Adaptive Remapping for High-Order Particle-in-Cell Methods

by

Colin Wahl

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Applied Science and Technology

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Phillip Colella, Chair
Professor Panayiotis Papadopoulos
Professor Katherine Yelick

Summer 2022

Adaptive Remapping for High-Order Particle-in-Cell Methods

Abstract

Adaptive Remapping for High-Order Particle-in-Cell Methods

by

Colin Wahl

Doctor of Philosophy in Applied Science and Technology

University of California, Berkeley

Professor Phillip Colella, Chair

We present an analytic framework for understanding the errors associated with deposition in particle-in-cell methods applied to kinetics problems in one configuration space dimension. We begin by considering a one-dimensional advection model problem. Results of the analysis includes a $O(1)$ error, also referred to as a barrier error, that is dependent on the deformation gradient of the particle locations. The presence of this barrier is confirmed by numerical experiments. The techniques developed in the analysis of the one-dimensional advection model problem are used to analyze a deposition model problem in the context of kinetics. This analysis shows an error that is similar to the error that arose in the one-dimensional advection model problem but is dependent only on the configuration space deformation gradient of the particle locations. A set of ODEs are derived for the Vlasov-Poisson system that can be used to compute the configuration space deformation gradient without explicit knowledge of particle neighbors. This allows for computation of the $O(1)$ error at each time-step. Remapping is then introduced to control this error. Numerical experiments confirm that remapping when the $O(1)$ error is comparable to the other errors inherent in the simulation reduces the "particle noise" phenomena that plagues many PIC methods and maintains the high-order convergence. With a fourth-order kernel this leads to an overall reduction in the number of remaps compared to a forward semi-Lagrangian method that remaps every few time steps. We also implement adaptive-in-time-and-space (local) remapping where only a portion of the phase-space is remapped and the $O(1)$ error along with the value of the distribution serve as an implicit boundary between the remapped and non-remapped regions. The adaptive-in-time and locally remapped PIC methods are then applied to the Dory-Guest-Harris instability which, unlike the other test problems considered, has two velocity space dimensions.

To my parents.

Contents

List of Figures

List of Tables

Acknowledgments

To begin, I would like to thank Professor Phillip Colella, my Ph.D. advisor. I still remember my first meeting with Phil and how excited and passionate he was about this project. That passion was infectious and I consider myself extremely privileged to have had the opportunity to work with Phil. I always enjoyed our meetings and appreciated his valuable insights into any roadblocks or rabbit holes that were slowing my progress. Without Phil's unwavering support, patience, and guidance this research would not have been possible.

I would also like to thank the entire Applied Numerical Algorithm Group at Lawrence Berkeley National Lab. I have enjoyed conversations on both numerical methods and non-work topics with so many in the group. A special thanks goes to my office mates over the years: Chris Chaplin, Boris Lo, Chris Gebhart, and Marissa Ramirez de Chanlatte. I picked up innumerable lessons and tidbits of wisdom from each of you and could not imagine better people to share a coffee break with. Additionally, my graduate school experience would not have been the same without Luke Corcos. Thank you for the many conversations about applied mathematics, research, and life while enjoying whatever sport was on TV.

I view this dissertation as the culmination of a decade of education. I would not have began on this journey if it were not for my high school Physics teacher and mentor Mitch Disch. His classes showed me how fulfilling it could be to apply mathematics to the world around me. Professor Thad Walker and the Atom Trainers, in particular Ibrahim Sulai and Zachary DeLand, introduced me to academic research; they showed me both how frustrating the process could be as well as how rewarding it is when things come together. Finally, Professor Saverio Spagnolie and Professor Jean-Luc Thiffeault nurtured my interest in applied mathematics and introduced me to many of the topics I went on to pursue in graduate school. Much of the researcher that I have become is thanks to their mentorship and guidance.

Early on in my graduate school career an older, wiser student told me to treat the process like a marathon; don't sprint out the gate but figure out how to make the process sustainable. For me, the outdoors have been how I unplug and give balance to my life. My adventure partners have helped me step away and come back to my research refreshed. This is especially true of Ben Werbner and Luis Barroso-Luque. They have been the best adventure partners and friends that I could ask for over these last 6 years. I'd also like to thank Philipp Gritsch and Chris Miller for the conversations and life advice over the many trips I took with each of them. To anyone else I shared a rope, crash pad, skin track, trail, or conversation in a beautiful place with, thank you for your friendship.

To my family, thank you for all the love and support that each of you have given me. I would not be the person that I am today without each of you. My parents, Dean and Julie, have shaped me in innumerable ways. The curiosity, tenacity, and continuous learning that they both have demonstrated throughout their own careers has been particularly influential during mine. Memories from my childhood and the stories they shared about when they were my age always gave me the confidence that I could accomplish anything I set out to do. My sister, Leah, has always cared deeply for the ones she loves and simultaneously balanced many different interests, friendships, and careers. Her example inspires and motivates me to be the best version of myself. Finally, I'd like to acknowledge my grandparents, John and Dotty, and my great-great Aunt, Sister Stephanie. I would not be who I am without the example they've set forth and their encouragement to pursue my passions.

Anyone who knows me knows that there is one person left who has been by my side for the entirety of this journey. I am so unbelievably lucky to call Meghan my fiancée and to have had her as my partner these last 10 years. She has supported me unconditionally and believed in me when I did not believe in myself. She has made the bad days tolerable and the good days so much sweeter. I am amazed at how much we have both grown individually and together. I am so proud of what we have accomplished so far and am excited to see what we do together in the many years to come.

This research was supported at the Lawrence Berkeley National Laboratory by the Office of Advanced Scientific Computing Research of the U.S. Department of Energy (DOE) under Contract Number DE-AC02-05CH11231.

Chapter 1

Introduction

1.1 Lagrangian Particle Methods

Advection equations are equations of the form

$$\frac{\partial f}{\partial t} + \nabla \cdot (\boldsymbol{u} f) = R \tag{1.1}$$

$$f = f(\boldsymbol{x}, t), \boldsymbol{x} \in \mathbb{R}^N. \tag{1.2}$$

Typically, $\boldsymbol{u}$ and R are related to f through

$$\boldsymbol{u} = L_u(f), R = L_R(f) \tag{1.3}$$

where L_u and L_R are integro-differential operators. To derive the equivalent Lagrangian equations, we define the Lagrangian trajectories

$$\boldsymbol{X}(\boldsymbol{\alpha}, t) \in \mathbb{R}^N \tag{1.4}$$

$$\boldsymbol{X}(\boldsymbol{\alpha}, 0) = \boldsymbol{\alpha} \tag{1.5}$$

$$\frac{d\boldsymbol{X}(\boldsymbol{\alpha}, t)}{dt} = \boldsymbol{u}(\boldsymbol{X}(\boldsymbol{\alpha}, t)). \tag{1.6}$$

It then follows that the evolution of f along the Lagrangian trajectories is written as

$$\frac{df(\boldsymbol{X}(\boldsymbol{\alpha}, t), t)}{dt} = \frac{\partial f}{\partial t} + \frac{d\boldsymbol{X}}{dt} \cdot \nabla f \tag{1.7}$$

$$= \frac{\partial f}{\partial t} + \boldsymbol{u}(\boldsymbol{X}(\boldsymbol{\alpha}, t)) \cdot \nabla f \tag{1.8}$$

$$= -f \ \nabla \cdot \boldsymbol{u}(\boldsymbol{X}(\boldsymbol{\alpha}, t)) + R. \tag{1.9}$$

Lagrangian particle methods are a class of numerical methods that solve advection dominated problems by discretizing the Lagrangian form of the equations. The equations for this discretization are:

$$f(\boldsymbol{x},t) \rightarrow \{f_k, \boldsymbol{X}_k\} \tag{1.10}$$

$$\frac{d\boldsymbol{X}_k}{dt} = \boldsymbol{u}(\boldsymbol{X}_k),\ \boldsymbol{X}_k(t=0) = \boldsymbol{\alpha}_k \tag{1.11}$$

$$\frac{df_k}{dt} = -f_k\ \nabla \cdot \boldsymbol{u}(\boldsymbol{X}_k) + R_k \tag{1.12}$$

where k is a particle index and now the distribution is represented by a discrete set of particles with strength f_k located at $\boldsymbol{X}_k$ rather than a continuous representation. Implicit in the particle representation of f is some shape function, ζ. This shape function can be used to approximate the continuous distribution by summing over all of the particles,

$$f(\boldsymbol{x},t) \approx \sum_k f_k(t)\zeta\left(\boldsymbol{x} - \boldsymbol{X}_k(t)\right). \tag{1.13}$$

Two separate classes of Lagrangian particle methods have evolved that are distinguishable by how they compute $L_u(f) = \boldsymbol{u}$ and $L_R(f) = R$. The first class solves for $\boldsymbol{u}$ and R by direct evaluation of (possibly regularized) convolutions. An example of a method that falls into this class is vortex methods which compute the velocity field at each particle by applying the Biot-Savart law and using fast multipole methods to compute the convolutions.

In this thesis we will focus on the second class of Lagrangian particle methods, particle-in-cell (PIC) methods. Particle-in-cell methods are different from the first class of methods in that they compute $\boldsymbol{u}$ and R by solving associated PDEs on rectangular grids. This is done by: interpolating from the particle locations to a rectangular grid using (1.13), computing $L_u(f) = \boldsymbol{u}$ and $L_R(f) = R$ on the rectangular grid, and then interpolating back to the particles using (1.13) to obtain $\boldsymbol{u}_k$ and R_k.

The particle-in-cell method was first developed in the 1950s to solve problems in fluid mechanics [27]. A review of the early applications of PIC methods to fluids is given in [31]. Particle-in-cell methods were further explored for solving systems that arise in fluid mechanics in [8] where the fluid-implicit particle (FLIP) method was first developed and compared to the method developed in [46] for flow over a step. FLIP was extended to continuum problems in solid mechanics with the material point method (MPM) which was first developed in [40].

During the late 50s and early 60s PIC was adopted by the plasma physics community to solve kinetics problems [10, 22]. Early use of PIC methods to solve the electrostatic Vlasov equation yielded insight into complex non-equilibrium phenomena that had been difficult to understand using theoretical or experimental techniques. Discussion of early work can be found in the review [23] and the books [32, 7]. PIC methods continue to be heavily utilized in plasma physics and cosmology where Vlasov-type PDEs are frequently encountered. A characteristic feature of PIC methods for Vlasov-type PDEs is that the distribution is a function of phase-space, $f = f(\boldsymbol{x}, \boldsymbol{v}, t)$, but the velocity is only dependent on the configuration space, $\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{x})$. This means that for a problem with six phase-space dimensions, three configuration space and three velocity dimensions, the rectangular grid where $\boldsymbol{u}$ is solved is three dimensions.

PIC has also been applied to the velocity-vorticity formulation of Euler's equations. This was first done by [14] where the vortex-in-cell (VIC) method was developed. VIC methods have been successfully applied to problems such as a vortex ring impinging on a cylinder in [16]. The advantages of VIC over vortex methods in this context is the ability to utilize fast Poisson solvers developed for cartesian grids. The method of local corrections, introduced in [4], is similar to VIC in that a Poisson solver is used to solve for the field on a grid; however, local interactions are computed using convolutions, similar to vortex methods. The method was expanded to three dimensional problems and applied adaptive mesh refinement to the field solve in [2].

1.2 Numerical Analysis of Lagrangian Particle Methods

The existing literature on the convergence theory for Lagrangian particle methods relies heavily on techniques developed to study the convergence of vortex methods. The convergence theory for vortex methods was developed in the late 70s and 80s in [30, 6, 3]. The error bound depends on the interpolation kernel and initial discretization of the distribution on a rectangular grid. This was determined by pulling back to the Lagrangian coordinates - an idea we will utilize. The convergence theory developed for vortex methods was extended to the electrostatic Vlasov system by [39, 41].

All of these convergence theories ignore the errors associated with interpolating from a deformed set of particles to a rectangular grid to solve for the fields. Throughout PIC literature there are references to issues arising from this but, to our knowledge, no formal error analysis exists. Often the errors that arise and the associated degradation of accuracy is said to be caused by "particle noise". One way of managing these errors is by remapping the distribution every few time-steps. Remapping is the interpolation of the distribution back to a rectangular grid where the Largrangian particles then begin for the next time-step. Controlling the regularity of the distribution of particles via remapping has a long history in both vortex methods and PIC methods. The form of remapping we use was introduced in [33]. It has been applied to achieve second-order accuracy in the context of kinetics in [45, 44]. This work was extended to high-order in [37] and a dark matter problem with singular initial data in [38]. Remapping has also been used to develop a high-order FLIP and MPM method in [26].

1.3 Eulerian vs Lagrangian Methods

The alternative to Lagrangian particle methods are Eulerian grid methods. Eulerian grid methods for advection dominated problems include finite difference methods, finite volume methods, and finite element methods. Grid methods work very well but there are situations in which they may need to be avoided:

1. The dimensionality of the problem is high. Typically, for kinetics the dimensionality is greater than 4. Representing a high dimensional space with unions of rectangular patches is inefficient at these dimensions.

2. Difficulty with representing multiple materials. Lagrangian particle methods are more apt at correctly representing material interfaces as they are naturally adaptive and implicitly track these boundaries. In kinetics this natural adaptivity is especially advantageous since large regions of the distribution are near-zero. In contrast, Eulerian methods for moving boundaries require an auxilary calculation of a representation of the moving boundary plus specialized discretizations on either side of the boundary [15].

3. Eulerian grid methods with an explicit time-stepping algorithm require that a Courant-Friedrichs-Lewy (CFL) condition like

   $$\frac{||\boldsymbol{u}||_\infty \Delta t}{\Delta x} < 1 \tag{1.14}$$

   be satisfied for stability. PIC does not have such a stability constraint but rather requires

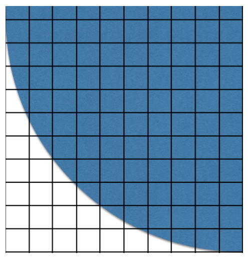
   $$\Delta t ||\nabla \boldsymbol{u}||_\infty < C \tag{1.15}$$

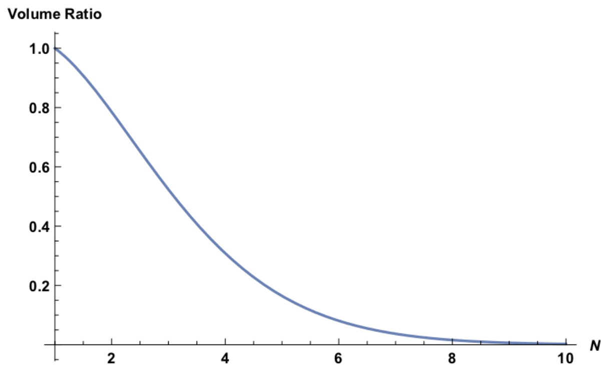
   where C depends on the integration scheme. For problems in which the velocity is a small perturbation of streaming, this condition leads to a much larger Δt than the classical CFL condition for Eulerian grid methods.

For Vlasov-type PDEs, all of these issues are present but issues 2 and 3 are unique in that they compound upon the first issue listed.

Curse of Dimensionality

The first issue listed is often called the curse of dimensionality. The curse of dimensionality refers to many different phenomena that appear when the dimensionality of the problem is high - where high is varied and depends on the specifics of the application. These phenomena are typically a consequence of the the volume of the space growing exponentially with dimension. For numerical PDEs, two classical curse of dimensionality phenomena apply that highlight the challenges with covering irregular regions in high dimensional spaces with unions of rectangular grids. The first is that the number of points required to uniformly sample an N-dimensional hypercube grows exponentially with the dimension N. The second is that these points concentrate away from the center of the hypercube.


(a) 1/4 of Unit ball covered by a cube in $N = 2$



(b) Volume of a unit ball versus unit cube in N dimensions

Figure 1.1: Covering a unit ball by a hypercube is progressively less and less efficient as the dimension increases. For a spherical distribution of data with $N \geq 4$, the efficiency is less than $\sim 30\%$ and is $\sim 8\%$ by $N = 6$.

To highlight why these two phenomena are specifically important when considering kinetics problems, where the phase-space can be six-dimensional and the mass of the distribution can be a concentrated in a small fraction of the total phase-space, consider the example where we cover a unit ball with a hypercube with a sampling in each unit direction of 0.01. To sample the hypercube with points separated by this spacing requires

$$N_p = 200^N \tag{1.16}$$

where N_p is the total number of points. Recall, for a unit ball and a covering hypercube the volumes are

$$V_{\text{ball}} = \frac{\pi^{N/2}}{\Gamma\left(\frac{N}{2}+1\right)},\ V_{\text{cube}} = 2^N \tag{1.17}$$

$$\tag{1.18}$$

where N is the dimension. It is clear from (1.16) that the total number of points grows exponentially with the dimension but it is less clear how the increase in dimension effects the efficiency. Figure 1.1b, plots the ratio of the volume of the unit ball compared the volume of the covering hypercube as a function of N. In 6D, the covering hypercube would require $N_p \sim 10^{13}$ points and the unit sphere would only be $\sim 8\%$ of the total volume of the hypercube. If one were interested in computing something where only the dynamics within the ball were relevant, using a union of rectangular grids would require an order of magnitude more memory than what is strictly necessary and similarly require an order of magnitude more computation (but it may be much worse if the algorithm does not scale linearly with N_p). This inefficiency can be even worse if the distribution is singular in multiple dimensions or more irregular than a unit ball, as they frequently are when modeling particle accelerators, dark matter, or plasma turbulence.

Lagrangian Methods

Forward semi-Lagrangian methods address issues 2 and 3. These methods advect particles along the Lagrangian trajectories for a time-step and then remap the distribution to a rectangular grid. This is similar to the PIC methods that apply remapping to reduce the impact of "particle noise". In fact, the remapped PIC method and forward semi-Lagrangian method are identical when remapping is applied every few time-steps (note the similarities in [45] and [21]). Forward semi-Lagrangian methods still frequently invoke a rectangular grid in phase-space as part of remapping. This is costly when the dimensionality of the problem is high for some of the same reasons that Eulerian grid based methods suffer from the curse of dimensionality.

We address this by developing an adaptively remapped PIC method. To develop a method that remaps minimally but does not suffer from the errors caused by deformations of the particle locations one first needs to develop an analytical framework that pulls back to the Lagrangian coordinates, similar to the convergence theory for vortex methods, and defines the requirements for convergence in terms of the initial rectangular grid. This framework allows for a principled high-order method to be developed and to understand how and when to apply remapping to control the error. Figure 1.2 summarizes the relationship of the different methods discussed in this section. It highlights some of the challenges that each method faces in reducing the effects of the curse of dimensionality.

1.4 Outline of Thesis

We begin to build the analytic framework by first considering a one-dimensional model problem of advection and interpolation to a rectangular grid in Chapter 2. This model problem and the techniques used to analyze it, specifically the use of the Faà di Bruno formula, were inspired by [17] which built a similar framework to understand a one-dimensional forward semi-Lagrangian method. The major result from the model problem we develop in Chapter 2 is that there is an $O(1)$ error associated with the interpolation of a deformed set of particle locations. We demonstrate that this has the potential to grow exponentially and also that if $\boldsymbol{u}$ depends on this solution that this error has the potential to get fed back into the algorithm, leading to an accumulation of error in the trajectories of the particles. Techniques from the one-dimensional model problem are then used to investigate deposition for the (1+1)D model problem in Chapter 3. We find that an $O(1)$ error appears in the deposition. This error is a function of the configuration space deformation of the particles. We then define a means of computing this deformation for kinetics and introduce remapping as a means of controlling the $O(1)$ error. This framework is then tested in Chapter 4. We find that it does in fact allow us to bound the $O(1)$ error, control the "particle noise" in these benchmark

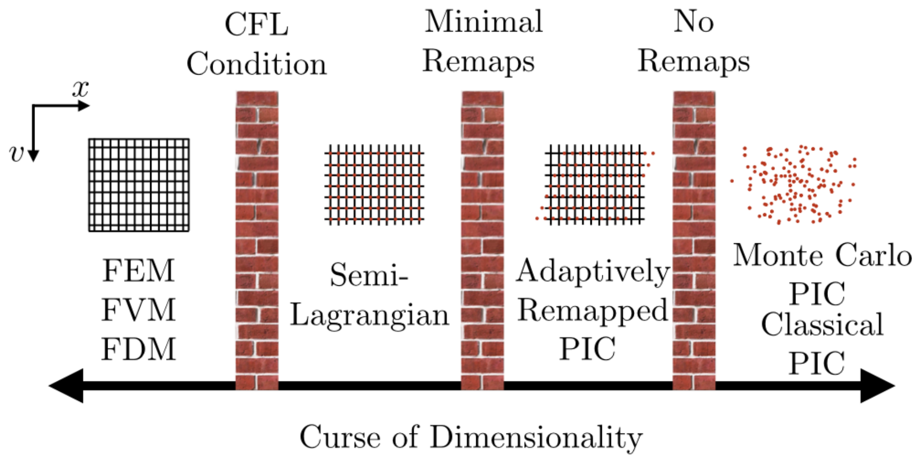


Figure 1.2: Spectrum of methods for solving kinetics problems. Methods to the left suffer more from the curse of dimensionality and methods on the right suffer less from the curse of dimensionality. This is important to kinetics problems as target applications are typically five- or six-dimensional. Brick walls separate the methods. Grid-based methods like finite element (FEM), finite volume (FVM), and finite difference methods (FDM) have to grid the entire phase-space domain. These methods are also limited by the Courant-Lewy-Freidrichs (CFL) condition in the time-step that they can take which exacerbates the curse of dimensionality issue. Semi-Lagrangian methods introduce particles (red dots) and advect the particles for up to a few time steps before interpolating the particles back to the phase-space grid - an operation we will refer to as remapping. This method can have a CFL > 1 and has a well understood convergence theory. Adaptively remapped particle-in-cell methods can reduce the number of remaps required to the minimal necessary for a given accuracy. This method can also be used to reduce the region of phase-space that needs to be remapped. This method is developed in Chapter 3 and Chapter 4. Classical PIC and Monte Carlo PIC methods never remap and thus only use a grid that is the dimension of the configuration space; however, the accuracy of these methods is less understood than the other methods.

problems, and achieve high-order convergence. In Chapter 5 we test this on a (1+2)D set of benchmark problems. Finally, in Chapter 6 we discuss a way of avoiding rectangular arrays in the remapping algorithm. We end with a discussion of future work and ways in which the results we presented provides intuition for solutions to those problems.

Chapter 2

Error Analysis of an Advection Model Problem

2.1 An Advection Model Problem

We begin by considering a model problem of advection and interpolation. Recall equation (1.1) for advection of a scalar quantity. For our model problem we will use the reduced equations:

$$\begin{aligned} \frac{\partial f}{\partial t} + \nabla \cdot (\boldsymbol{u} f) &= 0, && (2.1)\\ \boldsymbol{x} &\in \mathbb{R}^N, && (2.2)\\ f(\boldsymbol{x},t) = f : \mathbb{R}^N \times [0,T] &\rightarrow \mathbb{R}, && (2.3)\\ f(\boldsymbol{x},0) &= f_0(\boldsymbol{x}), && (2.4)\\ \boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{u} : \mathbb{R}^N &\rightarrow \mathbb{R}^N, && (2.5) \end{aligned}$$

where $\boldsymbol{u}(\boldsymbol{x})$ is the smooth velocity field at position $\boldsymbol{x}$. The Lagrangian form of these equations are given in equations (1.4) and (1.7). If the velocity field is divergence free, then f is constant along these trajectories, i.e. $\frac{df}{dt} = 0$. If the velocity field is not divergence free, we can still relate f to f_0 at an arbitrary time. Let V_0 be the initial volume of an infinitesimal parcel, the mass of that infinitesimal particle is fV_0. The change in volume along these trajectories is related through the deformation gradient $\boldsymbol{F} = \{F_{ij}, 1 \leq i,j \leq N\}$,

$$F_{ij} = \frac{\partial X_i(\boldsymbol{\alpha},t)}{\partial \alpha_j}, \tag{2.6}$$

as

$$V(t) = V_0 \det \boldsymbol{F}(\boldsymbol{\alpha},t). \tag{2.7}$$

Using this we see that

$$f(\boldsymbol{X}(\boldsymbol{\alpha},t),t) = \frac{f_0(\boldsymbol{\alpha})}{\det \boldsymbol{F}(\boldsymbol{\alpha},t)}. \tag{2.8}$$

is an exact solution to the advection problem if we have prior knowledge of the form of $\det(\boldsymbol{F}(\boldsymbol{\alpha}, t))$ or can compute it as a part of our system. Further discussion of this topic can be found in [13].

An example that we will use in this chapter that does allow us to use this exact solution is the case where the velocity field depends only on the initial position,

$$\boldsymbol{u}(\boldsymbol{X}(\boldsymbol{\alpha}, t)) = \boldsymbol{u}(\boldsymbol{\alpha}). \tag{2.9}$$

In that case, we can integrate (1.4) to get

$$\boldsymbol{X}(\boldsymbol{\alpha}, t) = \boldsymbol{\alpha} + t\, \boldsymbol{u}(\boldsymbol{\alpha}) \tag{2.10}$$

which can be used to compute the deformation gradient,

$$\boldsymbol{F}(\boldsymbol{\alpha}, t) = \boldsymbol{I} + t\, \nabla_{\boldsymbol{\alpha}} \boldsymbol{u}(\boldsymbol{\alpha}), \tag{2.11}$$

and derive the exact solution,

$$f(\boldsymbol{X}(\boldsymbol{\alpha}, t), t) = \frac{f_0(\boldsymbol{\alpha})}{\det\Big(\boldsymbol{I} + t\, \nabla_{\boldsymbol{\alpha}} \boldsymbol{u}(\boldsymbol{\alpha})\Big)}. \tag{2.12}$$

2.2 Particle Method for Advection Model Problem

To discretize the Lagrangian formulation, we sample the initial distribution, f_0, on a grid. Through out we will use a rectangular initial grid with grid spacing $\boldsymbol{h}_p$ between each particle. To sample the initial distribution on this grid, we utilize the shape function ζ,

$$f_0(\boldsymbol{x}) \approx \sum_k M_k \zeta\left(\boldsymbol{x} - \boldsymbol{\alpha}_k\right) \tag{2.13}$$

where k sums over our set of particles and $M_k(t = 0) = f_0(\boldsymbol{\alpha}_k) \prod_{n=1}^{N} h_{p_n}$ is the mass of a particle and

$$\zeta(\boldsymbol{x} - \boldsymbol{\alpha}) = \prod_{n=1}^{N} \zeta(x_n - \alpha_n). \tag{2.14}$$

Following [39, 12] one can show that given the approximation made in (2.13), a weak solution to (2.1) satisfies the system of ODEs:

$$\frac{dM_k}{dt} = 0 \tag{2.15}$$

$$\frac{d\boldsymbol{X}_k(\boldsymbol{\alpha}_k, t)}{dt} = \boldsymbol{u}(\boldsymbol{X}_k(\boldsymbol{\alpha}_k, t)). \tag{2.16}$$

To reconstruct $f(\boldsymbol{x},t)$ at a later time, we can utilize (2.13) but with our particles at their advected location $\boldsymbol{X}_k(\boldsymbol{\alpha}_k,t)$ in place of the initial position $\boldsymbol{\alpha}_k$ in (2.13). To do this in practice, we need to approximate the shape function in (2.13). We use the shape function $1/h_g W_{q,r}(x/h_g)$, where h_g is the grid spacing of the grid we are interpolating onto. $W_{q,r}$ is a one-dimensional kernel that satisfies q moment conditions,

$$\sum_i W_{q,r}\left(\frac{x_i - x}{h_g}\right) = 1 \tag{2.17}$$

$$\sum_i (x - x_i)^{q'} W_{q,r}\left(\frac{x_i - x}{h_g}\right) = 0 \text{ for } 1 \le q' \le q-1, \tag{2.18}$$

and is in C^r. Historically, particle-in-cell and vortex-in-cell methods have used C^0 or C^1 kernels for this approximation. For example, [37] used

$$W_{2,0}(x) = \begin{cases} 1 - |x|, & 0 \le |x| < 1 \\ 0 & 1 \le |x| \end{cases} \tag{2.19}$$

and

$$W_{4,0} = \begin{cases} 1 - \frac{1}{2}|x| - |x|^2 + \frac{1}{2}|x|^3 & |x| < 1 \\ 1 - \frac{11}{6}|x| + |x|^2 - \frac{1}{6}|x|^3 & 1 \le |x| < 2 \\ 0 & 2 \le |x| \end{cases} \tag{2.20}$$

for the deposition and force interpolation stage of second and fourth order particle-in-cell methods. Stencil size has been a concern for particle methods and thus using minimally smooth kernels allowed the use of smaller sized stencils due to the decrease in required degrees of freedom.

Forward semi-Lagrangian methods have began to use smooth kernels to achieve high-order accuracy [17, 19, 18]. Although this comes at an increased cost due to the larger stencil size associated with a smoother kernel, for memory-bound algorithms like particle methods this is a trade-off worth making due to the increased throughput for a given accuracy. Examples of some of the C^2 kernels that we consider are

$$W_{2,2} = \begin{cases} 1 - |x|^2 - \frac{9}{2}|x|^3 + \frac{15}{2}|x|^4 - 3|x|^5 & |x| < 1 \\ -4 + 18|x| - 29|x|^2 + \frac{43}{2}|x|^3 - \frac{15}{2}|x|^4 + |x|^5 & 1 \le |x| < 2 \\ 0 & 2 \le |x| \end{cases} \tag{2.21}$$

and

$$W_{4,2} = \begin{cases} 1 - \frac{5}{4}|x|^2 - \frac{35}{12}|x|^3 + \frac{21}{4}|x|^4 - \frac{25}{12}|x|^5 & |x| < 1 \\ -4 + \frac{75}{4}|x| - \frac{245}{8}|x|^2 - \frac{545}{24}|x|^3 + \frac{63}{8}|x|^4 + \frac{25}{24}|x|^5 & 1 \le |x| < 2 \\ 18 - \frac{153}{4}|x| + \frac{255}{8}|x|^2 - \frac{313}{24}|x|^3 + \frac{21}{8}|x|^4 - \frac{5}{24}|x|^5 & 2 \le |x| < 3 \\ 0 & 3 \le |x| \end{cases}, \tag{2.22}$$

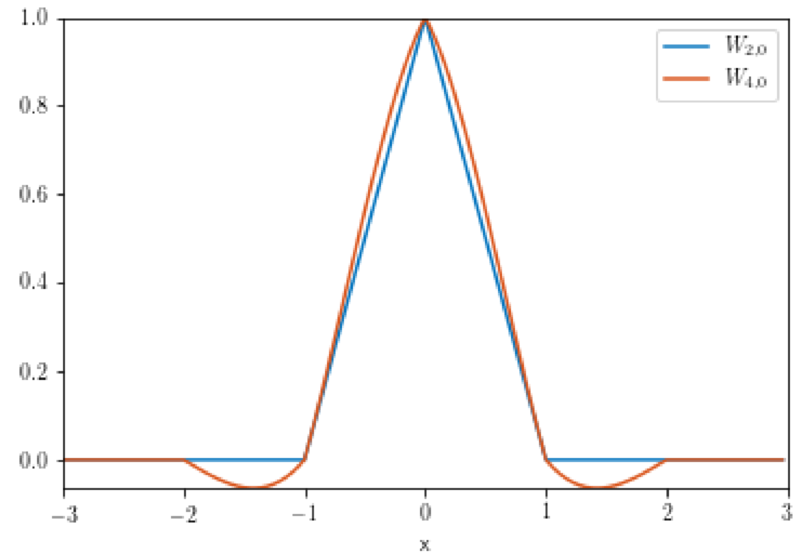


Figure 2.1: Plot of $W_{2,0}$ and $W_{4,0}$

which both can be found in [17]. We also consider the C^4 kernel

$$W_{4,4} = \begin{cases} 1 - \frac{5}{4}|x|^2 + \frac{1}{4}|x|^4 - \frac{100}{3}|x|^5 + \frac{455}{4}|x|^6 - \frac{295}{2}|x|^7 + \frac{345}{4}|x|^8 - \frac{115}{6}|x|^9 & |x| < 1 \\ -199 + \frac{5485}{4}|x| - \frac{32975}{8}|x|^2 + \frac{28425}{4}|x|^3 - \frac{61953}{8}|x|^4 + \frac{33175}{6}|x|^5 & \\ \quad - \frac{20685}{8}|x|^6 + \frac{3055}{4}|x|^7 - \frac{1035}{8}|x|^8 + \frac{115}{12}|x|^9 & 1 \le |x| < 2 \\ 5913 - \frac{89235}{4}|x| + \frac{297585}{8}|x|^2 - \frac{143895}{4}|x|^3 + \frac{177871}{8}|x|^4 - \frac{54641}{6}|x|^5 & \\ \quad + \frac{19775}{8}|x|^6 - \frac{1715}{4}|x|^7 + \frac{345}{8}|x|^8 - \frac{23}{12}|x|^9 & 2 \le |x| < 3 \\ 0 & 3 \le |x| \end{cases} \tag{2.23}$$

from [17].

