9-9-2010

An arbitrary curvilinear coordinate particle in cell method

Christopher Fichtl

Christopher A. Fichtl

Candidate

Chemical and Nuclear Engineering

Department

This dissertation is approved, and it is acceptable in quality and form for publication:

Approved by the Dissertation Committee:

[illegible], Chairperson

[illegible]

[illegible]

[illegible]

[illegible]

An Arbitrary Curvilinear Coordinate Particle In Cell Method

by

Christopher A. Fichtl

B.A., Physics, Hastings College, 2002

M.S., Nuclear Engineering, University of New Mexico, 2005

DISSERTATION

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

Engineering

The University of New Mexico

Albuquerque, New Mexico

July 1, 2010

Dedication

"This is presented as a work of fiction and dedicated to nobody"

- Bukowski (Post Office)

Acknowledgments

First and foremost, I want to extend a huge thank you to my family for their never-ending love and support during this process. To my wife Erin, for all the little things she has done to make this whole experience more bearable, especially during the last few months of late nights and early mornings at the office. To my children, Caden and Keira, who brighten my days just by being their crazy, adorable selves, and to my sister-in-law Lauren, who went out of her way to help me by watching the kids in the afternoons so I could put in a few more hours at work. I know I don't say it enough, but I love you all and appreciate everything you do for me.

I would especially like to thank my thesis advisor, John Finn, for taking me under his wing during trying times and then continuing to share his wealth of knowledge and love of learning for the next two and a half years. Your value to me as both a friend and mentor is beyond words. Huge thank-you's are also in order for Keith Cartwright and Luis Chacón for all their help with the numerous problems I have encountered during the completion of this work. I am also grateful to Gian Luca Delzanno for providing the initial ideas which eventually lead to this thesis and for all the help provided over the years. Special thanks are also extended to Vadim Royterstein for innumerable inciteful conversations about all things PIC-related and for correcting my horrible engrish. Finally, I would like to thank my committee members, Anil Prinja, Norm Roderick, Keith Cartwright, and Mark Gilmore for their efforts in reading this dissertation and providing insightful comments.

An Arbitrary Curvilinear Coordinate Particle In Cell Method

by

Christopher A. Fichtl

ABSTRACT OF DISSERTATION

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Doctor of Philosophy
Engineering

The University of New Mexico

Albuquerque, New Mexico

July 1, 2010

An Arbitrary Curvilinear Coordinate Particle In Cell Method

by

Christopher A. Fichtl

B.A., Physics, Hastings College, 2002
M.S., Nuclear Engineering, University of New Mexico, 2005

Ph.D., Nuclear Engineering, University of New Mexico, 2010

Abstract

A new approach to the kinetic simulation of plasmas in complex geometries, based on the Particle-in-Cell (PIC) simulation method, is explored. In this method, called the Arbitrary Curvilinear Coordinate PIC (ACC-PIC) method, all essential PIC operations are carried out on a uniform, unitary square logical domain and mapped to a nonuniform, boundary fitted physical domain.

We utilize an elliptic grid generation technique known as Winslow's method to generate boundary-fitted physical domains. We have derived the logical grid macroparticle equations of motion based on a canonical transformation of Hamilton's equations from the physical domain to the logical. These equations of motion are not seperable, and therefore unable to be integrated using the standard Leapfrog method. We have developed an extension of the semi-implicit Modified Leapfrog (ML) integration technique to preserve the symplectic nature of the logical grid particle mover. We constructed a proof to show that the ML integrator is symplectic for systems of arbitrary dimension. We have constructed a generalized, curvilinear coordinate formulation of Poisson's equations to solve for the electrostatic fields on the uniform logical grid. By our formulation, we supply the plasma charge density on the logical grid as a source term. By the formulations of the logical grid particle mover and the field equations, the plasma particles are weighted to the uniform logical grid and the self-consistent mean fields obtained from the solution of the Poisson equation are interpolated to the particle position on the logical grid. This process eliminates the complexity associated with the weighting and interpolation processes on the nonuniform physical grid.

In this work, we explore the feasibility of the ACC-PIC method as a first step towards building a production level, time-adaptive-grid, 3d electromagnetic ACC-PIC code. We begin by combining the individual components to construct a 1d, electrostatic ACC-PIC code on a stationary nonuniform grid. Several standard physics tests were used to validate the accuracy of our method in comparison with a standard uniform grid PIC code. We then extend the code to two spatial dimensions and repeat the physics tests on a rectangular domain with both orthogonal and nonorthogonal meshing in comparison with a standard 2d uniform grid PIC code. As a proof of principle, we then show the time evolution of an electrostatic plasma oscillation on an annular domain obtained using Winslow's method.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Traditionally, there have been two approaches to the computational modeling of plasmas. One emphasizes the fluid nature of the plasma system by solving the fluid equations assuming appropriate closures are chosen [1]. The other emphasizes the kinetic interactions between the plasma particles and electromagnetic fields. The fluid approach is more appropriate for treating large scale properties of plasmas involving mass, momentum, and flux transport, while the kinetic description provides a more accurate treatment of localized, short-scale processes in collisionless plasmas [2]. In this work, we are solely concerned with the kinetic approach to plasma simulation.

The usual basis for analytic treatment of the evolution of a collisionless plasma is the Vlasov equation [3]:

$$\frac{\partial f_\alpha}{\partial t} + \vec{v} \cdot \nabla_{\vec{x}} f_\alpha + \frac{q_\alpha}{m_\alpha}(\vec{E} + \vec{v} \times \vec{B}) \cdot \nabla_{\vec{v}} f_\alpha = 0, \tag{1.1}$$

where $f_\alpha(\vec{x}, \vec{v}, t)$ is the time-dependent six-dimensional phase space distribution function of the plasma particles of species α. The Vlasov equation is valid in the regime in which particle-particle and higher correlations (collisions) can be ignored, and therefore the electric and magnetic fields, $\vec{E}$ and $\vec{B}$, in Eqn. (1.1) are *mean* fields that obey Maxwell's equations. Mean field theory says that the electric and magnetic fields seen by the particle are due to the global charge density and currents, thus the Vlasov equation requires a long-range force field.

While there are numerous models for solving the Vlasov-Maxwell equations (see for example Refs. [4–6]), the most common approach is known as the Particle-in-Cell (PIC) method. The basic approach of the PIC model is to represent the phase space distribution f_α as a collection of *macroparticles*, each representing a given number of physical plasma particles. The macroparticles evolve self-consistently in time according to Maxwell's equations and the Newton-Lorentz force equations. As such, PIC is capable of retaining most of the physics involved in a collisionless plasma system, including kinetic and nonlinear effects which cannot be reproduced by fluid models.

Ironically, the basis for the PIC method was developed by Harlow [7] in 1955 as a method to study fluid problems. It was soon after extended to kinetic particle simulations by Buneman [8] and Dawson [9]. These precursors to the modern PIC method used point particles and calculated the forces on each particle via a direct summation of Coulomb's law. In the following two decades, improvements were made, many of which are still in use today. Perhaps the most fundamental improvement was the introduction of the spatial grid [10, 11] with which the particles are coupled. In the PIC method, charge and current densities are accumulated on the grid by *weighting* the particle charges and velocities to the grid with an interpolation function. Maxwell's equations are then solved on the grid using the charge and current density as source terms. The resulting mean fields are then *interpolated* from the grid to the particle positions so that forces can be calculated and the particle positions and velocities updated.

Throughout this thesis, we refer to the process of transferring information from the particles to the grid as either accumulation or weighting, whereas the process of transferring information from the grid to the particles is referred to as interpolation.

These two steps are crucial to the accuracy of the PIC method, but they are responsible for the introduction of "particle noise" into the system as we expect to have some statistical fluctuation in the number of particles per cell as our simulation advances in time. These fluctuations cause nonuniformities in the grid density, which are then perpetuated through the subsequent field solve and interpolation steps. Smoother (broader) weighting functions allow for more control over these density fluctuations as particles contribute a fraction of their charge to more cells, leading to smoother density distributions on the grid and reduced noise within the system.

PIC codes are generally designed using rectangular meshes in Cartesian geometry, but have been extended to cylindrical and spherical geometries. However, extension to arbitrary grid shapes has proven much more difficult. Jones [12] was among the first to develop a curvilinear-coordinate PIC method capable of operating on boundary-conforming grids tailored to accelerator and pulsed-power applications. Soon after, other codes were developed along similar lines in an effort to model ion diodes [13, 14] and microwave devices [15] more accurately. These methods involved generating a nonuniform initial grid based upon the physical boundaries of the system and running the PIC components on this physical grid. There are many benefits to this type of system, such as smoothly curved boundaries in contrast to the "stair-stepped" boundaries inherent to the rectangular-grid PIC approach. Furthermore, higher grid density can be placed in areas of interest within the system either statically or by allowing the grid to adapt dynamically [16, 17] to the problem by following a prespecified control function. Implemented wisely, such techniques should allow complex geometries to be simulated at a fraction of the cost associated with using a uniform grid code in which the entire mesh must assume the same resolution as is required to resolve the smallest physical features of the system.

These methods are not without their problems, however. Nonuniform grid cells make it difficult to locate macroparticles on the grid for the charge accumulation and field interpolation steps of the PIC method, as well as for enforcing the particle boundary conditions. In a standard PIC code, particle positions on the uniform grid are easily determined by dividing the current particle position by the length of the cells in the system. On a nonuniform grid, this location must be done iteratively [18], which increases the computational cost of the method. Furthermore, interpolation on a nonuniform grid becomes much more complicated, as the variations of the grid change the shape of the interpolation functions, often in a non-trivial way [19]. Finally, as the ratio of the largest to the smallest cell size increases and the number of particles per cell in the smallest cells becomes small, the amount of noise near the small cells also increases (assuming the charge per particle is constant). Thus, careful and often time-consuming gridding strategies must be utilized in such a way that the noise within the system is kept to a minimal threshold value while the structures of interest within the system are still resolved. It is therefore worth considering whether, for the problem of interest, nonuniform grid methods are sufficiently more accurate to justify their use over the alternate approach of gridding the hell out of a problem with a uniform grid code, inserting billions of particles to keep noise levels low, and running it on the largest computer you can find (with triply-periodic boundary conditions), but with great computer science techniques applied to make it run faster [20].

More recently, techniques have been devised which allow uniform grid PIC codes to model complex problems more efficiently by coupling them to other methods such as the Immersed-Boundary (IB) method [21] or adaptive mesh refinement (AMR) [22]. With the IB technique, a set of particles is introduced to fill a region (or form a boundary) occupied by an object. These particles are assigned properties suitable for describing the object they represent. In this way, the field equations can be solved everywhere on a uniform grid, thereby avoiding some of the troubles associated with grid generation techniques. Unfortunately, the IB method still requires that in order to resolve areas of interest, the entire grid must be refined. Furthermore, without the addition of some external grid adaptation strategy, the grid cells remain rectangular and fields still experience the effects of the stair-stepped boundary.

The AMR technique applies "patches" of finer gridding within the physical domain of the system. In this method, regions of interest can be more finely gridded while regions of less importance can be more coarsely gridded. If the areas of the domain that require higher resolution evolve in time, the AMR technique allows for automatic redistribution of the refined mesh patches following some specified criterion. However, since the AMR technique involves a jump from one level of resolution to another, it often leads to a spurious self-force on particles close to the patch boundaries. Furthermore, since the grid does not vary continuously, AMR can lead to lack of conservation of total charge within the system and electromagnetic models can see wave reflection off the boundary of the patch. Much work has been put into controlling and even eliminating these problems [23]. However, the AMR method, when it is used, is incorporated in standard uniform mesh PIC codes. Thus, complex boundaries are still modeled using uniformly-shaped grids, leading to stairstepping and its associated inaccuracies.

We consider the problems associated with these and other existing adaptive grid PIC approaches to be serious. As such, we have developed a new approach to the nonuniform grid PIC method, in which we attempt to adapt some of the best features of several existing methods to a revolutionary new idea for the implementation of the PIC method. Our ultimate goal is to design a arbitrary, curvilinear-coordinate (ACC) PIC code capable of operating efficiently and accurately on an arbitrary moving mesh with grid shapes in multiple geometries by implementing the main components of the PIC method–the charge accumulation, particle push, field solve and interpolation–on a uniform, unit square *logical* (or computational) domain and coupling them with state-of-the-art methods for moving mesh generation and particle handling.

