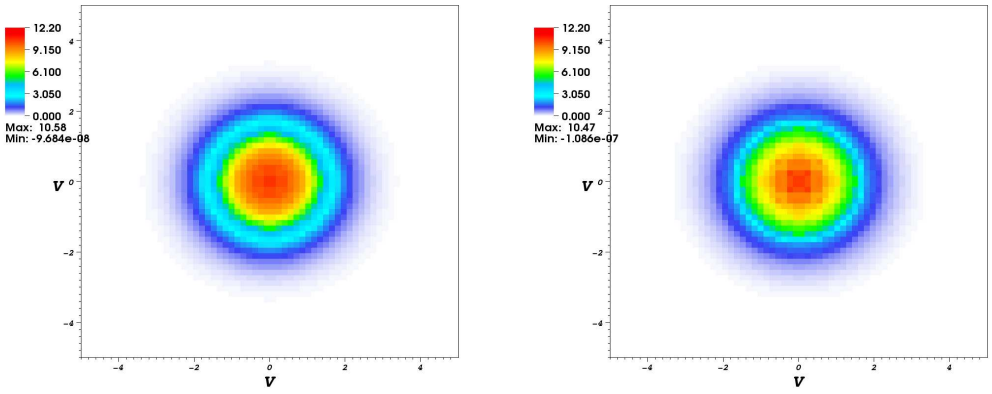
with infinite domain boundary conditions.

The beam is composed of ionized potassium. The physical parameters are the following: current $I = 0.2A$, beam velocity $v_b = 0.63 \times 10^6 m/s$, and the radius of the beam $a = 0.02m$. We have chosen the tune depression $\eta = 1/2$. For the numerical parameters, we choose $v_{\max} = 10$, $(L_x, L_y) = (-10, 10)$. $h_x = |L_x|/64, h_y = |L_y|/64, h_{v_x} = h_{v_y} = v_{\max}/64$. The time step is equal to $\Delta t_p = 0.010585$. We perform the simulation in one period, where $T = 2.71$.

Figure (7.30) to (7.35) show the snapshots of the projection on the v_x, v_y, x, v_x and x, y planes of the distribution function for the simulation without and with remapping. When comparing the projection on the x, v_x plane without remapping (Figure 7.31) and with remapping (Figure 7.34), we observe that remapping reduces the noise significantly.

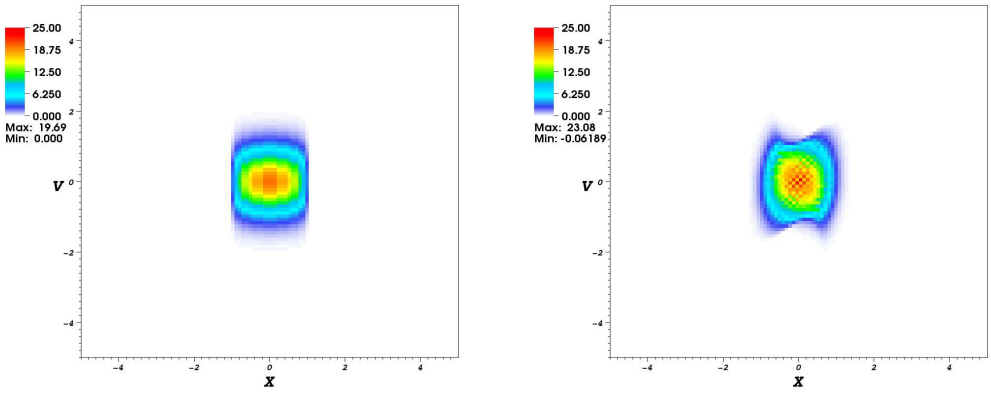


(a) $t = 0$ (b) $t = T/4$



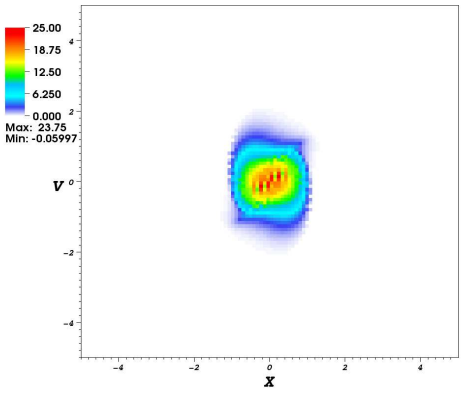
(c) $t = T/2$ (d) $t = T$

Figure 7.30: Time evolution of $v_x - v_y$ projection of the distribution function without remapping (a) t=0 (b)t=T/4 (c) t=T/2 (d) t=T.