The reconstruction of f can be written as

$$\tilde{f}_i = \sum_k \frac{M_k}{\prod_{n=1}^{N} h_{g_n}} W_{q,r}\left(\frac{\boldsymbol{x}_i - \boldsymbol{X}_k}{\boldsymbol{h}_g}\right) \tag{2.24}$$

where k indexes over the set of particles, i indexes over the uniform grid, x_i and X_k denote the position of the grid point and particle respectively,and M_k is the mass of the kth particle.

To our knowledge, little has been done to formalize how deformations in the set of particle positions effect the representation of a distribution on the rectangular grid. It was noted as a potential issue in [35] but was concluded that the worst case Monte Carlo estimate was

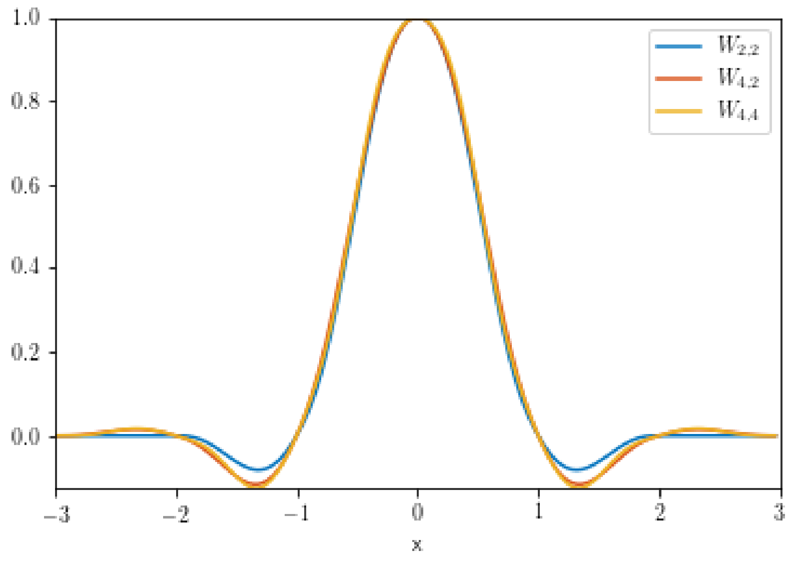


Figure 2.2: Plot of $W_{2,2}$, $W_{4,2}$, and $W_{4,4}$

unlikely to be achieved since the energy available in the systems was too low to cause large fluctuations. We approach this problem similar to [19] where we first examine the simplest one-dimensional case and then expand this theory to the kinetics problems in Section 3.3. The one-dimensional method advects a set of particles with a velocity that is dependent on the initial positions of the particles and computes the interpolation error on a rectangular grid.

2.3 Error Analysis of One-Dimensional Model Problem

We begin by analyzing the case where the spacing of the grid we are depositing onto, h_g, is equal to the initial spacing of the particles h_p. For clarity, we drop the subscript and denote the grid spacing as h. We evaluate at a fixed time t, thus suppress the explicit dependence on t. Furthermore, we assume that both the particle positions and their values of f are given by smooth functions of the Lagrangian coordinates α evaluated at grid points:

$$x_k = x(kh) \ , \ f_k = f(kh) \ ; \ f, x = f(\alpha), x(\alpha). \tag{2.25}$$

We assume $x(\alpha) = x_0 + \alpha + \delta(\alpha)$, where $\delta(\alpha)$ is a smooth function with $\delta(0) = 0$, and x_0 is the nearest Lagrangian grid point to the point $\bar{x}$ at which we are evaluating the interpolation function:

$$x_0 = x\Big(\Big\lfloor \frac{\alpha(\bar{x})}{h} \Big\rfloor h\Big). \tag{2.26}$$

Similar to [17], we define

$$\mathcal{I}(\{f_k, x_k\}, \bar{x}) = \sum_k f(kh)\mathcal{I}(k, kh), \tag{2.27}$$

$$\mathcal{I}(k, \alpha) \equiv \mathcal{W}\Big(\frac{\bar{x} - x_0}{h} - k - \frac{\delta(\alpha)}{h}\Big), \tag{2.28}$$

and for each summand, expand $f(\alpha)$ and $\mathcal{I}(k, \alpha)$ in a Taylor series about $\alpha = 0$:

$$f(kh) \to \sum_{L=0}^{L-1} \frac{d^l f}{d\alpha^l}\Big|_{\alpha=0} \frac{(kh)^l}{l!}, \tag{2.29}$$

$$\mathcal{I}(k, kh) \to \sum_{m=0}^{M-1} \frac{d^m}{d\alpha^m}\Big(\mathcal{I}(k, \alpha)\Big)\Big|_{\alpha=0} \frac{(kh)^m}{m!}. \tag{2.30}$$

Note that $\frac{d^m}{d\alpha^m}\mathcal{I}(k, \alpha)$ is an mth order derivative of a composition of functions. Therefore, we apply the Faà di Bruno formula [28],

$$\frac{d^N}{dx^N}(f(g(x)) = \sum_{n=1}^{N} \frac{d^n f}{dz^n}\Big|_{g(x)} \sum \frac{N!}{n_1! \dots n_N!} \prod_{j=1}^{N} \Big(\frac{1}{j!}\frac{d^j g}{dx^j}\Big)^{n_j}, \tag{2.31}$$

to compute the derivatives of $\mathcal{I}(k, \alpha)$:

$$\frac{d^N}{d\alpha^N}\Big(\mathcal{I}(k, \alpha)\Big) = \sum_{n=1}^{N} \frac{1}{h^n}\frac{d^n \mathcal{W}}{dz^n}\Big|_{\frac{\bar{x}-x_0}{h}-k} \sum \frac{N!}{n_1! \dots n_N!} \prod_{j=1}^{N} \Big(-\frac{1}{j!}\frac{d^j \delta}{d\alpha^j}\Big)^{n_j}. \tag{2.32}$$

For the case $n = N$, the inner sum has only one term, corresponding to the case $n_1 = N$, $n_j = 0$ for $j > 1$. From that we conclude

$$\frac{d^N}{d\alpha^N}\Big(\mathcal{I}(k, \alpha)\Big) = \frac{1}{h^N}\frac{d^N \mathcal{W}}{dz^N}\Big|_{\frac{\bar{x}-x_0}{h}-k}\Big(-\frac{d\delta}{d\alpha}\Big)^N + O(h^{-(N-1)}). \tag{2.33}$$

We replace $f(kh)$, $\mathcal{I}(k, kh)$ in (2.28) by their Taylor expansions (2.29), (2.30), keeping only the terms up to order $M - 1$:

$$\sum_k \sum_{l,m} \frac{1}{l!m!}(kh)^{l+m} \frac{d^l f}{d\alpha^l}\Big|_{\alpha=0} \frac{d^m \mathcal{W}}{dz^m}\Big|_{\frac{\bar{x}-x_0}{h}-k}\Big(-\frac{1}{h}\frac{d\delta}{d\alpha}\Big)^m, \tag{2.34}$$

where the inner sum is over $\{l, m \geq 0 : l + m < M\}$. Summing over k first, and using the differentiated moment relations [17],

$$\sum_{k \in \mathbb{Z}} k^l \frac{d^m \mathcal{W}}{dz^m}\Big|_{z=x-k} = \begin{cases} \frac{l!}{(l-m)!} x^{l-m} & \text{for } l \geq m \\ 0 & \text{for } l < m, \end{cases} \tag{2.35}$$

we obtain

$$\mathcal{I}(\{f_k, x_k\}, \bar{x}) = \sum_{l,m} \frac{1}{l!} (\bar{x} - x_0)^l \frac{d^l f}{d\alpha^l}\Big|_{\alpha=0} \Big(- \frac{d\delta}{d\alpha} \Big)^m \frac{(m+l)!}{l! m!} + O\Big(\frac{d\delta}{d\alpha} \Big)^M + O(h). \tag{2.36}$$

$$\tag{2.37}$$

Recall that from the definition of x_0,

$$|\bar{x} - x_0| \leq Ch. \tag{2.38}$$

Using this, we see that for all terms with $l > 0$ the expression is at least $O(h)$ and thus can be written as

$$\mathcal{I}(\{f_k, x_k\}, \bar{x}) = f(0) \sum_{m=0}^{M-1} \Big(- \frac{d\delta}{d\alpha} \Big)^m + O\Big(\frac{d\delta}{d\alpha} \Big)^M + O(h), \tag{2.39}$$

$$= \frac{f(0)}{1 + \frac{d\delta}{d\alpha}} + O\Big(\frac{d\delta}{d\alpha} \Big)^M + O(h). \tag{2.40}$$

Reintroducing the time variable, $\mathcal{I}(\{f_k, x_k\}, \bar{x})$ approximates the solution to the advection equation in one-dimension

$$\frac{\partial f}{\partial t} + \frac{\partial (uf)}{\partial x} = 0 \Leftrightarrow f(x(\alpha, t), t) = \frac{f(\alpha, 0)}{F}, \; F = \frac{\partial x}{\partial \alpha} \tag{2.41}$$

with an error given by

$$\epsilon = O\Big(\Big(F - 1 \Big)^M \Big) + O(h). \tag{2.42}$$

Note that the leading term in the error, $(F - 1)^M$, is an $O(1)$ error that is a barrier to the convergence. This leads to two different regimes: a regime where the h-error is dominant and the spline interpolation analysis holds, and a second regime where h is sufficiently small such that the $O\Big(\Big(F - 1 \Big)^M \Big)$ becomes the dominant error and thus the method will appear to no longer converge, and in fact, the error will appear to be independent of h. The only parameter that provides any control over either regime is the kernel choice. Once a kernel is chosen, the barrier error is determined by the dynamics of the problem. Barriers to the convergence of a numerical method are often addressed by setting a tolerance and controlling the barrier such that it stays below that tolerance. An example of a method that does this is the fast multipole method [29]. In Chapter 4 we will introduce a means of controlling the barrier error in the context of kinetics.

For the more general case, $h_g \neq h_p$, we define

$$N_p = \frac{h_g}{h_p} \tag{2.43}$$

where $N_p \in \mathbb{N}^+$. Using this, we can write down the deposition algorithm as

$$\mathcal{I}(\{f_k, x_k\}, \bar{x}) = \frac{h_p}{h_g} \sum_k \sum_{n_p=0}^{N_p-1} f(kh_g + n_p h_p) \mathcal{I}(k, kh_g, n_p h_p), \tag{2.44}$$

$$\mathcal{I}(k, \alpha, n_p h_p) \equiv \mathcal{W}\Big(\frac{\bar{x} - x_0 - n_p h_p}{h_g} - k - \frac{\delta(\alpha + n_p h_p)}{h_g}\Big). \tag{2.45}$$

Defining

$$\tilde{x}_{n_p} \equiv x_0 + n_p h_p,\ \ \tilde{f}_{n_p}(\alpha) \equiv f(\alpha + n_p h_p),\ \ \tilde{\delta}_{n_p}(\alpha) \equiv \delta(\alpha + n_p h_p)$$

allows us to rewrite (2.44) as

$$\mathcal{I}(\{f_k, x_k\}, \bar{x}) = \frac{h_p}{h_g} \sum_k \sum_{n_p=0}^{N_p-1} \tilde{f}_{n_p}(kh_g) \tilde{\mathcal{I}}(k, kh_g, n_p h_p), \tag{2.46}$$

$$\tilde{\mathcal{I}}(k, \alpha, n_p h_p) \equiv \mathcal{W}\Big(\frac{\bar{x} - \tilde{x}_{n_p}}{h_g} - k - \frac{\tilde{\delta}_{n_p}(\alpha)}{h_g}\Big), \tag{2.47}$$

which resembles (2.28) and allows us to apply the same expansions. Applying these expansions gives

$$\mathcal{I}(\{f_k, x_k\}, \bar{x}) = \frac{h_p}{h_g} \sum_{n_p=0}^{N_p-1} f(n_p h_p) \sum_{m=0}^{M-1} \Big(-\frac{d\delta}{d\alpha}\Big|_{n_p h_p}\Big)^m + O\Big(\frac{d\delta}{d\alpha}\Big|_{n_p h_p}\Big)^M + O(h_g), \tag{2.48}$$

$$= \frac{h_p}{h_g} \sum_{n_p=0}^{N_p-1} \left(\frac{f(n_p h_p)}{1 + \frac{d\delta}{d\alpha}\Big|_{n_p h_p}} + O\Big(\frac{d\delta}{d\alpha}\Big|_{n_p h_p}\Big)^M \right) + O(h_g), \tag{2.49}$$

which in the case of $f(\alpha) \equiv 1$ and $\delta = c\alpha$ simplifies further to give the same result as the case $h_g = h_p$. The reason we consider this special case is that it highlights some important features of the error for this problem. Namely, we can expect that for a fixed n_p, the resulting error should decrease at a rate of h_p/h_g but that the total error when summed over all n_p should be the same as if $h_p = h_g$. That is to say, we would not expect an improvement in the $O(1)$ error by using multiple particles per cell. We will further explore this in Section 2.4.

Recall the definition of F in equation (2.41) and differentiate it with respect to time. Applying the chain rule gives the ordinary differential equation

$$\frac{dF}{dt} = \frac{dv}{dx}F, F(\alpha, t = 0) = 1. \tag{2.50}$$

If the velocity is time-independent, then a solution to this system is

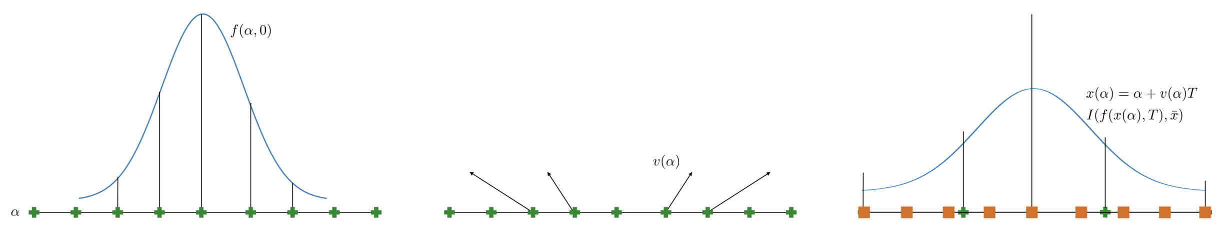
$$F(\alpha, t) = e^{ct},\ c = \frac{dv}{dx} \tag{2.51}$$

and the associated error would therefore be

$$\epsilon = O\Big((e^{ct} - 1)^M\Big) \tag{2.52}$$

which grows exponentially in time and highlights how there can be disastrous growth in the $O(1)$ error. If computing $\boldsymbol{u}$ depends on the representation of f on the rectangular grid, this $O(1)$ error can be fed back into the particle trajectories and lead to catastrophic losses in accuracy.

2.4 Numerical Results for One-Dimensional Model Problem


$f(\alpha,0)$
α
$v(\alpha)$
$x(\alpha) = \alpha + v(\alpha)T$
$I(f(x(\alpha),T),\bar{x})$

(a) Initial Distribution (b) Velocity Field (c) Interpolated Distribution

Figure 2.3: At initial particle positions (green plus signs) we sample a distribution (a), apply a velocity that is dependent on the initial particle position (b), and then interpolate the resulting distribution to a uniform grid (orange squares) where we evaluate the error (c).

To test the analysis in Section 2.3, we initially tried both a constant and Gaussian initial condition; however, the error for both distributions were dominated by the $O(1)$ barrier. To examine both the convergence and barrier, we found that the initial condition

$$f(\alpha) = \sin k\alpha \tag{2.53}$$

$$k = 4\pi \tag{2.54}$$

with $\alpha \in [-1/2, 1/2]$ captured both regimes for the kernels of interest. The particles are advected according to

$$x(\alpha) = \alpha + c\alpha + \frac{1}{\sqrt{2}} \tag{2.55}$$

and deposited on the initial grid where the error,

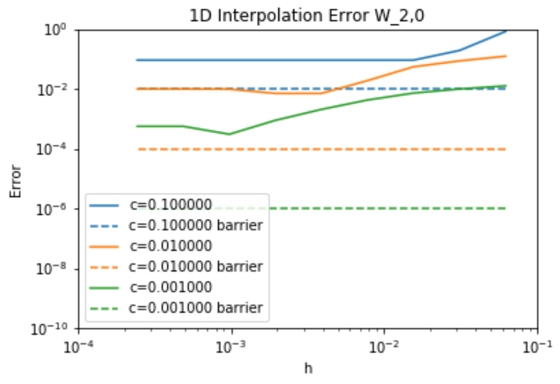
$$\epsilon = \max_i \left(\left| \mathcal{I}(\{f_k, x_k\}, (ih)) - \frac{1}{1+c} f\left(\frac{ih}{1+c}\right) \right| \right), \tag{2.56}$$

is computed. The $\frac{1}{\sqrt{2}}$ factor was included to mitigate the chances of cancellations that may come from the particles being initially aligned with the rectangular grid that we interpolate onto and use to evaluate the error. From above, given the form of the perturbation, the $O(1)$ barrier is c^M where the exponent M is determined by the order and smoothness of the interpolation kernel. Error plots for this 1D test are displayed in Figure 2.4. The parameters used in these tests are $c \in \{10^{-1}, 10^{-2}, 10^{-3}\}$ and $h = 1/N$ with $N = 2^M$ and M taking integer values in the interval $[4, 12]$.

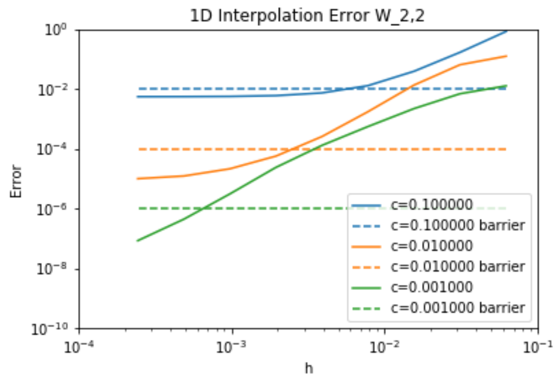
For all of the kernels we see that there are regions where the error is $O(h^p)$ and then reaches a $O(1)$ barrier that dominates the error as h continues to decrease. The order and smoothness of the kernels effect each of these regions differently. We see that for $W_{2,0}$ the kernels stay well above the barrier predicted by the number of moment conditions the kernel satisfies and is not exhibiting the predicted second order accuracy. This suggests that the barrier is being dominated by the smoothness rather than the order of the kernel. Where as $W_{2,2}$ converges at second order and has a barrier that scales like c^2, as suggested by the analysis. For $W_{4,2}$, we see a similar result as above, where the number of moment conditions that the kernel satisfies gives the order of convergence but the smoothness of the kernel determines the value of the $O(1)$ barrier. The c^2 barrier for the $W_{4,2}$ kernel may be problem specific as there is numerical evidence for (1+1)D Vlasov-Poisson kinetics simulations that controlling the $O(1)$ error assuming it scales like c^4 is enough to achieve fourth order convergence (see Chapter 4). Finally, $W_{4,4}$ has a convergence rate of h^4 and displays a barrier that scales as c^4. For $c = 10^{-3}$ a barrier appears earlier than expected. This is possibly due to the effects of roundoff. For high-order interpolation kernels the condition number can become a factor and may require using quadruple precision to compute the interpolation and achieve the expected accuracy [34].

Multiple Particles Per Cell

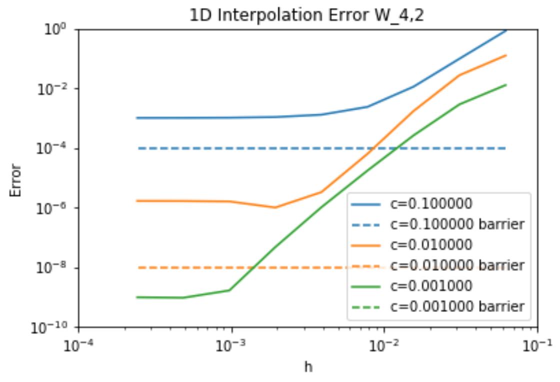
As noted in the analysis above, adding more particles per cell should not increase the accuracy of the method. This contradicts what we observe for the above one-dimensional simulations



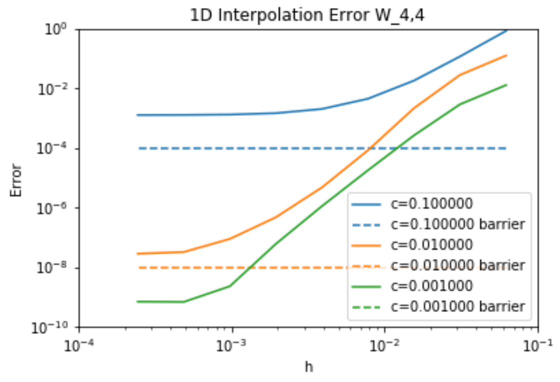
(a) $W_{2,0}$



(b) $W_{2,2}$



(c) $W_{4,2}$



(d) $W_{4,4}$

Figure 2.4: Log-log error plots for the 1D test problem described with one particle per cell. These plots clearly display the dependence of the barrier on both the number of moment conditions the kernel satisfies as well as the smoothness of the kernel.

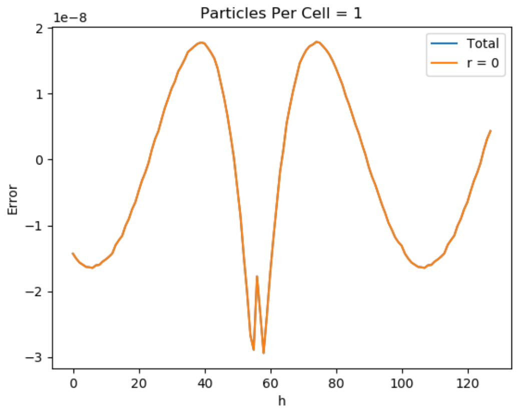
and what the community has long held as one of the primary tools for controlling accuracy in particle methods. To understand this discrepancy, we simplified the distribution to

$$f(\alpha) \equiv 1 \tag{2.57}$$

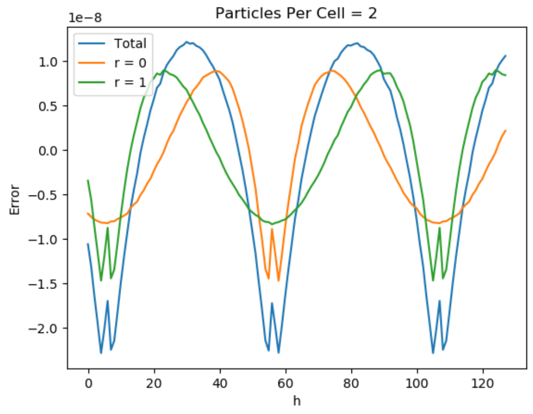
so that we could examine the contribution to the error of every n_pth particle separately. For N_{ppc} particles per cell, the contribution to the error from the set of every n_p-th particle is

$$\epsilon_{n_p}(x) = \sum_{k:\ (k \mod N_{\text{ppc}})=n_p} \mathcal{I}(\{f_k, x_k\}, (x)) - \frac{1}{1+c}\frac{1}{N_{\text{ppc}}}, \tag{2.58}$$

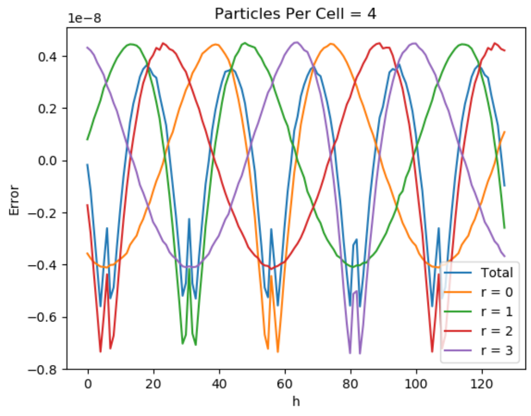
where $n_p \in \{0, 1, \ldots, N_p - 1\}$. Figure 2.5 shows the results for $W_{4,4}$ with $N_{\text{ppc}} = 1, 2, 4$ and 8. For $N_{\text{ppc}} = 2$ we see that the error contributions constructively add but for $N_{\text{ppc}} = 4$



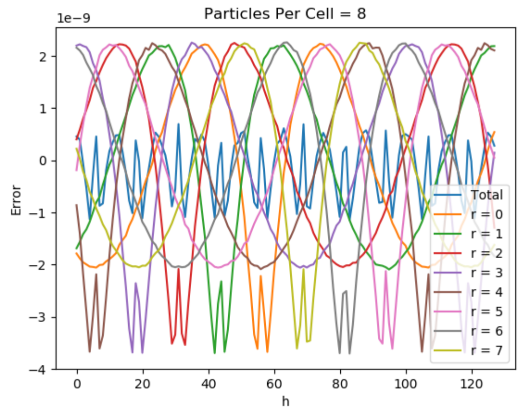
(a) $N_{\text{ppc}} = 1$



(b) $N_{\text{ppc}} = 2$



(c) $N_{\text{ppc}} = 4$



(d) $N_{\text{ppc}} = 8$

Figure 2.5: Error plots for the 1D test problem with a constant distribution described in Section 2.4. The interpolation kernel used is $W_{4,4}$. These plots show the dependence of the total error on the number of particles per cell and the way in which individual errors can accumulate destructively or constructively.

and $N_{\text{ppc}} = 8$ the errors add destructively. However, the contribution from each set of n_p-th particles decreases by half in magnitude with each increase in number of particles per cell. This is in line with the analysis that the total error is constant with increasing number of particles per cell; however, the analysis does not account for the possibility of the errors adding in a constructive or destructive manner. To fully understand when these errors add constructively or destructively appears to require a different approach to the analysis.

Chapter 3

Particle Methods for Kinetics

3.1 Vlasov-Poisson System

The Vlasov-Poisson system models a collisionless plasma in the absence of magnetic fields. The Vlasov-Poisson system is useful in modeling plasma phenomena such as Landau damping as well as in the study of the evolution of dark matter where instead of electrostatic forces there are gravitational forces.

The Vlasov-Poisson system can be written in the form of (1.1) if we take our "position" variable corresponding to $\boldsymbol{x}$ to be the position in phase-space $(\boldsymbol{x}, \boldsymbol{v})^T$. The velocity field is $\boldsymbol{u} = (\boldsymbol{v}, -\boldsymbol{E}(\boldsymbol{x}))^T$ where $\boldsymbol{E}(\boldsymbol{x})$ is the electric field and f is a function of the phase-space location $(\boldsymbol{x}, \boldsymbol{v})$. We do not have sources, sinks, or collisions which means that $R \equiv 0$. Making these substitutions, expanding the derivatives, and recognizing that the velocity field, $\boldsymbol{u}$, is divergence free gives the more common form

$$\frac{\partial f}{\partial t} + \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} f - \boldsymbol{E}(\boldsymbol{x}) \cdot \nabla_{\boldsymbol{v}} f = 0. \tag{3.1}$$

To close the system, we need some relationship between $f(\boldsymbol{x}, \boldsymbol{v}, t)$ and the electric field $\boldsymbol{E}(\boldsymbol{x})$. This relationship comes in the form of Poisson's equation which can be written as

$$\rho(\boldsymbol{x}) = \int f(\boldsymbol{x}, \boldsymbol{v}, t) d\boldsymbol{v} - 1, \tag{3.2}$$

$$\Delta\phi(\boldsymbol{x}) = \rho(\boldsymbol{x}), \tag{3.3}$$

$$\boldsymbol{E}(\boldsymbol{x}) = \nabla_{\boldsymbol{x}}\phi(\boldsymbol{x}). \tag{3.4}$$

To express this in the Lagrangian frame, we use the phase-space Lagrangian variables

$$(\boldsymbol{x}(\boldsymbol{\alpha}, t), \boldsymbol{v}(\boldsymbol{\alpha}, t))^T \in \mathbb{R}^{N_{x+v}} \tag{3.5}$$

where N_{x+v} is the phase-space dimension, which is the sum of the configuration space and velocity space dimension. Since the configuration space and velocity space dimension may differ, we will typically refer to the space as $(N_x + N_v)$D. The initial position of the Lagrangian variables are

$$(\boldsymbol{x}(\boldsymbol{\alpha}, 0), \boldsymbol{v}(\boldsymbol{\alpha}, 0))^T = (\boldsymbol{\alpha}_x, \boldsymbol{\alpha}_v)^T. \tag{3.6}$$

This gives the set of ODEs

$$\frac{d}{dt}(\boldsymbol{x}(\boldsymbol{\alpha}, t), \boldsymbol{v}(\boldsymbol{\alpha}, t))^T = (\boldsymbol{v}(\boldsymbol{\alpha}, t), -\boldsymbol{E}(\boldsymbol{x}(\boldsymbol{\alpha}, t), t))^T \tag{3.7}$$

$$\frac{d}{dt}f(\boldsymbol{x}(\boldsymbol{\alpha}, t), \boldsymbol{v}(\boldsymbol{\alpha}, t), t) = 0 \tag{3.8}$$

which are again augmented with (3.2). From these conditions, note that to define the charge density in configuration space one takes an integral over the velocity space dimensions. In (1+1)D this integral is

$$I_x(x) \equiv \int_{v_L}^{v_H} f(x, v, t)dv. \tag{3.9}$$

To express this in the Lagrangian coordinate system, we can express the differentials as

$$dx = \frac{\partial x}{\partial \alpha_x}d\alpha_x + \frac{\partial x}{\partial \alpha_v}d\alpha_v \tag{3.10}$$

$$dv = \frac{\partial v}{\partial \alpha_x}d\alpha_x + \frac{\partial v}{\partial \alpha_v}d\alpha_v. \tag{3.11}$$

Note, we are integrating for a fixed x and thus $dx = 0$. Using this simplifies the above differentials into a single expression for dv:

$$dv = \Big(\frac{\partial x}{\partial \alpha_x}\Big)^{-1}\Big(\frac{\partial x}{\partial \alpha_x}\frac{\partial v}{\partial \alpha_v} - \frac{\partial x}{\partial \alpha_v}\frac{\partial v}{\partial \alpha_x}\Big)d\alpha_v. \tag{3.12}$$

This further simplifies by recalling the incompressibility condition

$$\frac{\partial x}{\partial \alpha_x}\frac{\partial v}{\partial \alpha_v} - \frac{\partial v}{\partial \alpha_x}\frac{\partial x}{\partial \alpha_v} \equiv 1. \tag{3.13}$$

Using this change of variables, the integral defining the charge density can be written in the Lagrangian frame as

$$I_x(x) = \int_{v_L}^{v_H} f_0(\alpha_x(x, \alpha_v, t), \alpha_v)\Big(\frac{\partial x}{\partial \alpha_x}\Big)^{-1}d\alpha_v. \tag{3.14}$$

We can generalize this to higher dimensions using the implicit function theorem and the Schur determinant formula. Recall, the implicit function theorem states that there exists $\boldsymbol{A}(\boldsymbol{\alpha}_v)$ such that

$$\boldsymbol{x}(\boldsymbol{A}(\boldsymbol{\alpha}_v, t), \boldsymbol{\alpha}_v) = \text{const} \tag{3.15}$$

and

$$\frac{\partial \boldsymbol{A}}{\partial \boldsymbol{\alpha}_v} = -\left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\alpha}_x}\right)^{-1} \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\alpha}_v}. \tag{3.16}$$

The total derivative of $\boldsymbol{v}(\boldsymbol{\alpha}_x, \boldsymbol{\alpha}_v)$ can then be expanded

$$\begin{aligned} \frac{D\boldsymbol{v}}{D\boldsymbol{\alpha}_v} &= \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{\alpha}_v} + \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{\alpha}_x}\frac{\partial \boldsymbol{A}}{\partial \boldsymbol{\alpha}_v} && (3.17)\\ &= \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{\alpha}_v} - \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{\alpha}_x}\left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\alpha}_x}\right)^{-1} \frac{\partial \boldsymbol{x}}{\boldsymbol{\alpha}_v}. && (3.18)\end{aligned}$$

From the Schur determinant formula [47],

$$\begin{aligned} \det\left(\frac{D\boldsymbol{v}}{D\boldsymbol{\alpha}_v}\right) &= \det\left(\frac{\partial \boldsymbol{v}}{\partial \boldsymbol{\alpha}_v} - \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{\alpha}_x}\left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\alpha}_x}\right)^{-1} \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\alpha}_v}\right) && (3.19)\\ &= \det\left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\alpha}_x}\right)^{-1} \det\begin{pmatrix} \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\alpha}_x} & \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\alpha}_v} \\ \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{\alpha}_x} & \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{\alpha}_v}\end{pmatrix} && (3.20)\\ &= \det\left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\alpha}_x}\right)^{-1} && (3.21)\end{aligned}$$

where the final reduction uses the fact that the system is incompressible. As in the (1+1)D case, this allows us to write the deposition stage of the algorithm in the Lagrangian frame as

$$I_x(\boldsymbol{x}) = \int_{v_L}^{v_H} f_0(\boldsymbol{\alpha}_x(\boldsymbol{x}, \boldsymbol{\alpha}_v, t), \boldsymbol{\alpha}_v) \det\left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\alpha}_x}\right)^{-1} d\boldsymbol{\alpha}_v. \tag{3.22}$$

One interesting aspect of the above is the similarity in the integrand to the exact solution in Section 2.2. This should not be surprising though given that for a fixed α_v the particles on that line in configuration space are being advected with perturbations in the velocity causing deformations in the configuration space dimensions. The interpolation of the charge to the grid in a PIC method is then integrating contributions from each of these advected lines and can be pulled back to the Lagrangian frame using the exact solution of the advection equation but only the configuration space contribution. This connection to the one-dimensional problem is motivation to apply a similar analysis to examine the exact form of the error in the density caused by deformations in the configuration space. We do this for (1+1)D in Section 3.3 after introducing the discrete system and algorithm in Section 3.2.