The building blocks necessary for reaching our ultimate goal of a 3d, electromag-

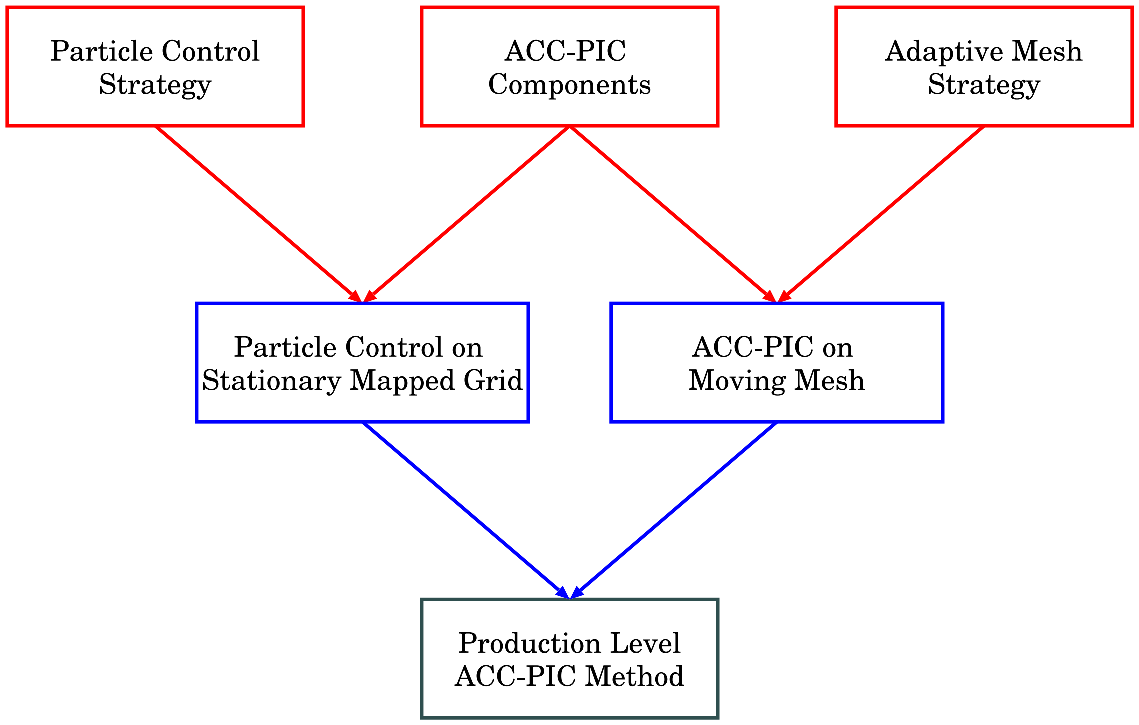

Particle Control Strategy
ACC-PIC Components
Adaptive Mesh Strategy
Particle Control on Stationary Mapped Grid
ACC-PIC on Moving Mesh
Production Level ACC-PIC Method

Figure 1.1: Path envisioned for development of production level ACC-PIC method.

netic, time-adaptive ACC-PIC code capable of handling large-scale simulations are shown schematically in Fig. 1.1. We envision several key areas for the development of this new method, in which a set of key components can initially be developed and benchmarked separately, then combined with the ultimate goal of developing a production level PIC code. We identify these key areas in the top row of Fig. 1.1.

As described above, nonuniform grids can lead to increased amounts of noise in the system due to differences in the number of particles populating the smallest grid cells. As such, a particle controlling strategy capable of efficiently and accurately control the particles within the system such that resolution is maintained in more important areas of the grid while also controlling the total number particles in the system must be developed. While several methods of particle and particle noise control currently exist [24–28], questions surround the accuracy and/or efficiency of each.

Another important ingredient is the development of an efficient and robust time adaptive mesh method. We require an adaptive mesh method capable of following a particular control parameter during the evolution of the physical system and automatically generating high quality (unfolded) meshes. Recent work on a Monge-Kantorovich grid adaptation approach by Delzanno, *et al.* [29] has shown great promise for satisfying our desired adaptive mesh qualities.

In this thesis, we concentrate on the keystone of our method: the development of the individual PIC components necessary for the ACC-PIC method. By implementing the PIC components on the logical grid, we expect to eliminate the particle location and interpolation problems that plagued the earlier boundary conforming non-uniform grid methods [12–15]. Particle locations are easily found on the logical grid using the same techniques as standard uniform grid PIC codes. Since the charge accumulation and field interpolation are both done on the logical grid, we are again able to use the same functions as are utilized in a uniform grid code. Furthermore, we develop a Hamiltonian-based, semi-implicit symplectic logical grid mover and apply it to the time advance of the particles that includes the effects of inertial forces. Since we are moving particles on a square grid, particle boundary conditions, which may be difficult to implement on a nonuniform physical grid, are fairly simple with our approach. Finally, we are able to solve the field equations on a simple square mesh in the logical domain rather than in the complex physical domain.

For this thesis, we have implemented Winslow's Laplacian method [30] as our grid generation tool. This method allows us to create a grid that not only exactly matches the boundary segments of the system, but also concentrates finer gridding in certain areas near the boundary, e.g. where the radius of curvature is smallest, via a smoothly varying grid. With methods such as this, we are able to avoid the discontinous jump in gridding that troubles the AMR approach but also retain the nonuniform gridding that is absent from the IB approach. Furthermore, since our method maps the nonuniform grid onto a uniform logical mesh, we are able to do all required differencing using simple central-difference techniques and allowing the mapping between our physical and logical grids to control the weighting of the difference scheme due to the nonuniformity of the grid. While we have chosen to implement a specific grid generation technique to demonstrate the feasibility of our method, in reality the initial grid can be generated by any means. We require only that a structured, non-folded initial physical grid be generated in such a way that it can be directly mapped onto the logical domain.

The remainder of this thesis is structured as follows. In Chapter 2, we provide a mathematical development of the standard PIC method for a one-dimenional electrostatic plasma system. This chapter is intended to give the reader an elementary understanding of the components of the PIC method to lay the foundation on which the rest of this thesis is built. Chapter 3 develops the methods central to our curvilinear PIC method. We detail our approach to grid generation, derive particle equations of motion on the logical grid, introduce the semi-implicit Modified Leapfrog (ML) integrator, and derive Poisson's equation on the logical grid. Chapter 4 incorporates the methods developed in Chapter 3 into an electrostatic, one dimensional (1d) curvilinear coordinate PIC code. Several standard tests are performed to validate our code. Chapter 5 extends the code developed in Chapter 4 to a 2d system, detailing the intricacies involved in extending the method to multiple dimensions. The standard tests are again applied as a validation tool. Chapter 6 provides our conclusions and some suggestions for future work on the method. In Appendix A, we provide the reader with some of the general background relations between Cartesian and curvilinear coordinates which are paramount to this thesis. In Appendix B, we provide a more detailed explanation of the staggered-grid interpolation and weighting schemes used throughout this thesis. Finally, in Appendix D, we provide pseudocode for the modified leapfrog mover as applied to the logical grid particle equations of motion in 2d.

Chapter 2

Mathematical Development of the Standard PIC Method

In a system in which the collective effects of long-range Coulomb interactions between plasma particles dominate the binary collision process, the evolution of the phase space distribution function $f_\alpha(\vec{x}, \vec{v}, t)$ for a given species α of plasma particles is governed by the Vlasov equation (Eqn. (1.1)) [2]. The electric and magnetic fields obey Maxwell's equations:

$$\nabla \cdot \vec{E} = \frac{\rho}{\epsilon_0} \tag{2.1a}$$

$$\nabla \cdot \vec{B} = 0 \tag{2.1b}$$

$$\nabla \times \vec{E} = -\frac{\partial \vec{B}}{\partial t} \tag{2.1c}$$

$$\nabla \times \vec{B} = \mu_0 \vec{J} + \mu_0 \epsilon_0 \frac{\partial \vec{E}}{\partial t}, \tag{2.1d}$$

where the charge and current density are given by

$$\rho = \sum_\alpha q_\alpha \int f_\alpha(\vec{x}, \vec{v}, t)\, d\vec{v} \tag{2.2a}$$

and

$$\vec{j} = \sum_{\alpha} q_{\alpha} \int \vec{v} f_{\alpha}(\vec{x}, \vec{v}, t)\, d\vec{v}, \tag{2.2b}$$

respectively. Every plasma particle moves according to the mean fields $\vec{E}$ and $\vec{B}$ according to the Newton-Lorentz force law:

$$\vec{F} = m_{\alpha} \frac{d\vec{v}}{dt} = q_{\alpha}(\vec{E} + \vec{v} \times \vec{B}). \tag{2.3}$$

To illustrate more clearly the fundamentals of the PIC method, we now restrict our system to a one dimensional, electrostatic plasma with no magnetic field such that the Vlasov-Maxwell system of equations (Eqns. (1.1) and (2.1)) described above reduces to the Vlasov-Poisson system, in which the electrostatic fields are updated via Poisson's equation:

$$\frac{d^2\Phi}{dx^2} = -\frac{\rho}{\epsilon_0}. \tag{2.4}$$

Here Φ is the electrostatic potential, related to the electric field by its gradient:

$$E^x = -\frac{d\Phi}{dx}. \tag{2.5}$$

2.1 Sampling Phase Space with Computational Macroparticles

In a particle-mesh (PM) method [11] such as PIC, computational macroparticles are used to represent a large number of plasma particles which are close to each other in phase space, thereby discretizing the Vlasov equation [31]. With this notion, we can define the macroparticle quantities

$$\begin{aligned} Q &= Kq \\ M &= Km \\ N &= \tfrac{n}{K}, \end{aligned} \tag{2.6}$$

where n is the plasma particle number density, N is the macroparticle number density, and K is the number of physical particles present in the element of phase space represented by a single macroparticle. For simplicity, we have assumed a single plasma species in which each macroparticle represents an equal number of physical particles, i.e. K is the same for all particles in the system. The use of macroparticles allows us to approximate the actual plasma distribution function as:

$$f(x,v,t) \approx \sum_p f_p(x,v,t), \tag{2.7}$$

with

$$K = \int f_p(x,v,t)dx\,dv. \tag{2.8}$$

One way of regarding these macroparticles is as finite sized clouds of "super-electrons" or "super-ions", with the position of the macroparticles being the center of mass of the charge cloud and their velocities being the mean velocities of the physical plasma particles. As such, we must assign a phase space shape to the macroparticles to sample accurately the plasma species distribution function:

$$f_p(x,v,t) = KS_x(x-x_p(t))S_v(v-v_p(t)). \tag{2.9}$$

Here S_x and S_v are the macroparticle shape functions in position and velocity, respectively. In the PIC method, the shape function in the momentum coordinate of phase space is chosen to be a Dirac delta function. We can then recast Eqn. (2.9) as

$$f_p(x,v,t) = KS_x(x-x_p(t))\delta(v-v_p(t)). \tag{2.10}$$

Since each macroparticle typically represents a large number of plasma particles, the binary collisional forces between two macroparticles are potentially much larger than those between two physical electrons or ions [32]. Thus, the choice of a non-delta function shape for the position coordinate, which for convenience we will write as $S(x)$, is governed by the need to soften these short-range interactions between macroparticles to the extent that they pass freely through each other, minimizing binary macroparticle interaction effects [32, 33] (not to be confused with binary collisional effects, which are not contained within the Vlasov description of the plasma).

At the same time, we also require that the support of the spatial shape function is kept relatively small in order to describe a small volume in phase space, meaning that the function $S(x)$ must go to zero outside a small range to retain the long-range Coulomb forces between macroparticles [34]. Furthermore, Eqn. (2.8) requires that the area of the shape function satisfies

$$\int_{-\infty}^{\infty} S(x)dx = 1. \tag{2.11}$$

Finally, Ockham's razor suggests that symmetric particle shapes should be chosen when possible [35]. We will return to this issue in more detail later.

While these rules do not seem to provide too much regulation on the shape functions chosen for particles, in practice only a small set of shape functions are generally used on uniform grids. The simplest shape function we can choose is known as the nearest grid point(NGP) method. However, as we will see in § 2.4, this shape function is discontinuous across grid cells, meaning it provides no direct connection between neighboring grid-cell quantities to reduce noise in the system due to discontinuities in the macroparticle density. As such, this shape function is rarely used in PIC codes [36]. In general, higher-order, more spatially continuous particle shape functions are generally chosen, based a series of consecutively smoother functions obtained from each other by a series of m convolutions with the NGP, or S_0 shape function [37]:

$$S_m(x) = S_0(x) * S_{m-1}(x) \tag{2.12}$$

where

$$S_0(x) = \frac{1}{\Delta x}\begin{cases} 1, & \text{for } |x| \le \frac{\Delta x}{2} \\ 0, & \text{otherwise} \end{cases} . \tag{2.13}$$

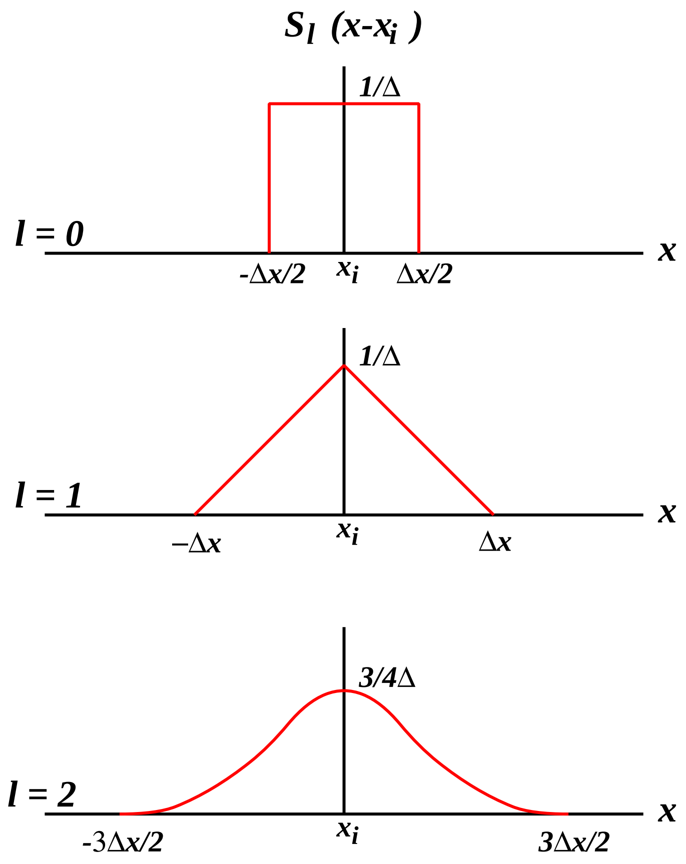


Figure 2.1: The first three macroparticle shape functions.