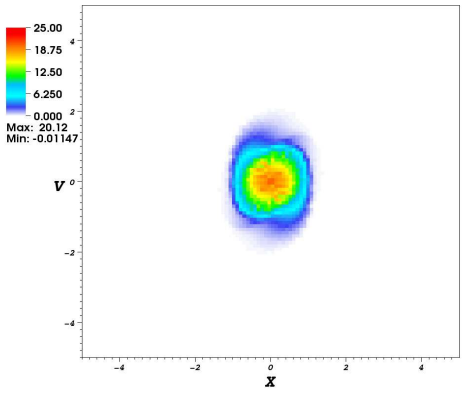


(a) $t = 0$

(b) $t = T/4$

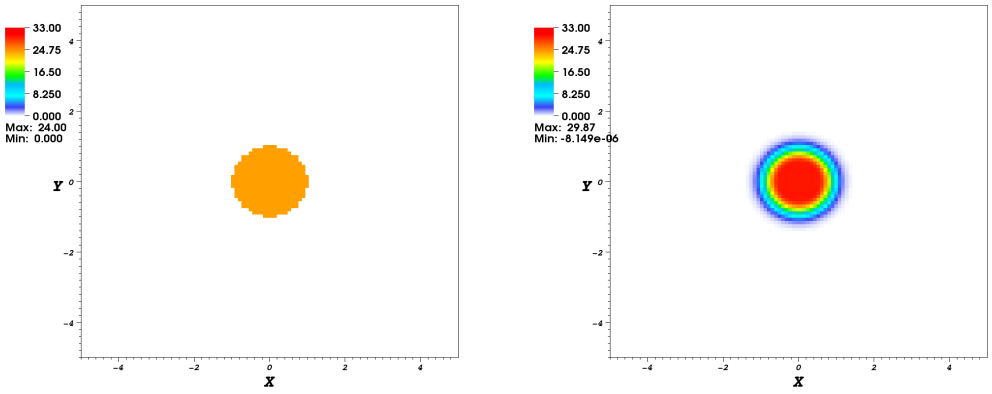


(c) $t = T/2$



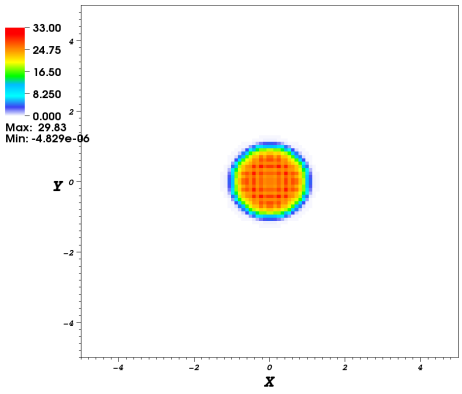
(d) $t = T$

Figure 7.31: Time evolution of $x - v_x$ projection of the distribution function without remapping (a) t=0 (b)t=T/4 (c) t=T/2 (d) t=T.

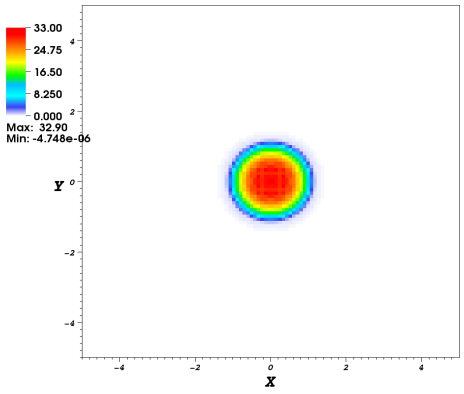


(a) $t = 0$

(b) $t = T/4$

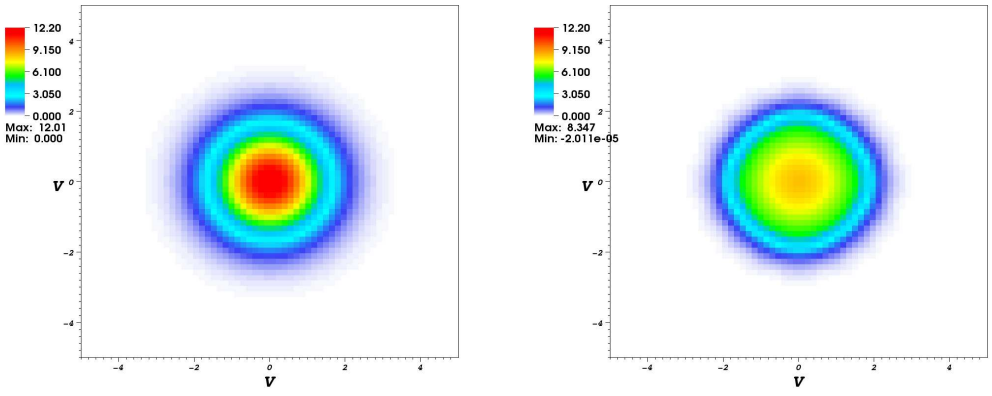


(c) $t = T/2$

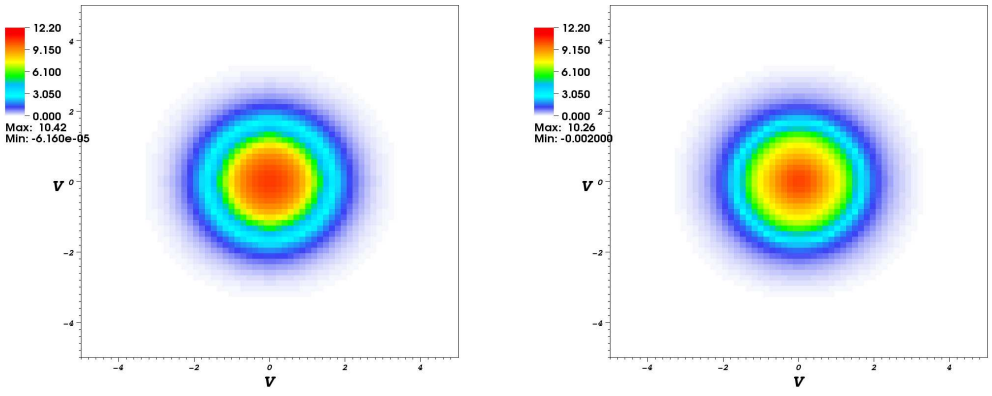


(d) $t = T$

Figure 7.32: Time evolution of $x-y$ projection of the distribution function with remapping (a) t=0 (b)t=T/4 (c) t=T/2 (d) t=T.