3.2 Particle-in-cell Method for Arbitrary Dimension Vlasov-Poisson

To discretize this system for one configuration space and one velocity dimension, we follow the procedure in Section 2.2. In this case, we are starting our particles on an initially uniform grid and define the mass (often called charge in the kinetics context) as

$$M_p = f(\boldsymbol{\alpha}_{x_p}, \boldsymbol{\alpha}_{v_p}, 0) h_x^{N_x} h_v^{N_v}, \tag{3.23}$$

where we assume, but are not limited to, a uniform grid size in each configuration space and velocity space dimension. With this mass and the shape function approximation (2.13), we can then advect the particles according to the Lagrangian trajectories in (3.7).

In practice, to solve this system, a particle-in-cell method traditionally has three components to the algorithm that makes up each stage of the integrator: deposition, a field solve, and force interpolation. We discuss our implementation of these components as well as the Runge-Kutta method we used.

Deposition

To compute the charge density on the uniform Cartesian grid, we sum over the set of particles as follows:

$$\rho_i = \sum_p \frac{M_p}{V_i} W_{q,r}\left(\frac{\boldsymbol{x}_i - \boldsymbol{x}_p}{\Delta x}\right) \tag{3.24}$$

where V_i is the volume of the Poisson cell (Δx^{N_x} in this case), and $W_{q,r}$ is a one-dimensional kernel that satisfies q moment conditions and is in C^r. The application to N_x dimensional vectors is defined in equation (2.14). Through out this manuscript we will use kernels like those presented in Section 2.2. We found that it was beneficial to use kernels that have, at a minimum, a continuous first derivative to achieve high orders of accuracy.

Field Solve

Once the density has been computed, each node uses a spectral method based on the FFTW package to solve

$$\Delta\phi = -\rho.$$

A fourth-order finite difference stencils is then applied to compute $\boldsymbol{E}$. We chose to have each rank solve this step independently since [45] found that this step of the PIC algorithm was $\sim$10% of the total computation time.

Force Interpolation

To determine the effective electric field felt by each particle, we use the following scheme:

$$\boldsymbol{E}_p = \sum_i \boldsymbol{E}_i W_{q,r}\left(\frac{\boldsymbol{x}_i - \boldsymbol{x}_p}{\Delta x}\right) \tag{3.25}$$

where to avoid self-forcing we require $W_{q,r}$ to be the same as used in the deposition.

Integration

To move the particles, we use high-order Runge-Kutta schemes. In [37], a second- and fourth-order scheme that reduced the number of required stages was presented. For our simulations, we use a the three-stage fourth-order integrator

$$\boldsymbol{x}^{n+1} = \boldsymbol{x}^n + \boldsymbol{v}^n \Delta t + \frac{1}{6}(\boldsymbol{k}_1 + 2\boldsymbol{k}_2)\Delta t^2, \tag{3.26}$$

$$\boldsymbol{v}^{n+1} = \boldsymbol{v}^n + \frac{1}{6}(\boldsymbol{k}_1 + 4\boldsymbol{k}_2 + \boldsymbol{k}_3)\Delta t \tag{3.27}$$

where

$$\boldsymbol{k}_1 = \boldsymbol{E}\left(\boldsymbol{x}^n\right), \tag{3.28}$$

$$\boldsymbol{k}_2 = \boldsymbol{E}\left(\boldsymbol{x}^n + \frac{1}{2}\boldsymbol{v}^n \Delta t + \frac{1}{8}\boldsymbol{k}_1 \Delta t^2\right), \tag{3.29}$$

$$\boldsymbol{k}_3 = \boldsymbol{E}\left(\boldsymbol{x}^n + \boldsymbol{v}^n \Delta t + \frac{1}{2}\boldsymbol{k}_2 \Delta t^2\right). \tag{3.30}$$

Note, we can use the fact that $\boldsymbol{k}_1 = \boldsymbol{E}\left(\boldsymbol{x}^n\right)$ needs to be computed for this method and thus could be reused as part of a forward Euler integrator of any quantity that relies on the electric field.

3.3 Error analysis of deposition model problem in (1+1)D

We consider a model problem similar to the one developed in Chapter 2. The ODEs we integrate are given in (3.7) and the closure equations are given in (3.2). We assume the electric field is some known function that we can integrate. Perturbations to the streaming dynamics will be caused by a change in acceleration (the electric field in this case). For the purposes of this model problem, we evaluate the error in charge density on the grid and do not consider how this may get fed back into the system via the electric field. A perturbation to the velocity of this form will manifest in perturbed streaming equations as

$$x = \alpha_x + x_0(\alpha_v, t) + \alpha_v t + \delta(\alpha_x, \alpha_v, t) \tag{3.31}$$

$$v = \alpha_v + \dot{\delta}(\alpha_x, \alpha_v, t) \tag{3.32}$$

$$\frac{d\delta}{dt} = \dot{\delta}, \;\; \delta(\alpha_x, \alpha_v, 0) = 0. \tag{3.33}$$

Note that in deposition the velocity does not explicitly appear and thus the perturbation in the velocity, $\dot{\delta}$, does not explicitly appear in our analysis below.

From (3.24), deposition in (1+1)D can be written in a discrete form as

$$\mathcal{I}_x(\{f_k, x_k, v_k\}, \bar{x}) \equiv \sum_k f_k h_x h_v \frac{1}{\Delta x} W^{\Delta x}(\bar{x} - x_k) \tag{3.34}$$

where k indexes over particles that initially sampled the distribution f_0. The summation over k can be broken into two separate components, one where we sum over all the particles with $v(t=0) = \alpha_v$ fixed, and then sum in the velocity dimension. This allows us to write deposition as

$$\mathcal{I}_x(\{f_k, x_k, v_k\}, \bar{x}) = \sum_{j_v} \sum_{j_x} f_{j_x, j_v} h_x h_v \frac{1}{\Delta x} W^{\Delta x}(\bar{x} - x_{j_x}) \tag{3.35}$$

where for simplicity of notation we write $f_0(j_x h_x, j_v h_v) = f_{j_x, j_v}$. Throughout the remainder of the analysis we will write $f_0 = f$. We want to estimate the error,

$$\epsilon = \mathcal{I}_x(\{f_k, x_k, v_k\}, \bar{x}) - \int_{v_L}^{v_H} f(\bar{x}, v, t) dv \tag{3.36}$$

where we have already shown the transformation of the integral into the Lagrangian coordinates in (3.14).

Initially, suppose that we have one particle per cell ($h_x = \Delta x$). The discrete formulation of the coordinates for the perturbed streaming case are $x_k = x_0(\bar{x}, h_x, j_v, h_v, t) + j_x h_x + j_v h_v t + \delta(j_x h_x, j_v h_v, t)$. Here, x_0 is used to define the origin of the coordinate system as the nearest grid point to the translated $\bar{x}$ for each j_v distribution. This amounts to choosing

$$x_0(\bar{x}, h_x, j_v, h_v, t) = x\Big(\Big\lfloor \frac{\alpha_x(\bar{x} - j_v h_v t)}{h_x} \Big\rfloor h_x\Big), \tag{3.37}$$

which we can do without a loss of generality. The first expansion performed is $f(\cdot, v)$ around the origin,

$$\mathcal{I}_x = h_v \sum_{j_v} \sum_{j_x} \sum_{l=0}^{L-1} f^{(l,0)}(0, j_v h_v) \frac{(j_x h_x)^l}{l!} W\Big(\frac{\bar{x} - x_0 - j_v h_v t}{h_x} - j_x - \frac{\delta(j_x h_x, j_v h_v, t)}{h_x}\Big). \tag{3.38}$$

Next we expand W around the origin

$$W(j_x, j_v, t) = \sum_{m=0}^{M-1} \frac{d^m}{d\alpha_x^m} \Big(W(j_x, j_v, t)\Big)\Big|_{\alpha_x = 0} \frac{(j_x h_x)^m}{m!} + O\Big(\Big(h_x^M \frac{\partial^M}{\partial \alpha_x^M} W\Big)\Big) \tag{3.39}$$

and use the Faa di Bruno formula to express, the Nth derivative as

$$\frac{d^N}{d\alpha_x^N}\Big(W(j_x, j_v, t)\Big) = \sum_{n=1}^{N} \frac{1}{h_x^n} \frac{d^n W}{dz^n}\Big|_{\frac{\bar{x} - x_0 - j_v h_v t}{h_x} - j_x} \sum \frac{N!}{n_1! \ldots n_N!} \prod_{j=1}^{N} \Big(-\frac{1}{j!} \frac{d^j \delta}{d\alpha_x^j}\Big|_{\alpha_x = 0}\Big)^{n_j}. \tag{3.40}$$

For the case $n = N$, the inner sum has only one term, corresponding to the case $n_1 = N$, $n_j = 0$ for $j > 1$. From that we conclude

$$\frac{d^N}{d\alpha_x^N}\Big(W(j_x, j_v, t)\Big) = \frac{1}{h_x^N}\frac{d^N W}{dz^N}\Big|_{\frac{\bar{x}-x_0-j_v h_v t}{h_x}-j_x}\Big(-\frac{d\delta}{d\alpha_x}\Big|_{\alpha_x=0}\Big)^N + O((1/h_x)^{N-1}). \tag{3.41}$$

Substituting this into the full expression gives

$$\begin{aligned}\mathcal{I}_x = h_v \sum_{j_v}\sum_{j_x}\sum_{l=0}^{L-1}\sum_{m=0}^{M-1} f^{(l,0)}(0, j_v h_v)\frac{(j_x h_x)^{l+m}}{l!m!}\frac{d^m W}{dz^m}\Big|_{\frac{\bar{x}-x_0-j_v h_v t}{h_x}-j_x}\Big(-\frac{1}{h_x}\frac{d\delta}{d\alpha_x}\Big|_{\alpha_x=0}\Big)^m \\ + O(h_x) + O\Big(f(0, j_v h_v)\frac{d^M W}{dz^M}\Big(\frac{d\delta}{d\alpha_x}\Big)^M\Big)\end{aligned} \tag{3.42}$$

and can be simplified by rearranging the order of the sums,

$$\begin{aligned}\mathcal{I}_x = h_v \sum_{j_v}\sum_{l=0}^{L-1}\sum_{m=0}^{M-1} f^{(l,0)}(0, j_v h_v)\frac{h_x^l}{l!m!}\Big(-\frac{d\delta}{d\alpha_x}\Big)^m \sum_{j_x} j_x^{l+m}\frac{d^m W}{dz^m}\Big|_{\frac{\bar{x}-x_0-j_v h_v t}{h_x}-j_x} \\ + O(h_x) + O\Big(f(0, j_v h_v)\frac{d^M W}{dz^M}\Big(\frac{d\delta}{d\alpha_x}\Big)^M\Big),\end{aligned} \tag{3.43}$$

and recognizing the sum over j_x as the moment conditions for the kernels. Making this substitution further simplifies the expression

$$\begin{aligned}\mathcal{I}_x = h_v \sum_{j_v}\sum_{m=0}^{M-1}\sum_{l=0}^{L-1} f^{(l,0)}(0, j_v h_v)\frac{1}{l!}\Big(-\frac{d\delta}{d\alpha_x}\Big)^m\frac{(l+m)!}{l!m!}\Big(\bar{x}-x_0-j_v h_v t\Big)^l \\ + O(h_x) + O\Big(f(0, j_v h_v)\frac{d^M W}{dz^M}\Big(\frac{d\delta}{d\alpha_x}\Big)^M\Big)\end{aligned} \tag{3.44}$$

Recall that from the definition of x_0,

$$|\bar{x} - x_0 - j_v h_v t| \leq C h_x. \tag{3.45}$$

Using this, we see that for all terms with $l > 0$ the expression is at least $O(h_x)$ and thus can be written as

$$\mathcal{I}_x = h_v \sum_{j_v} f(0, j_v h_v)\sum_{m=0}^{M-1}\Big(-\frac{d\delta}{d\alpha_x}\Big)^m + O(h_x) + O\Big(f(0, j_v h_v)\frac{d^M W}{dz^M}\Big(\frac{d\delta}{d\alpha_x}\Big)^M\Big) \tag{3.46}$$

which further simplifies to

$$\begin{aligned}\mathcal{I}_x = h_v \sum_{j_v}\frac{f(\bar{x}-x_0-j_v h_v t, j_v h_v)}{1+\frac{d\delta}{d\alpha_x}} \\ + O\Big(f(\bar{x}-x_0-j_v h_v t, j_v h_v)\Big(\frac{d\delta}{d\alpha_x}\Big)^M\Big) \\ + O\Big(f(\bar{x}-x_0-j_v h_v t, j_v h_v)\frac{d^M W}{dz^M}\Big(\frac{d\delta}{d\alpha_x}\Big)^M\Big) + O(h_x).\end{aligned} \tag{3.47}$$

Note, the similarity to the error for the one-dimensional model problem given in (2.42). As already discussed, this is because each individual fixed α_v line is streaming in configuration space with there own perturbations and then are summed over to compute the charge density. Introducing the configuration space deformation gradient,

$$F_x = \frac{dx}{d\alpha_x}, \tag{3.48}$$

we can write this as

$$\begin{aligned}\mathcal{I}_x = h_v \sum_{j_v} &\frac{f(\bar{x} - x_0 - j_v h_v t, j_v h_v)}{F_x(\bar{x} - x_0 - \alpha_v t, \alpha_v, t)} \\ &+ O\Big(f(\bar{x} - x_0 - j_v h_v t, j_v h_v)\Big(F_x - 1\Big)^M \Big(1 + \frac{d^M W}{dz^M}\Big)\Big) + O(h_x).\end{aligned} \tag{3.49}$$

To estimate the error in the sums representation of the integral, we utilize the Euler-MacLaurin formula. First, for simplicity of notation define the integrand as,

$$\tilde{f}(x, \alpha_v, t) = \frac{f(x - \alpha_v t, \alpha_v)}{F_x(x - \alpha_v t, \alpha_v, t)}, \tag{3.50}$$

then we can rewrite the sum as

$$\begin{aligned}h_v \sum_{j_v} \tilde{f}(\bar{x} - x_0, j_v h_v, t) = \int_{v_L}^{v_H} &\tilde{f}(\bar{x} - x_0, \alpha_v, t) d\alpha_v \\ &+ \frac{h_v}{2}(\tilde{f}(\bar{x} - x_0, v_L, t) + \tilde{f}(\bar{x} - x_0, v_H, t)) \\ &+ \sum_{i=1}^{N} \frac{B_{2i}}{(2i)!} h_v^{2i} \Big(\frac{d^{2i-1}}{d\alpha_v^{2i-1}} \tilde{f}(\bar{x} - x_0, \alpha_v, t)\Big)\Big|_{v_L}^{v_H} \\ &+ \frac{h_v^{2N+2}}{(2N+2)!} \int_{v_L}^{v_H} \bar{B}_{2N+2}\Big(\frac{\alpha_v - v_L}{h_v}\Big) \frac{d^{2N+2}}{d\alpha_v^{2N+2}} \tilde{f}(\bar{x} - x_0, \alpha_v, t) d\alpha_v\end{aligned} \tag{3.51}$$

where it is assumed that $\tilde{f} \in C^{2N+2}$, B_{2i} are Bernoulli numbers, and $\bar{B}_i(x)$ is the periodic extension of the Bernoulli polynomial $B_i(x)$ [5]. For typical kinetics problems $f(\alpha_x, \alpha_v) \in C^\infty$ and has tails that decrease exponentially in the velocity dimension. Therefore,

$$h_v \sum_{j_v} \tilde{f}(\bar{x} - x_0, \alpha_v, t) = \int_{v_L}^{v_H} \tilde{f}(\bar{x} - x_0, \alpha_v, t) d\alpha_v + O(h_v^p) \tag{3.52}$$

for arbitrarily large values of p. Finally, combining the results of the application of the Euler-Maclaurin with the rest of our analysis gives an approximation of

$$\epsilon = O\Big(\Big(F_x(\alpha_x, \alpha_v, t) - 1\Big)^M f(\alpha_x, \alpha_v)\Big(1 + \frac{d^M W}{dz^M}\Big)\Big) + O(h_v^p) + O(h_x). \tag{3.53}$$

Using the same transformation as in Section 2.2 for multiple particles per cell, one can show that this error estimate holds for the case more frequently encountered in PIC where $\Delta x/h_x \in \mathbb{N}^+$.

The first term is an $O(1)$ error that will have a similar effect on the convergence of the PIC method as the barrier error we encountered in Chapter 2 for the one-dimensional advection model problem. Unlike the model problem, we do not get to specify this quantity exactly for $(1+1)D$ kinetics test problems. Instead, this error is determined by the dynamics of the individual test problems. This means that instead of appearing to dominate after a certain resolution, if this error grows sufficiently large, we expect it to eventually dominate the grid based errors for all resolutions. We expect to see the effect on the finest level first but eventually for this error to dominate all levels and appear independent of the resolution.

3.4 Numerical Results

Typically, Landau damping and the two-stream instability are used to test the validity and accuracy of (1+1)D kinetics code. Here we consider both of these and use them to benchmark the accuracy of any method to control the $O(1)$ error. The initial distribution for linear Landau damping is

$$f(x,v) = \frac{1}{2\pi} e^{-v^2/2} \left(1 + \alpha \cos(kx)\right)$$

where $x \in [0, L]$, $v \in [-v_{\max}, v_{\max}]$, $L = 2\pi/k$, with $k = 1/2$ and $\alpha = 0.01$. Typically for (1+1)D tests, $v_{\max}$ is chosen such that the velocity space distribution is small at the boundary. A sufficient choice for the problems we consider is $v_{\max} = 10$. Analytic results show that the norm of the electric field is damped at rate $\gamma = 0.1533$ and oscillates with a frequency $\omega = 1.416$ [37]. Landau damping suffers from a recurrence phenomena that was first discussed in [11]. This is a consequence of fine scale structures developing that are smaller than the h_v resolution. The time that the solution is expected to degrade due to the fine scale structures overwhelming the resolution is

$$T_{\text{rec}} = \frac{\pi}{k h_v}.$$

For the resolutions considered in our convergence studies, this phenomena will begin to dominate around $t \sim 20, 40, 80$, respectively.

Using the same domain parameters and perturbation as above, the initial distribution for the two-stream instability is

$$f(x,v) = \frac{1}{2\pi} v^2 e^{-v^2/2} \left(1 + \alpha \cos(kx)\right).$$

This change in the velocity space distribution leads this problem to have small scale features in the phase-space distribution that start to appear after $t = 18$ and typically get smaller, and more difficult to resolve, for the remainder of the simulation. At $t = 20$ a phase-space vortex begins to rotate and the image of the phase-space distribution at this time is often reported in literature.

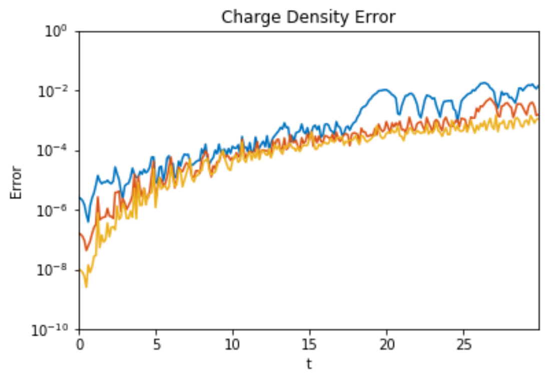
To compute the error for different grid sizes, we used Richardson extrapolation,

$$e = \max \left| \rho^{M_g} - \rho^{M_g+1} \right| \tag{3.54}$$

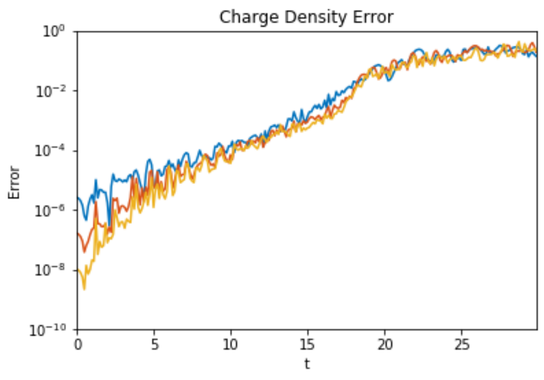
where the grid parameters for level M_g are $N_x = 2^{M_g}$, $N_x^p = 2^{M_g+1}$, $N_v^p = 2^{M_g+2}$, and $dt = 1/2^{M_g-1}$. For all convergence studies we used $M_g \in \{4, 5, 6, 7\}$ and for plots where we are changing values of C we use $M_g \in \{6, 7\}$ to compute the error in the charge density of the $M_g = 6$ level.

For all simulations presented here, we utilize a fourth-order Runge-Kutta scheme to advance the particles, a spectral field solver to obtain the electric potential from the charge density, and a fourth-order finite difference scheme to compute the electric field from the potential. A single kernel will be used for deposition and field interpolation.

In Section 3.3 we showed that the error in the deposition is dependent on the deformation gradient. Figure 3.1 and Figure 3.2 plot the results of computing the error for linear Landau damping and the two-stream instability. These plots show that for both $W_{2,2}$ and $W_{4,4}$ that for a short time the method is converging but that eventually there is a collapse of all the computed errors onto each other. This is indicative of an error that is independent of the mesh spacing, like the $O(1)$ error in (3.53). The plot of the norm of the electric field for linear Landau damping is plotted in Figure 3.3 for $M_g = 6$ with the kernel $W_{4,4}$. The accumulation of the $O(1)$ error in the charge density eventually leads to noise appearing in the solution around $t \sim 30$ and eventually causes the numerical solution to no longer be damped but rather begin to grow by $t \sim 35$. From the theory, we expect the numerical solution to capture the correct damping rate until $t \sim 80$.

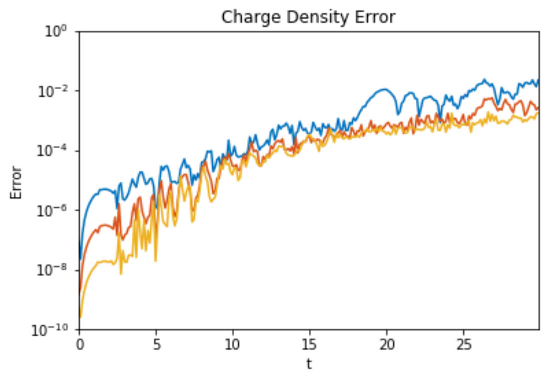


(a)

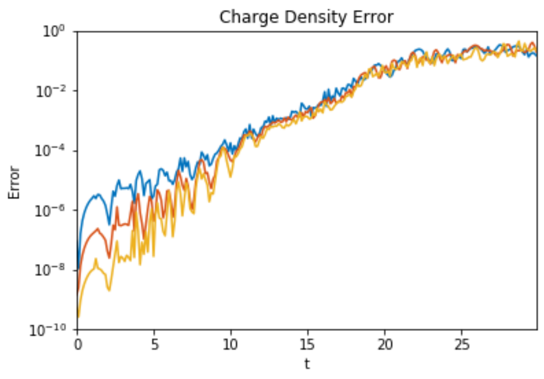


(b)

Figure 3.1: These plots show what happens when the $O(1)$ error is allowed to grow uncontrolled for the linear Landau damping (a) and two-stream instability (b) test problems. $W_{2,2}$ was used for deposition and force interpolation. Three different resolutions are used: $M_g = 4$(blue), $M_g = 5$(red), and $M_g = 6$(yellow).

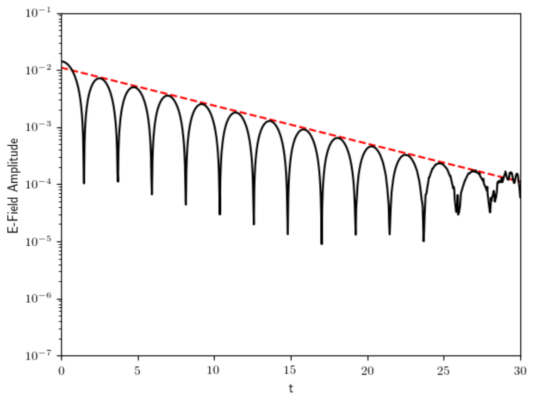


(a)

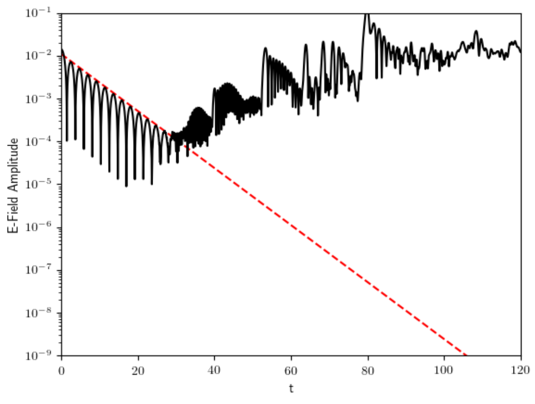


(b)

Figure 3.2: Similar to Figure 3.1, these plots show what happens when the $O(1)$ error is allowed to grow uncontrolled for the linear Landau damping (a) and two-stream instability (b) test problems. $W_{4,4}$ was used for deposition and force interpolation. Three different resolutions are used: $M_g = 4$(blue), $M_g = 5$(red), and $M_g = 6$(yellow). Note, that for early times we see the onset of the noise and collapse of the order of accuracy later for $W_{4,4}$ than for $W_{2,2}$ as is expected from the theory above.



(a) $W_{2,2}$



(b) $W_{4,4}$

Figure 3.3: Plot of the norm of the electric field (black), the analytic damping rate of 0.1533 (red dashed) for linear Landau damping with $M_g = 6$ and the kernel $W_{4,4}$. In (a) we see that around $t \sim 30$ the solution begins to show spurious artifacts. This continues for $t > 30$, as seen in (b), and begins to effect the damping of the problem that is expected to continue to at least $t \sim 80$.

Chapter 4

Adaptive Remapping for (1+1)D Kinetics

4.1 Computation of the Deformation Gradient

To be able to control the $O(1)$ error, we first need a way of computing the error. Initially, it may seem like the easiest way to compute the configuration space deformation gradient,

$$F_x = \frac{\partial x}{\partial \alpha_x}, \tag{4.1}$$

would be to keep track of neighboring particles relative positions. This method would be expensive and is not something that is available in PIC algorithms. Instead, let us introduce another variable,

$$G_x = \frac{\partial v}{\partial \alpha_x}. \tag{4.2}$$

Taking time derivatives of these variables gives us the system of equations

$$\frac{\partial F_x}{\partial t} = G_x \tag{4.3}$$

$$\frac{\partial G_x}{\partial t} = -\frac{dE}{dx} F_x \tag{4.4}$$

that can be evolved with the following initial conditions

$$F_x(\alpha_x, \alpha_v, t = 0) = 1 \tag{4.5}$$

$$G_x(\alpha_x, \alpha_v, t = 0) = 0. \tag{4.6}$$

$$\tag{4.7}$$

This system can be discretized so that the pair of variables $\{F_x, G_x\}$ can be computed and evolved for each of the particles. Furthermore, this system of equations generalizes to higher dimensions and could be used to track the deformation gradient in any (N_x+N_v)D electrostatic kinetics problem.

One more cost saving aside is to recognize that we are simply using this to approximate the deformation. These quantities do not feed into the PIC algorithm presented in Section 3.2 and thus can be computed using low-order methods. Using the electric field coming out of the first stage of the algorithm, we can evolve this system using a first-order forward Euler integrator.

4.2 Remapping

Remapping is a means of projecting the distribution onto a rectangular grid in phase-space where particles then begin from on the next time-step. For the pth particle on the rectangular phase-space grid, the remapped charge, M_p^*, can be computed by

$$M_p^* = \sum_s M_s W_{q,r}\left(\frac{\boldsymbol{x}_s - \boldsymbol{x}_p}{h_x}\right) W_{q,r}\left(\frac{\boldsymbol{v}_s - \boldsymbol{v}_p}{h_v}\right) \tag{4.8}$$

where s indexes the original particle set and $W_{q,r}$ is a kernel similar to those used for deposition and interpolation. For forward semi-Lagrangian methods, the order and smoothness of the remapping kernel dictates the order of the method.

Unlike the other operations in a PIC algorithm remapping is a phase-space operations. This means that the work per particles increases by a factor of $\mathrm{Supp}(W_{q,r})^{N_v}$ compared to deposition and interpolation. This makes it the most expensive part of the algorithm and why, in practice, forward semi-Lagrangian methods do not remap every time step. Another factor contributing to the expense of remapping in forward semi-Lagrangian methods is the fact that if remapping occurs on the order of Δt, the remapping kernel $W_{q,r}$ must be at least one degree higher order than the depositions kernel since we lose a degree of accuracy by remapping at every time-step. In the next section we will provide a metric to determine when remapping is required based on the error analysis in Chapter 3. This metric is independent of Δt and therefore we propose using a kernel of the same order and smoothness as the deposition and force interpolation kernel.