Here Δx is the spacing of our uniform 1d grid. As a result of the series of convolutions with S_0, the support of each consecutive shape function S_m increases in width by Δx and the particle has its order of continuity increased by one. Shape functions of this type are known as "splines," as they are based upon the same principal as the *b*-spline functions [38]. As can be seen in Fig. 2.1, the S_0 particle shape is a narrow, discontinuous function in position. In a system of many macroparticles, this discontinuity can lead to noise, which is then propogated thoughout the system as the simulation progresses. While higher-m shape functions may not increase the accuracy of the simulation, it is often more appropriate to use a higher-m shape function than S_0 due to the smoothing properties associated with their progressively wider supports and smoother (i.e. higher-order in continuity) shapes. [36]

Whereas the $S_0(x)$ particle shape (NGP) is $O(\Delta x)$, the linear shape function $S_1(x)$ is $O(\Delta x)^2$, and is constructed using $S_1(x) = S_0(x) * S_0(x)$. The shape function for this scheme is triangular, given by

$$S_1(x) = \frac{1}{\Delta x} \begin{cases} 1 - \frac{|x|}{\Delta x}, & \text{for } |x| \leq \Delta x \\ 0, & \text{otherwise} \end{cases} . \tag{2.14}$$

Note that while this piecewise linear function is continuous, its derivative not. For calculations requiring more smoothing, i.e. a continuous derivative, it is necessary to use higher order weighting. One of the inherent advantages of is that we can control the "smoothness" of the weighted quantity within the host cells by changing the order of the shape function spline. The second order, or quadratic, spline shape function ($S_2(x) = S_0(x) * S_0(x) * S_0(x) = S_1(x) * S_0(x)$) is piecewise parabolic and globally C^1 differentiable, and is given by

$$S_2(x) = \frac{1}{\Delta x} \begin{cases} \frac{3}{4} - \left(\frac{|x|}{\Delta x}\right)^2, & \text{for } |x| \leq \frac{\Delta x}{2} \\ \frac{1}{2}\left(\frac{3}{2} - \frac{|x|}{\Delta x}\right)^2, & \text{for } \frac{\Delta x}{2} \leq |x| \leq \frac{3\Delta x}{2} \\ 0, & \text{otherwise} \end{cases} . \tag{2.15}$$

Similarly, we can construct a piecewise cubic and globally C^2 differentiable third order (cubic) spline shape function:

$$S_3(x) = \frac{1}{\Delta x} \begin{cases} \frac{2}{3} - \left(\frac{|x|}{\Delta x}\right)^2 + \frac{|x|^3}{2\Delta x^3}, & \text{for } |x| \leq \Delta x \\ \frac{1}{6}\left(2 - \frac{|x|}{\Delta x}\right)^3, & \text{for } \Delta x \leq |x| \leq 2\Delta x \\ 0, & \text{otherwise} \end{cases} . \tag{2.16}$$

Notice that for successive shape function orders, the weighted quantity has its order of continuity increased by one. However, it is important to keep in mind that the order of spline weighting is related to the smoothness of the resulting quantity, not the accuracy. This means that while the weighting error is smoother within a cell for higher-order weighting schemes, they do not improve the accuracy of the numerical solver, except during the jump from the S_0 to S_1 shape functions. In fact, using higher order weighting methods uses grid cells beyond the nearest grid point, and thus smoothes the results by lowering the resolution at that particular point. [36]

It should also be noted that while we have defined the particle shape functions above in one dimension, they can be easily generalized to any spatial dimension by simply multiplying by the same shape function in each direction.

2.2 Macroparticle Equations of Motion

Vlasov's equation (Eqn. (1.1)) can be written in terms of each individual macroparticle as:

$$\frac{\partial \hat{f}_p}{\partial t} + v\frac{\partial \hat{f}_p}{\partial x} + \frac{Q_\alpha E}{M_\alpha}\frac{\partial \hat{f}_p}{\partial v} = 0 \tag{2.17}$$

where $\hat{f}_p$ denotes a single macroparticle of shape $S(x)$ whose center of mass is located at some position x_p. In § 2.1, we noted that the PIC method chooses a delta-function representation of the macroparticle velocity-space coordinate. By Eqn. (2.17), we see that by this particular choice, each macroparticle moves as a single particle, and the macroparticle shape is not distorted in time due to its finite size. E is the total mean electric field due to the positions of all the macroparticles.

Applying the method of characteristics [39] to Eqn. (2.17), it can be shown that the macroparticle orbits in phase-space are given by

$$\frac{dx_p}{dt} = v_p, \tag{2.18a}$$

and

$$\frac{dv_p}{dt} = \frac{Q_p E_p}{M_p}. \tag{2.18b}$$

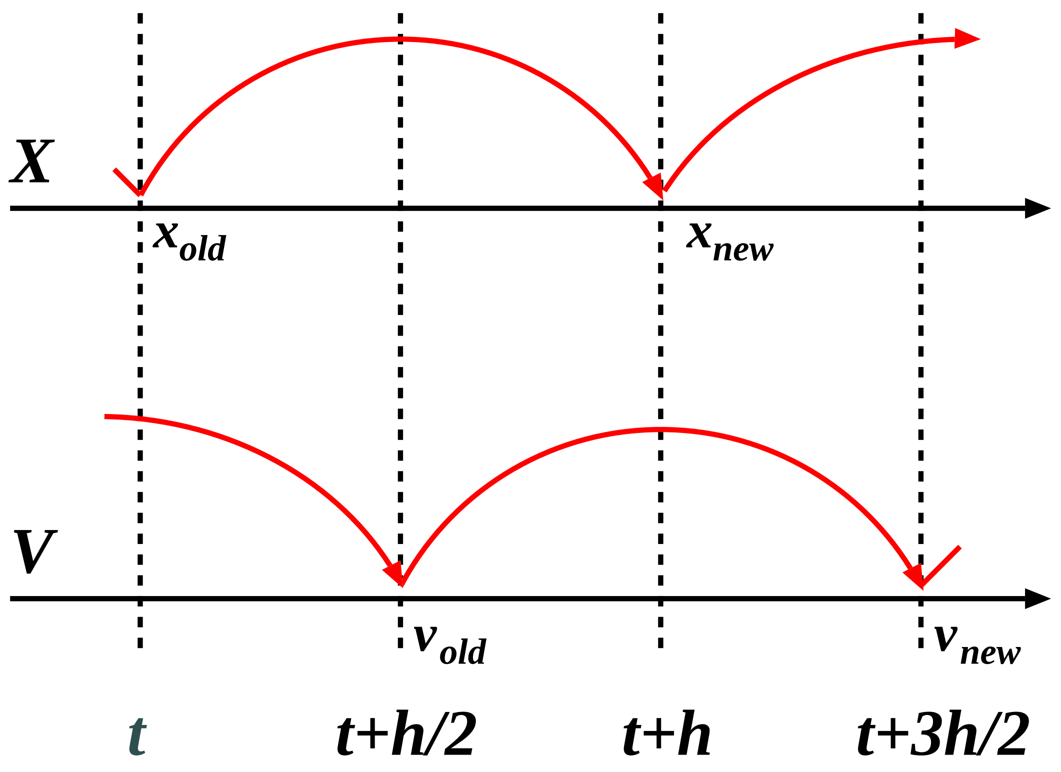


Figure 2.2: Sketch of the time-centered nature of the leapfrog scheme.

The term E_p in Eqn. (2.18b) is the electric field interpolated to the particle position, x_p. This interpolation will be discussed in more detail in § 2.4, but for the purposes of the current discussion we simply point out that this term couples the particles moving on the continuum to the discrete fields given on the grid.

2.2.1 Integration of Equations of Motion

The set of first-order ordinary differential equations (ODE's) in Eqns. (2.18) can be integrated in a number of ways. Perhaps the simplest and most widely used method is the 2$^{\text{nd}}$-order accurate leapfrog (LF) integration scheme, which is based on staggering the time levels of the macroparticle's position and velocity by a half timestep and rewriting Eqns. (2.18) in finite-difference form [10]:

$$\frac{x_p(t+\Delta t) - x_p(t)}{\Delta t} = v_p(t + \frac{\Delta t}{2}) \tag{2.19a}$$

$$\frac{v_p(t+\frac{3\Delta t}{2}) - v_p(t+\frac{\Delta t}{2})}{\Delta t} = \frac{Q_p E(x_p(t+\Delta t))}{M_p}. \tag{2.19b}$$

The LF integrator can then be written as a composition of two maps, $M_{\Delta t} = V_{\Delta t} \circ X_{\Delta t}$, where

$$X_{\Delta t} : \begin{cases} x_{p\,1} = x_p + \Delta t \, v_p \\ v_{p\,1} = v_p \end{cases}, \quad V_{\Delta t} : \begin{cases} x'_p = x_{p\,1} \\ v'_p = v_{p\,1} + \Delta t \, \frac{Q_p E(x_{p\,1})}{M_p} \end{cases}, \tag{2.20}$$

so that it is a composition of two maps which are trivially area preserving in (x, v)-phase space. As illustrated in Fig. 2.2, LF is a time-centered scheme, and even though the positions and velocities exist at different times, both will be updated from some time t_{old} to a new time t_{new} while preserving the accuracy of the integrator. However,the LF algorithm can also be symmetrized to keep the positions and velocities colocated in time after a full timestep by composing $\tilde{M} = X_{\Delta t/2} \circ V_{\Delta t} \circ X_{\Delta t/2}$. This symmetrized formulation retains the 2nd order accuracy of the standard LF scheme. We also note here that the LF algorithm is both symplectic (area preserving for 2d phase space) and time-reversible, properties that other integration techniques such as forward-Euler and Runge-Kutta lack.

2.3 Field Equations

The Poisson equation (Eqn. (2.4)) can be solved in a variety of ways, with the majority of PIC codes relying on fast Fourier transforms (FFT's), finite differences (FD), or the finite volume (FV) method [10, 11, 34]. In the following, we present a description of the finite difference method applied to Poisson's equation on our uniform 1d grid.

Eqn. (2.4) can be written in discretized form as the difference between a forward difference on the electrostatic potential at a grid point, Φ_i, and a backward difference, leading to a three-point formula in 1d [11]:

$$\frac{\Phi_{i+1} - 2\Phi_i + \Phi_{i-1}}{\Delta x^2} = -\frac{\rho_i}{\epsilon_0}. \tag{2.21}$$

The charge density and electrostatic potential are colocated on the grid, and thus the cell index i can refer to our choice of either cell-centers or vertices. As a result of our choice of discretization, with appropriate boundary conditions it can easily be seen that Eqn. (2.21) leads to a symmetric matrix (see § 4.2.1 a more detailed analysis of the properties of this matrix). In 1d, Eqn. (2.21) is easily solved using a fast direct solver, but in multiple dimensions it must be solved iteratively. Due to the properties of the discretization, this can be done quite efficiently with a fast iterative solver such as the conjugate gradient method (CG) [40].

Once we have obtained the electrostatic potential, we again use a centered difference scheme to calculate the electric field:

$$E_i = \frac{\Phi_{i-1} - \Phi_{i+1}}{2\Delta x}. \tag{2.22}$$

Here we are assuming a fully colocated grid, meaning that all grid-based quantities (ρ, Φ, E) exist at the same points on the grid. The methodology behind this choice will be discussed in § 2.4. The colocated grid approach is presented here for clarity, but for our codes we have chosen to use a staggered mesh as detailed in Appendix B.

Alternatively, specific to a 1d system, we can obtain the electric field directly from ρ using Gauss' law:

$$\frac{dE}{dx} = \frac{\rho}{\epsilon_0}. \tag{2.23}$$

Eqn. (4.1) can be discretized on a colocated grid using central differences:

$$\frac{E_{i+1} - E_{i-1}}{2\Delta x} = \frac{\rho_i}{\epsilon_0}, \tag{2.24}$$

with the electric field then obtained by an integration over ρ. An extension of this method on a staggered mesh is presented in § 4.2.2. We note here that while it would appear that would are able to combine Eqns. (2.22) and (2.24) to get Eqn. (2.21) directly, it cannot be done on the colocated grid due to stencil spreading. The staggered grid presents no such problems, and is thus our approach in Chapters 4 and 5.

2.4 Charge Assignment and Field Interpolation

Having fully explored all the properties relating to the macroparticles on the continuum and the fields on the discrete grid, it now becomes necessary to couple the two. This is done by assigning charge from the particles to discrete positions on the grid and interpolation of the discrete electric field to the particle positions on the continuum (to obtain $E(x = x_p)$).

Starting with Eqn. (2.10), we can obtain the charge density on the continuum using Eqn. (2.2a):

$$\begin{aligned}\rho(x) &= \sum_\alpha q_\alpha \int f_\alpha(x, v, t)dv \\ &= \sum_p Q_p S(x - x_p),\end{aligned} \tag{2.25}$$

where we have returned to the original notation assuming multiple plasma species and Q_p is the macroparticle charge for whichever species. We can therefore obtain the charge density on the discrete grid point i directly from the macroparticles' positions using

$$\rho_i = \rho(x_i) = \sum_p Q_p S(x_i - x_p). \tag{2.26}$$

We then choose to use the mirror image of the shape function used in Eqn. (2.26) to define the electric field:

$$E(x_p) = \sum_i E_i S(x_p - x_i). \tag{2.27}$$

2.4.1 Conservation of Momentum

We can check the effect of our choice of using symmetric shape functions in weighting the particles to the grid and interpolating the fields to the particles by looking examining the self-force (force on the macroparticle due to itself). If we have chosen correctly, this quantity should be zero for every macroparticle in our system.

Following Ref. [10], we now constuct a simple proof to show that on a uniform, colocated grid, if the above property holds, then for a symmetric shape function, momentum in a 1d periodic system is conserved and self-forces are eliminated. For more clarity, we define the matrix notation $S_{ip} \equiv S(x_i - x_p)$ and $S_{pi} \equiv S(x_p - x_i)$ such that $S_{pi} = S_{ip}^T$ for our system.