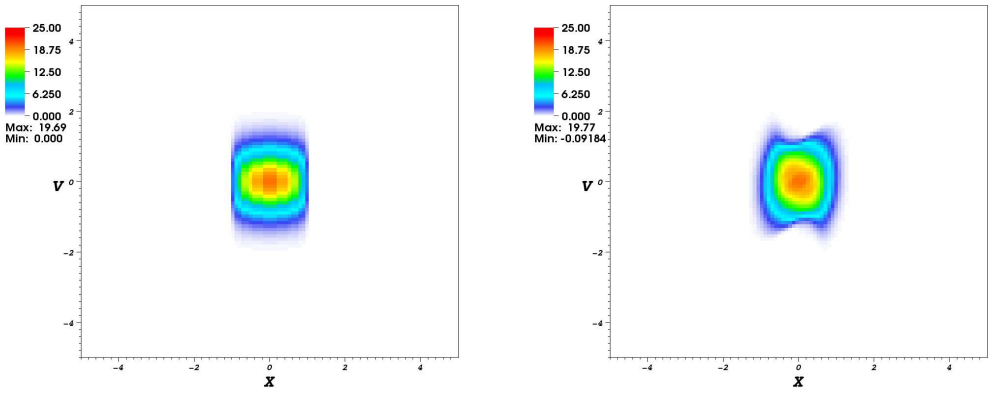


(a) $t = 0$ (b) $t = T/4$



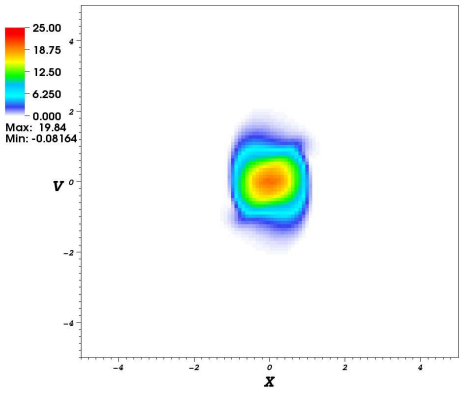
(c) $t = T/2$ (d) $t = T$

Figure 7.33: Time evolution of $v_x - v_y$ projection of the distribution function with remapping (a) t=0 (b)t=T/4 (c) t=T/2 (d) t=T. Compare with Figure 7.30.

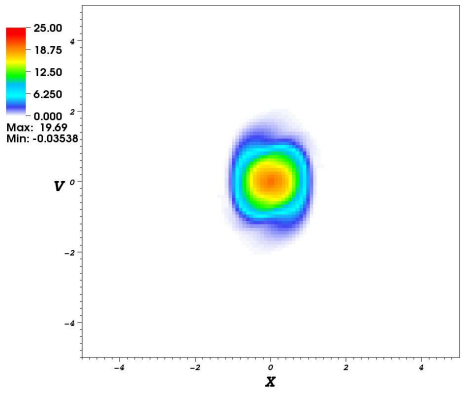


(a) $t = 0$

(b) $t = T/4$

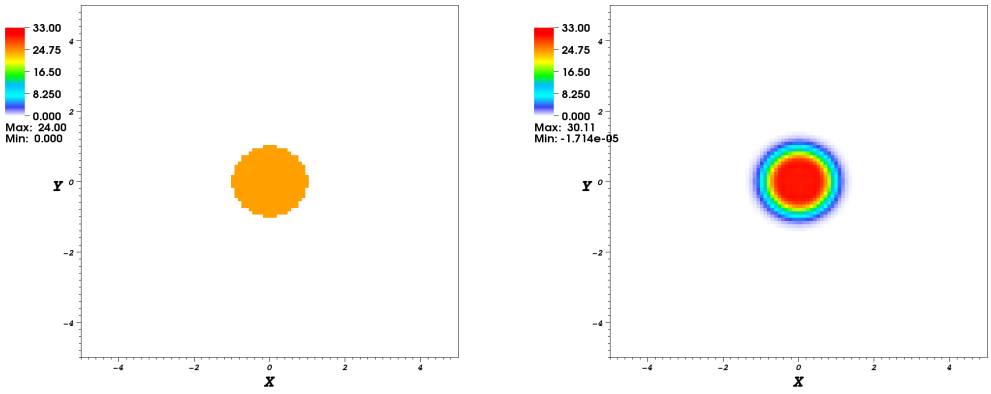


(c) $t = T/2$



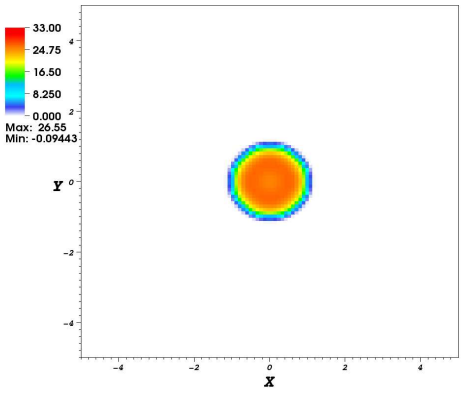
(d) $t = T$

Figure 7.34: Time evolution of $x - v_x$ projection of the distribution function with remapping (a) t=0 (b)t=T/4 (c) t=T/2 (d) t=T. Compare with Figure 7.31.