Now that there is a means of computing the $O(1)$ error for any (N_x+N_v)D electrostatic kinetics test problem, we need to control the error once it approaches a level that is deemed unacceptable. As we have already discussed, adding more particles will not decrease the error. The only way to reduce the error is to sample the current distribution and restart the simulation on a rectangular phase-space grid. This can be done via remapping. This reduces the $O(1)$ error since for a rectangular grid $F_x \equiv 1$. This operation is something already done by forward semi-Lagrangian methods and is typically referred to as remapping. Remapping has traditionally been used in particle methods as a means to regularize the distribution and reduce the "particle noise" observed in particle methods; however, when remapping is applied is typically not based on the numerical analysis. Remapping is typically done every few time-steps but can be done as often as every time-step [21, 45, 37].

4.3 Adaptive-in-time Remapping

We have demonstrated a way to compute the $O(1)$ error that arises in (3.53) and have provided a way of reducing the error once it reaches an unacceptable level. Similar to adaptive Runge-Kutta methods, we can apply these tools to control the error. The analog to decreasing the size of the time-step for PIC methods is to remap the particles when the $O(1)$ error is greater than some threshold value, C. In practice, the bound we use throughout is

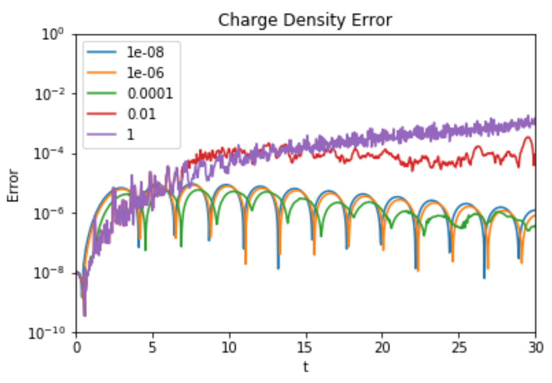
$$\Big(F_x(\alpha_x, \alpha_v, t) - 1\Big)^M f(\alpha_x, \alpha_v) < C \tag{4.9}$$

since $d^M W/dz^M$ is bounded as long as the number of derivatives does not exceed the smoothness of the kernel. That is to say that the number of moment conditions and the smoothness of the kernel match. For all results in this section, we will use $W_{2,2}$ and $W_{4,4}$ listed in Section 2.2. For these kernels, $M = 2$ or $M = 4$ in (4.9) respectively. In Section 4.6, we examine kernels with mixed orders of moment conservation and smoothness as well as examine using one kernel for deposition and force interpolation and another for remapping.

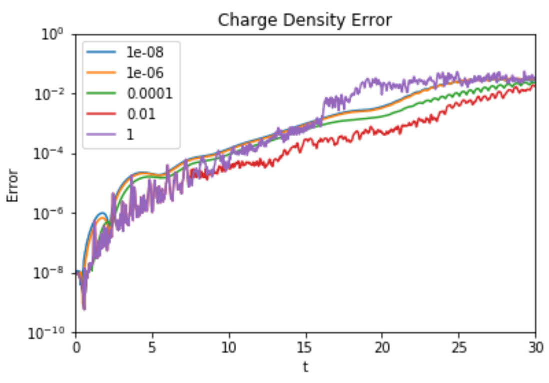
For all simulations presented here, we utilize the fourth-order Runge-Kutta scheme to advance the particles, a spectral field solver to obtain the electric potential from the charge density, and a fourth-order finite difference scheme to compute the electric field from the potential. The same kernel will be used for deposition, field interpolation, and remapping. We utilize the deposition kernel for remapping since for a fixed value of C the number of remaps is independent of Δt. Deposition, force interpolation, and the field solve are discussed further in Section 3.2. We will use the linear Landau damping and two-stream instability test problem introduced at the end of Chapter 3 to benchmark this method. We will focus on the error in the charge density through out this section. We compute this error by Richardson extrapolation.

To highlight the difference that changing C makes on the error, we plot the charge density error, which is computed using (3.54), for $M_g = 6$ in Figure 4.1a. Note that for $C = 1$ the error saturates at the same value as if no remapping was done. This is due to the fact that the $F_x - 1$ term saturates and does not continue to grow. For $C = 1$ no remappings were triggered for the duration of the simulation and thus the error is identical to the finest resolution plotted in Figure 3.1a. For $C < 10^{-4}$ the discretization errors dominate the $O(1)$ error and thus there is no marked improvement in the error for these tighter bounds.

Similar to the second-order kernel, the control of the $O(1)$ error directly influences the errors at late times for the $W_{4,4}$ kernel as seen in Figure 4.2a. Since the error inherent to the method is lower by approximately two orders of magnitude, we expand the list of bounds tested over that tested for $W_{2,2}$. Although the error for $C = 10^{-6}$ is approximately the same as those for smaller values of C, we do see some noise and a structurally different error. Therefore, for most simulations with $W_{4,4}$ we utilize $C = 10^{-8}$ as this appears to be approximately the error from the convergent portions of the method and there do not seem to be large improvements for smaller values of C. The number of remaps for each

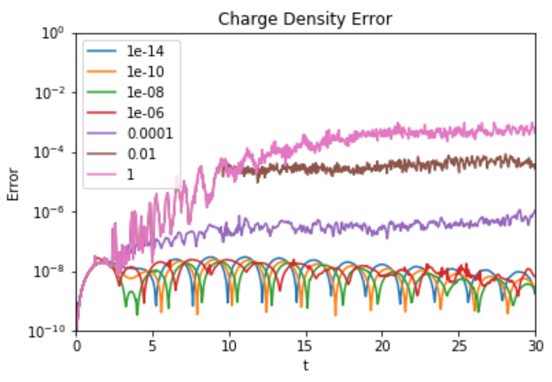


(a)

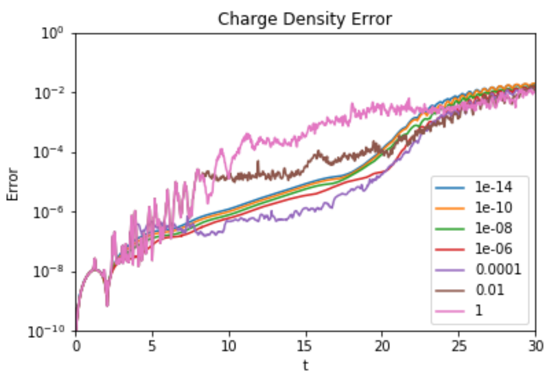


(b)

Figure 4.1: Using the initial conditions for linear Landau damping, (a) shows the error in the charge density for different values of C where $f|F_x - 1|^2 < C$ for the $W_{2,2}$ kernel. These plotted results are for the finest level of resolution. The equivalent plot using two-stream instability initial conditions is shown in (b).



(a)



(b)

Figure 4.2: Using the initial conditions for linear Landau damping, (a) shows the error in the charge density for different values of C where $f|F_x - 1|^4 < C$ for the $W_{4,4}$ kernel. These plotted results are for the finest level of resolution. Given that the initial error for the finest level is $\sim 10^{-8}$, this is the largest value of C that would likely be considered acceptable. The equivalent plot using two-stream instability initial conditions is shown in (b). Note that even though the initial error starts at 10^{-8} like linear Landau damping, the growth in the error for the two-stream instability means that using a smaller C does not improve the error at the end of the test beyond the results for $C = 0.01$. The results are clearly improved for early times with the results for $C \leq 10^{-8}$ looking nearly identical.

Number of Remaps for Linear Landau Damping		
C	$W_{2,2}$	$W_{4,4}$
1	0	1
10^{-2}	1	1
10^{-4}	7	3
10^{-6}	24	6
10^{-8}	72	11
10^{-10}	-	18
10^{-14}	-	57

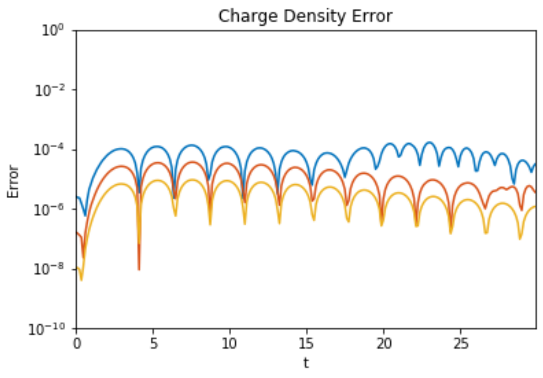
Table 4.1: The number of remaps for linear Landau damping with differing values of C. For all simulations, the total number of time steps was 960. Remapping every 5 time steps would require a total of 192 remaps as done in [45, 37]

value of C are listed in Table 4.1. For each of these simulations, the number of remaps is listed for $M_g = 6$ and thus the total number of time steps taken is 960. For the smallest value of C, the number of remaps required is reduced by a factor of approximately 2.7 and 3.6 compared to remapping every 5 time steps for $W_{2,2}$ and $W_{4,4}$, respectively. For the $W_{4,4}$ kernel using $C = 10^{-8}$ requires a factor of 17.5 fewer remaps than remapping every 5 time steps. This is a significant reduction in the number of remaps and clearly shows the performance opportunities available by tracking the deformation gradient and remapping only when needed.

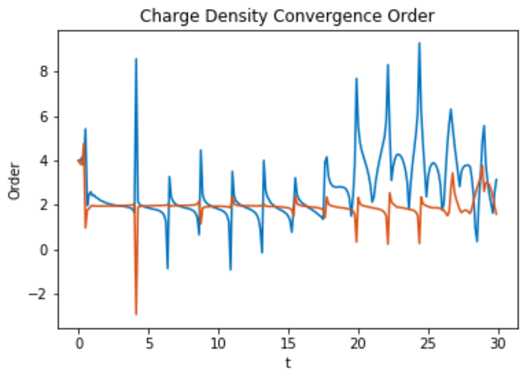
Similar results are shown for the two-stream instability for $W_{2,2}$ in Figure 4.1b and for $W_{4,4}$ in Figure 4.2b. Note though that the $F_x - 1$ term does not saturate at late times and instead continues to grow, unlike linear Landau damping. This means that even the largest value of C does eventually require a remap. For these large values of C, after a short initial start up time, the error becomes dominated by the $O(1)$ term causing noisy solutions and errors. Eventually though, the grid errors overwhelm this bound and become the dominant term. Thus, the error appears to end independent of the initial value of C.

For the two-stream instability, the number of remaps for $W_{2,2}$ and $W_{4,4}$ for different values of C are listed in Table 4.2. Due to the deformations in the phase-space distribution after $T = 18$, we see that significantly more remaps are required than for linear Landau damping. For the finest levels, the number of remaps required increased by a factor of 1.8 and 1.7 for $W_{2,2}$ and $W_{4,4}$ respectively when compared to remapping every 5 time steps. For $W_{4,4}$ with $C = 10^{-8}$, we do still see a reduction in the number of remaps by a factor of 1.9. One thing to note about remapping based on the $O(1)$ error is that the distribution of remaps is not uniform. For linear Landau damping, many of the remaps come early on in the simulation and for the two-stream instability the majority of remaps come much later in the simulation.

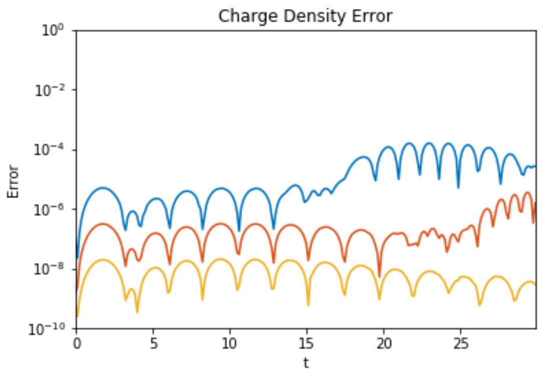
Figure 4.3 and 4.4 examines how the second- and fourth-order kernels converge with a bounded $O(1)$ error for linear Landau damping and the two-stream instability, respectively. For these plots we set $C = 10^{-8}$ for both kernels. As discussed above, this is stricter



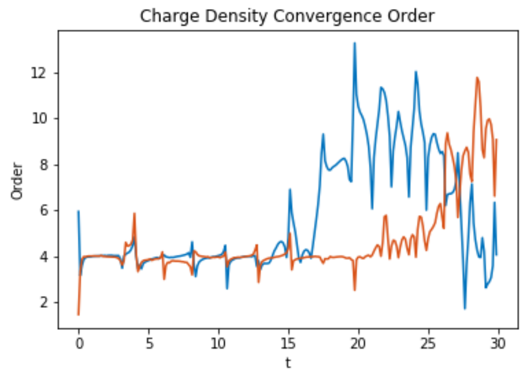
(a) $W_{2,2}$



(b) $W_{2,2}$

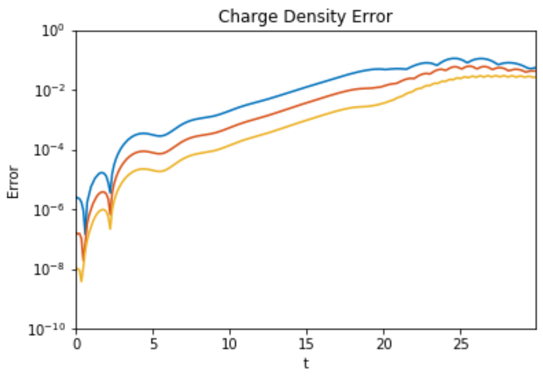


(c) $W_{4,4}$

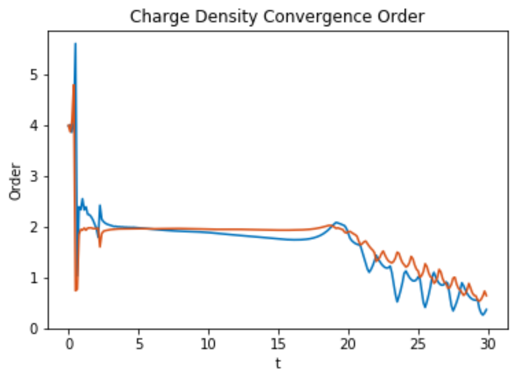


(d) $W_{4,4}$

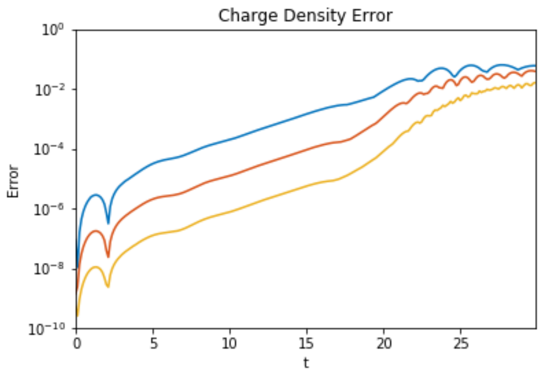
Figure 4.3: Plots of error and convergence order for both $W_{2,2}$ and $W_{4,4}$ for the linear Landau damping test problem. (a) shows the error for three different resolutions: $M_g = 4$(blue), $M_g = 5$(red), and $M_g = 6$(yellow). (b) shows the order of convergence with $\log_2(e^4/e^5)$ in blue and $\log_2(e^5/e^6)$ in red. For all tests we bound the $O(1)$ error by 10^{-8}. As is expected, for the second-order kernel we see second-order convergence and for the fourth-order kernel we see fourth-order convergence. At late times the error increases for the coarsest grids due to the recurrence phenomenon.



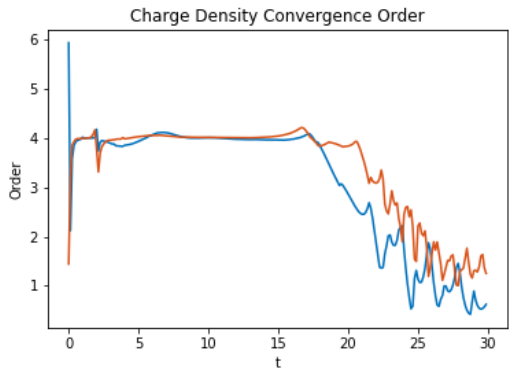
(a) $W_{2,2}$



(b) $W_{2,2}$



(c) $W_{4,4}$



(d) $W_{4,4}$

Figure 4.4: Plots of error and convergence order for both $W_{2,2}$ and $W_{4,4}$ for the two-stream instability test problem. (a) shows the error for three different resolutions: $M_g = 4$(blue), $M_g = 5$(red), and $M_g = 6$(yellow). (b) shows the order of convergence with $\log_2(e^4/e^5)$ in blue and $\log_2(e^5/e^6)$ in red. For all tests we bound the $O(1)$ error by 10^{-8}. As is expected, for the second-order kernel we see second-order convergence and for the fourth-order kernel we see fourth-order convergence. At late times the error increases for both methods and the methods convergence order decreases. This is due to the filamentation in the velocity dimension and has been previously reported in [45].

Number of Remaps for Two-Stream Instability		
C	$W_{2,2}$	$W_{4,4}$
1	7	11
10^{-2}	24	20
10^{-4}	73	35
10^{-6}	192	60
10^{-8}	352	100
10^{-10}	-	156
10^{-14}	-	331

Table 4.2: The number of remaps for the two-stream instability with differing values of C. As for linear Landau damping, all simulations required 960 time steps and remapping every 5 time steps would require a total of 192 remaps.

than what is necessary for $W_{2,2}$. For both test problems and kernels we see the expected convergence behavior since we remap to keep the barrier error below the error of the finest level. For linear Landau damping, there is an increase in the order of convergence caused by the coarsest level encountering the recurrence phenomena inherent to any method discretizing the velocity dimension of this problem. For the two-stream instability we have a decrease in accuracy at late times due to similar filamentation issues in the velocity dimension.

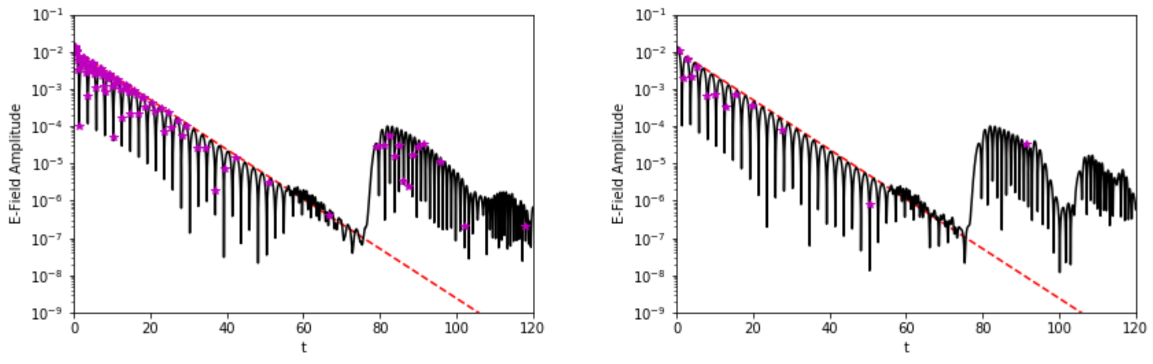
Number of Remaps				
	Linear Landau Damping		Two-Stream Instability	
M_g	$W_{2,2}$	$W_{4,4}$	$W_{2,2}$	$W_{4,4}$
4	55	10	114	72
5	63	11	204	92
6	72	11	352	100
7	78	11	565	108

Table 4.3: The number of remaps for the linear Landau damping and two-stream instability for $C = 10^{-8}$.

Table 4.3 provides the number of remaps for each level of refinement in the convergence study. For $W_{4,4}$ we see that for linear Landau damping the number of remaps is almost constant for different levels of refinement and for the two-stream instability that it is converging. The second-order kernel shows convergent behaviour in the number of remaps for linear Landau damping while it grows rapidly for the two-stream instability. This is a time where it would be beneficial to choose a bound closer to $C = 10^{-4}$ or $C = 10^{-6}$ for $W_{2,2}$ as the current condition appears to be too strict and forces remapping at nearly every time-step at late times.

This highlights the somewhat strange relationship of the order of the kernel and the remapping condition. Since $F_x - 1 < 1$, the higher the order and smoothness of the kernel used the fewer remaps that are required. This means that as one pushes the order and smoothness of the kernel higher, which corresponds to increasing M in (4.9), not only will the accuracy increase, but the number of remaps necessary will actually decrease for a constant C. We see this phenomena in Table 4.3. For a given level, independent of the test problem, $W_{4,4}$ requires fewer remaps than $W_{2,2}$. Furthermore, if one fixes a level of accuracy there is even a larger disparity in the number of remaps required for $W_{2,2}$ when compared to $W_{4,4}$. For example, the error in $M_g = 6$ for $W_{2,2}$ and in $M_g = 4$ for $W_{4,4}$ are approximately equal. For this fixed level of accuracy, $W_{4,4}$ requires 7.2 and 4.9 times fewer remaps for linear Landau damping and two-stream instability, respectively, when compared to $W_{2,2}$. Granted, for $C = 10^{-4}$ we see a much closer alignment in number of remaps required for $W_{2,2}$ compared to $W_{4,4}$ for a similar level of accuracy; however, a significant difference in the number of particles. The tradeoff is a larger stencil size for deposition, force interpolation, and remapping. Given that particle methods are memory-bound, this is a trade-off worth making since we do achieve increased accuracy for the increased work per particle.

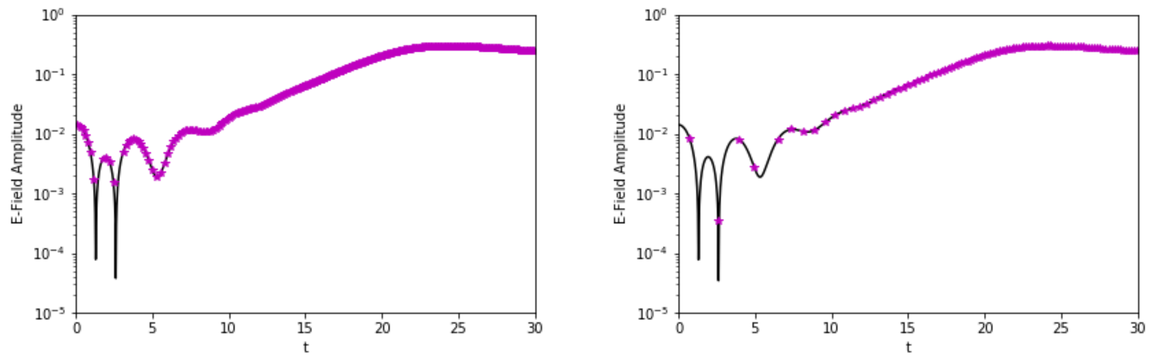
Having a higher order method that converges with the expected order is only useful if it converges to the correct solution. For linear Landau damping, we have a few different analytic predictions to check our method against. Figure 4.5 shows the analytic damping



(a) $W_{2,2}$ (b) $W_{4,4}$

Figure 4.5: Plot of the norm of the electric field (black), the analytic damping rate of 0.1533 (red dashed), and the remap times (magenta stars) for linear Landau damping with $C = 10^{-8}$. There is a recurrence effect at $t \sim 80$ as predicted.

rate (red dashed line) and the amplitude of the electric field (solid black line) for $M_g = 6$ and $t \in [0, 120]$. The magenta stars indicate when remaps occur and, as previously mentioned, they tend to occur early in the simulation for linear Landau damping. For our parameters, the analytic damping rate is $\gamma = 0.1533$ [37]. We see that both $W_{2,2}$ and $W_{4,4}$ correctly capture this damping rate until the recurrence phenomenon at $t \sim 80$ in Figure 4.5. Using the interval $t \in [0, 55]$ the computed damping rate is 0.1543 for $W_{2,2}$ and 0.1539 for $W_{4,4}$, which correspond to a relative error of 0.700% and 0.447% respectively. The analytic frequency for this set of parameters is 1.416 [37]. Using the same time interval, the computed frequency of the damped electric field in the same time interval is 1.410 for $W_{2,2}$ and 1.409 for $W_{4,4}$. These correspond to relative errors of 0.371% and 0.432%. Finally, we see that both tests have recurrence phenomenon of $t \sim 80$ which matches well with the theoretical results.



(a) $W_{2,2}$ (b) $W_{4,4}$

Figure 4.6: Plot of the norm of the electric field (black) and the remap times (magenta stars) for the two-stream instability with $C = 10^{-8}$. Note that at late times there are significantly more remaps. This is a consequence of the phase-space distribution getting distorted, which leads to large configuration space deformation gradients.

Figure 4.6 is a plot of the amplitude of the electric field for the two-stream instability. Contrasting this with Figure 4.5, it is easy to see how the distribution of remaps is heavily weighted toward late in the simulation for two-stream instability. Figure 4.7 is a plot of the phase-space distribution at $t = 20$ for the two different kernels. The remapped distributions are in good agreement with previous work [45, 37]. Figure 4.8 shows the charge density at $t = 20$ for the phase-space distributions in Figure 4.7. The accumulated noise is evident in Figure 4.8a when compared to the two remapped charge densities. The non-uniform distribution of this noise leads to similar looking errors in the electric field which then feeds back into the particles position and velocity.

The benefits of bounding the $O(1)$ error term from Section 3.3 are obvious. We see that different bounds lead to different total errors until a floor is reached where the dominant portion of the error comes from the time step, the phase-space discretization, and the configuration space discretization. We showed that bounding this error leads to a method that converges to a solution that is in good agreement with the analytic values and converges at the order predicted by the kernels used. Next, we investigate how remapping, accuracy,

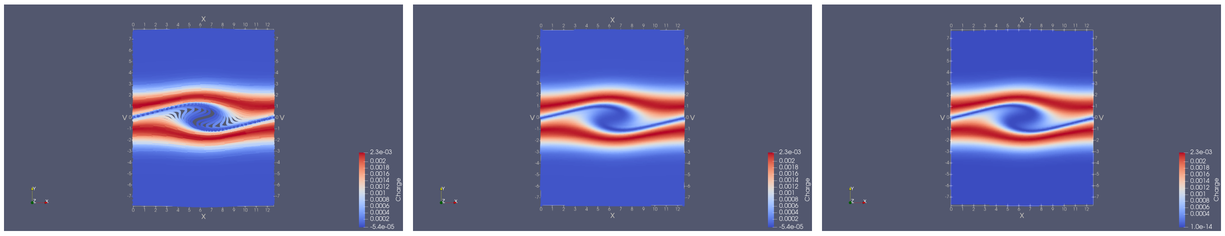

(a) $W_{4,4}, C = \infty$ (b) $W_{2,2}, C = 10^{-8}$ (c) $W_{4,4}, C = 10^{-8}$

Figure 4.7: Plot of phase-space distribution at time t=20 for the two-stream instability test problem. All tests use grid parameters and time step parameters with $M_g = 6$. The large deformations in phase-space are noticeable in (a). This leads to difficulty correctly resolving fine features as well as large errors in the charge density, which can be seen in Figure 4.8

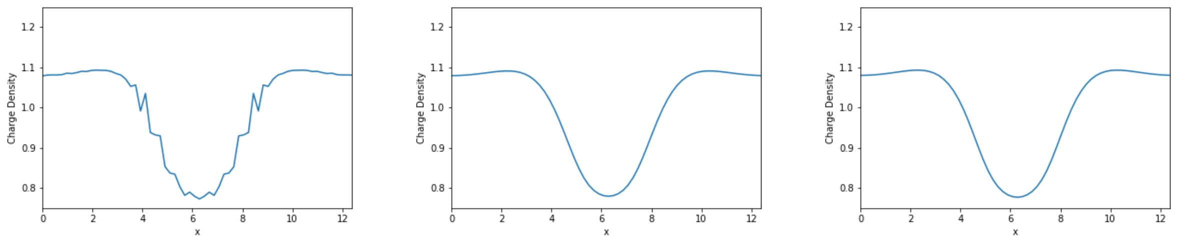
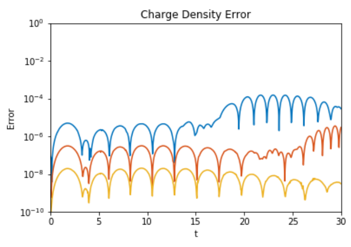

(a) $W_{4,4}, C = \infty$ (b) $W_{2,2}, C = 10^{-8}$ (c) $W_{4,4}, C = 10^{-8}$

Figure 4.8: Plot of charge density, ρ, at time t=20 for the two-stream instability test problem. All tests use grid parameters and time step parameters with $M_g = 6$. The noise is not uniformly distributed in the non-remapped charge density. This leads to noisy solutions to Poissons equation that then feed back into the positions and velocities of the particles. This accumulation of noise causes the disastrous losses of accuracy that we see in Figure 3.2b.

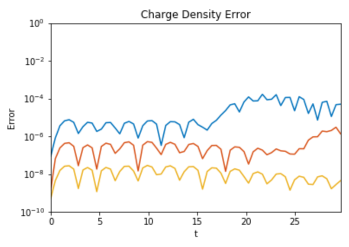
and order of convergence are effected by using both large and small time-steps. We then show that remapping on an implicit function of the $O(1)$ term and the distribution allows us to remap adaptively in space as well. Finally, we will discuss how mixed kernels - either kernels that have differing degrees of smoothness for a given number of moments or using different kernels for remapping and deposition - effect the value of M in (4.9) and the order of convergence.

4.4 Adaptive-in-time with Varying CFL Conditions

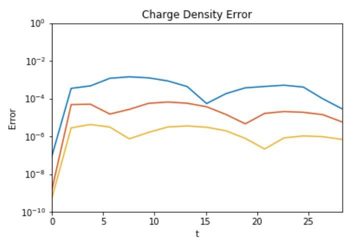
Thus far we have only considered time steps relative to the grid size of $\Delta t = 1/2^{M_g - 1}$. One significant advantage that PIC methods have over traditional grid based methods is that the



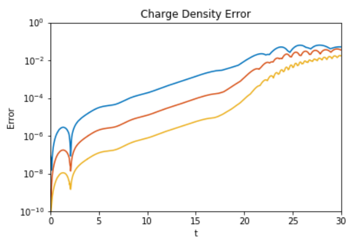
(a) CFL= 3/4



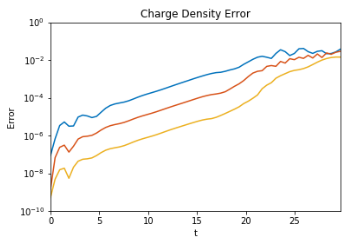
(b) CFL= 12



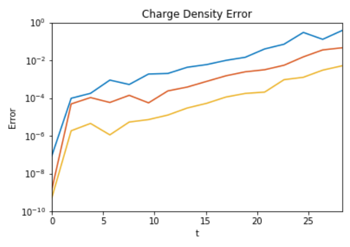
(c) CFL= 48



(d) CFL= 3/4



(e) CFL= 12



(f) CFL= 48

Figure 4.9: Plots of the error for different CFL conditions for three different resoulutions: $M_g = 4$(blue), $M_g = 5$(red), and $M_g = 6$(yellow). Plots in the top row depict the computed errors for linear Landau damping. Plots in the bottom row are for two-stream instability. The error for a CFL condition less than 12 have approximately the same initial and final errors for all levels. For a CFL of 48 there is an increase in the error of all levels due to the $O(\Delta t)$ error increase.

Courant-Fredrichs-Lewy (CFL) condition, which is defined as

$$C_{\text{CFL}} = \frac{v\Delta t}{h_x}, \tag{4.10}$$

can be larger than 1 and typically the time-step restriction is a function of the strain in the velocity field and has no relation to the configuration or velocity space grid. It is worth noting that this restriction on the time-step for Lagrangian particle methods is an accuracy condition and not a stability condition like the CFL condition is for grid methods. Many forward semi-Lagrangian methods utilize CFLs significantly larger than one. To show that the above works well with larger CFLs, we compare the results for CFLs of 3/4, 12, and 48 with our standard time step, which equates to a CFL of $10/\pi$.