Defining the total momentum,

$$P = \sum_p M_p v_p,$$

such that with Eqn. (2.27) we can express Newton's equation of motion in terms of the force on each macroparticle:

$$\frac{dP}{dt} = \sum_p F(x_p) = \sum_p Q_p \sum_i E_i S_{pi}. \tag{2.28}$$

Assuming a symmetric particle shape, $S_{pi} = S_{ip}$, and changing the order of the sums we have:

$$\frac{dP}{dt} = \sum_i E_i \sum_p Q_p S_{ip}. \tag{2.29}$$

However, from Eqn. (2.26), we know that $\sum_p Q_p S_{ip}$ is ρ_i, thus

$$\frac{dP}{dt} = \sum_i \rho_i E_i, \tag{2.30}$$

with no shape function present. This means that if proper boundary conditions are chosen, the discretizations used in Eqns. (2.21) and (2.22) or Eqn. (2.24) telescope and we will have no self-forces on the particles (to computer precision). For example, if we rewrite Eqn. (2.24) as

$$\rho_i \propto E_{i+1} - E_{i-1},$$

where we have ignored the constants Δx and ϵ_0 for now, it is easy to see that Eqn. (2.30) becomes

$$\frac{dP}{dt} \propto \sum_i E_i(E_{i+1} - E_{i-1}), \tag{2.31a}$$

which telescopes over i such that

$$\frac{dP}{dt} = 0, \tag{2.31b}$$

assuming appropriately applied boundary conditions.

Chapter 3

Development of a Logical Grid-Based PIC code

3.1 Grid Generation Strategy

Techniques utilizing systems of partial differential equations (PDE's) to derive coordinate transformations are very popular in structured grid generation. We have chosen to base our grid generation strategy on a system of elliptic equations, as this strategy possesses two built-in properties that make it attractive for our grid generation efforts. First, elliptic systems adhere to the extremum principle, meaning that the extrema of their solutions are constrained to the system boundaries. Thus, grids derived from elliptic systems have less tendency to fold, resulting in a "one-to-one" mapping. Furthermore, these systems of equations exhibit an inherent smoothness of solution, meaning that boundary slope discontinuities are not propogated into the interior of the domain, provided proper boundary conditions are used. The combination of these two properties allows grids to be generated for virtually any configuration without grid folding.

Admittedly, there are some disadvantages to elliptic systems, namely that a system of PDE's must be solved in order to generate the coordinate system. This becomes important in simulations in which the grid is changed at each timestep (which is not the case in this thesis). Since the most commonly applied equations are nonlinear in the transformed space, an iterative solver must be used, thereby increasing the computational cost with respect to other methods of grid generation. Nevertheless, the relative simplicity associated with these systems is usually enough to overcome the added computational cost, making them very popular.

We have chosen to implement Winslow's Laplace method [30], in which a set of uncoupled Laplace equations are solved on a two-dimensional (2d) logical domain, $\xi, \eta \in [0:1]$. With Winslow's Laplace method, the finest gridding is concentrated in the regions of highest curvature, e.g. around an object at the center of the domain. While this gridding method is rather primitive in that we have no control over where the grid is concentrated away from the boundary, we have chosen this method for its simplicity and suitability for the problems in which we are interested. More advanced methods such as Winslow's variable diffusion method [41] allow for greater control of the grid through an adjustable diffusion coefficient.

The Laplace equations can be written in any coordinate system, but for simplicity we have chosen to implement Winslow's method in the Cartesian coordinate system in physical space. In the physical domain, Laplace's equations take the form

$$\nabla_x^2 \xi^\alpha = \frac{\partial}{\partial x^\beta}\frac{\partial \xi^\alpha}{\partial x^\beta} = 0, \quad \alpha, \beta = 1, \cdots, n. \tag{3.1}$$

Unless otherwise noted, we are summing over repeated indices here and in the rest of this thesis. (For example, we prefer to solve the Laplace equations on a uniform grid in logical space (ξ, η) rather than directly gridding Eqn. (3.1) on the physical space, solving for $\xi(x, y), \eta(x, y)$ and inverting.) In Eqn. (3.1), ξ^α are the dependent variables and x^β are the independent variables. However, we are interested in a set of PDE's such that x^β are the *dependent* variables and ξ^α are the *independent* variables. In the following, we follow the method outlined by Liseikin [42] to provide some details of this transformation.

As shown in Appendix A, we can define the Jacobi matrix

$$j_{\alpha\beta} \equiv \frac{\partial x^{\alpha}}{\partial \xi^{\beta}}, \tag{3.2a}$$

and its inverse

$$k^{\alpha\beta} \equiv \frac{\partial \xi^{\alpha}}{\partial x^{\beta}}, \tag{3.2b}$$

such that

$$j_{\alpha\beta} k^{\beta\gamma} = \delta^{\gamma}_{\alpha}. \tag{3.2c}$$

We can then write Eqn. (3.1) as

$$\frac{\partial k^{\alpha\beta}}{\partial x^{\beta}} = 0. \tag{3.3}$$

Now, by multiplying both sides by $j_{\nu\alpha}$, we have

$$j_{\nu\alpha} \frac{\partial k^{\alpha\beta}}{\partial x^{\beta}} = 0, \tag{3.4}$$

which we recognize to be of the form

$$\frac{\partial}{\partial x^{\beta}} \left(j_{\nu\alpha} k^{\alpha\beta} \right) - k^{\alpha\beta} \frac{\partial j_{\nu\alpha}}{\partial x^{\beta}} = 0. \tag{3.5}$$

From the definition of the Jacobi matrix and its inverse, the first term in Eqn. (3.5) is zero. Expanding the spatial derivative in the second term such that

$$\frac{\partial}{\partial x^{\beta}} = \frac{\partial \xi^{\mu}}{\partial x^{\beta}} \frac{\partial}{\partial \xi^{\mu}} = k^{\mu\beta} \frac{\partial}{\partial \xi^{\mu}}, \tag{3.6}$$

and using the identity

$$g^{\alpha\mu}(\vec{x}) \equiv \frac{\partial \xi^{\alpha}}{\partial x^{\beta}} \frac{\partial \xi^{\mu}}{\partial x^{\beta}} = k^{\alpha\beta} k^{\mu\beta}, \tag{3.7}$$

we can write:

$$g^{\alpha\mu}\frac{\partial^2 x^\nu}{\partial\xi^\alpha\partial\xi^\mu} = 0, \quad \alpha, \gamma, \nu = 1, \cdots, n. \tag{3.8}$$

By its definition, the contravariant metric tensor is a function of the dependent variable $\vec{x}$, i.e. $g^{\alpha\gamma} = g^{\alpha\gamma}(\vec{x})$, meaning that we would have to treat ξ^α as the *dependent* variables in order to calculate $g^{\alpha\gamma}$. In order to solve this system of equations then, we must utilize the covariant metric tensor as defined in Appendix A:

$$g_{\alpha\beta}(\vec{\xi}) \equiv \frac{\partial x^\gamma}{\partial\xi^\alpha}\frac{\partial x^\gamma}{\partial\xi^\beta} = j_{\gamma\alpha}j_{\gamma\beta}, \tag{3.9}$$

such that x^γ are the dependent variables. The conversion from the contravariant metric tensor to the covariant form in 2d can be done using the identity

$$g^{\alpha\beta} = (-1)^{\alpha+\beta}\,\frac{g_{3-\alpha,3-\beta}}{g_{\text{cov}}}, \quad \alpha, \beta = 1, 2, \tag{3.10}$$

where $g_{\text{cov}} = g_{11}g_{22} - g_{12}^2$ is the determinant of the covariant metric tensor. This allows us to write the final system of equations to be solved:

$$\begin{aligned} g_{22}\tfrac{\partial^2 x}{\partial\xi^2} - 2g_{12}\tfrac{\partial^2 x}{\partial\xi\partial\eta} + g_{11}\tfrac{\partial^2 x}{\partial\eta^2} &= 0 \\ g_{22}\tfrac{\partial^2 y}{\partial\xi^2} - 2g_{12}\tfrac{\partial^2 y}{\partial\xi\partial\eta} + g_{11}\tfrac{\partial^2 y}{\partial\eta^2} &= 0. \end{aligned} \tag{3.11}$$

While Eqns. (3.11)(a) and (b) appear to be the same, they actually possess opposite boundary conditions along each segment of the grid boundary. For example, for any segment of the boundary, we might have Neumann boundary conditions for ξ and Dirichlet boundary conditions for η, both expressed in terms of $x(\xi, \eta)$ and $y(\xi, \eta)$. Furthermore, from Eqn. (3.1) and these boundary conditions, we know that $\xi(x, y)$ and $\eta(x, y)$ are conjugate harmonic functions. Thus, Winslow's method forces the grid lines to be orthogonal ($\nabla\xi \cdot \nabla\eta = g^{12} = 0$) in the physical space.

However, since we solve Eqns. 3.11 numerically, we know that *exact* orthogonality is very difficult to achieve. We are able to quantify the amount of nonorthogonality, or "skewness," of our generated grids using the Cauchy-Schwartz inequality [43]. By this inequality, any two points $\vec{p}$ and $\vec{q}$ in n-space must satisfy

$$\vec{p} \cdot \vec{q} \leq |\vec{p}||\vec{q}|. \tag{3.12}$$

Writing Eqn. (3.7) in the form

$$g^{\alpha\beta}(\vec{x}\,) = \nabla\xi^{\alpha} \cdot \nabla\xi^{\beta} \tag{3.13}$$

such that

$$\begin{aligned} g^{11} &= |\nabla\xi|^2 \\ g^{12} &= \nabla\xi \cdot \nabla\eta \\ g^{22} &= |\nabla\eta|^2. \end{aligned} \tag{3.14}$$

Eqn. (3.12) can then be rewritten as

$$\nabla\xi \cdot \nabla\eta \leq |\nabla\xi||\nabla\eta|. \tag{3.15}$$

Squaring both sides and rearranging allows us to define the local grid skewness factor

$$S(\xi,\eta) = \frac{(g^{12})^2}{g^{11}g^{22}}, \tag{3.16}$$

where $0 \leq S \leq 1$. In the limit of $S(\xi,\eta) \to 0$, the grid is orthogonal at (ξ,η), whereas for $S(\xi,\eta) \to 1$, the grid becomes singular, i.e. the grid folds there. Furthermore, we can define the maximum grid skewness,

$$S_{\max} = \max\left[S(\xi,\eta)\right], \tag{3.17}$$

such that we can use a single parameter to characterize the maximum nonorthogonality of the generated grid.

Furthermore, while Eqns. (3.11) appear to be a system of linear equations at first glance, we recognize that by the definition of the covariant metric tensor that this is indeed a more complicated system of equations. Since the covariant terms $g_{\alpha\beta}$ depend on $\frac{\partial x}{\partial \xi}$ and the solution we seek is x, the system of equations is indeed nonlinear.

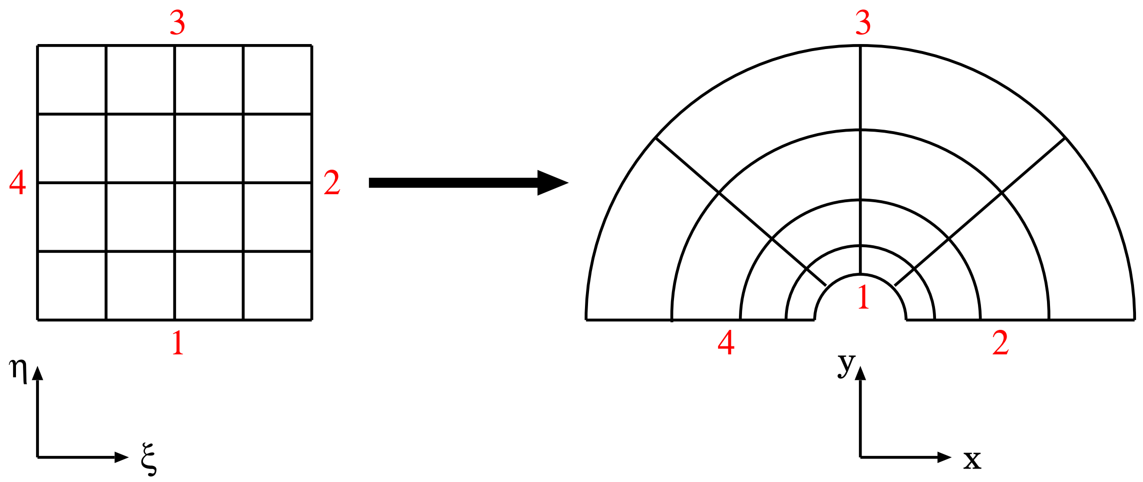


Figure 3.1: Schematic for mapping from the logical to the physical grid in the annulus case. The numbers along the boundaries of the square logical domain are used to indicate which boundary each is mapped to on the physical domain.

3.2 Example Grids

Eqns. (3.11) are discretized using a 2nd order accurate finite-difference method and, because of their non-linear form, are iteratively solved with an inexact Newton-Krylov solver [29]. A vast array of physical grids can be generated from the uniform logical grid by simply specifying the desired boundary conditions along each boundary segment on the logical grid as shown in Fig. (3.1). The grids presented in Fig. (3.2) are of some of the physical grids with circular objects along the inner boundary which we have generated using Winslow's method. As alluded to in § 3.1, one variable of ξ and η satisfies Dirichlet boundary conditions and the other satisfies Neumann boundary conditions on each boundary segment. We have also generated grids with elliptically shaped objects, and are capable of generating objects of fairly general shape and size on any boundary segment.