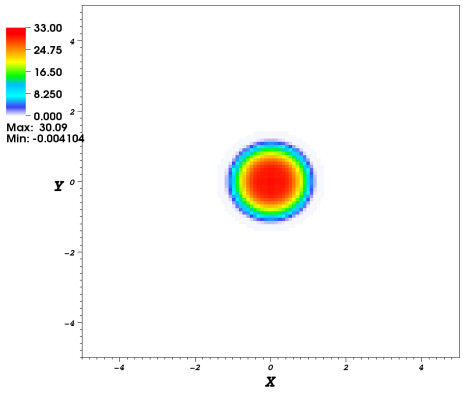


(a) $t = 0$

(b) $t = T/4$



(c) $t = T/2$



(d) $t = T$

Figure 7.35: Time evolution of $x - y$ projection of the distribution function with remapping (a) t=0 (b)t=T/4 (c) t=T/2 (d) t=T. Compare with Figure 7.32.

Chapter 8

Software Implementation

We implement the algorithm using Chombo framework, a C++ and FORTRAN library for solving partial differential equations on block-structured locally refined Cartesian grids with finite volume discretization. It has been developed by the Applied Numerical Algorithm Group at Lawrence Berkeley National Laboratory.

For implementing the remapped PIC algorithm in Chombo, we make use of the existing AMR elliptic solver and the particle tools. However, some new programming models are introduced. In this chapter, we will first briefly describe the existing AMR elliptic solver and the particle tools in Chombo. Then we will discuss the new programming model to implement our algorithm. This includes a AMRPIC class, a AMRRemap class and a simplified Fokker-Planck solver class.

8.1 Chombo Package

8.1.1 AMR Elliptic Solver

The AMR elliptic solvers in Chombo are provided by a set of classes. The `AMRNodeSolve` class executes geometric multigrid method over an AMR hierarchy of node-centered grids. It manages the hierarchy of levels and provides an interface to the elliptic solver. The algorithm of the AMR multigrid method

is given in Section 5.1.3. The user needs to pass a `NodeLevelOp` class and a `LinearSolve` class to the `AMRNodeSolve`. `NodeLevelOp` provides elliptic operator on a level assuming that coarse-fine boundary conditions may be required. For example, `NodePoissonOp` is a derived class from `NodeLevelOp` that provides a Laplacian operator. `LinearSolve` represents a solver class on a uniform level. In `AMRNodeSolve`, it provides the bottom solver on the coarsest level. Usually, `NodeBiCGStabSmoother` class, a derived class from `LinearSolve`, is provided to `AMRNodeSolve` as a node-centered BiCGStab implementation of the bottom solver.

Chombo provides both cell-centered and node-centered version of Poisson solver. We use the node-centered version as described above for the solution of the Poisson equation. The corresponding cell-centered version of the classes are: `AMRMultiGrid`, `AMRLevelOp`, `AMRPoissonOp` and `BiCGStabSolver`. Our simplified Fokker-Planck solver is built on the top of those cell-centered classes for elliptic solver. We will discuss it in Section 8.2.3.

8.1.2 Particle Tools

In Chombo, the class `BinItem` is the basic particle class from which all other particles can be defined. It contains a position, along with functions to set and modify the position. We develop a `Charge` class, a derived class from `BinItem`. `Charge` contains other particle properties, such as field, velocity and charge value, and their associated functions.

`BinFab` is the data holder for the `BinItem` as a form of `List` on each grid cell on a given box. For example, the particles on a box is given as `BinFab<List<Particle> >`. `BinFab` is derived from `BaseFab`. In Chombo, a `BaseFab<T>` class is the data holder for data type `<T>` defined on a each grid cell of the box given by `BaseFab<T>::box()`. Since each box is assigned to a processor, the data on different boxes map naturally onto distributed memory.

8.2 The Algorithm Implementation

The implementation of a multi-dimensional PIC method with adaptive phase space remapping and a grid-based collisional model based on the existing Chombo package will involve four main additions to the current codes:

1. The PIC algorithm in physical space. This includes interpolating the particle charge on a physical space grid, solving the Poisson equation, interpolating the field from grid points back to particle locations and updating the particle properties with numerical integration, such as Runge Kutta. This process is implemented in class `AMRPIC`.

2. The transformation between physical space `Charge` and phase space `PhaseCharge` and their data holder. This process uses the MultiDimPhase library in Chombo. We extend MultiDimPhase to include the capability of dimension reduction for phase space `PhaseCharge` and `BinFab<PhaseCharge>` and dimension increasing for physical space `Charge` and `BinFab<Charge>` .