Figure 4.9 displays error plots for the three different CFL conditions that can be compared to Figure 4.3c and Figure 4.4c. For each CFL and test problem we use the grid and domain parameters given previously with $M_g \in \{4, 5, 6, 7\}$. For all tests we use $W_{4,4}$ as the deposition, interpolation, and remapping kernel as well as the associated remapping condition

$$f\left(F_x - 1\right)^4 < 10^{-8}. \tag{4.11}$$

CFLs of 3/4, $10/\pi$, and 12 have no noticeable increase in the error due to the differing time-step sizes for either the linear Landau damping test problem or the two-stream instability test problem. Increasing the CFL further to 48 does noticeably increase the error. This is due to the fact that the time-stepping error is now dominant. Figure 4.10 highlights that

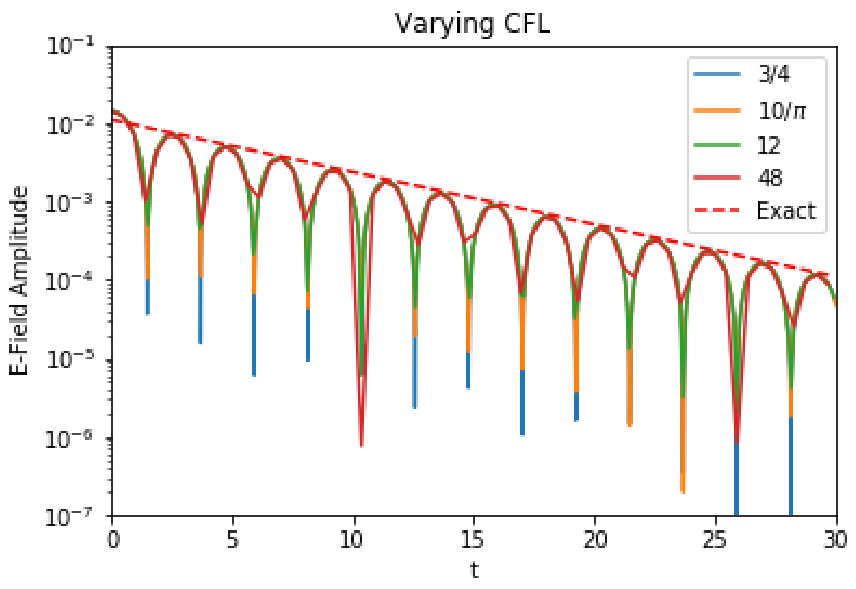


Figure 4.10: All CFL conditions correctly capture the analytic damping rate. The larger the CFL the more sub-sampled the solution is; however, the norm of the electric field is consistent across the CFL conditions.

all three CFLs have the same damping rate and, furthermore, the larger CFLs appear to be a sub-sampled version of smallest CFL.

The number of remaps for each CFL for $M_g = 6$ are listed in Table 4.4. Similar to how the time step restrictions are a function of the dynamics of the problem and not related to the grid size, the deformation gradient does not depend on the parameters used to discretize the computational domain. This leads one to expect that the number of remaps should be relatively consistent between CFLs. This holds true for linear Landau damping which has a variation of two remaps between the smallest and largest CFL condition. For the two-stream instability, the number of remaps does make large changes for different CFLs. This is due to the fact that there are many fewer time steps as the CFL increases and that many of the remaps occur for $t \in [20, 30]$. For the largest CFL we only have 64 total time steps, which is less than even the next closest CFLs total number of remaps. Thus, we cannot expect agreement in the number of remaps between all the CFLs for this problem.

CFL Comparison			
CFL	Number of Time Steps	Number of Remaps	
		Linear Landau Damping	Two-Stream Instability
$3/4$	4076	11	114
$10/\pi$	960	11	100
12	256	10	77
48	64	9	27

Table 4.4: The number of time steps and remaps for the two different test problems and four CFL conditions for the resolution $M_g = 6$.

4.5 Adaptive-in-time-and-space (Local) Remapping

In Section 4.1 we demonstrated that for each particle we could compute the contribution to the $O(1)$ error. We then presented remapping as a way of controlling this. The remapping method that we presented remaps the entire phase-space. This does not take full advantage of the locality of the computation of the error metric. If we just apply the remapping algorithm given in (4.8) to the regions that have a larger $O(1)$ error than the tolerance, we will be left with a distribution with discontinuities along the boundary of the region that degrades the accuracy of the method. We propose smoothly connecting these regions by using a compactly supported mollifier. The pseudocode for the implementation is as follows,

Algorithm 1 Local Remapping

```
Input:
Particles

Boxes ← getBoxes(Particles)                ▷ Defines regions of phase-space to remap

for Box ∈ Boxes do

    transition particles ← (1 - $\chi$(Box))(Particles ∉ Box)
    remapped particles ← remap($\chi$(Box)(Particles ∈ Box) )

    Particles ← remapped particles $\bigcup$ transition particles
end for
return Particles
```

where the remap function computes M_p^*, the remapped particle charge, as described in Section 4.2 and $\chi(x)$ is our mollifier function. Here the order of operations is important to the smooth transition from the old to the remapped portions of the distribution.

To understand the constraints on the mollifier, we follow the error analysis for global remapping in [45, 20]. Clearly, the error outside of the remapped rectangle will not be disturbed by the remapping and thus has no error associated with remapping. A pictorial representation of local remapping can be seen in Figure 4.11; the black box is the phase-space domain for the simulation, the red dots are particles, the green box is the rectangle where we remap and the support of the mollifier function, the orange is the region where the mollifier function is the identity. This means that between the orange and green boxes we will have both remapped and non-remapped particles.

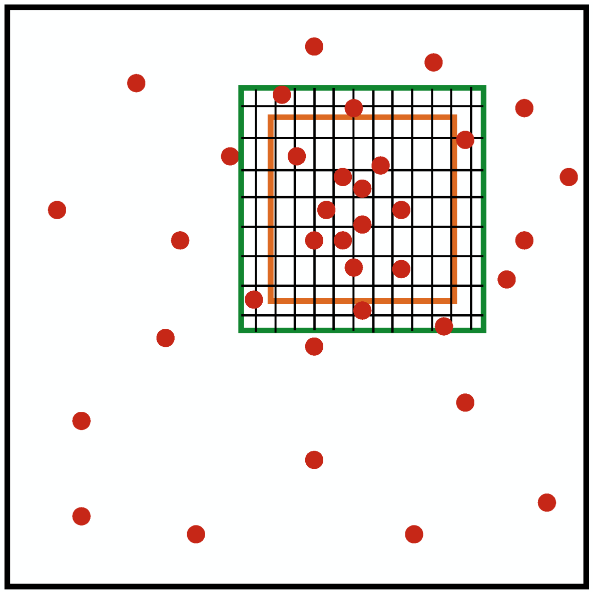


Figure 4.11: The orange box is where the mollifier function is identically one and the green is the region that must be remapped.

Recall, the definition of the remapping error in [20],

$$E = \sum_k \tilde{M}_k \delta(x - \tilde{x}_k) - \sum_i M_i \delta(x - x_i). \tag{4.12}$$

Following the analysis, we multiply by a test function $\phi(x)$. Making the substitution for M_i from the definition of remapping and defining I_{grid} as the set of indices for which $x_i = ih$

and x_i is within the green box, then we see that

$$\begin{aligned}
E &= \sum_k \tilde{M}_k \phi(\tilde{x}_k) - \sum_k (1-\chi(\tilde{x}_k))\, \tilde{M}_k \phi(\tilde{x}_k) - \sum_{i\in I_{\text{grid}}} \sum_k \chi(\tilde{x}_k)\tilde{M}_k W_{q,r}\left(\frac{x_i - \tilde{x}_k}{h}\right)\phi(x_i) \\
&= \sum_k \chi(\tilde{x}_k)\tilde{M}_k \phi(\tilde{x}_k) - \sum_{i\in I_{\text{grid}}} \sum_k \chi(\tilde{x}_k)\tilde{M}_k W_{q,r}\left(\frac{x_i - \tilde{x}_k}{h}\right)\phi(x_i) \\
&= \sum_k \tilde{M}_k \chi(\tilde{x}_k)\left[\phi(\tilde{x}_k) - \sum_{i\in I_{\text{grid}}} W_{q,r}\left(\frac{x_i - \tilde{x}_k}{h}\right)\phi(x_i)\right]
\end{aligned}$$

which allows us to define

$$f(x) = \phi(x) - \sum_{i\in I_{\text{grid}}} W_{q,r}\left(\frac{x_i - x}{h}\right)\phi(x_i).$$

Recall that $W(x)$ obeys the following moment condition,

$$\sum_{i\in I_{\text{grid}}} W_{q,r}\left(\frac{x_i - x}{h}\right) = 1$$

where x is in the green box, I_{grid} is representing the grid framed by the green box. Using this we see

$$\begin{aligned}
f(x) &= \sum_{i\in I_{\text{grid}}} (\phi(x) - \phi(x_i))\, W_{q,r}\left(\frac{x_i - x}{h}\right) && (4.13) \\
&= \sum_{i\in I_{\text{grid}}} \left(\sum_{\alpha=1}^{\infty} \frac{1}{\alpha!}(x-x_i)^{\alpha} \left.\frac{d^{\alpha}\phi(x)}{dx^{\alpha}}\right|_{x=x_i}\right) W_{q,r}\left(\frac{x_i - x}{h}\right) && (4.14)
\end{aligned}$$

where we have Taylor expanded $\phi(x)$. From this, if $W_{q,r}$ satisfies the moment conditions

$$\sum_{i\in K}(x-x_i)^{q'} W_{q,r}\left(\frac{x_i - x}{h}\right) = 0 \text{ for } 1 \le |q'| \le q-1 \tag{4.15}$$

then

$$E = O(h^q). \tag{4.16}$$

Note, this does not place any restrictions on χ but one acceptable choice would smooth step functions like those found in [25].

This analysis suggests that we can accurately represent a distribution as a union of two distributions: one that contains old particles, $f_1 = (1-\chi)f$, and one made up of new remapped particles , $f_2 = R(\chi f)$ where $R(\cdot)$ is the remapping operator. Although we can accurately represent the distribution this way, how does it interact with our deposition error analysis from Chapter 3?

Recall from (3.49) that we were left with a sum of the form,

$$h_v \sum_{j_v} \tilde{f}(\bar{x} - x_0, \alpha_v, t). \tag{4.17}$$

When we remap locally and represent the distribution as the union of two distributions then we need to replace $\tilde{f}$ by a sum of the two distributions. Doing this and replacing $\tilde{f}$ with

$$\tilde{f} = \tilde{f}_1 + \tilde{f}_2 = (1-\chi)\tilde{f} + R(\chi\tilde{f}) \tag{4.18}$$

we see then that the sum can be written as

$$\begin{aligned} h_v \sum_{j_v} \tilde{f}(\bar{x} - x_0, j_v h_v, t) = h_v &\sum_{j_v = v_L/h_v}^{J_{RL}} (1-\chi)\tilde{f}(\bar{x} - x_0, j_v h_v, t) \\ &+ h_v \sum_{i_v} R(\chi\tilde{f})(\bar{x} - x_0, \tilde{j}_v h_v, t) \\ &+ h_v \sum_{j_v = J_{RH}}^{v_H/h_v} (1-\chi)\tilde{f}(\bar{x} - x_0, j_v h_v, t) \end{aligned} \tag{4.19}$$

where i_v denotes the sum over the remapped region contained by the box with j_v boundary indices $[J_{RL}, J_{RH}]$ where RL and RH denote the low and high edge of the remapped box that contains the support of χ, respectively. Recalling the argument for the approximation of the respective integrals for each of these sums, we note that we can no longer apply the argument that $\tilde{f}, \tilde{f}', \tilde{f}''$ are small at the boundary of the integral since the distribution may not be small at these boundaries. Only the distribution multiplied by the deformation gradient will be bounded by C at the boundary of the box.

This is a complicating factor and suggests using a metric that accounts for the value the distribution will take at the boundary of the remapped region. One potential choice would be to remap in regions where

$$\max\Big(\Big(F_x(\alpha_x, \alpha_v, t) - 1\Big)^M f(\alpha_x, \alpha_v), f(\alpha_x, \alpha_v)\Big) > C. \tag{4.20}$$

Using such a condition guarantees that at the boundary of the remapped region $f(\alpha_x, \alpha_v) < C$ and thus we can apply the argument used to obtain equation (3.52) to equation (4.19) and achieve the same accuracy. This condition also allows us to implicitly define the regions that need to be remapped and does not require us to actually define boxes in phase-space. If we let

$$R_c(\alpha_x, \alpha_v, t) = \max\Big(\Big(F_x(\alpha_x, \alpha_v, t) - 1\Big)^M f(\alpha_x, \alpha_v), f(\alpha_x, \alpha_v)\Big) \tag{4.21}$$

then the pseudocode becomes

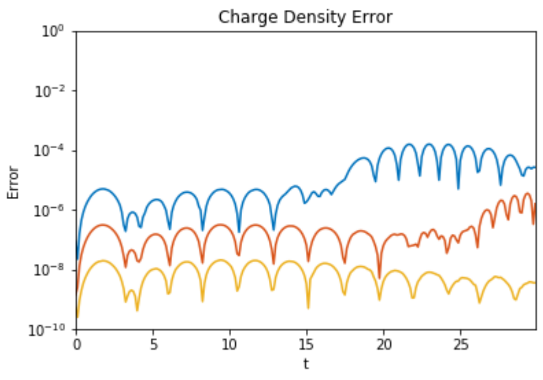
Algorithm 2 Implicit Local Remapping

```
  Input:
  Particles


  transition particles ← Particles
  transition particles ← (1 - $\chi(R_c$(Particles)))transition particles
  remapped particles ← remap($\chi(R_c$(Particles))Particles)

  eraseSmallParticles(transition particles)
  eraseSmallParticles(remapped particles)

  Particles ← remapped particles $\bigcup$ transition particles

  return Particles
```

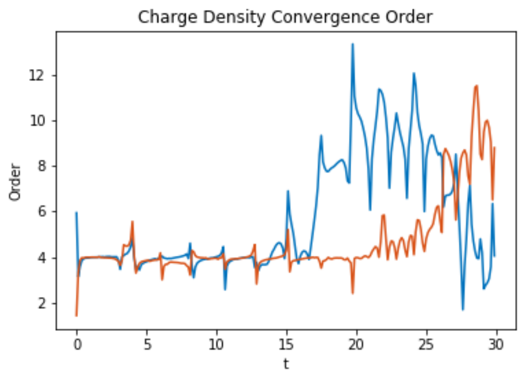
which has the advantage of requiring only local information. This makes it much simpler to implement and parallelize as there is no need for a particle to determine where it exists relative to a box in phase-space or for ranks to compute and communicate boxes that may span multiple different ranks. The actual implementation of remapping is discussed in further detail in Chapter 6.

Numerical Results

We test the remapping condition (4.21) and the form of local remapping described in Algorithm 2 by applying it to the two test problems used to test the adaptive-in-time algorithm. For the linear Landau damping test problem with $C = 10^{-8}$, Figure 4.12 shows that the error and order of convergence is similar to the results for adaptive-in-time remapping for $W_{4,4}$. We find that this remapping condition leads to the exact same number of remaps as when the phase-space distribution is globally remapped. We expect the number of remaps to be the same as for global remapping or increase. The reason one expects the possibility of an increase is because regions where $f(F_x - 1)^4$ is just less than the remapping condition will need to be remapped in the near future and may trigger a local remap before a remap would have been triggered had we used global remapping. Using this condition, Figure 4.13 does

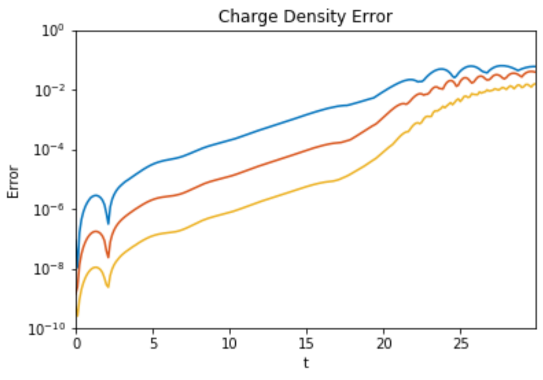


(a) Charge Density Error

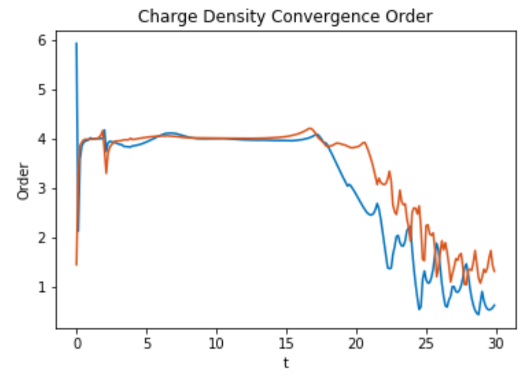


(b) Convergence Order

Figure 4.12: Plots of error and convergence order for $W_{4,4}$ for the linear Landau damping test problem. (a) shows the error for three different resolutions: $M_g = 4$(blue), $M_g = 5$(red), and $M_g = 6$(yellow). (b) shows the order of convergence with $\log_2(e^4/e^5)$ in blue and $\log_2(e^5/e^6)$ in red. We remap when $f(F_x - 1)^4 > 10^{-8}$ and remap the region where this condition is satisfied as well as all particles with $f > 10^{-8}$. As is expected, we see fourth-order convergence and error characteristics similar to Figure 4.3.



(a) Charge Density Error



(b) Convergence Order

Figure 4.13: Plots of error and convergence order for $W_{4,4}$ for the two-stream instability test problem. (a) shows the error for three different resolutions: $M_g = 4$(blue), $M_g = 5$(red), and $M_g = 6$(yellow). (b) shows the order of convergence with $\log_2(e^4/e^5)$ in blue and $\log_2(e^5/e^6)$ in red. We remap when $f(F_x - 1)^4 > 10^{-8}$ and remap the region where this condition as satisfied as well as all particles with $f > 10^{-8}$. As is expected, we see fourth-order convergence and error characteristics similar to Figure 4.4.

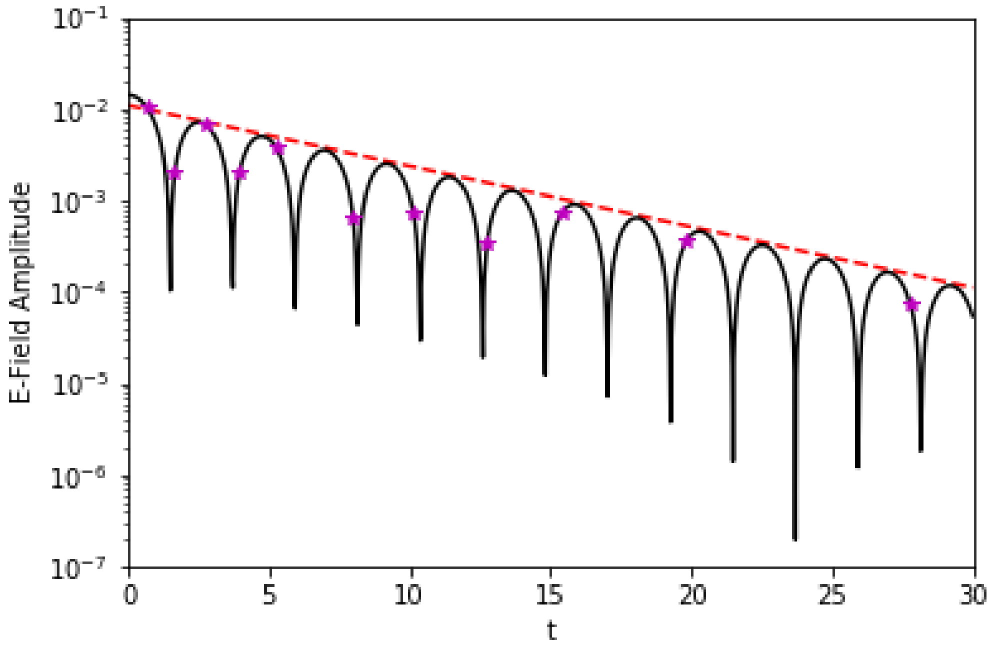


Figure 4.14: Plot of the norm of the electric field (black), the analytic damping rate of 0.1533 (red dashed), and the remap times (magenta stars) for linear Landau damping with $C = 10^{-8}$.

not show any substantial difference in the accuracy or order of convergence when compared to Figure 4.4.

Remapping in this manner produces solutions that converge towards analytic answers for these test problems and compare well with results previously reported in literature for these problems. The amplitude of the electric field is plotted in Figure 4.14 against that analytic damping rate of $\gamma = 0.1533$. The computed damping rate is $\gamma = 0.1559$ and has a relative error of 1.72%. The computed frequency is 1.414 which gives a relative error of 0.12% compared to the expected frequency of 1.416.

Figure 4.15 shows the phase-space distribution at $t = 20$ for the two-stream instability. The plots compare well to those from Figure 4.7. When we do not increase the particle radii for plotting purposes, the transition region between the remapped and not remapped regions of phase-space becomes apparent. The region being remapped is generally $v \in [-6, 6]$ with transition regions of $v \in [6, 7]$ and $v \in [-6, -7]$.

Remapping all particles whose initial distribution is greater than 10^{-8} is somewhat restrictive and it is worth exploring whether this restriction can be somewhat backed off. To

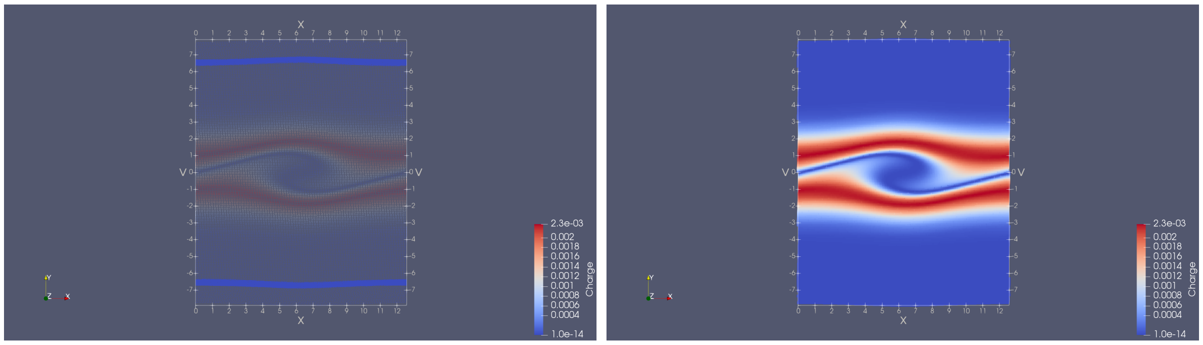

(a) Particle radii = 2 (b) Particle radii = 10

Figure 4.15: Plot of phase-space distribution at time t=20 for the two-stream instability test problem using grid parameters and time step parameters with $M_g = 6$ and $C = 10{-}8$. (a) uses small particle radii for plotting to highlight the number of particles accumulated in the tails of the distribution. This is caused by the requirement that the boundaries have small distribution values so that the Euler-Maclaurin formula can be applied. (b) is plotted with normal particle radii that allow this plot to be compared to those in Figure 4.7.

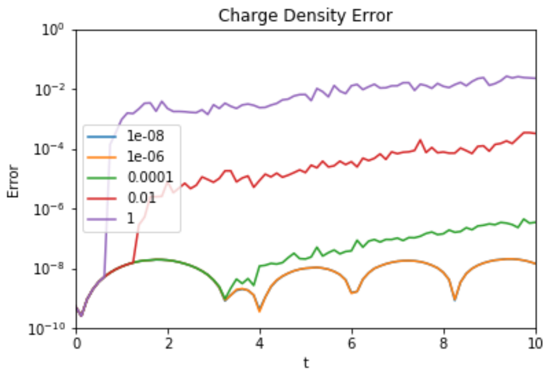
do this, we can define the remapping region as

$$\max\left(\left(F_x - 1\right)^M f(\alpha_x, \alpha_v)/C, f(\alpha_x, \alpha_v)/C_{\text{dist}}\right) > 1 \tag{4.22}$$

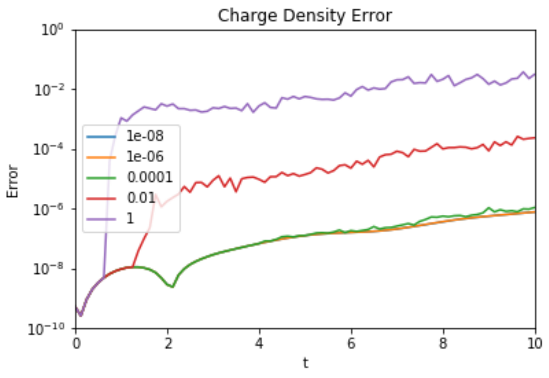
where $C = 10^{-8}$ is the normal condition and we vary C_{dist}. We tested for values of $C_{\text{dist}} \in \{10^{-8}, 10^{-6}, 10^{-4}, 10^{-2}, 1\}$. Figure 4.16 shows how large the error can become from using too large of a C_{dist} value. These errors are catastrophic to the accuracy and furthermore cause instabilities that lead to remapping being required almost every time-step. For these test problems it appears that a better choice of C_{dist} would be 10^{-6} although 10^{-4} might be sufficient for the two-stream instability. The convergence plots for $C_{\text{dist}} = 10^{-6}$ are shown in Figure 4.17.

To test the bounds of the C_{dist} parameter, we examine whether the method still converges with the expected order for $C_{\text{dist}} = 10^{-4}$. Figure 4.18a shows that the errors for the coarsest and middle levels do not display spikes like those seen at the finest level. In fact, the convergence is still approximately fourth-order up to $t = 20$ where the convergence order decays as previously seen.

To compare the distributions at $t = 20$ for $C_{\text{dist}} = 10^{-4}$ we plotted the distribution in Figure 4.19. Note that like $C_{\text{dist}} = 10^{-8}$ we get an accumulation of transition particles around $v = \pm 6$; however, now this accumulation begins near $v = \pm 4$. Another difference is that we also have an accumulation of particles along the vortex and the filament where the distribution function is now sufficiently small when compared to $C_{\text{dist}} = 10^{-4}$. Figure 4.20 shows this phenomena better and is a plot of the remapping condition 10 time steps before



(a) Linear Landau Damping



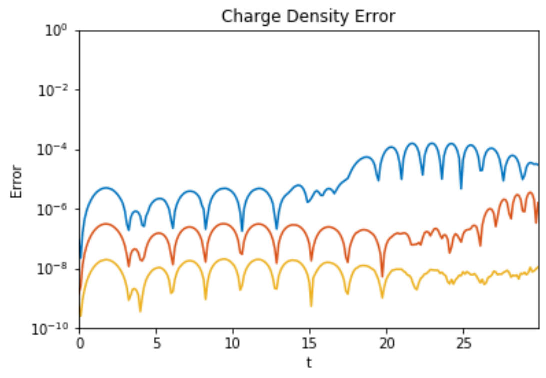
(b) Two-Stream Instability

Figure 4.16: Plot of error with varying C_{dist} values for the two test problems. From these tests, it is clear that $C_{\text{dist}} < 10^{-6}$ is required to achieve convergence similar to the globally remapped tests for the $W_{4,4}$ kernel. For the two-stream instability, a $C_{\text{dist}} = 10^{-4}$ has an error similar to the global remapping and could be used for only this test problem.

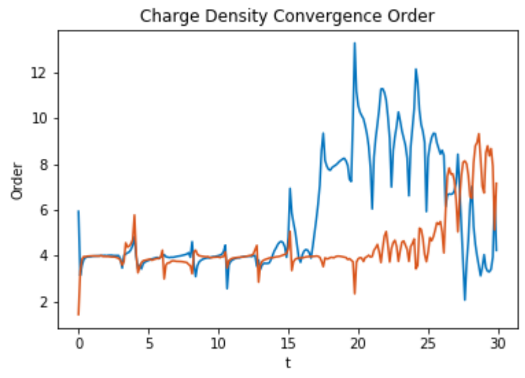
$t = 20$. We see a non-trivial remapping condition and accumulation of particles along the filament in this region.

Remapping locally in phase-space is potentially another way to reduce the cost of remapping. This provides local phase-space control of the $O(1)$ term that is detrimental to the accuracy. The advantages of the implicit definition are that we can do this for arbitrarily complex contours and all of the information required for remapping a particle is local. The cost of doing local remapping in phase-space is that particles are added to the simulation as both remapped and non-remapped particles exist in the transition region. The growth in the total number of particles can be bounded by occasionally remapping globally in place of a local remap when the total number of particles is too large.

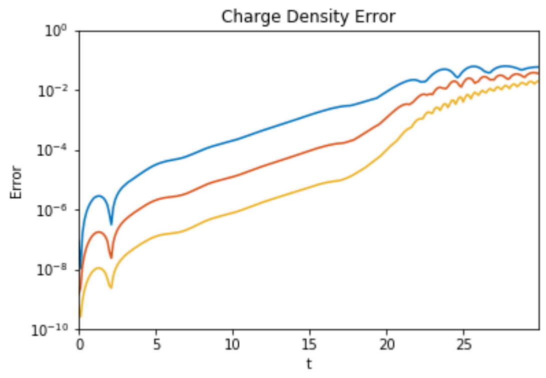
A fair way to compare global remapping to the local remapping method is to track the number of new particles generated by remapping. This is a proxy to the phase-space volume that is being remapped, which would be an ideal comparison but is hard to compute from the implicitly defined curves used to remap locally in phase-space. Based on the plots in Figures 4.15, 4.20a, and 4.19 we expect that the number of newly generated particles for local remapping to be less than the number of particles when remapping globally. This turns out to depend on the choice of C_{dist}. The number of particles generated and the difference between the number of particles generated for varying C_{dist} parameters during remapping is shown in Figure 4.21. We plot the results for all C_{dist} parameters that gave acceptable convergence results for $M_g = 6$ and $C = 10^{-8}$. For linear Landau damping it is clear that using the largest C_{dist} parameter possible gives the largest reduction in number of particles generated during remapping. From Figure 4.21c it is easy to see that using a larger C_{dist}



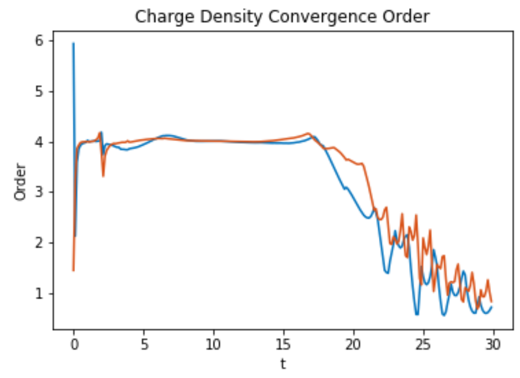
(a) Linear Landau Damping



(b) Linear Landau Damping

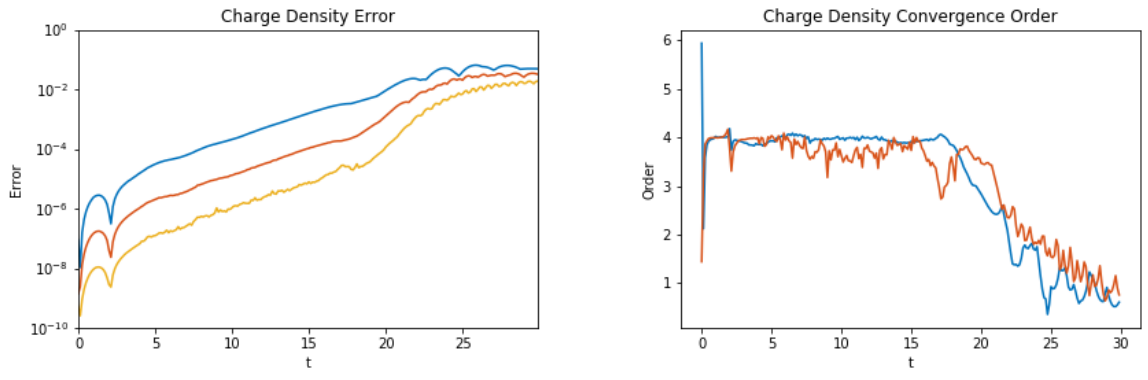


(c) Two-Stream Instability



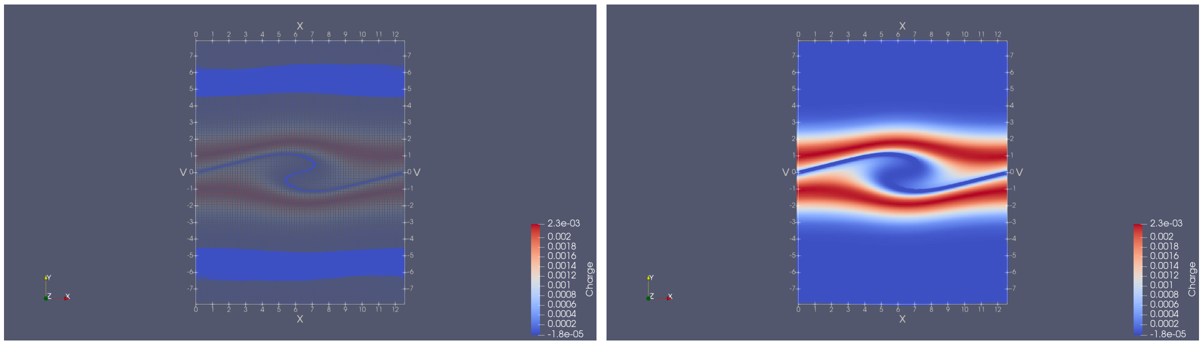
(d) Two-Stream Instability

Figure 4.17: Plots of error and convergence order with $C_{\text{dist}} = 10^{-6}$ for both linear Landau damping and the two-stream instability test problem with the $W_{4,4}$ kernel. (a) and (c) shows the error for three different resolutions: $M_g = 4$(blue), $M_g = 5$(red), and $M_g = 6$(yellow). (b) and (d) shows the order of convergence with $\log_2(e^4/e^5)$ in blue and $\log_2(e^5/e^6)$ in red. For both tests we bound the $O(1)$ error by 10^{-8}. We achieve similar levels of accuracy and convergence as for $C_{\text{dist}} = 10^{-8}$.