Furthermore, as shown in Fig. (3.3), we are fully capable of moving an interior object within the physical system. In the example grids provided here, we make full

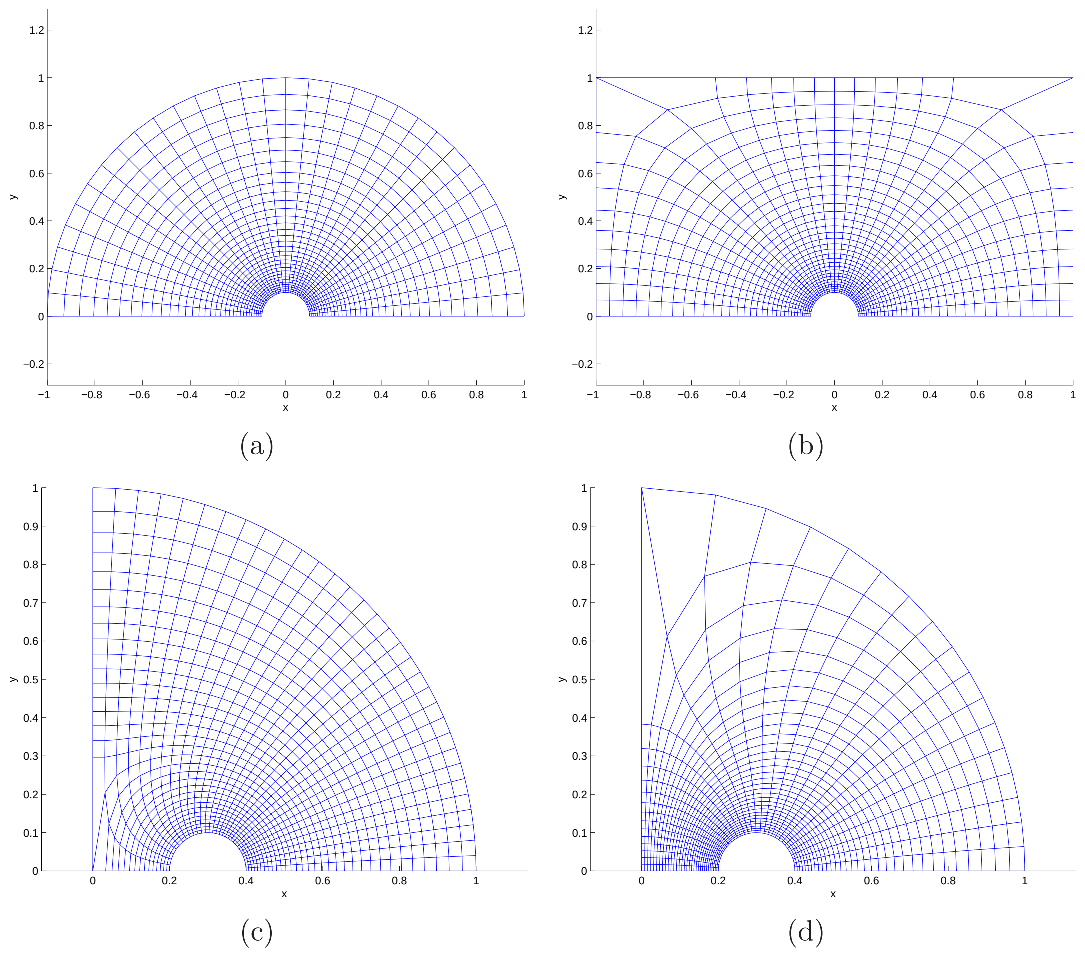


Figure 3.2: Grids generated using Winslow's method with a circular object along the inner physical grid boundary. Here we have used a 32×32 grid in order to show the mapping more clearly. The appearance of nonorthogonal grid lines is due to the straight lines used by the plotting tool.

use of the symmetry of the problem, which allows us to simulate half the physical system in Figs. (3.2a),(3.2b), and (3.3), and only a quarter of the physical system in Figs. (3.2c) and Figs. (3.2d). Note that by simulating only a quarter of the system in Figs. (3.2c) and Figs. (3.2d), we are constrained to simulate two objects of equal size and shape. Alternatively, by modeling two objects in a "half-domain", we are able to fix this problem such that we can specify different sizes and shapes for each.

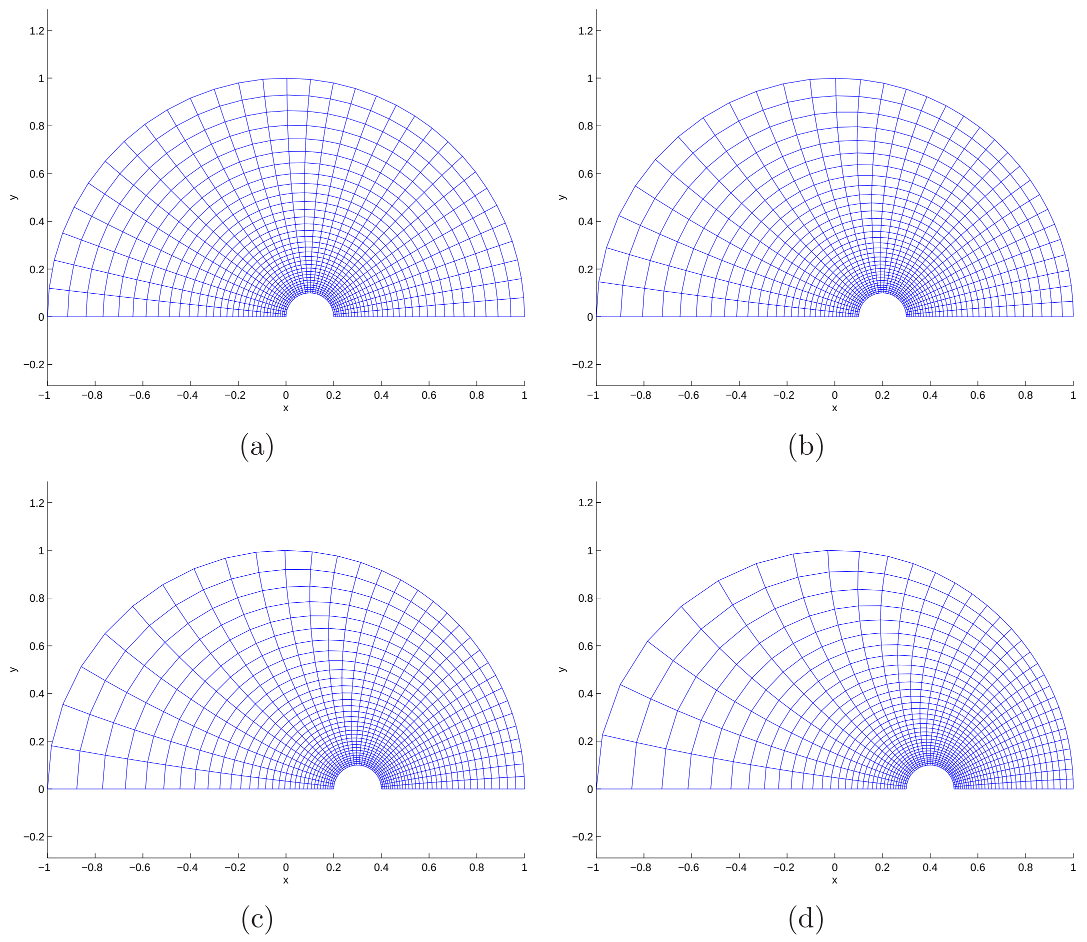


Figure 3.3: Examples of how the positioning of the object along inner boundary changes the grid within the physical domain using Winslow's method. Here we have used a 32×32 grid in order to show the mapping more clearly. The appearance of curved grid lines near the inner semicircle is due to the straight lines used by the plotting tool.

We note here that for the annular grid generated in Fig. 3.2, the maximum grid skewness as defined by Eqn. (3.17) is $S_{\max} \approx 10^{-14}$ for a 32×32 grid. For the offset (eccentric) annulus cases shown in Fig. 3.3, the maximum skewness occurs in Fig. 3.3(d), with $S_{\max} \approx 10^{-6}$ for a 32×32 grid. The general trend for these grids is that the skewness increases as the offset of the inner annulus is increased, and this parameter scales with 2^{nd}-order accuracy in $\Delta\xi$. The same scaling holds true for the "half-domain" cases shown in Fig. 3.2(c) and (d), where the $S_{\max} \approx 10^{-4}$ for the grid shown in Fig. 3.2(c) and $S_{\max} \approx 10^{-5}$ for Fig. 3.2(d) for a 32×32 grid. Thus, the lines that appear be curved in these figures are due to the graphics used for plotting them, as they are very nearly exactly orthogonal for all cases.

3.3 Logical Grid Equations of Motion

Since we are interested in building a code capable of modeling a wide variety of domain and cell shapes and sizes in multiple geometries, we have chosen to forgo a physical grid particle mover as used in many standard PIC codes. We instead focus our attention on developing a logical grid particle mover. This also eliminates the need for a complex search algorithm to locate particles on the non-uniform physical grid [18, 19].

3.3.1 Transformed Newton-Lorentz Equations of Motion

The most obvious method for implementing our mover in logical space would seem to be by simply converting the Newton-Lorentz equations of motion from the physical to the logical grid as follows. Eqns. (2.18a) and (2.18b) can be written in multi-dimensional form as:

$$\frac{dx^\alpha}{dt} = v^\alpha \tag{3.18a}$$

and

$$\frac{dv^{\alpha}}{dt} = -\frac{q}{m}\frac{\partial \Phi}{\partial x^{\alpha}}. \tag{3.18b}$$

Using Eqn. (3.2a), the physical velocity can be expressed in logical variables as

$$v^{\alpha} = j_{\alpha\beta}\frac{\partial \xi^{\beta}}{\partial t}. \tag{3.19}$$

Then, by Eqn. (3.2b),

$$\frac{d\xi^{\gamma}}{dt} = k^{\gamma\alpha}v^{\alpha} \equiv u^{\gamma}, \tag{3.20}$$

where we have used Eqn. (A.5). Inserting Eqn. (3.19) into Eqn. (3.18b) and expanding the $\frac{\partial \Phi}{\partial x^{\alpha}}$ term using Eqn. (3.6), we can write

$$\frac{d}{dt}\left(j_{\alpha\beta}u^{\beta}\right) = -\frac{q}{m}\left(k^{\mu\alpha}\frac{\partial \Phi}{\partial \xi^{\mu}}\right). \tag{3.21}$$

We can then expand the time derivative,

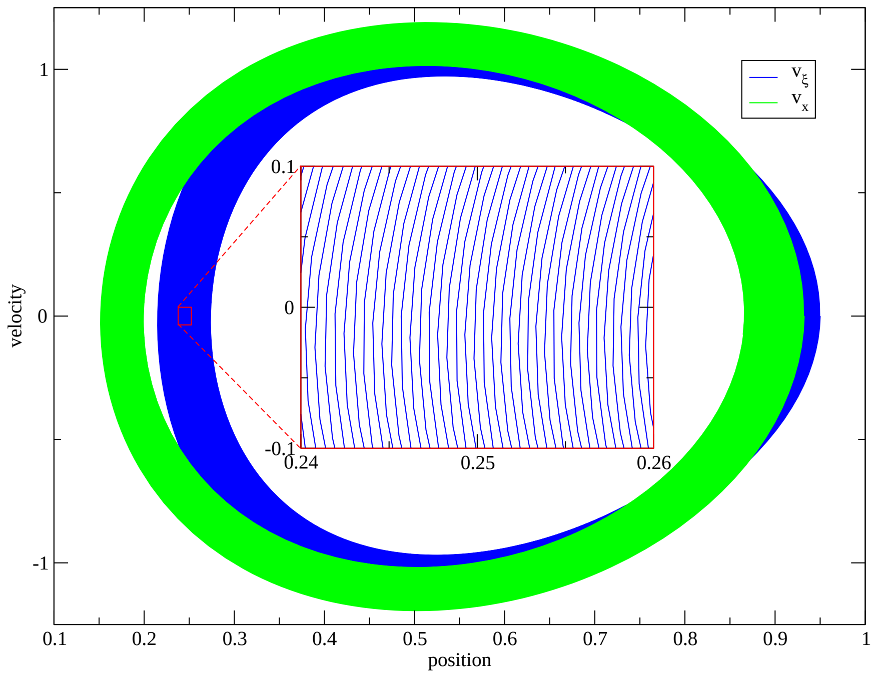
$$j_{\alpha\beta}\frac{du^{\beta}}{dt} + u^{\beta}\frac{\partial j_{\alpha\beta}}{\partial \xi^{\nu}}u^{\nu} = -\frac{q}{m}\left(k^{\mu\alpha}\frac{\partial \Phi}{\partial \xi^{\mu}}\right), \tag{3.22}$$

where we have used $\frac{d}{dt}\left(j_{\alpha\beta}\right) = \frac{\partial j_{\alpha\beta}}{\partial \xi^{\nu}}u^{\nu}$. We now multiply Eqn. (3.22) by $k^{\gamma\alpha}$ and use Eqns. (A.5) and (3.7) to obtain the "velocity" update equations in the transformed coordinates:

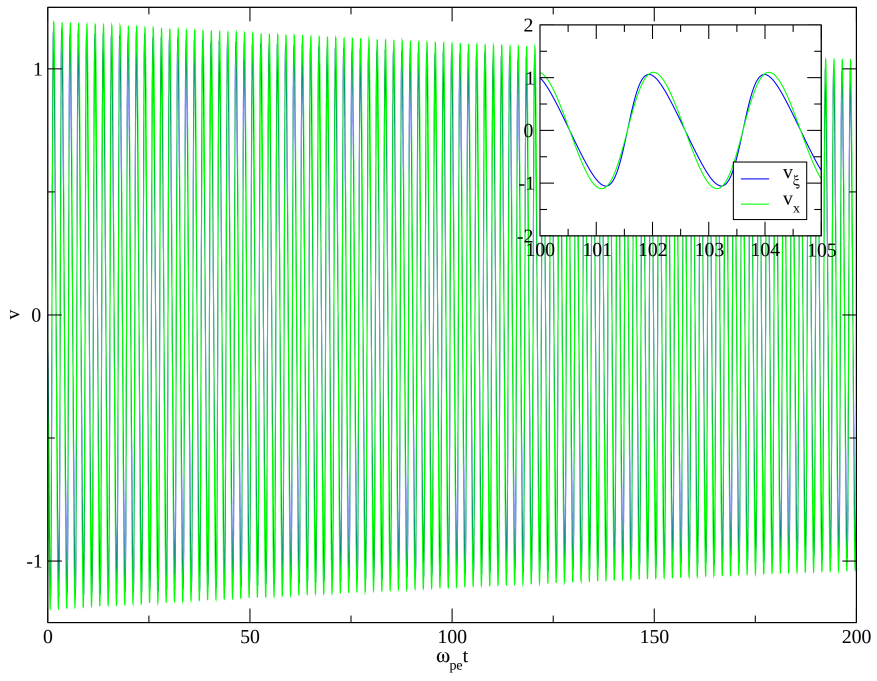
$$\frac{du^{\gamma}}{dt} = -\left[u^{\beta}k^{\gamma\alpha}\frac{\partial j_{\alpha\beta}}{\partial \xi^{\nu}}u^{\nu} + \frac{q}{m}\left(g^{\gamma\mu}\frac{\partial \Phi}{\partial \xi^{\mu}}\right)\right]. \tag{3.23}$$

As expected, the right-hand side of Eqn. (3.23) is comprised of a field force term as well as an inertial force term. We note here that the field force term contains $g^{\gamma\mu}(\vec{x})$, whereas our grid generation techniques provide us with $g_{\alpha\beta}(\vec{\xi})$. This is not a problem once the grid is generated, however, as we can then transform between them freely using Eqns. (A.10) and (A.11).

However, closer inspection of Eqns. (3.20) and (3.23) reveals a problem with our formulation. To more clearly illustrate this point, we now rewrite Eqns. (3.20)



(a)



(b)

Figure 3.4: Phase-space (a) and position vs. time (b) plots for a single particle in an externally applied harmonic oscillator potential given by $\Phi = Ax^2$ on a 1d non-uniform grid $x = \frac{\xi+\epsilon\xi^2}{1+\epsilon}$. Both the physical (blue) and logical (green) phase space areas spiral inward with time.

and (3.23) in their 1d forms as:

$$\begin{aligned} \dot{\xi} &= u \qquad (3.24a) \\ \dot{u} &= -\frac{J'u^2}{J} - \frac{q}{m}\frac{1}{J^2}\Phi', \qquad (3.24b) \end{aligned}$$

where $J \equiv \frac{dx}{d\xi}$ in 1d, an overdot represents a time derivative, and the $'$ symbol represents a derivative with respect to ξ. Taking the formal divergence of Eqns. (3.24a),

$$\left.\frac{\partial\dot{\xi}}{\partial\xi}\right|_u + \left.\frac{\partial\dot{u}}{\partial u}\right|_\xi = -\frac{2J'u}{J}, \qquad (3.25)$$

reveals that the Newton-Lorentz equations of motion are in fact *not* divergence free under this transformation of variables. As such, a time integration of these equations of motion with a "naive" LF integrator as in § 2.2.1 will *not* preserve phase space area, and the particle orbits will spiral (either inward or outward since we do not know the divergence of Eqn. (3.25) in general) with time. This effect is demonstrated for a single particle in an externally applied harmonic oscillator potential well in Fig. 3.4

3.3.2 Hamiltonian Approach to Logical Grid Equations of Motion

Clearly, the fact that we have not introduced canonical variables for the Newton-Lorentz equations of motion in logical variables in § 3.3.1 leads to the lack of (ξ, u) phase space area conservation. We have therefore constructed a logical grid particle mover based on Hamilton's equations by using a canonical transformation with an $F_2(\vec{x}, \vec{P}, t)$ generating function [44], thereby assuring divergence-free equations of motion on the logical grid. Under the canonical transformation we have

$$p^\alpha = \frac{\partial F_2}{\partial x^\alpha} \qquad (3.26a)$$

and

$$\xi^\alpha = \frac{\partial F_2}{\partial P^\alpha} \qquad (3.26b)$$

where $p^\alpha = mv^\alpha$ is the physical space momentum, and P^α is the logical space momentum. We have chosen the F_2 generating function such that x^α and P^α are considered independent, thus

$$K = H + \frac{\partial F_2}{\partial t}. \tag{3.27}$$

We do not yet have a time-adaptive grid so we can ignore the time-dependent part of Eqn. (3.27). Specializing to the contact transformation $\vec{\xi} = \vec{\xi}(\vec{x})$, we can write the F_2 generating function as

$$F_2(\vec{x}, \vec{P}) = \xi^\beta(\vec{x}) P^\beta, \tag{3.28}$$

such that by Eqns. (3.26), we have

$$p^\alpha = k^{\beta\alpha} P^\beta \tag{3.29a}$$

and

$$\xi^\alpha = \xi^\alpha(\vec{x}). \tag{3.29b}$$

In the absence of a static background magnetic field $\vec{B}$, the physical grid Hamiltonian is given in general form by

$$\begin{aligned} H &= T(\vec{p}) + V(\vec{x}) \\ &= \frac{p^\alpha p^\alpha}{2m} + q\Phi(\vec{x}). \end{aligned} \tag{3.30}$$

Notice that Eqn. (3.30) is fully separable. For a problem with azimuthal (z) symmetry, the physical Hamiltonian can be written

$$\begin{aligned} H_{\text{azi}} &= \frac{p_r^2}{2m} + \frac{p_\phi^2}{2mr^2} + \frac{p_z^2}{2m} + q\Phi(\vec{x}) \\ &= \frac{p_x^2 + p_y^2}{2m} + \frac{p_z^2}{2m} + q\Phi(\vec{x}) \\ &= \frac{p_i p_i}{2m} + \underbrace{\frac{p_z^2}{2m} + q\Phi(\vec{x})}_{\tilde{V}(\vec{x})} \qquad i = 1, 2, \end{aligned} \tag{3.31}$$

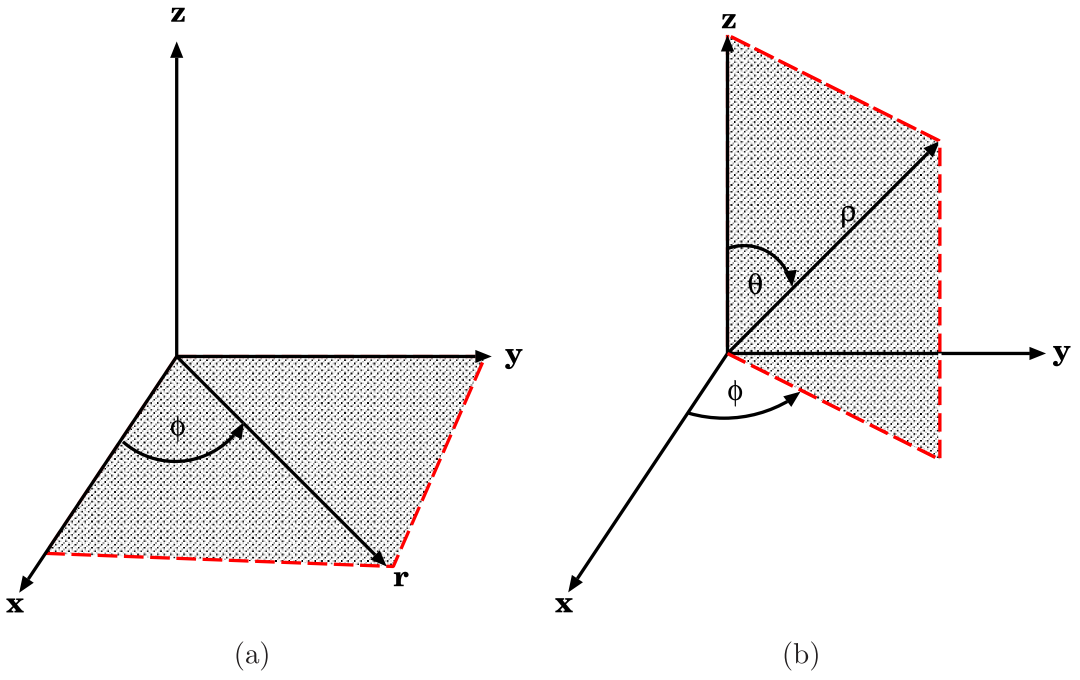


Figure 3.5: Azimuthal (a) and axisymmetic (b) coordinate conversions.

where $\frac{\partial\Phi(\vec{x})}{\partial z} = 0$ implies that $\frac{p_z^2}{2m}$ is a constant of the particle motion on in the physical space perpendicular to the plane of reference as shown in Fig. (3.5a). Here $\tilde{V}(\vec{x})$ is the effective potential in the physical space.

Likewise, for an axisymmetric problem,

$$\begin{aligned} H_{\text{axi}} &= \frac{p_\rho^2}{2m} + \frac{p_\phi^2}{2m\rho^2\sin^2\theta} + \frac{p_\theta^2}{2m\rho^2} + q\Phi(\vec{x}) \\ &= \frac{p_x^2+p_y^2}{2m} + \frac{p_\phi^2}{2mx^2} + q\Phi(\vec{x}) \\ &= \frac{p_i p_i}{2m} + \underbrace{\frac{p_\phi^2}{2mx^2} + q\Phi(\vec{x})}_{\tilde{V}(\vec{x})} \qquad i = 1,2, \qquad (3.32) \end{aligned}$$

where $\frac{\partial\Phi(\vec{x})}{\partial\phi} = 0$ implies that p_ϕ is a constant of motion and thus the term $\frac{p_\phi^2}{2mx^2}$ can be thought of as a (non-constant) contribution to the effective potential, due to the curved motion of the particle perpendicular to the shaded plane in Fig. (3.5b).

Eqn. (3.30) can then be used in conjunction with Eqns. (3.29) to construct the logical grid Hamiltonian:

$$\begin{aligned} K &= \frac{1}{2m}\left(k^{\beta\alpha}P^{\beta}k^{\gamma\alpha}P^{\gamma}\right) + \tilde{V}(\vec{x}(\vec{\xi})) \\ &= \frac{1}{2m}\left(g^{\beta\gamma}P^{\beta}P^{\gamma}\right) + V(\vec{\xi}). \end{aligned} \tag{3.33}$$

Thus, on the logical grid the Hamiltonian can be written

$$K = T(\vec{\xi}, \vec{P}) + V(\vec{\xi}), \tag{3.34}$$

meaning that we have transformed the *separable* physical grid Hamitonian (Eqn. (3.30)) to an equivalent, *non-separable* system. Applying Hamilton's equations, $\dot{\xi}^{\alpha} \equiv \frac{\partial K}{\partial P^{\alpha}}$ and $\dot{P}^{\alpha} \equiv -\frac{\partial K}{\partial \xi^{\alpha}}$ to Eqn. (3.33) gives the logical grid equations of motion:

$$\dot{\xi}^{\mu} = \frac{g^{\gamma\mu}P^{\gamma}}{m} \tag{3.35a}$$

$$\dot{P}^{\mu} = -\frac{1}{2m}\frac{\partial g^{\beta\gamma}}{\partial \xi^{\mu}}P^{\beta}P^{\gamma} - \frac{\partial V}{\partial \xi^{\mu}}. \tag{3.35b}$$

The term quadratic in $\vec{P}$ represents the inertial force in these coordinates.

3.4 Modified Leapfrog Integrator

Whereas the physical space Hamiltonian leads to separable equations of motion which can be integrated by a "naive" LF integrator (§ 2.2.1), in the logical space this is no longer true. Since Eqns. (3.35a) and (3.35b) are not separable, integration by the naive LF method will not conserve phase space area.