3. Charge remapping on a hierarchy of cell-centered grids in phase space. This is implemented in class `ChargeRemapFullMR` and `ChargeRemapVelMR`. `ChargeRemapFullMR` has uniform refinement ratio in all dimensions. `ChargeRemapVelMR` has non-uniform refinement ratio with refinement in velocity directions only.

4. Solving the collisional model in velocity space. The collisional model and its grid-based discretization are discussed in Chapter 4. We develop a Fokker-Planck operator, `FokkerPlanckOp` class, based on the current Chombo variable coefficient Poisson operator, `VCAMRPoissonOp2` class.

8.2.1 Class: AMRPIC

The `AMRPIC` class serves two main purposes. First, given a set of particle charges, it computes the self-consistent electric field created by the particle distribution. Second, it updates the particle properties, such as position and velocity, using numerical integration, e.g. 2^{nd} order and 4^{th} order Runge Kutta method.

In addition, we provide a simplified version of James' algorithm (see Section 5.1.4) to solve the Poisson equation with infinite boundary conditions in 2D.

8.2.2 Class: ChargeRemap

`ChargeRemapBase` is the base class for charge remapping on a hierarchy of grids. It provides the basic mechanics of building a hierarchy of grids in phase space, loading a set of disturbed charges to the hierarchy of grids for remapping, and unloading a set of regularized charges after remapping. The positive and conservative remapping algorithm is implemented in the derived classes `ChargeRemapFullMR` and `ChargeRemapVelMR`.

8.2.3 Class: FokkerPlanckOp

For implementing the simplified Fokker-Planck solver on a hierarchy of locally refined velocity space with the algorithm we discussed early (see Chapter 4), we make fully use of the current Chombo elliptic solver and the second-order L_0 stable implicit time integrator, `AMRMultiGrid`, `VCAMRPoissonOp2` and `AMRTGA`. The `AMRTGA` class implements the TGA algorithm (Chapter 4) to advance the data on all levels to the next time step with a second-order L_0 stable implicit method. `AMRMultiGrid` is passed as a parameter to `AMRTGA` to solve the linear system at the second and third step of TGA algorithm. We develop a Fokker-Planck operator, `FokkerPlanckOp`, through deriving existing `VCAMRPoissonOp2` class. The `FokkerPlanckOp` class is then passed to `AMRMultiGrid` that provides a elliptic operator for the multigrid algorithm.

The `FokkerPlanckOp` along with `AMRMultigrid` and `AMRTG` solves the variable coefficients Fokker-Planck system with second-order accuracy using finite difference discretization:

$$\underbrace{[\kappa a(v)I + \gamma\nabla\cdot(b(v)\nabla(c(v)) + d(v)c(v))]}_{A}\phi = \rho \tag{8.1}$$

$a(v)$ and $c(v)$ are cell-centered variable coefficients. $b(v)$ and $d(v)$ are edge-centered variable coefficients. In a simplified Fokker Planck equation, $a(v) = 1$; $b(v) = \beta$; $c(v) = \alpha$ and $d(v) = v_d$. κ and γ are scalar values

determined by TGA steps. For example, in the second step of TGA, $\kappa = 1$ and $\gamma = -\Delta t \mu_2$.

8.3 Parallelization

In Chombo, data are defined on a union of rectangles with each rectangle mapped to a single processor. The communication of data between different processors are through ghost cells.

The current parallel implementation of the algorithm is straightforward based on physical space domain decomposition. Physical space is decomposed into M patches. Given a parallel machine with N processors, each patch is assigned to a processor cyclically. Particles are assigned to each processor according to their physical space positions. The current implementation is not a linear scaled algorithm because of decomposition in physical space only. Figure (8.1) shows a example of processor assignment based on physical space domain decomposition. Each cell in physical space along with its velocity space grid, is assigned to one processor. The potential issue of this implementation is that as the problem size increases, since the number of processors is scaled in proportion to the problem size in physical space, the computational time and memory will increase in proportion to the problem size in velocity space. In the worst case, the processor will be out of memory. An alternative implementation based on phase space domain decomposition will be an important extension of our current research.

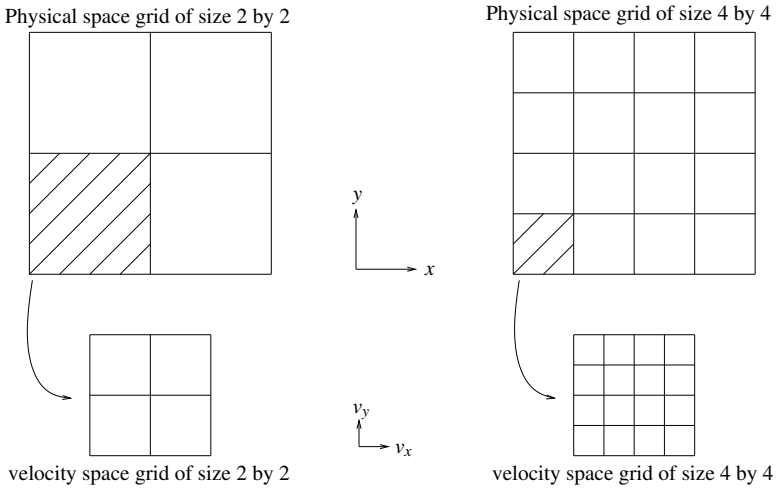


Figure 8.1: Physical space domain decomposition. The shaded regions are an example of the domain assigned to one processor (one cell in the example). Each physical space cell includes a full velocity space grid. As we increase the resolution by a factor of 2 in all dimensions in phase space (left to right plot), even the number of processors is scaled as a factor of 4, the computational load in one processor is still increasing by a factor of 4 due to velocity space grid.