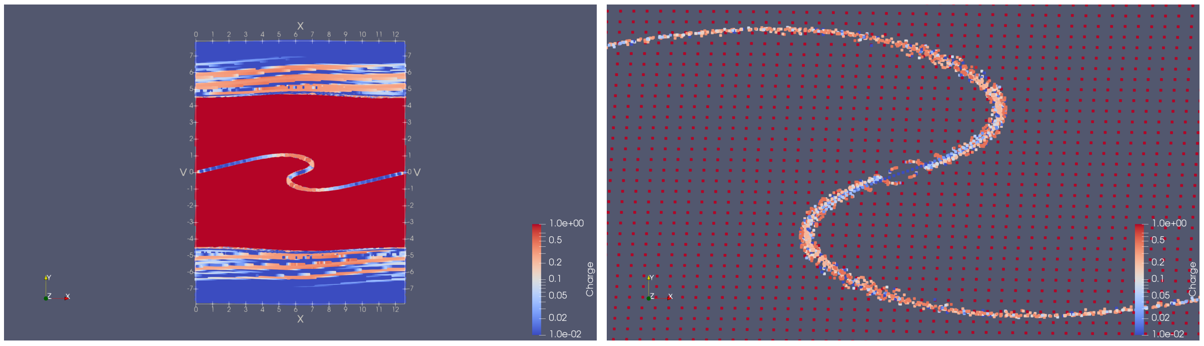
(a) Two-Stream Instability (b) Two-Stream Instability

Figure 4.18: Plots of error and convergence order with $C_{\text{dist}} = 10^{-4}$ for the two-stream instability test problem. (a) shows the error for three different resolutions: $M_g = 4$(blue), $M_g = 5$(red), and $M_g = 6$(yellow). (b) shows the order of convergence with $\log_2(e^4/e^5)$ in blue and $\log_2(e^5/e^6)$ in red. We bound the $O(1)$ error by 10^{-8}. We achieve similar levels of accuracy and convergence as for $C_{\text{dist}} = 10^{-8}$ although do notice some noise in the finest level.



(a) Particle radii = 2 (b) Particle radii = 10

Figure 4.19: Plot of phase-space distribution at time t=20 for the two-stream instability test problem using grid parameters and time step parameters with $M_g = 6$ and $C = 10{-}8$. We set $C_{\text{dist}} = 10^{-4}$. (a) uses small particle radii for plotting to highlight the number of particles accumulated in the tails of the distribution. This is caused by the requirement that the boundaries of the have small distribution values so that the Euler-Maclaurin formula can be applied. (b) is plotted with normal particle radii that allow this plot to be compared to those in Figure 4.7.



(a) Remapping Condition (b) Zoomed in Remapping Condition

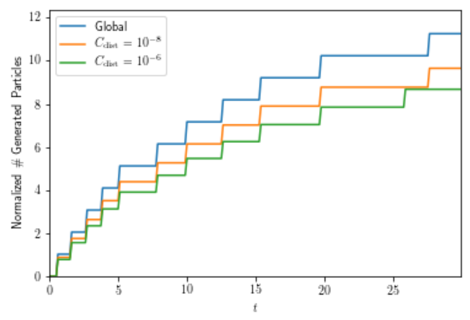
Figure 4.20: Plot of remapping condition 10 time steps before t=20 for the two-stream instability test problem using grid parameters and time step parameters with $M_g = 6$ and $C = 10{-}8$. We set $C_{\text{dist}} = 10^{-4}$. (a) shows the full distribution. We transition smoothly between the region 1 and 0.1. (b) is zoomed in on the center to show that transition particles exist in the vortex of the distribution and along the filament.

parameter can effect when a remap is required. Specifically, for $C_{\text{dist}} = 10^{-6}$ the final remap occurs before either of the other tests. This is due to a region that was previously not remapped because it satisfied $f(\alpha_x, \alpha_v) < C_{\text{dist}}$ and $(F_x - 1)^M f(\alpha_x, \alpha_v) < C$ triggering the remap by no longer satisfying $(F_x - 1)^M f(\alpha_x, \alpha_v) < C$. For the two-stream instability this increased frequency eventually eats away at the efficiency gained early in the simulation for $C_{\text{dist}} = 10^{-4}$ and $C_{\text{dist}} = 10^{-6}$. The number of remaps for each of these tests are listed in Table 4.5. These plots show the delicate balance that needs to be struck via the choice of the

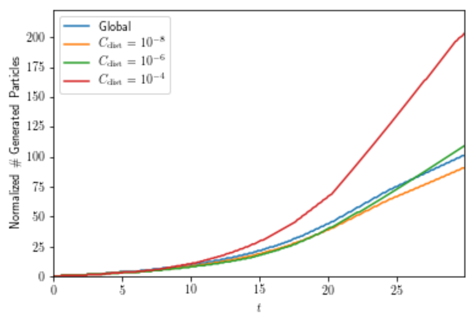
Number of Remaps		
C_{dist}	Linear Landau Damping	Two-Stream Instability
10^{-4}	-	236
10^{-6}	11	126
10^{-8}	11	100

Table 4.5: The number of remaps for the linear Landau damping and two-stream instability where remapping occurs when $f(F_x - 1)^4 > 10^{-8}$ and the region where this condition is satisfied as well as all particles with $f > C_{\text{dist}}$. All tests use $M_g = 6$. This table highlights how by remapping more locally in phase-space can lead to many more remaps and cause more phase-space volume to be remapped than in global remapping.

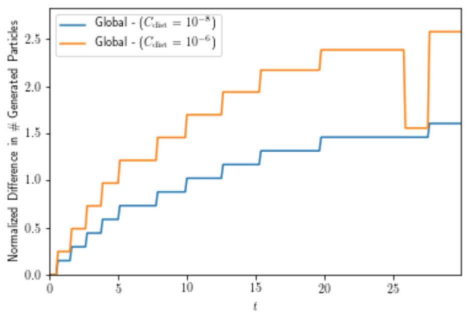
C_{dist} parameter between the frequency of remapping and the region of remapping to achieve optimal performance.



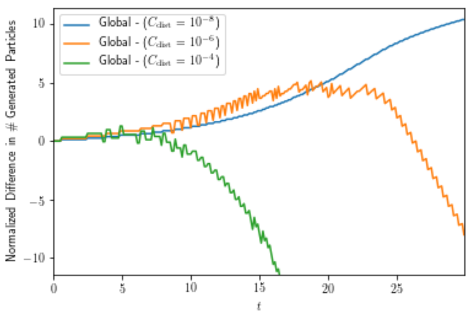
(a) Linear Landau Damping



(b) Two-Stream Instability



(c) Linear Landau Damping



(d) Two-Stream Instability

Figure 4.21: Plots of the number of particles generated in remapping normalized by the initial number of particles. Figures (a) and (b) show the normalized number of remapped particles for the two tests. Figures (c) and (d) show the normalized difference between the number of particles generated by remapping globally and the number generated for each C_{dist}. We test all of the C_{dist} parameters that gave acceptable convergence results. For these tests $C = 10^{-8}$ and $M_g = 6$ which corresponds to the finest resolution error in the convergence tests. These plots highlight how the C_{dist} parameter impacts the performance of locally remapping in phase-space. For linear Landau damping it is advantageous to use the largest C_{dist} that gives acceptable convergence results. At the end of the simulation, $C_{\text{dist}} = 10^{-6}$ leads to a remapping savings of 2.5x the initial number of particles. Unlike linear Landau damping, for two-stream instability we found that if the C_{dist} parameter was too large this advantage disappeared. This is due to the increase in frequency of remapping required by the larger C_{dist} parameters since regions that have a deformation gradient that will need to be remapped at the next time step may not get remapped in the current remapping step. This phenomena is seen in the last remap of the linear Landau damping test problem as the $C_{\text{dist}} = 10^{-6}$ test remaps before either of the other tests that remap a larger portion of the distribution at each remap. We found that for two-stream instability using $C_{\text{dist}} = 10^{-8}$ lead to a savings approximately equal to doing 10 fewer remaps.

4.6 Mixed Kernels

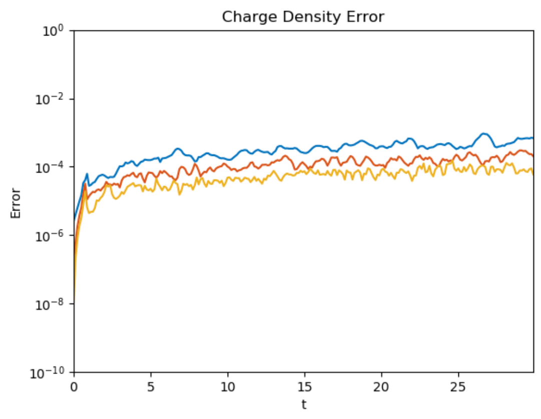
Thus far, we have considered only two kernels, $W_{2,2}$ and $W_{4,4}$. There are numerous other examples of kernels that one could choose. We require that $W_{q,r}(0) = 1$. This reduces the dissipation from remapping and, in our experience, retains better accuracy and convergence properties. Other kernels that satisfy this property and conserve the first four moment conditions are $W_{4,0}$, and $W_{4,2}$. For these kernels we only consider the case where the remapping condition has the power $M = 4$ and again we bound the $O(1)$ term by $C = 10^{-8}$. Figure 4.22 shows the results for these two kernels for both linear Landau damping and the two-stream instability test problem. The C^0 kernel, $W_{4,0}$, has a large increase in the error after only a short time. Late in the simulation when the error increases for the two-stream instability, it does begin to resemble the error for $W_{4,4}$ (see Figure 4.4c for comparison). The C^2 kernel $W_{4,2}$ has an error that is nearly identical to $W_{4,4}$ for both test problems. For the linear Landau damping test problem, we do see some noise in the error of the most refined level. This is especially obvious at late times in the simulation. This noise is not present in the two-stream instability results.

The main advantage of using $W_{4,2}$ instead of $W_{4,4}$ is that it has lower order polynomials and thus is more robust to errors in round off. For the levels of accuracy we have examined in these test problems the polynomials up to the ninth order in $W_{4,4}$ have not caused issues; however, in Section 2.2 we did see a barrier appearing earlier than was expected. We suspect this was caused by the condition number of the $W_{4,4}$ kernel. Both kernels have the same support size and thus there is only a marginal computational savings of approximately eight multiplication and four addition operations per kernel call by using $W_{4,2}$ instead of $W_{4,4}$.

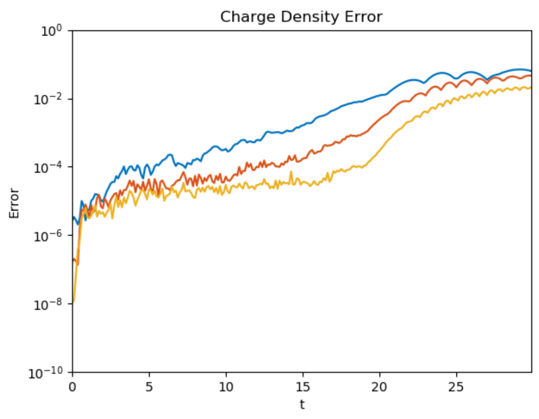
Recall that we must use the same kernel for deposition and force interpolation to avoid self-forcing errors. This does not restrict us from using a different kernel for remapping though. Since deposition and force interpolation occur significantly more frequently than remapping, it would be advantageous from a computational stand point to use low-order kernels for deposition and force interpolation and high-order kernels for remapping. Previous works like [45, 44, 43, 38, 37] have used high-order kernels for remapping to achieve the accuracy predicted by the deposition and force interpolation kernel due to the fact that they lose an order of convergence by remapping at an $O(\Delta t)$ interval. It begs the question then, do we stand to gain anything from remapping with a high-order kernel? In theory, we should be limited by the low-order kernel used in deposition; however, by reconstructing a high-order representation of the distribution occasionally the high-order of convergence could in theory be retained.

We choose to investigate deposition and force interpolation kernels $W_{2,0}$ and $W_{2,2}$ with the remapping kernel $W_{4,4}$. From Section 2.2, $M = 2$ would be the appropriate choice for these deposition kernels but we also consider $M = 4$ with the $W_{2,2}$ kernel. For all cases, we use the same condition that we have used thus far, namely the $O(1)$ term is bounded by $C = 10^{-8}$.

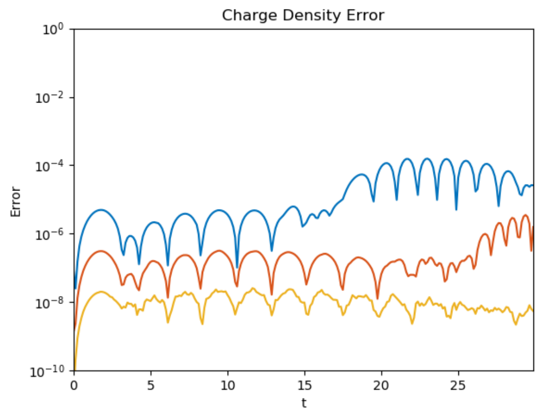
Figure 4.23 shows the results for $W_{2,0}$. The convergence plots appear to have similar characteristics to the the plots for $W_{2,2}$ but with a noticeable addition of small oscillations.



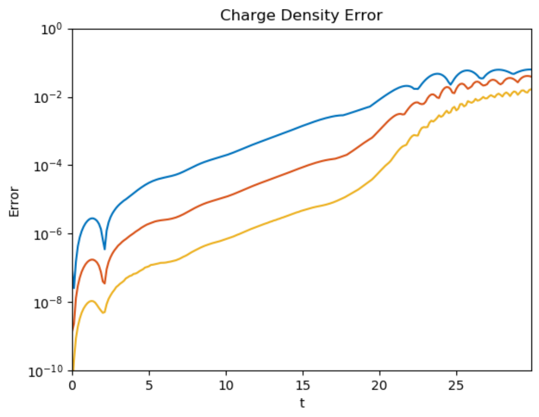
(a) $W_{4,0}$ Linear Landau Damping



(b) $W_{4,0}$ Two-Stream Instability

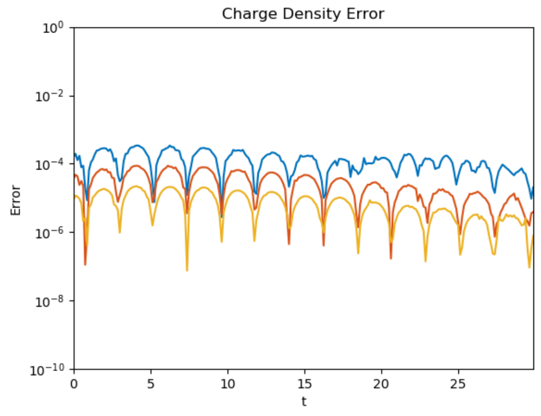


(c) $W_{4,2}$ Linear Landau Damping

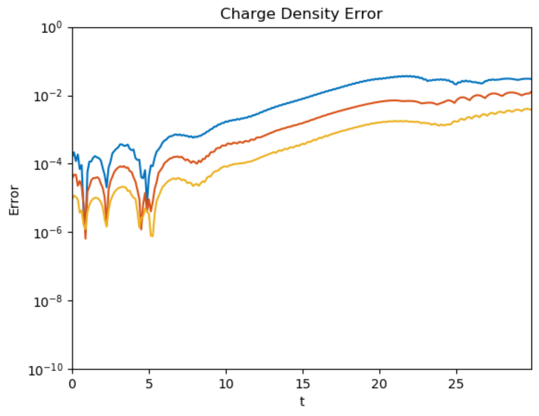


(d) $W_{4,2}$ Two-Stream Instability

Figure 4.22: Plots of error for both $W_{4,0}$ and $W_{4,2}$ for the linear Landau damping and two-stream instability test problems. Three resoultions are plotted:$M_g = 4$(blue), $M_g = 5$(red), and $M_g = 6$(yellow). For all tests we bound the $O(1)$ error by 10^{-8} and the power $M = 4$. We see that the C^0 kernel has a much higher error after a few time steps than the C^2 kernel and when compared to previous results for $W_{4,4}$ in Figure 4.3c and Figure 4.4c. The $W_{4,2}$ kernel closely resembles the error plots for $W_{4,4}$ except for some small additional spikes that are particularly noticeable on the most refined level of the linear Landau damping test problem.



(a) Linear Landau Damping



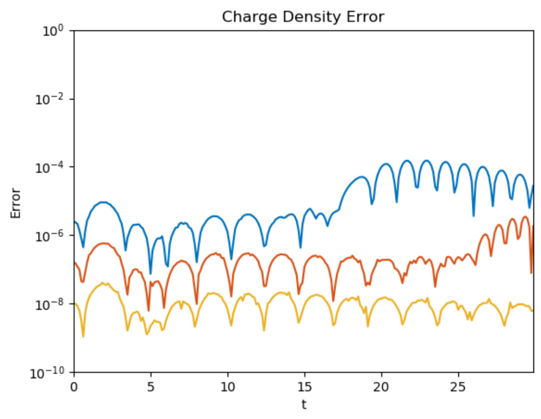
(b) Two-Stream Instability

Figure 4.23: Plots of error for the linear Landau damping and two-stream instability test problems where $W_{2,0}$ was used as the deposition and force interpolation kernel and $W_{4,4}$ was used as the remapping kernel. For all tests we bound the $O(1)$ error by 10^{-8} and the power $M = 2$. Three resolutions are plotted: $M_g = 4$(blue), $M_g = 5$(red), and $M_g = 6$(yellow). We see that compared to the results for $W_{2,2}$ in Figure 4.3a and Figure 4.4a the errors are quite similar except for a small level of oscillations persistent in the results here.

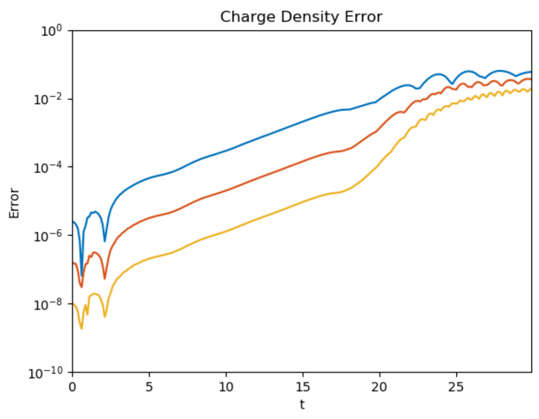
It is interesting that both test problems appear to converge with second-order accuracy. These results do not fit neatly into the theory we have developed thus far. It should be noted though that for this test we use $C = 10^{-8}$ which is a much tighter bound than what would be required by the errors seen. This overly aggressive remapping may mean that the error due to the C^0 knot conditions may be relatively small and we are evaluating errors between knots where the interpolation functions are C^∞.

Figure 4.24 continues to push the bounds of the theory developed in Section 3.3. In this case we see fourth-order convergence while using $W_{2,2}$ as the deposition and force interpolation kernel. This holds for $M = 2$ and $M = 4$. Although the $M = 4$ solutions have noticeable anomalous spikes that appear to get larger and more prevalent for the finer levels. One possibility for why we see higher orders of convergence than we initially expect is that $W_{2,2}$ might not exactly preserve the third and fourth moment conditions but it might be a small value for the moderately deformed phase-space distributions. This would fit with what we see in Figure 4.3a and Figure 4.4a where for a very short time the errors are small and then grow. In those cases the reason that the high-order convergence does not continue is that we remap the distribution with $W_{2,2}$, which is second-order.

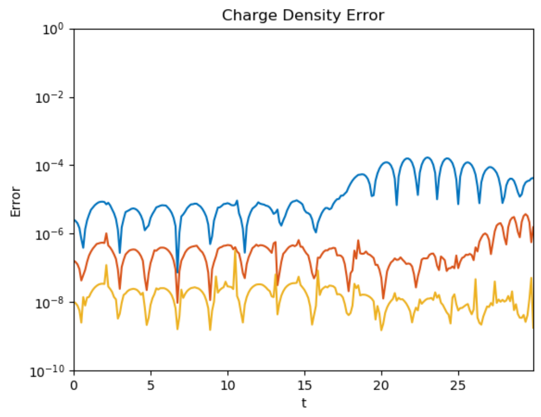
Using $W_{2,2}$ for deposition and force interpolation and $W_{4,4}$ for remapping has non-negligible computational savings as the support of $W_{2,2}$ is $[-2, 2]$ where it is $[-3, 3]$ for $W_{4,4}$. This increased convergence order should not be relied upon for general problems or mapped geometries where the cancellations we benefit from here may not be present.



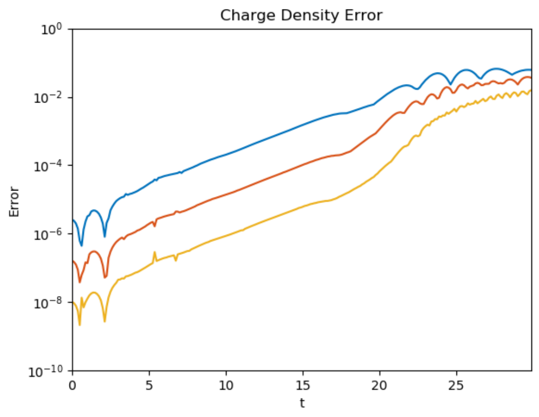
(a) $M = 2$, Linear Landau Damping



(b) $M = 2$, Two-Stream Instability



(c) $M = 4$, Linear Landau Damping



(d) $M = 4$, Two-Stream Instability

Figure 4.24: Plots of error and convergence order with $W_{2,2}$ used for deposition and interpolation and $W_{4,4}$ used for remapping for the linear Landau damping and two-stream instability test problems. Three resolutions are plotted:$M_g = 4$(blue), $M_g = 5$(red), and $M_g = 6$(yellow). For all tests we bound the $O(1)$ error by 10^{-8}. We tested for both $M = 2$ and $M = 4$ to see how reducing the number of remaps effected the regularity of the error. We see that in both cases we achieve fourth order convergence, which is higher than expected. In Figures 4.3b and 4.4b for a short time there is fourth order convergcene. By remapping with a high order kernel we are able to preserve this convergence for long times and get fourth order convergence even though the kernel only exactly preserves the first two moment conditions.

Chapter 5

Dory-Guest-Harris Instability

5.1 Problem Description

We have shown that bounding the $O(1)$ error by remapping when it grows large compared to the discretization errors reduces the "noise" that is often seen in non-remapped particle methods. Although this applies for an arbitrary number of velocity dimensions, thus far we have only considered (1+1)D problems. An electrostatic Vlasov system that has analytic predictions and can act as a benchmark for more than one velocity dimension is the Dory-Guest-Harris (DGH) instability. The existence of unstable electrostatic waves traveling perpendicular to the equilibrium magnetic field ($k_{\|} = 0$) was first noted in [24] after previously being thought to not exist. Dispersion relations were developed in [42] for waves of the form $e^{i(k_{\perp}x-\omega t)}$ where ω has a real, ω_r, and imaginary, ω_i, component that are the oscillation frequency and growth rate, respectively. This gives analytic estimates for the growth and frequency that can be compared to the numerical results. This served as a benchmark for a high-order finite-volume method and here we will use it to benchmark both remapped and non-remapped particle-in-cell methods with two velocity dimensions.

The Vlasov equation for the Dory-Guest-Harris instability can be written as

$$\frac{\partial f}{\partial t} + \frac{\partial}{\partial x}(v_x f) + \frac{Z_e}{M_e}\frac{\partial}{\partial v_x}\left(\left[E_x + \frac{\omega_c}{\omega_p}v_y\right] f\right) + \frac{Z_e}{M_e}\frac{\partial}{\partial v_y}\left(-v_x\frac{\omega_c}{\omega_p}f\right) = 0 \tag{5.1}$$

where similar to the (1+1)D case we relate the phase-space distribution and the electric field via the Poisson equation

$$\rho = \int\int f dv_x dv_y - 1 \tag{5.2}$$

$$\frac{\partial^2\phi}{\partial x^2} = \rho \tag{5.3}$$

$$E_x = \frac{\partial\phi}{\partial x}. \tag{5.4}$$

From this, we can derive the Lagrangian equations of motion

$$\frac{dM^k}{dt} = 0 \tag{5.5}$$

$$\frac{dx^k}{dt} = v_x \tag{5.6}$$

$$\frac{dv_x^k}{dt} = \frac{Z_e}{M_e}\left[E_x + \frac{\omega_c}{\omega_p}v_y^k\right] \tag{5.7}$$

$$\frac{dv_y^k}{dt} = -\frac{Z_e}{M_e}\frac{\omega_c}{\omega_p}v_x^k. \tag{5.8}$$

where again the charge, M, is defined as

$$M^k = f_0(\alpha_x^k, \boldsymbol{\alpha}_v^k)h_x h_v^2 \tag{5.9}$$

and k indexes over particles. From these then, we can derive the evolution of the deformation gradient

$$F_x = \frac{\partial x}{\partial \alpha_x} \tag{5.10}$$

$$\frac{\partial F_x}{\partial t} = \frac{\partial v_x}{\partial \alpha_x} = G_x \tag{5.11}$$

$$\frac{\partial G_x}{\partial t} = F_x \frac{\partial}{\partial x}\frac{\partial v_x}{\partial t} \tag{5.12}$$

$$= \frac{Z_e}{M_e}F_x\frac{\partial}{\partial x}\left[E_x + \frac{\omega_c}{\omega_p}v_y\right] \tag{5.13}$$

$$= \frac{Z_e}{M_e}F_x\frac{\partial E_x}{\partial x} \tag{5.14}$$

which are identical to the (1+1)D system except for the charge-mass ratio factor. For this test problem, we use normalized units and thus $Z_e/M_e = -1$.

Unlike the (1+1)D Vlasov-Poisson System we introduced in Section 3.1, the Lagrangian trajectory equations are no longer equivalent to a second-order system and thus we cannot apply the three-stage fourth-order Runge-Kutta method. This means that we revert back to a standard four-stage fourth-order Runge-Kutta method. For memory purposes, it is advantageous to use a Runge-Kutta method with zeroes everywhere except the sub-diagonal of the tableau so that only one set of increments needs to be stored for each particle at any given stage. For this reason, we use the standard fourth order Runge-Kutta [1].

Other than the change to the Runge-Kutta method, the algorithm for advancing the particle positions for the DGH characteristic equations is the same as in Section 3.2. The remapping algorithm is the only algorithm that is explicitly changed by the two velocity dimensions. The remapping condition is the same as in the (1+1)D test problems,

$$(F_x - 1)^M f < C. \tag{5.15}$$

The justification for using this condition is discussed in Section 3.1 where the change of variables for deposition is generalized to arbitrary configuration and velocity dimensions.

The initial distribution for the DGH instability is

$$f_0(x, v_x, v_y) = \frac{1}{\pi\alpha_\perp^2 j!}\left(\frac{v_x^2+v_y^2}{\alpha_\perp^2}\right)^j \exp\left(-\frac{v_x^2+v_y^2}{\alpha_\perp^2}\right)\left(1+\epsilon\sin\left(4\theta - \frac{\tilde{k}\Omega_c}{v_{\perp 0}}x\right)\right) \tag{5.16}$$

$$\tilde{k} = \frac{2\pi v_{\perp 0}}{L}\frac{\omega_p}{\omega_c},\ \theta = \arctan\left(v_y/v_x\right) \tag{5.17}$$

where $x \in [0, L]$, $v_x, v_y \in [-v_{\max}, v_{\max}]$. This initial distribution was found optimal to excite the dominant mode in the perturbation in [42]. For all tests, $\epsilon = 10^{-4}$ and $\alpha_\perp = \sqrt{2/j}$ so that $v_{\perp 0} = \sqrt{2}$. The length of the domain, L, is computed from the chosen $\tilde{k}$ and ω_c/ω_p.

To maintain charge neutrality, it is important to choose $v_{\max}$ sufficiently large for the given j so that

$$1 = \int_{-v_{\max}}^{v_{\max}}\int_{-v_{\max}}^{v_{\max}} \frac{1}{\pi\alpha_\perp^2 j!}\left(\frac{v_x^2+v_y^2}{\alpha_\perp^2}\right)^j \exp\left(-\frac{v_x^2+v_y^2}{\alpha_\perp^2}\right) dv_x dv_y \tag{5.18}$$

is satisfied to a sufficient precision. For $j = 6$, choosing $v_{\max} = 4$ satisfies this condition to within 10^{-15}. However, for $j = 2$, choosing $v_{\max} = 4$ only satisfies this condition to 10^{-6}. Choosing $v_{\max} = 6$ for $j = 2$ gives a difference of 10^{-14} which is sufficiently close considering we remove particles with a strength of 10^{-14} or less. This is a different means of solving this to [42] which fixed $v_{\max} = 4$ and scaled the integral by a factor that was dependent on the value of j chosen. Doing this would then require that scale factor be used in deposition. Although a larger $v_{\max}$ increases the computational cost, the test problem that we consider with $j = 2$ has relatively few remaps and thus the increase in the number of particles in the velocity dimension is not prohibitively expensive.