As such, we have chosen to implement an extension of the semi-implicit *modified leapfrog* (ML) integrator originally developed by Finn and Chacón [45] for integrating solenoidal flows in fluid dynamics and MHD codes. Rewriting Eqns. (3.35a) and (3.35b) as $\dot{\vec{P}} = \vec{V}$ and $\dot{\vec{\xi}} = \vec{U}$, respectively, where $\frac{\partial U_i}{\partial \xi_i} + \frac{\partial V_i}{\partial P_i} = 0$. Denoting explicit (implicit) updates with a superscript e (i), the ML integrator can be written as $\vec{M}_{\Delta t} = \vec{P}^e_{\Delta t} \circ \vec{\xi}^i_{\Delta t}$, where

$$\xi^i_{\Delta t} : \begin{cases} \vec{\xi}_1 = \vec{\xi} + \Delta t \vec{U}(\vec{\xi}_1, \vec{P}) \\ \vec{P}_1 = \vec{P} \end{cases}, \quad P^e_{\Delta t} : \begin{cases} \vec{\xi}' = \vec{\xi}_1 \\ \vec{P}' = \vec{P}_1 + \Delta t \vec{V}(\vec{\xi}_1, \vec{P}_1) \end{cases}. \tag{3.36}$$

Combining, we have

$$\vec{\xi}' = \vec{\xi} + \Delta t\, \vec{U}(\vec{\xi}', \vec{P}), \quad \vec{P}' = \vec{P} + \Delta t\, \vec{V}(\vec{\xi}', \vec{P}). \tag{3.37}$$

The map $\xi^i_{\Delta t}$ is implicit and must be done by means of Newton or Picard iterations. The map $P^e_{\Delta t}$ is explicit, and can therefore be applied directly. It can be shown that, unlike the standard non-time-centered LF scheme described in § 2.2.1, this non-time-centered formulation of the ML scheme results in only 1st-order accuracy in Δt. To achieve 2nd-order accuracy in time, we simply symmetrize the ML scheme by composing $\vec{\xi}^e_{\Delta t/2} \circ \vec{P}^i_{\Delta t/2} \circ \vec{P}^e_{\Delta t/2} \circ \vec{\xi}^i_{\Delta t/2}$. The implicit-followed-by-explicit ordering in each pair of mappings retains the area preserving nature of the integrator as shown in a 2d phase space in Ref. [45]. (The volume preserving nature, and its dependence on the explicit followed by implicit nature, is shown below). The alternation of the steps in $\vec{\xi}$ and $\vec{P}$ gives second-order accuracy in Δt. The logical flow of the ML integrator as implemented on Eqns. (3.35a) and (3.35b) is

$$\vec{\xi}^i_{\Delta t/2} \rightarrow \vec{P}^e_{\Delta t/2} \rightarrow \vec{P}^i_{\Delta t/2} \rightarrow \vec{\xi}^e_{\Delta t/2}.$$

We note here that the charge density is accumulated on the grid and the mean fields are solved for after the implicit position update step of the symmetrized ML mover. While this requires us to pass through the particle array twice per timestep, it allows us to accumulate the charge density and solve for the fields only once per timestep.

3.4.1 Proof of ML as a Symplectic Integrator

We have constructed a Hamiltonian system of equations to describe the particle motion on the logical grid, therefore it is important that our integrator is symplectic. In the following, we prove that the ML integrator is symplectic. For simplicity we outline the proof for the non-symmetrized ML integrator (Eqn. (3.36)). As a starting point, we can first show ML to be volume preserving by finding the Jacobians of both the implicit and explicit maps. For the integrator to be volume preserving, the product of these two maps as implemented in Eqn. (3.37) must be one. Defining

$$A_{\alpha\beta} = \frac{\partial U^\alpha}{\partial P^\beta} = \frac{\partial^2 K}{\partial P^\alpha \partial P^\beta} \tag{3.38a}$$

$$B_{\alpha\beta} = \frac{\partial V^\alpha}{\partial \xi^\beta} = -\frac{\partial^2 K}{\partial \xi^\alpha \xi^\beta} \tag{3.38b}$$

$$M_{\alpha\beta} = \frac{\partial U^\alpha}{\partial \xi^\beta} = \frac{\partial^2 K}{\partial \xi^\beta \partial P^\alpha} \tag{3.38c}$$

$$N_{\alpha\beta} = \frac{\partial V^\alpha}{\partial P^\beta} = -\frac{\partial^2 K}{\partial \xi^\alpha \partial P^\beta} \tag{3.38d}$$

we can write the Jacobians of the implicit map

$$\boldsymbol{\mathcal{J}}^i = \frac{\partial\left(\xi^i, P^i\right)}{\partial\left(\xi, P\right)} = \begin{bmatrix} \left(\boldsymbol{I} - \Delta t\, \boldsymbol{M}\right)^{-1} & \left(\boldsymbol{I} - \Delta t\, \boldsymbol{M}\right)^{-1} \Delta t\, \boldsymbol{A} \\ 0 & \boldsymbol{I} \end{bmatrix} \tag{3.39}$$

as well as the explicit map

$$\boldsymbol{\mathcal{J}}^e = \frac{\partial\left(\xi^e, P^e\right)}{\partial\left(\xi^i, P^i\right)} = \begin{bmatrix} \boldsymbol{I} & 0 \\ \Delta t\, \boldsymbol{B} & \boldsymbol{I} + \Delta t\, \boldsymbol{N} \end{bmatrix}, \tag{3.40}$$

in a simplified form. Here I is the identity matrix. We note also that by our definitions, $A^{\alpha\beta}$ and $B^{\alpha\beta}$ are symmetric matrices, whereas $M^{\alpha\beta} = -(N^T)^{\alpha\beta}$. The determinants of Eqns. (3.39) and (3.40) are simply

$$\det(\boldsymbol{\mathcal{J}}^i) = \det\left(\frac{1}{\boldsymbol{I} - \Delta t\, \boldsymbol{M}}\right) \tag{3.41a}$$

and

$$\begin{aligned} \det(\boldsymbol{\mathcal{J}}^e) &= \det(\boldsymbol{I} + \Delta t\, \boldsymbol{N}) \\ &= \det(\boldsymbol{I} - \Delta t\, \boldsymbol{M}), \end{aligned} \tag{3.41b}$$

thus their products can be trivially shown to be unity. The subtlety in our formulation is that $\boldsymbol{M}$ and $\boldsymbol{N}$ here depend on the variables $\vec{\xi}_1$ and $\vec{P}$, such that the implicit followed by explicit mapping order retains the volume preserving nature of the integrator as described in § 3.4. It can be easily shown that by reversing this order, the product of the determinants of the maps is indeed not unity, and volume preservation is lost.

In one degree of freedom (i.e. a 1d system), area preservation is enough to show that an integrator is symplectic [45]. However, in order to prove that an integrator is symplectic for two or more degrees of freedom, one must prove the Poisson bracket relation $[\xi', P'] = [\xi, P]$, or equivalently, $\boldsymbol{\mathcal{J}}\boldsymbol{\mathcal{E}}\boldsymbol{\mathcal{J}}^T = \boldsymbol{\mathcal{E}}$. Here $\boldsymbol{\mathcal{J}} = \boldsymbol{\mathcal{J}}^e\boldsymbol{\mathcal{J}}^i$, $\boldsymbol{\mathcal{J}}^T = \boldsymbol{\mathcal{J}}^{iT}\boldsymbol{\mathcal{J}}^{eT}$, and

$$\boldsymbol{\mathcal{E}} = \begin{bmatrix} 0 & \boldsymbol{I} \\ -\boldsymbol{I} & 0 \end{bmatrix} \tag{3.42}$$

is the unit antisymmetric rotation tensor. Thus,

$$\begin{aligned} \boldsymbol{\mathcal{J}} &= \begin{bmatrix} \boldsymbol{I} & 0 \\ \Delta t\, \boldsymbol{B} & \boldsymbol{I} + \Delta t\, \boldsymbol{N} \end{bmatrix} \begin{bmatrix} (\boldsymbol{I} - \Delta t\, \boldsymbol{M})^{-1} & \Delta t\ (\boldsymbol{I} - \Delta t\, \boldsymbol{M})^{-1}\boldsymbol{A} \\ 0 & \boldsymbol{I} \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{C}^{-1} & \Delta t\, \boldsymbol{C}^{-1}\boldsymbol{A} \\ \Delta t\, \boldsymbol{B}\boldsymbol{C}^{-1} & \Delta t^2\, \boldsymbol{B}\boldsymbol{C}^{-1}\boldsymbol{A} + \boldsymbol{C} \end{bmatrix} \end{aligned} \tag{3.43}$$

where, for simplicity, we have defined $\boldsymbol{C}^{-1} = (\boldsymbol{I} - \Delta t\, \boldsymbol{M})^{-1}$. Since $\left(\boldsymbol{R}^{-1}\right)^T = \left(\boldsymbol{R}^T\right)^{-1}$, we also have

$$\boldsymbol{C}^{-T} = \left(\boldsymbol{I} - \Delta t\, \boldsymbol{M}^T\right)^{-1}, \tag{3.44}$$

where $\boldsymbol{R}$ is any arbitrary matrix. Using these definitions, we can also calculate

$$\boldsymbol{\mathcal{J}}^T = \begin{bmatrix} \boldsymbol{C}^{-T} & \Delta t\, \boldsymbol{C}^{-T}\boldsymbol{B} \\ \Delta t\, \boldsymbol{A}\boldsymbol{C}^{-T} & \Delta t^2\, \boldsymbol{A}\boldsymbol{C}^{-T}\boldsymbol{B} + \boldsymbol{C}^T \end{bmatrix}, \tag{3.45}$$

where we have used $\boldsymbol{A}^T = \boldsymbol{A}$ and $\boldsymbol{B}^T = \boldsymbol{B}$. All that remains is to take the matrix products and simplify as necessary. The first product is given by:

$$\boldsymbol{\mathcal{J}}\boldsymbol{\mathcal{E}} = \begin{bmatrix} -\Delta t\, \boldsymbol{C}^{-1}\boldsymbol{A} & \boldsymbol{C}^{-1} \\ -\Delta t^2\, \boldsymbol{B}\boldsymbol{C}^{-1}\boldsymbol{A} - \boldsymbol{C} & \Delta t\, \boldsymbol{B}\boldsymbol{C}^{-1} \end{bmatrix}. \tag{3.46}$$

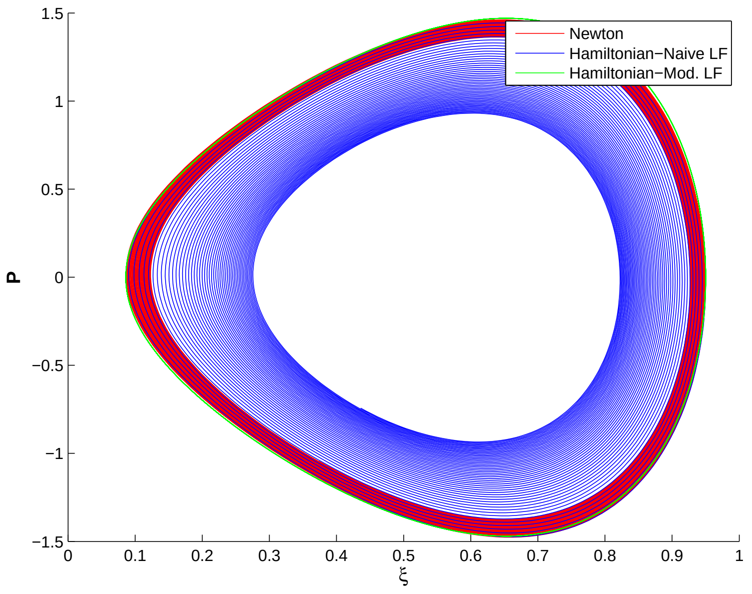
Finally,

$$\begin{aligned} \boldsymbol{\mathcal{J}}\boldsymbol{\mathcal{E}}\boldsymbol{\mathcal{J}}^T &= \begin{bmatrix} -\Delta t\, \boldsymbol{C}^{-1}\boldsymbol{A} & \boldsymbol{C}^{-1} \\ -\Delta t^2\, \boldsymbol{B}\boldsymbol{C}^{-1}\boldsymbol{A} - \boldsymbol{C} & \Delta t\, \boldsymbol{B}\boldsymbol{C}^{-1} \end{bmatrix} \times \\ &\qquad \begin{bmatrix} \boldsymbol{C}^{-T} & \Delta t\, \boldsymbol{C}^{-T}\boldsymbol{B} \\ \Delta t\, \boldsymbol{A}\boldsymbol{C}^{-T} & \Delta t^2\, \boldsymbol{A}\boldsymbol{C}^{-T}\boldsymbol{B} + \boldsymbol{C}^T \end{bmatrix} \\ &= \begin{bmatrix} 0 & \boldsymbol{C}^{-1}\boldsymbol{C}^T \\ -\boldsymbol{C}\boldsymbol{C}^{-T} & \Delta t\, \left(\boldsymbol{B}\boldsymbol{C}^{-1}\boldsymbol{C}^T - \boldsymbol{C}\boldsymbol{C}^{-T}\boldsymbol{B}\right) \end{bmatrix} \\ &= \begin{bmatrix} 0 & \boldsymbol{I} \\ -\boldsymbol{I} & 0 \end{bmatrix} \equiv \boldsymbol{\mathcal{E}}, \end{aligned} \tag{3.47}$$

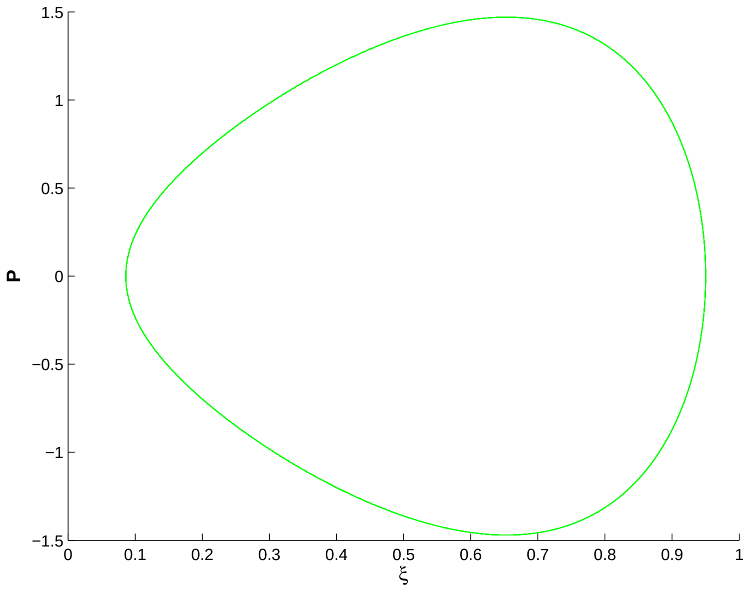
revealing that *our ML integrator is indeed symplectic for a system of arbitrary dimension*. This proof works equally well when the order of the updates is reversed $\vec{\xi}^e_{\Delta t} \circ \vec{P}^i_{\Delta t}$ rather than $\vec{P}^e_{\Delta t} \circ \vec{\xi}^i_{\Delta t}$, so that the symmetrized ML integrator is also symplectic.