Chapter 9

Conclusions

9.1 Summary

In this thesis, an accurate and efficient PIC method has been developed. We provide a conservative, high-order and positive remapping scheme for reducing numerical noise of the classical PIC method. Meanwhile, the remapped PIC method has been extended to include mesh refinement. The initial experiments on a set of classical plasma problems are very promising. The new method significantly reduces the numerical noise and results in a more consistent second-order method than the standard PIC method.

At the same time, we have developed a multi-dimensional, parallel Vlasov-Poisson solver based on Chombo framework. The parallelization is based on domain decomposition of the Poisson grid. For modeling beam problems, an implementation of the infinite boundary conditions for the Poisson solver based on James' algorithm is included.

Finally, we have provided an error analysis of the PIC method based on Cottet and Raviart's work [1] in particle methods for Vlasov-Poisson system. We extend their approach to the PIC method for the one-dimensional Vlasov-Poisson equation. The error analysis provides guidance for the design of the PIC method and for the choice of optimal parameters.

As we develop the method, we found that our method is similar to the forward semi-Lagrangian

method of Crouseilles [9]. Both methods advance the position of a particle along the characteristics forwards in time and periodically reconstruct the distribution function. But there are three major differences. First, we use a high-order modified B-spline function for remapping instead of standard B-splines. This avoids the solving of a matrix system to determine the new particle weights. Second, we provide a local mass redistribution algorithm to preserve the positivity of the high-order interpolation function. Third, we introduce mesh refinement for remapping to reduce the total number of small strength particles. The developed algorithm is second order accurate in the electric field error.

9.2 Future Work

This research is an initial investigation of remapping algorithm for the PIC method. There remain two areas. First is adaptivity: the adaptivity of the remapping frequency, and adaptivity in creating a hierarchy of locally refined grids. Currently, we apply remapping periodically with a constant frequency. Ideally, we would like the remapping to be applied as needed. That is, when the exponential factor grows over some threshold. Our current algorithm creates the finer level grid on a fixed sub-domain since the problems we have studied are well-known. For more general problems, we hope that finer grids are created dynamically by selecting some refinement criterion. For example, each particle on the grid carries approximately the same weight. Second is the treatment of complex geometry. In the current implementation, we only consider rectangular geometry (2D). It is worth extending the algorithm for complex geometry on real physical experiments. An attractive option is to use Cartesian grid embedded boundary method by Johansen [55].

We have demonstrated the algorithm on a set of classical plasma problems for the Vlasov-Poisson equation. An important extension of the current research is to apply it on magnetized plasmas or laser-plasma interaction problems. This requires us to solve Maxwell's equation instead of Poisson's equation. Chilton's ongoing thesis work on "A fourth-order adaptive mesh refinement solver for Maxwell's equations"is the immediate extension of this research.

Finally, we would like to address the issue of the scalability of the developed solver. As mentioned

in Chapter 8, domain decomposition based on phase space will be a great option to achieve scalability of the algorithm. Particles are assigned to a processor according to their phase space. The disadvantage of this implementation is that charges on the same physical space node position might be scattered to different processors. Since the Laplacian operator is linear, we can choose to solve the Poisson equation separately on different processors. The total fields are then obtained by gathering the solution of the Poisson equation from different processors.

Bibliography

1. G.-H. Cottet and P. A. Raviart. Particle methods for one-dimensional Vlasov-Poisson equations. *SIAM J. Numer. Anal.*, 21:52–76, 1984.

2. O. Buneman. Dissipation of currents in ionizzed media. *Phys. Rev.*, 115:503–517, 1959.

3. J. M. Dawson. One-dimensional plasma model. *Phys. Fluids*, 5:445–459, 1963.

4. G. Knorr. *Naturfosch*, 18a:1304, 1963.

5. T. Armstrong and D. Montgomery. Numerical study of weakly unstable electron plasma oscillation. *Phys. Fluids*, 12:2094–2098, 1969.

6. C. Z. Cheng and G. Knorr. The integration of the Vlasov equation in configuration space. *J. Comput. Phys.*, 22:2330–2351, 1976.

7. E. Sonnendrucker, J. Roche, P. Bertrand, and A. Ghizzo. The semi-Lagrangian method for the numerical resolution of Vlasov equations. *J. Comput. Phys.*, 149:201–220, 1998.

8. T. Nakamura and T. Yabe. Cubic interpolated propagation scheme for solving the hyper-dimensional Vlasov-Poisson equation in phase space. *Comput. Phys. Comm.*, 120:122–154, 1999.

9. N. Crouseilles, T. Respaud, and E. Sonnendrucker. A forward semi-lagrangian method for the numerical solution of the vlasov equation. *Comput. Phys. Comm.*, 180:1730–1745, 2009.

10. J. P. Boris and D. L. Book. Flux-corrected transport. i:shasta, a fluid transport algorithm that works. *J. Comput. Phys.*, 11:38–69, 1973.