For this test problem, we use similar parameters to those used in Chapter 4. The grid parameters for level M_g are $N_x = 2^{M_g}$, $N_x^p = 2^{M_g+1}$, $N_v^p = 2^{M_g+2}/3$, and given the grid parameters, we choose Δt such that the CFL condition,

$$C_{\text{CFL}} = \frac{v_{\max}\Delta t}{h_x}, \tag{5.19}$$

satisfies $C_{\text{CFL}} = 10$. For all convergence studies we used $M_g \in \{5, 6, 7\}$. We reduced the the number of particles used to sample the velocity dimension since the velocity domain shrank from $v_{\max} = 10$ for the (1+1)D test problems to $v_{\max} = 4$ or $v_{\max} = 6$ depending on the value of j for the problem. In [42], for the convergence studies, the three velocity resolutions used were $h_v = 0.125, 0.1, 0.0833$. We use a slightly larger range with $h_v = 0.1875, 0.09375, 0.046875$.

5.2 Numerical Results

We focus on three sets of parameters that cover the three possibilities for a perturbation: exponential growth, oscillatory exponential growth, and a stable perturbation. With $j = 6$, $\tilde{k} = 3.15$, $\omega_p/\omega_c = 20$, and $v_{\max} = 4$ the perturbation exponentially grows without oscillations. The log of the energy of the electric field,

$$U_E = \frac{1}{2}\int_0^L E^2(x)dx, \tag{5.20}$$

for this perturbation is the blue line in Figure 5.1. By using the same parameters but $\tilde{k} = 4.65$ we get a perturbation that grows exponentially but in a manner that is oscillatory. This is the orange line in Figure 5.1. Finally, choosing $j = 2$, $\tilde{k} = 2.12$, $\omega_p/\omega_c = 10$, and $v_{\max} = 6$ gives a stable electrostatic wave that propagates through the plasma without exponential gain or loss in energy. This is the green line in Figure 5.1. For all three cases, we see good agreement with the results obtained in [42] displayed in Figure 7.

Of interest is to compare the remapped and non-remapped solutions for these three perturbations. In (1+1)D we saw that not remapping could be disastrous for the accuracy of the simulation. Figure 5.2 and Figure 5.3 show this comparison for the kernels $W_{2,2}$ and $W_{4,4}$, respectively. For all tests we used $M_g = 5$ and $C = 10^{-8}$ as the criteria to remap globally in phase-space. The remapped solution is the solid blue line while the non-remapped solution is the orange dashed line.

At late times, $t > 400$ we see disagreement between the solutions for all cases; however, for the instabilities we see that the remapped and non-remapped solutions both capture the correct behavior until this time. The non-remapped method does struggle to correctly resolve the stable perturbation for both kernels. This would be the most problematic use of the non-remapped method as it could lead one to believe that this case has oscillatory exponential growth rather than being a stable perturbation. The remapped solutions are not immune from issues. For example, using $W_{2,2}$ for the oscillatory exponential growth case(Figure 5.2b), the kernel appears to be too dissipative for this resolution. The second-order method in [42] displayed similar issues for coarse grids, and as we will show shortly, the method does converge to the correct solution for refined grids. Figure 5.3c also shows issues at late times; however, unlike the non-remapped method it still correctly captures the stability of this perturbation.

Beyond capturing the correct behavior, it is important that the adaptively remapped PIC method converges for these test problems. Since it is difficult to resolve the oscillation period for the stable perturbation, we only test convergence for the two unstable perturbations - as was done in [42]. For both of these perturbations we compute the slope of the exponential growth of the electric energy. We compute this by taking the log of U_E and then fitting a linear approximation to this line. For the oscillatory growth, we compute the linear fit based on the local maxima. To compare with the theoretical imaginary part of the frequency, we

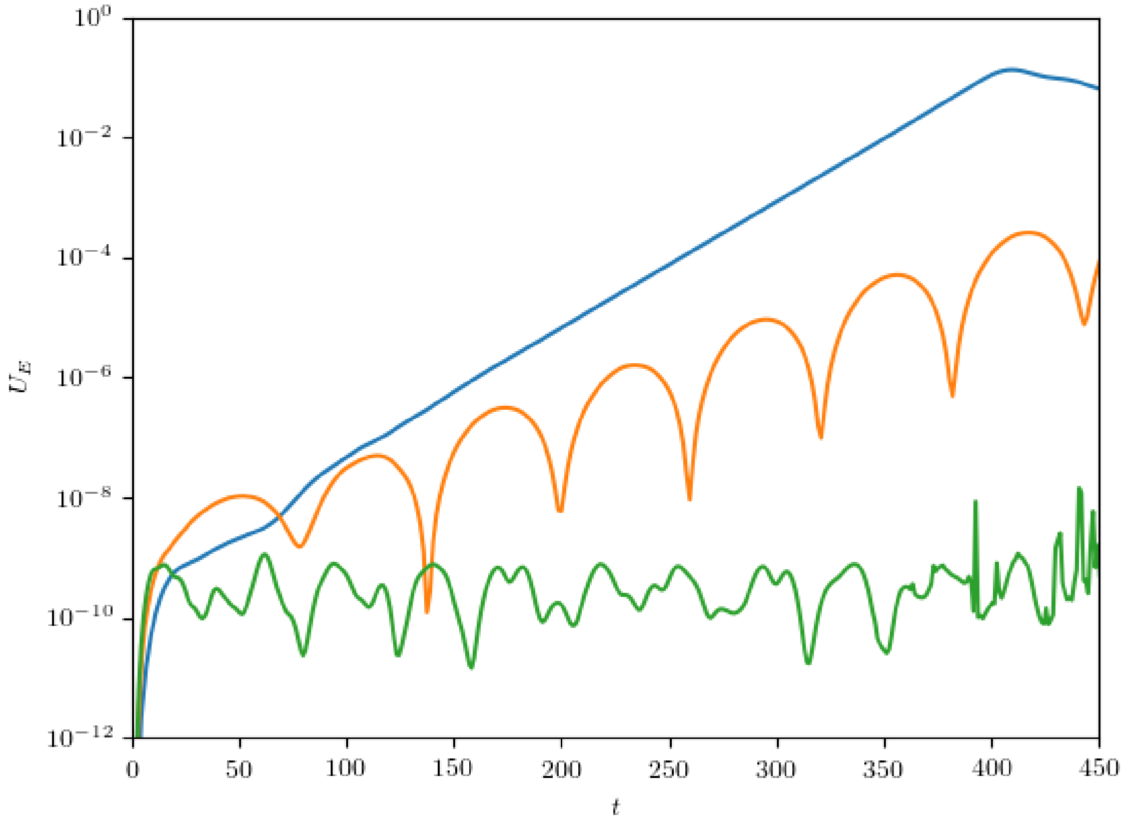
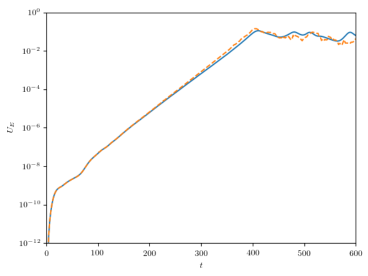
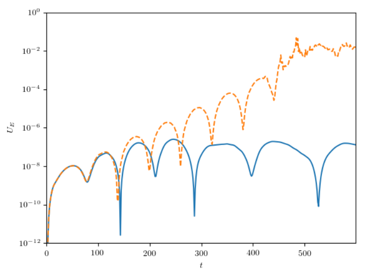


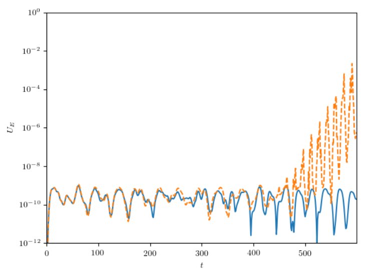
Figure 5.1: Plot of three different stability regimes for the Dory-Guest-Harris instability. These plots were generated by using the kernel $W_{4,4}$ for deposition, force interpolation, and remapping. For this test we used $M_g = 5$ and remapped when the $O(1)$ condition was larger than $C = 10^{-8}$. Blue depicts pure exponential growth of the electric field and is initialized with $j = 6$, $\tilde{k} = 4.65$, $\omega_p/\omega_c = 20$, and $v_{\max} = 4$. Orange shows oscillatory exponential growth in the electric field energy that comes from using the same initial condition as the pure exponential except for $\tilde{k} = 4.65$. Finally, green is the electric field energy of a stable electrostatic wave propagating through the plasma. The electric field energy does not grow in time. The initial conditions for this stable wave is $j = 2$, $\tilde{k} = 2.12$, $\omega_p/\omega_c = 10$, and $v_{\max} = 6$. The $v_{\max}$ was extended to ensure charge neutrality.



(a) $j = 6$, $\tilde{k} = 3.15$, $\omega_p/\omega_c = 20$, $v_{\max} = 4$



(b) $j = 6$, $\tilde{k} = 4.65$, $\omega_p/\omega_c = 20$, $v_{\max} = 4$



(c) $j = 2$, $\tilde{k} = 2.12$, $\omega_p/\omega_c = 10$, $v_{\max} = 6$

Figure 5.2: Plot of three different stability regimes for remapped (blue solid) and non-remapped (orange dashed) for the Dory-Guest-Harris instability with $W_{2,2}$, $M_g = 5$, and $C = 10^{-8}$. (a) depicts pure exponential growth of the electric field and we see that both the remapped and non-remapped methods capture the growth and only differ significantly in the nonlinear regime. (b) shows oscillatory exponential growth in the electric field energy. The remapped solution appears to be dissipative and does not capture the behavior correctly at this resolution. (c) shows the electric field energy of a stable electrostatic wave propagating through the plasma. The electric field energy should not grow in time but rather oscillate for the duration of the simulation. We see agreement in the solution until $t \sim 200$ and then complete divergence in the solutions after $t = 400$. In this case, the non-remapped versions electric field energy grows in a manner that is exponential and oscillatory. This is the incorrect behavior and in contrast to the lack of growth in the remapped implementation.

have to scale the slope, S, by

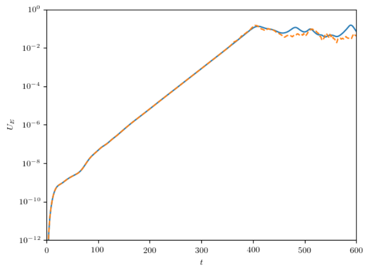
$$\tilde{\omega}_i = \frac{1}{2}\frac{\omega_p}{\omega_c}S. \tag{5.21}$$

For the oscillatory exponential growth we also use these local maxima to compute the period which can then be compared to theoretical results. To compute the real part of the frequency from the period, T, we compute

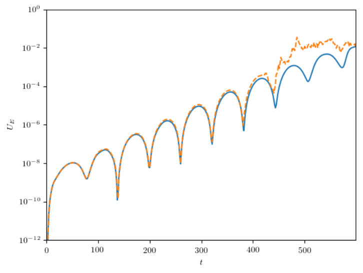
$$\tilde{\omega}_r = \frac{\pi}{T}\frac{\omega_p}{\omega_c}. \tag{5.22}$$

For $\tilde{k} = 3.15$, we compute the slope for $t > 138$ and for $\tilde{k} = 4.65$ we compute the slope and period for $t > 150$. Figure 5.4 plots U_E for $M_g \in \{5, 6, 7\}$ versus time for the two test cases and both kernels. In all plots blue is the coarsest level and green is the most refined.

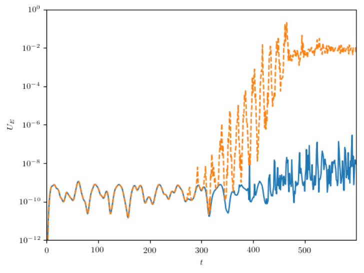
As is evident from these plots, for $W_{2,2}$, the coarsest level appears to be too dissipative. This may be exacerbated by the tight remapping bound of $C = 10^{-8}$. Table 5.1 lists large relative errors in the coarsest level for all test problems. The finest level has much more reasonable relative errors of all less than 2%. Figure 5.5 plots the results as well as the linear



(a) $j = 6$, $\tilde{k} = 3.15$, $\omega_p/\omega_c = 20$, $v_{\max} = 4$



(b) $j = 6$, $\tilde{k} = 4.65$, $\omega_p/\omega_c = 20$, $v_{\max} = 4$



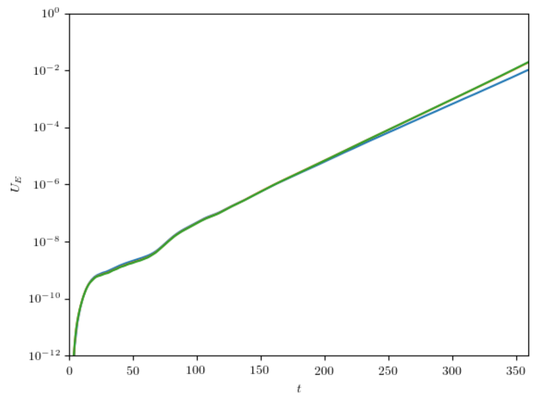
(c) $j = 2$, $\tilde{k} = 2.12$, $\omega_p/\omega_c = 10$, $v_{\max} = 6$

Figure 5.3: Plot of three different stability regimes for remapped (blue solid) and non-remapped (orange dashed) for the Dory-Guest-Harris instability with $W_{4,4}$, $M_g = 5$, and $C = 10^{-8}$. (a) depicts pure exponential growth of the electric field and we see that both the remapped and non-remapped methods capture the growth and only differ significantly in the nonlinear regime. (b) shows oscillatory exponential growth in the electric field energy. The remapped solution and non-remapped solution begin to noticeably diverge around $t = 200$ and the non-remapped solution no longer captures the oscillatory behavior after $t = 400$. (c) shows the electric field energy of a stable electrostatic wave propagating through the plasma. The electric field energy should not grow in time but rather oscillate for the duration of the simulation. We see agreement in the solution until $t \sim 300$ and then complete divergence in the solutions after that time. In this case, the non-remapped versions electric field energy grows in a manner that is exponential and oscillatory. This is the incorrect behavior and in contrast to the lack of growth in the remapped implementation. We see a noisier solution for after $t = 400$ for the remapped solution; however, the solution does not grow dramatically like the non-remapped solution.

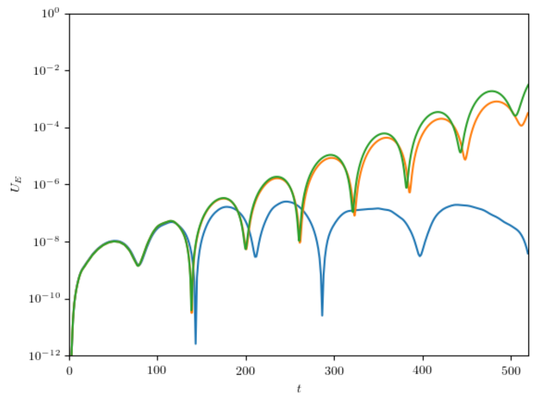
fit. Using Richardson extrapolation we achieved between third and fourth order convergence for all test problems. We plotted this and computed the converged value with $(h_v/h_v^c)^4$ rather than $(h_v/h_v^c)^2$.

For the fourth-order kernel, $W_{4,4}$, we obtained much better relative errors. Table 5.2 lists the relative errors for each level of refinement as well as the converged value. The worst relative error is 6.224%. The finest level of refinement has relative errors all less than 0.9%. With this kernel, the computed orders of convergence are fourth-order for the growth rates. We see less than the expected order of convergence for the frequency; however, for this test problem the relative error for every level is less than 1%. This is in good agreement with what [42] found using a fourth-order finite volume method.

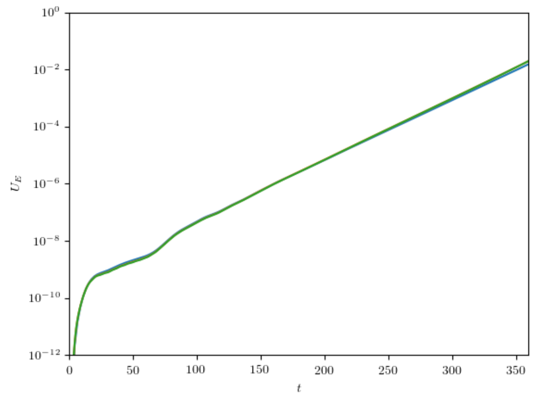
For the test with oscillatory growth, we plot slices of the phase-space distribution in Figure 5.7 at late times. The distribution begins to deform substantially here as we get features reminiscent of the two-stream instability in the (1+1)D case. This explains why the non-remapped method struggles more at late times to correctly capture the behavior for this



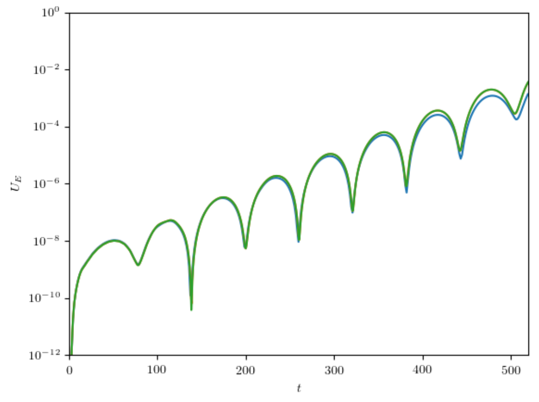
(a) $W_{2,2}$, $\tilde{k} = 3.15$



(b) $W_{2,2}$, $\tilde{k} = 4.65$

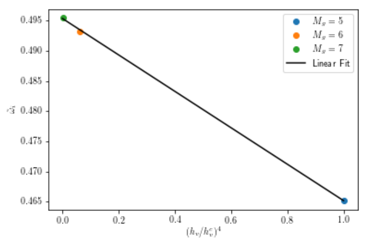


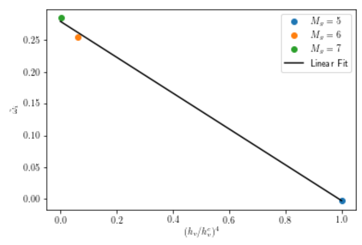
(c) $W_{4,4}$, $\tilde{k} = 3.15$



(d) $W_{4,4}$, $\tilde{k} = 4.65$

Figure 5.4: (a) and (b) show convergence plots for the two different test problems in Figure 5.1 with $W_{2,2}$. Blue, yellow, and green show the solutions for the parameters $M_g \in \{5, 6, 7\}$ respectively. For all tests the initial conditions are $j = 6$, $\omega_p/\omega_c = 20$, and $v_{\max} = 4$ with $\tilde{k}$ varying for the two tests. As discussed in Figure 5.2a, for the coarsest grid we see issues with dissipation for this kernel in (b). (c) and (d) show the same tests for $W_{4,4}$. Unlike for the second-order kernel, we see good agreement for all levels for both test problems.


(a) $\tilde{\omega}_i$, $\tilde{k} = 3.15$


(b) $\tilde{\omega}_i$, $\tilde{k} = 4.65$

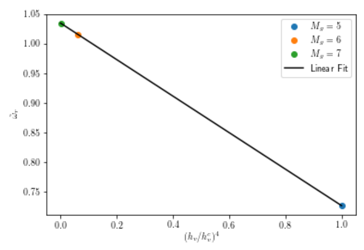
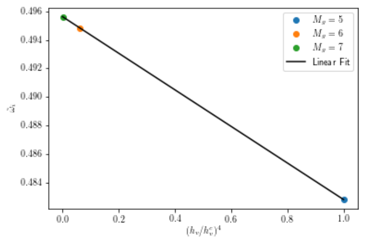
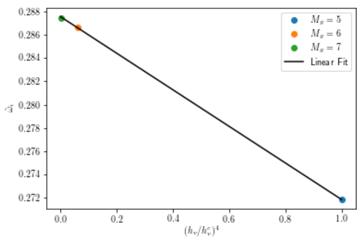

(c) $\tilde{\omega}_r$, $\tilde{k} = 4.65$

Figure 5.5: (a) and (b) show convergence of the linear growth rates for the two different test problems with $W_{2,2}$. (c) plots the convergence of the oscillation frequency. For all plots, the blue, yellow, and green dots show the computed values for $M_g \in \{5, 6, 7\}$ respectively. For all tests the initial conditions are $j = 6$, $\omega_p/\omega_c = 20$, and $v_{\max} = 4$ with $\tilde{k}$ varying for the two tests. For $W_{2,2}$ we see between third and fourth order convergence for the different tests and exact solutions. For this reason, we chose to fit and plot with h_v/h_v^c taken to the fourth power as the plots with the expected order displays this behaviour.


(a) $\tilde{\omega}_i$, $\tilde{k} = 3.15$


(b) $\tilde{\omega}_i$, $\tilde{k} = 4.65$

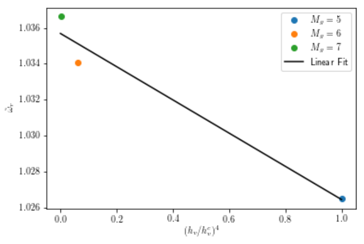

(c) $\tilde{\omega}_r$, $\tilde{k} = 4.65$

Figure 5.6: (a) and (b) show convergence of the exponential growth rates for the two different test problems with $W_{4,4}$. (c) plots the convergence of the oscillation frequency. For all plots, the blue, yellow, and green dots show the computed values for $M_g \in \{5, 6, 7\}$ respectively. For all tests the initial conditions are $j = 6$, $\omega_p/\omega_c = 20$, and $v_{\max} = 4$ with $\tilde{k}$ varying for the two tests. For the linear growth rates, we see fourth-order convergence while for the oscillatory frequency computed for $\tilde{k} = 3.15$ is lower than expected but all solutions have a computed frequency within less than 1% of the theoretical value.

$\widetilde{k}$		Theoretical	$M_g = 5$	$M_g = 6$	$M_g = 7$	Extrapolated
3.15	$\tilde{\omega}_i$	0.4912	0.4652	0.4933	0.4954	0.4954
	% error	-	5.295	0.4214	0.8622	0.8470
4.65	$\tilde{\omega}_i$	0.2899	-0.002	0.2550	0.2846	0.2794
	% error	-	100.7	12.03	1.814	3.628
4.65	$\tilde{\omega}_r$	1.0361	0.7256	1.0166	1.0341	1.0356
	% error	-	29.96	1.883	0.1951	0.0466

Table 5.1: Computed growth, $\tilde{\omega}_i$, and frequency, $\tilde{\omega}_r$, for unstable test problems using $W_{2,2}$. The % error is a relative error to the theoretical solution. Data is plotted in Figure 5.5. Some of the extrapolated and finest resolution values are reported as equivalent. This is an artifact of rounding to the precision that the theoretical value is reported with in [42].

$\widetilde{k}$		Theoretical	$M_g = 5$	$M_g = 6$	$M_g = 7$	Extrapolated
3.15	$\tilde{\omega}_i$	0.4912	0.4828	0.4948	0.4956	0.4956
	% error	-	1.712	0.7382	0.8954	0.9037
4.65	$\tilde{\omega}_i$	0.2899	0.2719	0.2866	0.2874	0.2876
	% error	-	6.224	1.125	0.8501	0.8082
4.65	$\tilde{\omega}_r$	1.0361	1.0265	1.0341	1.0366	1.0357
	% error	-	0.9254	0.1951	0.0507	0.0397

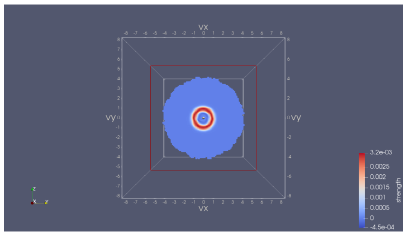
Table 5.2: Computed growth, $\tilde{\omega}_i$, and frequency, $\tilde{\omega}_r$, for unstable test problems using $W_{4,4}$. The % error is a relative error to the theoretical solution. The extrapolated values, which utilize the linear fit to compute the value for $M_g \to \infty$, are all within 1% of the theoretical value. This is in good agreement with what [42] found using a fourth-order finite volume method. Data is plotted in Figure 5.6. Some of the extrapolated and finest resolution values are reported as equivalent. This is an artifact of rounding to the precision that the theoretical value is reported with in [42].

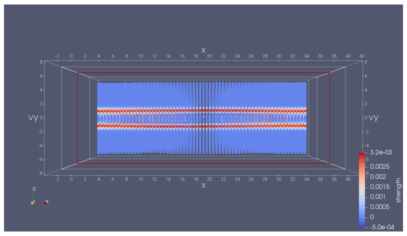
test problem.

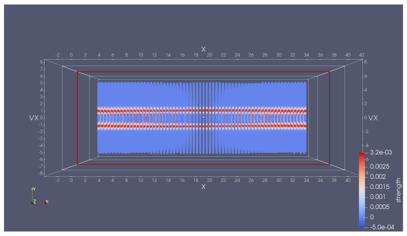
We also tested the local remapping method using the contours defined by

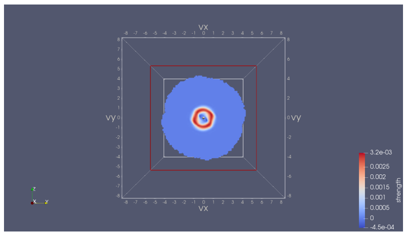
$$\max\left(\left(F_x - 1\right)^M f(\alpha_x, \alpha_v), f(\alpha_x, \alpha_v)\right) > C \tag{5.23}$$

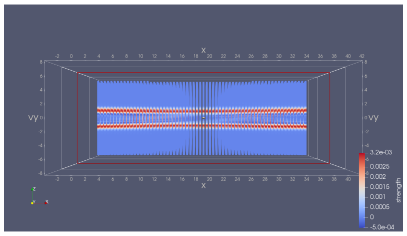
with $C = 10^{-8}$ and the transition function specified in Chapter 6. There was no noticeable degradation in the solution when compared to the globally remapped method; however, remapping locally led to many more remaps. This is the expected behaviour as particles with remapping conditions slightly less than C, for example $\left(F_x - 1\right)^M f = 10^{-10}$, would not be remapped but needed to be remapped within a few time steps (or even the next time step). Figure 5.8 shows the solution for $t \in [0, 600]$ and Table 5.3 lists the comparative number of

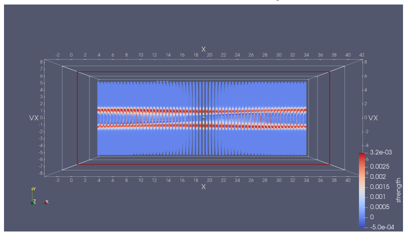

(a) $t = 500.12, x = L/2$

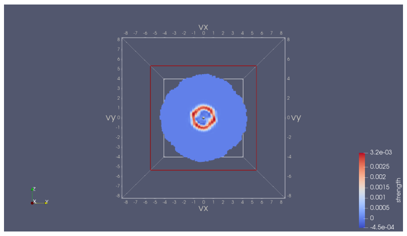

(b) $t = 500.12$, $v_x = 0$

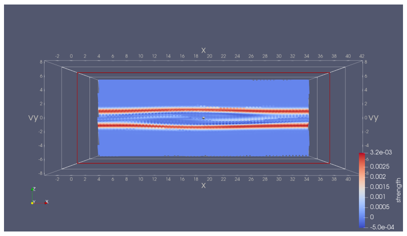

(c) $t = 500.12$, $v_y = 0$


(d) $t = 550.88$, $x = L/2$


(e) $t = 550.88$, $v_x = 0$


(f) $t = 550.88$, $v_y = 0$


(g) $t = 598.65$, $x = L/2$


(h) $t = 598.65$, $v_x = 0$

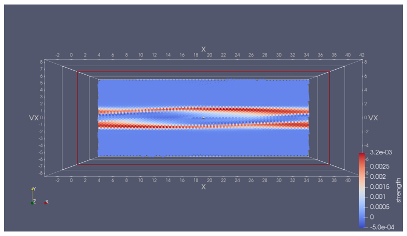

(i) $t = 598.65$, $v_y = 0$

Figure 5.7: Slices of the phase-space distribution at multiple times and in multiple planes for $j = 6$, $\tilde{k} = 4.65$, $\omega_p/\omega_c = 20$, and $v_{\max} = 4$. For these plots we use $W_{4,4}$, $C = 10^{-8}$, and $M_g = 5$. Notable is the deformation in the ring that occurs between $t = 500$ and $t = 600$.

remaps for adaptive-in-time and adaptive-in-time-and-space for the three test problems.

The adaptively remapped PIC method performs as expected in (1+2)D for both adaptive-in-time and adaptive-in-time-and-space remapping. We see that the method correctly captures the behavior where the non-remapped method fails. For the second-order kernel the convergence is between third- and fourth-order while for the fourth-order kernel we converge at fourth-order for the growth rates. The finest resolution shows good agreement with the theoretical results for all kernels.

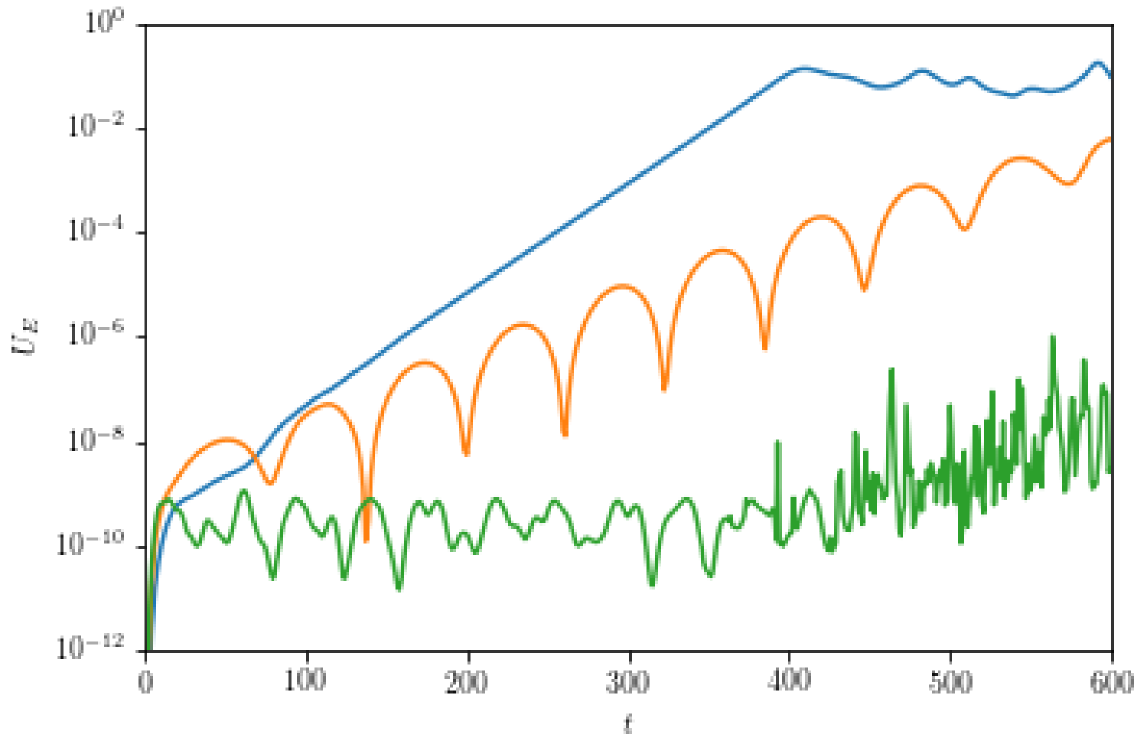


Figure 5.8: Plot of three different stability regimes for the Dory-Guest-Harris instability with the kernel $W_{4,4}$, $M_g = 5$, $C = 10^{-8}$, and adaptive-in-time-and-space remapping. Blue depicts pure exponential growth of the electric field and is initialized with $j = 6$, $\tilde{k} = 4.65$, $\omega_p/\omega_c = 20$, and $v_{\max} = 4$. Orange shows oscillatory exponential growth in the electric field energy that comes from using the same initial condition as the pure exponential except for $\tilde{k} = 4.65$. Finally, green is the electric field energy of a stable electrostatic wave propagating through the plasma. The electric field energy does not grow in time. The initial conditions for this stable wave is $j = 2$, $\tilde{k} = 2.12$, $\omega_p/\omega_c = 10$, and $v_{\max} = 6$. The $v_{\max}$ was extended to ensure charge neutrality. This is comparable to plots in Figure 5.3. The results are not noticeably different than the global remapping solution.

Number of Remaps			
$\widetilde{k}$	Number of Time Steps	Adaptive-in-time	Adaptive-in-time-and-space
3.15	272	90	153
4.65	401	89	229
2.12	549	9	10

Table 5.3: The number of remaps for the three different modes of stability for global and adaptive-in-space remapping with $t \in [0, 600]$. Each test is denoted by the associated $\tilde{k}$. For $\tilde{k} = 3.15$ and $\tilde{k} = 4.65$ the other parameters used to initiate the distribution are $j = 6$, $\omega_p/\omega_c = 20$, and $v_{\max} = 4$. For $\tilde{k} = 2.12$, the parameters are $j = 2$, $\omega_p/\omega_c = 10$, and $v_{\max} = 6$. There increase in number of remaps is expected since regions that are not remapped will likely need to be remapped soon.

Chapter 6

Software Implementation of Remapping

6.1 Implementation of Global Remapping

In Chapter 1 we introduced the curse of dimensionality in the context of PIC and showed how inefficient a rectangular grid becomes in dimensions that are encountered in kinetics. However, there are clear advantages in the error analysis to using a rectangular discretization. Namely, the application of the Euler-Maclaurin formula allows for the integration over the velocity dimension necessary for deposition to be done with arbitrary order accuracy. This suggests that we need to utilize a rectangular grid but that an explicit representation of this grid, like using rectangular arrays, must be avoided. One way of doing this of doing this is by using a hash function based on Morton ordering of the phase-space to implicitly define the grid [36]. Figure 6.1 shows the Morton ordering of the (1+1)D phase-space from the test problems in Chapter 4. When remapping, this hash function allows us to uniquely define the particle at a specific phase-space location and add it to a map without allocating non-zero entries. This hashing (or a different hashing metric) can then be used to define ownership of particles to ranks. For this work we used the Morton ordering keys to partition phase-space among the ranks. This may not be the most efficient splitting since from the remapping condition it is easy to see that large changes in the velocity dimension for a set of particles with the same initial velocity, α_v, will lead to the deformation gradient rapidly growing. This suggests that distributing the particles between ranks based on α_v will maintain the locality of the particles for a longer time.

The pseudocode for how to implement this using the Berkeley Container Library (BCL), developed in [9], is shown in Algorithm 3. The advantage of using BCL for this problem is that it allows us to use queues that can be written into just as a C++ map would be while also being able to be flushed asynchronously. This allows some of the computation and communication to be overlapped. Note that if a particle is already generated, the particle charge is summed. This prevents the generation of the same particle by every original particle

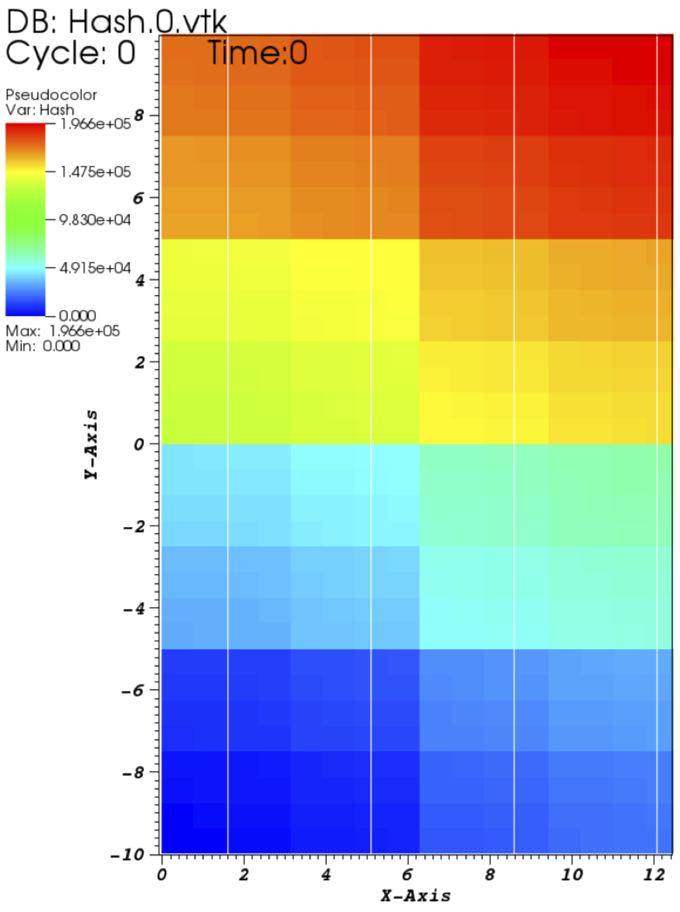


Figure 6.1: Plot of (1+1)D Morton order hash key. At each x-v location we plot the value of the hash key from interlacing the location. This provides a unique key for each location. A limitation of this approach is that for large dimensional problems the allowable resolution in each dimension decreases.

within the particles support.

Algorithm 3 Parallel Remapping

```
for p ∈ Particles do
    support_box ← InterpolationKernel.support(p)
    for $(\boldsymbol{x}, \boldsymbol{v})$ ∈ support_box do
        key ← MortonOrder($\boldsymbol{x}, \boldsymbol{v}$)
        BCL_FastQueue( key).charge += Interpolate(p);
        Push circular queue if larger than specified size
    end for
end for
Push out the rest of the particles
Process received particles
return Particles
```

The exact implementation of this is shown in the C++ code below. The input to the global remapping function is a vector of particles. This vector could be the entire set of particles, as it is for global remapping, or it could be a subset of the particles, like seen in the local remapping(see Algorithm 2 in Chapter 4). As noted in the pseudocode, if the particle we are writing to exists we increment the strength. We do have the possibility of generating more than one particle at a location in the queue if the particles are pushed at different times. This is a tradeoff that is made to allow the queues to be pushed asynchronously. We have not done any performance analysis or tuned any of the parameters listed. We have typically ran on small clusters of up to 16 MPI processes. We do believe though that to do remapping in a performance oriented manner that one needs to use a map structure like BCL's FastQueues.

```cpp
void ParticleSet::globalRemap(vector<Particle>& t_particles)
{
    int Npx = 1./m_hx;
    int Npv = 1./m_hv;
    int total_num_particles;
    if (VDIM > 1)
       total_num_particles = t_particles.size()*1.5;
    else
       total_num_particles = Power(Npx, XDIM)*Power(2*Npv, VDIM);
    int numParticles = t_particles.size();


    // Tuning parameters
    const int message_size = max(floor(numParticles/BCL::nprocs()),
            floor(total_num_particles/BCL::nprocs()/BCL::nprocs()))*2;
    int num_messages_per_rank;
    num_messages_per_rank = 2;
    const size_t bucket_size = max(message_size*BCL::nprocs()*↩
       num_messages_per_rank,
            total_num_particles/BCL::nprocs()*num_messages_per_rank);

    //Initialize FastQueues
    std::vector <BCL::FastQueue <Particle2Send> > queues;
    for (int rank = 0; rank < BCL::nprocs(); rank++) {
        queues.push_back(BCL::FastQueue<Particle2Send>(rank, bucket_size));
    }

    //Initialize futures for fastqueues
    list<BCL::future<vector<Particle2Send>>> futures;

    // Worth trying both of these
    std::vector <std::unordered_map<unsigned long long int, Particle2Send>> ↩
       message_queues(BCL::nprocs());

    // Computation of particle interpolation on phase-space grid
    for (int j = 0; j< t_particles.size(); j++)
    {
        array<int,XDIM> iposLowX, iposHighX;
        array<int,VDIM> iposLowV, iposHighV;
        array<double,XDIM> posX;
        array<double,VDIM> posV;
        //find particles phase space "bin"
        for (int l = 0; l < XDIM; l++)
        {
            posX[l] = t_particles[j].m_x[l];
            t_particles[j].m_F[l] = 1.0;
            t_particles[j].m_G[l] = 0.0;
            iposLowX[l] = floor(posX[l]/(m_hx*m_L));
            iposHighX[l] = ceil(posX[l]/(m_hx*m_L));
        }
        for (int l = 0; l < VDIM; l++)
        {
            posV[l] = t_particles[j].m_v[l];
            iposLowV[l] = floor(posV[l]/(m_hv*m_vmax));
            iposHighV[l] = ceil(posV[l]/(m_hv*m_vmax));
        }

        //Builds the support box in phase space
        Point<XDIM> ShiftX;
        ShiftX = ShiftX.getUnitv(0);
        ShiftX *= 0;
        ShiftX += 1;
        ShiftX *= (m_W_Remap.supportSize() - 1);
        Point<VDIM> ShiftV;
        ShiftV = ShiftV.getUnitv(0);
        ShiftV *= 0;
        ShiftV += 1;
        ShiftV *= (m_W_Remap.supportSize() - 1);

        Point<XDIM> HighCornerX(iposHighX);
        Point<XDIM> LowCornerX(iposLowX);
        Point<VDIM> HighCornerV(iposHighV);
        Point<VDIM> LowCornerV(iposLowV);
        Point<XDIM> LCX = LowCornerX - ShiftX;
        Point<XDIM> HCX = HighCornerX + ShiftX;
        Point<VDIM> LCV = LowCornerV - ShiftV;
        Point<VDIM> HCV = HighCornerV + ShiftV;
        DBox<XDIM> SupportBoxX(LCX, HCX);
        DBox<VDIM> SupportBoxV(LCV, HCV);

        // loops over support box in phase space
        for (Point<VDIM> sV = SupportBoxV.getLowCorner(); SupportBoxV.↩
            notDone(sV); SupportBoxV.increment(sV))
        {
            Point<VDIM> moV = sV;
            moV += Npv;
       pointVelocitySpaceMod(moV);
            for (Point<XDIM> sX = SupportBoxX.getLowCorner(); SupportBoxX.↩
                notDone(sX); SupportBoxX.increment(sX))
            {
                //compute key and rank for phase space grid point
                Point<XDIM> moX = sX;
                pointRealSpaceMod(moX);
                unsigned long long int Key = m_MortonOrder(moX, moV);
                int rank = computeHashRank(Key);
                //compute interpolated value at phase space grid point
                double KernelProduct = 1.0;
                for (int dir =0; dir<XDIM; dir++)
                {
                    int regionX = min(abs(sX[dir] - LowCornerX[dir]), abs(sX↩
                        [dir] - HighCornerX[dir]));
                    double valX = abs(sX[dir]*(m_hx*m_L) - posX[dir])/(m_hx*↩
                        m_L);
                    KernelProduct *= m_W_Remap.apply(valX, regionX);
                }
                for (int dir =0; dir<VDIM; dir++)
                {
                    int regionV = min(abs(sV[dir] - LowCornerV[dir]), abs(sV↩
                        [dir] - HighCornerV[dir]));
                    double valV = abs(sV[dir]*(m_hv*m_vmax) - posV[dir])/(↩
                        m_hv*m_vmax);
                    KernelProduct *= m_W_Remap.apply(valV, regionV);
                }

                // read, increment, write Particle2Send
                //If Particle does not exist, it member data is initialized ↩
                    to 0
                Particle2Send p2s = message_queues[rank][Key];
                p2s.strength += t_particles[j].strength*KernelProduct;
```

```cpp
                p2s.key = Key;

                for (int dir=0; dir<XDIM; dir++)
                {
                    p2s.m_x[dir]=moX[dir]*m_hx*m_L;
                }
                for (int dir=0; dir<VDIM; dir++)
                {
                    p2s.m_v[dir]=sV[dir]*m_hv*m_vmax;
                }
                message_queues[rank][Key] = p2s;

                // push local_queue if it is larger than some specified size.
                if (message_queues[rank].size() >= message_size)
                {
                    vector<Particle2Send> local_buffer;
                    local_buffer.reserve(message_queues[rank].size());
                    for (auto iter = message_queues[rank].begin(); iter!= message_queues[rank].end(); ++iter)
                        local_buffer.push_back(iter->second);
                    auto future = queues[rank].push(std::move(local_buffer));
                    if (!future.has_value())
                    {
                        throw std::runtime_error("error: Queue on " + std::to_string(rank) +
                                "out of space");
                    }
                    message_queues[rank].clear();
                    futures.emplace_back(std::move(future.value()));
                }
            }
        }
    }


    // Flush any remaining queues.
    for (int rankIter = 0; rankIter < message_queues.size(); rankIter++) {
        vector<Particle2Send> local_buffer;
        local_buffer.reserve(message_queues[rankIter].size());
        for (auto iter = message_queues[rankIter].begin(); iter!= message_queues[rankIter].end(); ++iter)
            local_buffer.push_back(iter->second);
        auto future = queues[rankIter].push(std::move(local_buffer));
        if (!future.has_value()) {
            throw std::runtime_error("error: Queue on " + std::to_string(rankIter) +
                    " out of space");
        }
        futures.emplace_back(std::move(future.value()));
    }

    for (auto& future: futures) {
        future.get();
    }

    BCL::barrier();

    *(m_RemapPtr[BCL::rank()].local()) = false;
    std::vector <Particle2Send> my_bucket = queues[BCL::rank()].as_vector();

    // Places particles into morton-ordered vector
    map<unsigned long long int, Particle> map2ZOrder;
    for(int k = 0; k<my_bucket.size(); k++)
    {
        map2ZOrder[my_bucket[k].key].strength += my_bucket[k].strength;
        map2ZOrder[my_bucket[k].key].key = my_bucket[k].key;
        map2ZOrder[my_bucket[k].key].m_x = my_bucket[k].m_x;
        map2ZOrder[my_bucket[k].key].m_v = my_bucket[k].m_v;
        map2ZOrder[my_bucket[k].key].alpha_x = my_bucket[k].m_x;
        map2ZOrder[my_bucket[k].key].alpha_v = my_bucket[k].m_v;
    }
    // Replaces input particles by remapped particles
    my_bucket.clear();
    t_particles.clear();
    t_particles.reserve(map2ZOrder.size());
    for(auto iter=map2ZOrder.begin(); iter!=map2ZOrder.end(); ++iter)
        t_particles.push_back(iter->second);

    // Tracks the total number of new particles generated by remapping
    t_numParticlesRemapped += t_particles.size();
}
```

Recall the pseudocode for the local remapping algorithm from Chapter 4. Utilizing this global remapping function, remapping based on the implicit transition function is quite simple. We give the code for this below where the transition function used for all tests is $\chi((x-0.1)/0.9)$ with

$$\chi(x) = \begin{cases} x^3(x(6x-15)+10), & 0 < x < 1 \\ 1 & x \geq 1 \\ 0 & x \leq 0 \end{cases} \tag{6.1}$$

where x is the remapping condition for a particle normalized by the bound C or C_{dist}. The remapping region is 0 when this normalized remapping condition is less than 0.1 and 1 when the condition is 1 or greater. This means that the transition function goes from 0 to 1 in the span of an order of magnitude of the remapping condition.

```
void ParticleSet::contourRemap(double zeroTFVal, double oneTFVal)
{
   vector<Particle> oldParticles = m_particles;

    // charge <- (1 - transistionFunction)*charge
   for(auto& p : oldParticles)
   {
      m_TransitionFunction(oneTFVal, zeroTFVal, p, true);
   }
   eraseSmallParticles(oldParticles);

    // charge <- transistionFunction*charge
   for(auto& p : m_particles)
   {
      m_TransitionFunction(oneTFVal, zeroTFVal, p);
   }
   eraseSmallParticles(m_particles);
   globalRemap(m_particles);
   copy(oldParticles.begin(), oldParticles.end(), back_inserter(m_particles)↩
       );
   oldParticles.clear();
   eraseSmallParticles(m_particles);
   //eraseNegativeParticles(m_particles);
   *(m_RemapPtr[BCL::rank()].local()) = false;
};
```

Chapter 7

Conclusion

Previous error estimates for PIC methods have provided no insight into how the deformation of a set of particles initialized on a rectangular grid leads to errors that can accumulate in the trajectories of the particles. We have developed a framework that begins to understand these errors. We now understand the errors that arise in the deposition stage of the PIC algorithm due to deformations in the configuration space of the particles for $N_x = 1$. From analysis of the (1+1)D model problem, we found that an $O(1)$ error, given in full in (3.53), of the form $(F_x - 1)^M f$ appeared where M depends on the interpolation kernel used for deposition. The form this error takes indicates that regions of phase-space which carry more mass must continue to be approximately on a rectangular grid in configuration space. When the $O(1)$ error became too large relative to other errors in the simulation, we applied remapping. Remapping allowed us to control the source of the $O(1)$ error by interpolating the distribution to a rectangular grid in phase-space and thus set the source of the error to 0 for the next iteration of deposition. We did not invent remapping; however, it has always been used in an ad-hoc way that was not based on a mathematically systemic analysis of the error, such as provided here. This typically meant that remapping was applied with a fixed frequency that was determined empirically for the application. Our principled approach allowed us to not only remap only when necessary but also remap locally in phase-space. By remapping to control the $O(1)$ error, we were able to eliminate the "particle noise" in the test problems. In fact, by varying the bound on the $O(1)$ error we demonstrated that we could control the degradation in accuracy for the (1+1)D test problems.

For the remainder of the concluding remarks we will focus on what lessons we have learned in the course of this work that can be applied to future work. There are four orthogonal directions that could expand upon this work:

1. Understand errors associated with deposition in $N_x > 1$
2. Investigate the performance implications and bottlenecks of high-order PIC with adaptive remapping
3. Determine errors from deformation of the particles for deposition of current

4. Understand the stability implications of (2.52)

Points one and two need to be addressed to understand the errors associated with PIC for Vlasov-Poisson in (3+3)D and efficiently control these errors. The first three points would need to be addressed to handle the more general PIC application, Vlasov-Maxwell in full (3+3)D phase-space. The fourth point is required to fully understand the mechanism with which the $O(1)$ error feeds back into the positions and velocities of the particle through the field solve and force interpolation stages of the PIC algorithm.

To consider kinetics problems in (2+2)D and (3+3)D, this framework needs to be expanded to understand deposition in multiple configuration space dimensions. We have shown that the change of variables for the continuous analog of deposition still depends on the determinant of the configuration space deformation gradient, independent of dimensionality of the problem. This indicates that for $N_x > 1$ that this quantity is likely to arise in the $O(1)$ error, similar to how it arose in $N_x = 1$. Having more than one configuration space dimension provides more degrees of freedom for the distribution to stray from the initial representation on a rectangular grid. Developing a full suite of model problems that addresses these will be instrumental in understanding the deposition error in $N_x > 1$. Early results for a two dimensional model problem similar in spirit to the one developed in Chapter 2 is already bearing fruit in on going work. The analysis of these model problems suggests even rigid-body rotation and incompressible strain in configuration space (so $\det(\boldsymbol{F}_x) = 1$) lead to errors that must be controlled through remapping. This suggests that even these seemingly benign transformations of the initial particle configuration will lead to errors in the charge density for PIC problems with two or more configuration space dimensions.

We are able to track the configuration space deformation gradient, F_x, for each particle and able to compute the $O(1)$ error for an individual particle. This allows us to define regions of phase-space for which the $O(1)$ error is above some threshold and remap locally in that region. Stitching the distribution of particles that are not remapped and the particles remapped locally back into one distribution needs to be done carefully. Specifically, the boundaries of these regions cannot cut through places where f is of similar magnitude to our $O(1)$ error. This is a direct consequence of the application of the Euler-Maclaurin formula to the approximation of the deposition integral. If we do not respect this condition we lose the arbitrary order accuracy for deposition. This lead us to define contours that control both the $O(1)$ error as well as the error coming from the Euler-Maclaurin formula (see (4.22)). We found that remapping locally based on these contours could reduce the cost of remapping when compared to global remapping. However, this was not true in all cases that we tested. We found that when the total number of remaps was significantly more for local remapping than global remapping that the volume of phase-space remapped over the duration of a simulation was greater for local remapping than global remapping. Adaptively controlling the C_{dist} parameter in the definition of the contours would allow for more consistent performance increases for local remapping. One way to get insight into when local remapping is beginning to require many more remaps than global remapping would be to use an extra set of particles that are sub-sampled and are evolved based on the electric field generated by the full simulation but globally remapped whenever the $O(1)$ error reaches the specified threshold. These particles would not feed into the algorithm but would allow for the tracking of the number of global remaps required for a simulation without having to run a test with global remapping. This could then be compared to the number of local remaps being performed in the simulation for the full set of particles. The C_{dist} parameter could be decreased based on whether using local remapping is requiring significantly more remaps than global remapping. Optimizing local remapping will become increasingly important as the dimension is increased and the of cost of remapping grows.

For both local and global remapping we demonstrated that using an implicit representation of the phase-space grid is feasible and will be important as the dimension of the problem increases. We did not study the performance of our current implementation. Comparing and contrasting it to other means of implementing an implicit rectangular phase-space grid would be valuable. Other partitions of the phase-space among ranks and methods of implementing a distributed map are specific areas that we have not studied here but will impact performance of remapping. This investigation needs to be a portion of a larger investigation into the performance of an adaptively remapped PIC method. PIC methods, as described in Chapter 3, are memory-bound. We have shown that we can in fact increase the order of an adaptively remapped PIC method by increasing the number of moment conditions that the deposition kernel satisfies. These high-order kernels have a larger support and therefore increase the arithmetic intensity for the deposition, interpolation, and remapping. This can be used to increase the throughput for a given precision on modern systems by moving the algorithm further up and to the right on the memory-bound region or, ideally, moving the algorithm into the computationally-bound regime.

The robustness of this theory in the (1+1)D and (1+2)D numerical experiments and how similar the $O(1)$ errors appeared to those in the one-dimensional model problem suggests that the framework is building upon the correct ideas. The most important being that pulling back into the Lagrangian coordinates to evaluate the errors is the correct way to approach the analysis for deposition in PIC methods. To extend this theory to the Vlasov-Maxwell system, a similar model problem and pull back to the Lagrangian frame would be necessary for the current deposition stage. The main difference between the deposition of charge and current is a factor of v in the integrand. One issue that is unique to current deposition is that direct deposition does not automatically satisfy the continuity equation. For direct current deposition this has lead to the use of the Boris correction. This analysis framework would be extremely useful in judging other methods of depositing current that are charge conserving.

The model problem presented in Chapter 2 does not address the effects that the $O(1)$ error has on stability in the context of PIC algorithms. In this example the velocity was computed exactly; however, a better approximation to a PIC algorithm would be to advect the particles for Δt and then recompute the velocity by doing a particle-to-grid and grid-to-particle interpolation. One could imagine solving Burgers' equation like this and computing the error for each Δt. This would give insights into how the errors cause by repeated deposition accumulates in the velocity and position of the particles. Extending this to PIC for kinetics requires understanding the effect that solving Poisson's equation has on this feedback loop. Beyond the theoretical importance of understanding the stability, this is important since it will have implications on how often remapping is required. The smoothing properties of solving Poisson's equation might make much larger $O(1)$ errors tolerable than would be expected based on (3.53).

All of these points need to be addressed to implement a high-order Vlasov-Maxwell solver in (3+3)D that controls the errors arising from deformations to the set of particles. We believe that this work is an important first step and provides valuable insight into how to take the next necessary steps.

Bibliography

[1] Milton Abramowitz and Irene A Stegun. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. Vol. 55. US Government printing office, 1964.

[2] Ann S Almgren, Thomas Buttke, and Phillip Colella. “A fast adaptive vortex method in three dimensions”. In: *Journal of computational Physics* 113.2 (1994), pp. 177–200.

[3] Christopher Anderson and Claude Greengard. “On vortex methods”. In: *SIAM journal on numerical analysis* 22.3 (1985), pp. 413–440.

[4] Christopher R Anderson. “A method of local corrections for computing the velocity field due to a distribution of vortex blobs”. In: *Journal of Computational Physics* 62.1 (1986), pp. 111–123.

[5] Kendall E Atkinson. *An introduction to numerical analysis*. John Wiley & Sons, 2008.

[6] J Thomas Beale and Andrew Majda. “High order accurate vortex methods with explicit velocity kernels”. In: *Journal of Computational Physics* 58.2 (1985), pp. 188–208.

[7] Charles K Birdsall and A Bruce Langdon. *Plasma physics via computer simulation*. CRC press, 2018.

[8] Jeremiah U Brackbill and Hans M Ruppel. “FLIP: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions”. In: *Journal of Computational physics* 65.2 (1986), pp. 314–343.

[9] Benjamin Brock, Aydın Buluç, and Katherine Yelick. “BCL: A cross-platform distributed data structures library”. In: *Proceedings of the 48th International Conference on Parallel Processing*. 2019, pp. 1–10.

[10] Oscar Buneman. “Dissipation of currents in ionized media”. In: *Physical Review* 115.3 (1959), p. 503.

[11] José Canosa, Jenö Gazdag, and JE Fromm. “The recurrence of the initial state in the numerical solution of the Vlasov equation”. In: *Journal of Computational Physics* 15.1 (1974), pp. 34–45.

[12] Alina Chertock. “A Practical Guide to Deterministic Particle Methods”. In: *Handbook of Numerical Analysis*. Vol. 18. Elsevier, 2017, pp. 177–202.

[13] Stephen Childress. *An introduction to theoretical fluid mechanics*. Vol. 19. American Mathematical Soc., 2009.

[14] IP Christiansen. “Numerical simulation of hydrodynamics by the method of point vortices”. In: *Journal of Computational Physics* 13.3 (1973), pp. 363–379.

[15] Phillip Colella. “Volume-of-fluid methods for partial differential equations”. In: *Godunov Methods*. Springer, 2001, pp. 161–177.

[16] G-H Cottet and Philippe Poncet. “Advances in direct numerical simulations of 3D wall-bounded flows by vortex-in-cell methods”. In: *Journal of Computational Physics* 193.1 (2004), pp. 136–158.

[17] G-H Cottet et al. “High order semi-Lagrangian particle methods for transport equations: numerical analysis and implementation issues”. In: *ESAIM: Mathematical Modelling and Numerical Analysis* 48.4 (2014), pp. 1029–1060.

[18] Georges-Henri Cottet. “Semi-Lagrangian particle methods for high-dimensional Vlasov–Poisson systems”. In: *Journal of Computational Physics* 365 (2018), pp. 362–375.

[19] Georges-Henri Cottet and Petros Koumoutsakos. “High order semi-lagrangian particle methods”. In: *Spectral and High Order Methods for Partial Differential Equations ICOSAHOM 2016*. Springer, 2017, pp. 103–117.

[20] Georges-Henri Cottet and Petros D Koumoutsakos. *Vortex methods: theory and practice*. Cambridge university press, 2000.

[21] Nicolas Crouseilles, Thomas Respaud, and Eric Sonnendrücker. “A forward semi-Lagrangian method for the numerical solution of the Vlasov equation”. In: *Computer Physics Communications* 180.10 (2009), pp. 1730–1745.

[22] John Dawson. “One-dimensional plasma model”. In: *The Physics of Fluids* 5.4 (1962), pp. 445–459.

[23] John M Dawson. “Particle simulation of plasmas”. In: *Reviews of modern physics* 55.2 (1983), p. 403.

[24] RA Dory, GE Guest, and EG Harris. “Unstable electrostatic plasma waves propagating perpendicular to a magnetic field”. In: *Physical Review Letters* 14.5 (1965), p. 131.

[25] David S Ebert et al. *Texturing & modeling: a procedural approach*. Morgan Kaufmann, 2003.

[26] Essex Edwards and Robert Bridson. “A high-order accurate particle-in-cell method”. In: *International Journal for Numerical Methods in Engineering* 90.9 (2012), pp. 1073–1088.

[27] Martha W Evans, Francis H Harlow, and Eleazer Bromberg. *The particle-in-cell method for hydrodynamic calculations*. Tech. rep. LOS ALAMOS NATIONAL LAB NM, 1957.

[28] Francesco Faa di Bruno. “Sullo sviluppo delle funzioni”. In: *Annali di scienze matematiche e fisiche* 6 (1855), pp. 479–480.

[29] Leslie Greengard and Vladimir Rokhlin. "A fast algorithm for particle simulations". In: *Journal of computational physics* 73.2 (1987), pp. 325–348.

[30] Ole H Hald. "Convergence of vortex methods for Euler's equations. II". In: *SIAM Journal on Numerical Analysis* 16.5 (1979), pp. 726–755.

[31] Francis H Harlow. "The particle-in-cell computing method for fluid dynamics". In: *Methods Comput. Phys.* 3 (1964), pp. 319–343.

[32] Roger W Hockney and James W Eastwood. *Computer simulation using particles.* crc Press, 2021.

[33] Petros Koumoutsakos and A Leonard. "High-resolution simulations of the flow around an impulsively started cylinder using vortex methods". In: *Journal of Fluid Mechanics* 296 (1995), pp. 1–38.

[34] Boris Lo and Phillip Colella. "An adaptive local discrete convolution method for the numerical solution of Maxwell's equations". In: *Communications in Applied Mathematics and Computational Science* 14.1 (2019), pp. 105–119.

[35] Joe J Monaghan. "Why particle methods work". In: *SIAM Journal on Scientific and Statistical Computing* 3.4 (1982), pp. 422–433.

[36] Guy M Morton. "A computer oriented geodetic data base and a new technique in file sequencing". In: (1966).

[37] Andrew Myers, Phillip Colella, and B Van Straalen. "A 4th-Order Particle-in-Cell Method with Phase-Space Remapping for the Vlasov–Poisson Equation". In: *SIAM Journal on Scientific Computing* 39.3 (2017), B467–B485.

[38] Andrew Myers, Phillip Colella, and Brian Van Straalen. "The convergence of particle-in-cell schemes for cosmological dark matter simulations". In: *The Astrophysical Journal* 816.2 (2016), p. 56.

[39] Pierre-Arnaud Raviart. "An analysis of particle methods". In: *Numerical methods in fluid dynamics.* Springer, 1985, pp. 243–324.

[40] Deborah Sulsky, Shi-Jian Zhou, and Howard L Schreyer. "Application of a particle-in-cell method to solid mechanics". In: *Computer physics communications* 87.1-2 (1995), pp. 236–252.

[41] HD Victory Jr and Edward J Allen. "The convergence theory of particle-in-cell methods for multidimensional Vlasov–Poisson systems". In: *SIAM journal on numerical analysis* 28.5 (1991), pp. 1207–1241.

[42] GV Vogman, Phillip Colella, and Uri Shumlak. "Dory–Guest–Harris instability as a benchmark for continuum kinetic Vlasov–Poisson simulations of magnetized plasmas". In: *Journal of Computational Physics* 277 (2014), pp. 101–120.

[43] Bei Wang. "A Particle-in-cell Method with Adaptive Phase-space Remapping for Kinetic Plasmas". PhD thesis. University of California at Davis, 2011.

[44] Bei Wang, Greg Miller, and Phil Colella. “An Adaptive, High-Order Phase-Space Remapping for the Two Dimensional Vlasov–Poisson Equations”. In: *SIAM Journal on Scientific Computing* 34.6 (2012), B909–B924.

[45] Bei Wang, Gregory H Miller, and Phillip Colella. “A particle-in-cell method with adaptive phase-space remapping for kinetic plasmas”. In: *SIAM Journal on Scientific Computing* 33.6 (2011), pp. 3509–3537.

[46] Paul Woodward and Phillip Colella. “The numerical simulation of two-dimensional fluid flow with strong shocks”. In: *Journal of computational physics* 54.1 (1984), pp. 115–173.

[47] Fuzhen Zhang. *The Schur complement and its applications*. Vol. 4. Springer Science & Business Media, 2006.