11. D. L. Book and J. P. Boris. Flux-corrected transport. ii:generalizationsof the method. *J. Comput. Phys.*, 18:248–283, 1975.

12. J. P. Boris and D. L. Book. Flux-corrected transport. iii: Minimal-error fcf algorithms. *J. Comput. Phys.*, 20:397–431, 1976.

13. E. Fijalkow. A numerical solution to the Vlasov equation. *Comput. Phys. Comm.*, 116:1999, 319–328.

14. P. Colella, M. R. Dorr, J. A. F. Hittinger, and D. F. Martin. High-order finite-volumn methods in mapped coordinates. *Submitted to J. Comput. Phys.*, 2010.

15. F. Filbet, E. Sonnendrucker, and P. Bertrand. Conservative numerical schemes for the Vlasov equation. *J. Comput. Phys.*, 172:166–187, 2001.

16. S. I. Zaki, L. R. T. Gardner, and T. J. M. Boyd. A finite lement code for the simulation of one-dimensional Vlasov plasmas. i: Theory. *J. Comput. Phys.*, 79:184–199, 1988.

17. G.-H. Cottet and P. D. Koumoutsakos. *Vortex Methods: Theory and Practice*. Cambridge University Press, Cambridge CB2 2RU, UK, 2000.

18. A. K. Chaniotis, D. Poulikakos, and P. Koumoutsakos. Remeshed smoothed particle hydrodynamics for the simution of viscous and heat conducting flows. *J. Comput. Phys.*, 182:67–90, 2002.

19. F. Filbet and E. Sonnendrucker. Comparison of eulerian vlasov solvers. *Comput. Phys. Comm.*, 150:247–266, 2003.

20. F. Filbet and E. Sonnendrucker. Numerical methods for the vlasov equation. 2011.

21. Leslie Greengard. *The rapid evaluation of potential fields in particle systems*. PhD thesis, Yale University, 1987.

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

23. C. K. Birdsall and A. B. Langdon. *Plasma Physics via Computer Simulation*. Institute of Physics Publishing, Bristol BS16BE, UK, 1991.

24. P. Colella and P. C. Norgaard. Controlling self-force errors at refinement boundaries for $amr - pic$. *J. Comput. Phys.*, 229:947–957, 2010.

25. I. J. Schoenberg. *Cardinal Spline Interpolation*. SIAM, Philadelphia, 1973.

26. J. J. Monaghan. Why particle methods work. *SIAM J. Sci. Stat. Comput.*, 3:422–433, 1982.

27. A. M. Dimits and W. W. Lee. Partially linearized algorithms in gyrokinetic particle simulation. *J. Comput. Phys.*, 107(2):309–323, 1993.

28. G. Hu and J. A. Krommes. Generalized weighting schemes for δf particle-simulation method. *Phys. Plasmas*, 1(4):863–874, 1993.

29. R. E. Denton and M. Kotschenreuther. δf algorithm. *J. Comput. Phys.*, 119(2):283–294, 1995.

30. P. D. Koumoutsakos. Inviscid axisymmetrization of elliptic vortex. *J. Comput. Phys.*, 138:821–857, 1997.

31. J. Denavit. Numerical simulation of plasma with periodic smoothing in phase space. *J. Comput. Phys.*, 9:75–98, 1972.

32. Srinath Vadlamani, Scott E. Parker, Yang Chen, and Charlson Kim. The particle-continuum method: an algorithmic unification of particle-in-cell and continuum methods. *Comput. Phys. Comm.*, 164:209–213, 2004.

33. Y. Chen, S. E. Parker, G. Rewoldt, S. Ku, G. Y. Park, and C-S. Chang. Coarse-graining the eletron distribution in turbulence simulations of tokamak plasmas. *Phys. Plasmas*, 15, 2008.

34. J. T. Beale and A. Majda. High order accurate vortex methods with explicit velocity kernels. *J. Comput. Phys.*, 58:188–208, 1985.

35. J. J. Monaghan. Extrapolating B Splines for interpolation. *J. Comput. Phys.*, 60:253–262, 1985.

36. S. Borve, M. Omang, and J. Trulsen. Regularized smoothed particle hydrodynamics: A new approach to simulating magnetohydrodynamic shocks. *The Astrophysical Journal*, 561:82–93, 2001.

37. S. Borve, M. Omang, and J. Trulsen. Regularized smoothed particle hydrodynamics with improved multi-resolution handling. *J. Comput. Phys.*, 208:345–367, 2005.

38. J. J. Monaghan. Particle methods for hydrodynamics. *Comput. Phys. Rep.*, 3:71–124, 1985.

39. S. T. Zalesak. Fully multidimensional flux-corrected transport algorithms for fluids. *J. Comput. Phys.*, 31:335–362, 1978.

40. I.-L. Chern and P. Colella. A conservative front tracking method for hyperbolic conservation laws. Technical report, Lawrence Livermore National Laboratory, 1987. UCRL-97200.

41. J. Hilditch and P. Colella. A projection method for low Mach number fast chemistry reacting flow. In *Proc. AIAA Aerospace Sciences Meeeting, Reno, NV.*, 1997.

42. C. E. Rathmann and J. Denavit. Simulation of collisional effects in plasmas. *J. Comput. Phys.*, 18:165–187, 1975.

43. E. H. Twizell, A. B. Gumel, and M. A. Arigu. Second-order, l_0-stable methods for the heat equation with time-dependent boundary conditions. *Advances in Comput. Math.*, 6:333–352, 1996.

44. J. L. Vay, P. Colella, J. W. Kwan, P. McCorquodale, D. B. Serafini, A. Friedman, D. P. Grote, G. Westenskow, J.-C. Adam, A. Heron, and I. Haber. Application of adaptive mesh refinement to particle-in-cell simulations of plasmas and beams. *Phys. of Plasmas*, 11:2928–2934, 2004.

45. David B. Serafini, Peter McCorquodale, and Phillip Colella. *Advanced 3D Poisson solvers and particle-in-cell methods for accelerator modeling*, volume 16. Joural of Phys.:Conference Series, 2005.

46. P. McCorquodale, P. Colella, D. Grote, and J-L. Vay. A node-centered local refinement algorithm for Poisson's equation in complex geometries. *J. Comput. Phys.*, 201:34–60, 2004.

47. Ann S. Almgren, John B.Bell, Phillip Colella, Louis H. Howell, and Michael L. Welcome. A conservative adpative projection method for the variable density incompressible Navier-Stokes equations. *J. Comput. Phys.*, 142:1–46, 1998.

48. R. A. James. The solution of Poisson's equation for isolated source distributions. *J. Comput. Phys.*, 25:71–93, 1977.

49. M. J. Berger and P. Collela. Local adaptive mesh refinement for shock hydrodynamics. *J. Comput. Phys.*, 82(1):64–84, 1989.

50. W. L. Briggs, V. E. Henson, and S. F. McCormick. *A Multigrid Tutorial*. SIAM, Philadelphia, 2000.

51. D. Martin and K. Cartwright. Solving Poisson's equation using adaptive mesh refinement. *U.C. Berkeley Electronics Research Laboratory report No. UCB/ERL M96/66*, 1996.

52. P. McCorquodale, P. Colella, G. T. Balls, and S. B. Baden. A scalable parallel Poisson solver in three dimensions with infinite-domain boundary conditions. *In* 7^{th} *International Workshop on High Performance Scientific and Engineering Computing*, pages 814–822, 2005.

53. P. McCorquodale, P. Colella, G. T. Balls, and S. B. Baden. A local corrections algorithm for solving Poisson's equation in three dimensions. *Comm. App. Math. and Comp. Sci.*, 2:57–81, 2007.

54. Dan Martin. *An adaptive cell-centered projection method for the incompressible Euler equations*. PhD thesis, Uiversity of California at Berkeley, 1998.

55. H. Johansen. *Cartesian grid embeded boundary finite difference method for elliptic and parabolic partial differential equations on irregular domains*. PhD thesis, University of California at Berkeley, 1997.

56. O. Hald. The convergence of vortex methods, II. *SIAM J. Numer. Anal.*, 16:726–755, 1979.

57. J. T. Beale and A. Majda. Vortex methods I: convergence in three dimensions. *Math. Comp.*, 39:1–27, 1982.

58. J. T. Beale and A. Majda. Vortex methods II: high order accuracy in two and three dimensions. *Math. Comp.*, 39:29–52, 1982.

59. C. A. Anderson and C. Greengard. On vortex methods. *SIAM J. Numer. Anal.*, 22:413–440, 1985.

60. H. Dym and H. P. McKean. *Fourier Series and Integrals*. Academic Press, New York, 1972.

61. H. D. Victory, Jr. and E. J. Allen. The convergence theory of particle-in-cell methods for multidimensional Vlasov-Poisson systems. *SIAM J. Numer. Anal.*, 28:1207–1241, 1991.

62. T. Utsumi, T. Kunugi, and J. Koga. A numerical method for solving the one-dimensional Vlasov-Poisson equation in phase space. *Comput. Phys. Comm.*, 108:159–179, 1998.

63. N. Besse and E. Sonnendrucker. Semi-Lagrangian schemes for the Vlasov equation on an unstructured mesh of phase space. *J. Comput. Phys.*, 191:341–376, 2003.

64. Jose Canosa, Jeno Gazdag, and J.E. Fromm. The recurrence of the inital state in the numerical solution of the Vlasov equation. *J. Comput. Phys.*, 15:34–45, 1974.

65. N. Crouseilles, M. Gutnic, G. Latu, and E. Sonnendrucker. Comparision of two eulerian solvers for four-dimensional *v*lasov equation: Part ii. *Communications in nonlinear science and numerical simulation*, 13:94–99, 2008.

66. N. Crouseilles, G. Latu, and E. Sonnendrucker. A parallel vlasov solver based on local cubic spline interpolation on patches. *J. Comput. Phys.*, 228:1429–1446, 2009.

67. F. Filbet and E. Sonnendrucker. Modeling and numerical simulation of space charged dominated beams in the paraxial approximation. *Math Mod. Meth. Appl. Sci.*, 16:763–791, 2006.

68. E. Sonnendrucker, F. Filbet, A. Friedman, E. Oudet, and J.-L. Vay. Vlasov simulations of beams with a moving grid. *Comput. Phys. Comm.*, 164:390–395, 2004.