3.4.2 Validation of the Logical Grid Particle Mover

The harmonic oscillator potential well test was again performed using the both the ML mover and a "naive" LF integration of Eqns. (3.35a) and (3.35b). As can be seen in Figs. (3.6), the new mover exactly preserves phase-space area, while the naive LF

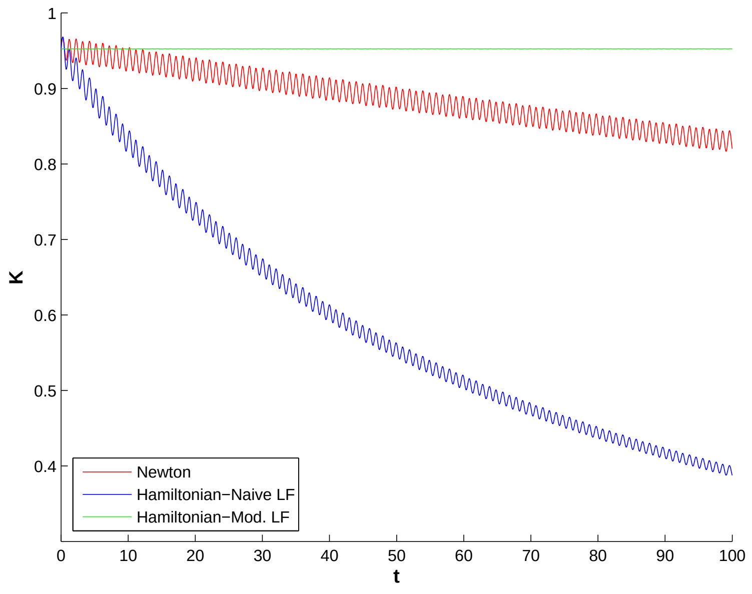


(a)

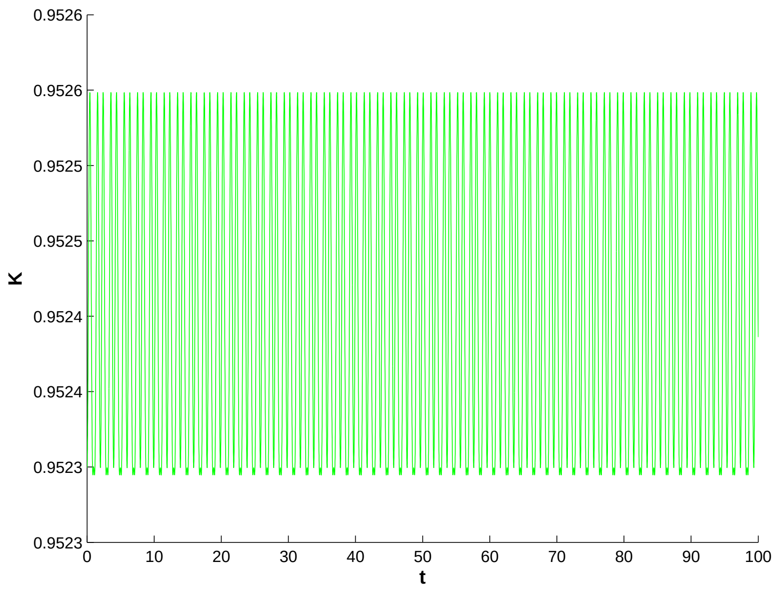


(b)

Figure 3.6: Single particle in an externally applied harmonic oscillator potential given by $\Phi = Ax^2$. Particle phase space area in ξ, P-space for all three mover schemes in (a) and a view of the phase space area of only the ML mover is shown in (b).



(a)



(b)

Figure 3.7: Single particle in an externally applied harmonic oscillator potential given by $\Phi = Ax^2$. Energy evolution in time is compared for all three movers in (a) and a highly-zoomed view of the ML mover is shown in (b). Here we do not use interpolations for either fields or grid quantities as both are prescribed everywhere in the domain.

mover actually spirals inward at a faster rate than the original transformed Newton-Lorentz LF mover. Fig. (3.7) shows the evolution of the energy ($K(\vec{\xi},\vec{P}) = H(\vec{x},\vec{p})$) in time for all three movers in (a), and offers a highly zoomed view of the same evolution of the ML mover in (b). The very small amplitude oscillations evident in Fig. (3.7b) are at the harmonic oscillator frequency. These are due to the fact that orbits in the ML mover follow approximate surfaces of a quantity K' which is an approximation to the energy K in phase-space, and are thus expected.

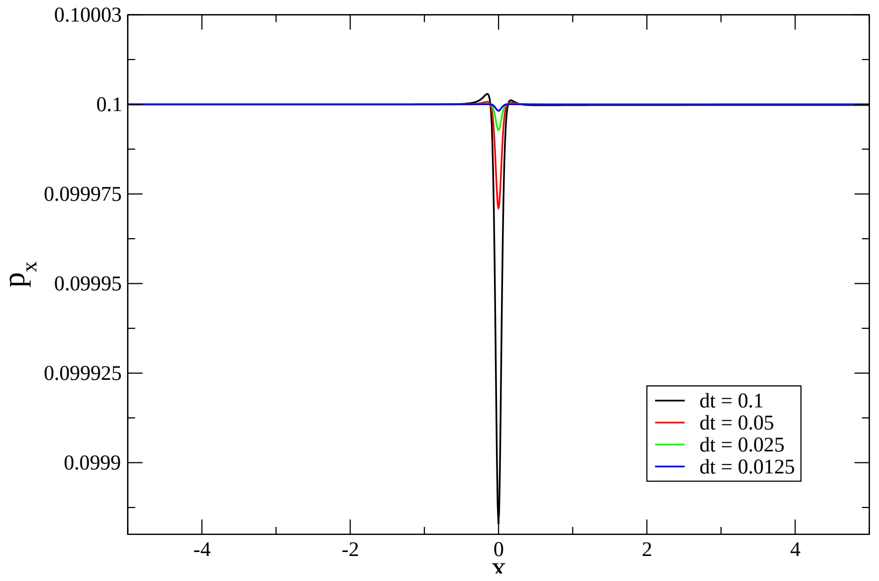
3.4.3 ML Mover Applied to Plane-Polar Coordinate Transformation

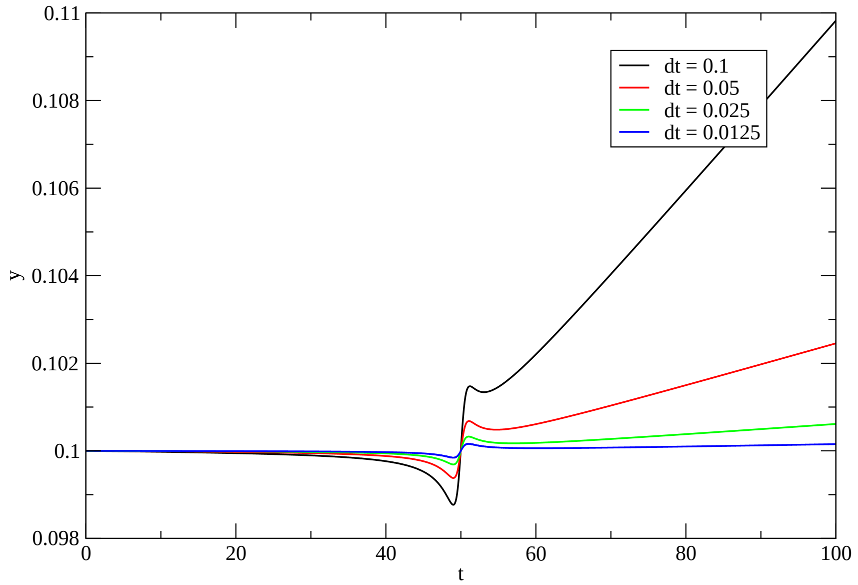
As a 2d test of how accurate our ML mover is, we have chosen to implement a particularly stringent test by running a single particle with zero field forces ($\Phi = 0$), which should give a straight line trajectory given by v_{x0}. In plane polar coordinates, this is a particularly stringent test if the particle passes just above the origin. There is no grid in this system, so we are isolating the time stepping errors. An exact (in Δt) integration method would allow this particle to pass all the way across our system with the same momentum, $p^x(t) = p^x_0$ and $p^y(t) = 0$. Since we do not have an exact integrator, we know there will be some deviation as the particle comes close to the origin, and this test quantifies that error. Below we develop the method used for this test.

Assuming a plane-polar system, such that

$$x = r\cos\theta \tag{3.48a}$$

$$y = r\sin\theta \tag{3.48b}$$





(b)

Figure 3.8: Timestep comparison for a single particle moving across a polar-coordinate grid starting at $x = -5$, $y = 0.1$ with an initial velocity $p_{x0} = 0.1$, $p_{y0} = 0$, showing (a) p_x vs x and (b) y-position vs time as the particle moves across the system in x.

we are simultaneously making the transformations

$$(x, y, p_x, p_y) \rightarrow (\xi, \eta, P_\xi, P_\eta) \tag{3.49}$$

$$(x, y, p_x, p_y) \rightarrow (r, \theta, P_r, P_\theta) \tag{3.50}$$

such that $\xi \rightarrow r$, $\eta \rightarrow \theta$, $P_\xi \rightarrow p_r$, and $P_\eta \rightarrow p_\theta$. By the definition of our contact transformation, we know that $K = H$. In polar coordinates, the Hamiltonian takes the form

$$H(r, \theta, p_r, p_\theta) = \frac{1}{2m}\left(p_r^2 + \frac{p_\theta^2}{r^2}\right) + q\Phi(\vec{r}), \tag{3.51}$$

where we are assuming the the potential term $q\Phi(\vec{r}) = 0$. Applying Hamilton's equations to Eqn. (3.51) gives the particle equations of motion:

$$\dot{r} = \frac{p_r}{m} \tag{3.52a}$$

$$\dot{\theta} = \frac{p_\theta}{mr^2} \tag{3.52b}$$

$$\dot{p_r} = \frac{p_\theta^2}{mr^3} \tag{3.52c}$$

$$\dot{p_\theta} = 0. \tag{3.52d}$$

Notice that the $\dot{p_r}$ term is an inertial force term, equivalent to that of Eqn. (3.35b) for this system.

Fig. 3.8 shows that our ML integrator is 2nd order accurate in time, as we expected. The results in Fig. 3.8 indicate that small variations in the particle momentum are to be expected as the particle moves across the grid in our full PIC simulations. However, we know that this variation is bounded in timestep, scaling as Δt^2.

3.5 Generalized Curvilinear Coordinate Poisson Solver

In order to solve the electrostatic field equation on the logical grid, we must construct the generalized curvilinear coordinate formulation of Poisson's equation,

$$\nabla^2\Phi = -\frac{\rho^x}{\epsilon_0} \tag{3.53}$$

on the logical grid. In this notation, ρ^x is the physical charge density. Using this formulation, we are able to solve the field equation in the physical grid geometry and coordinate system of our choice. This logical grid solver also removes the necessity of writing and maintaining multiple complex-geometry Poisson solvers for the various coordinate systems we wish to model.

We begin by rewriting Eqn. (3.53) as the divergence of the gradient of the potential in generalized coordinates:

$$\frac{1}{f}\nabla \cdot f\nabla\Phi = \frac{1}{f}\frac{\partial}{\partial x^\alpha} \cdot f\frac{\partial\Phi}{\partial x^\alpha} = -\frac{\rho}{\epsilon_0}, \tag{3.54}$$

where f is a *geometry factor* allowing us to switch between azimuthal and axially-symmetric systems as in Eqns. (3.31) and (3.32). For instance, if we want the physical coordinate system to be x, y with $\frac{\partial}{\partial z} = 0$, we set $f = 1$. For r, z coordinates with $\frac{\partial}{\partial \phi} = 0$, we set $f = r$.

In matrix notation we can write $\nabla\Phi$ as

$$\begin{aligned} \nabla\Phi &= \frac{\partial\Phi}{\partial\xi^m}\nabla\xi^m \\ &= A_m\nabla\xi^m \end{aligned} \tag{3.55}$$

where A_m are the covariant components of the 2d vector

$$\vec{A} = A_m\nabla\xi^m = A_1\nabla\xi + A_2\nabla\eta. \tag{3.56}$$

Likewise, we can represent $\vec{A}$ by its contravariant components, A^m:

$$\vec{A} = A^1 \nabla\eta \times \hat{z} + A^2 \hat{z} \times \nabla\xi, \tag{3.57}$$

such that we can write

$$A_1 \nabla\xi + A_2 \nabla\eta = A^1 \nabla\eta \times \hat{z} + A^2 \hat{z} \times \nabla\xi. \tag{3.58}$$

We can now find the direct relationship between the covariant and contravariant components of $\vec{A}$. For example, taking the inner product of Eqn. (3.58) with $\nabla\eta \times \hat{z}$ we find:

$$A^1 \left(\nabla\eta \times \hat{z}\right) \cdot \nabla\xi = A_1 |\nabla\xi|^2 + A_2 \nabla\eta \cdot \nabla\xi. \tag{3.59}$$

By Eqn. (3.7),

$$\nabla\xi^\beta \cdot \nabla\xi^\gamma = g^{\beta\gamma}, \tag{3.60}$$

allowing us to rewrite Eqn. (3.59) as

$$A_1 g^{11} + A_2 g^{12} = A^1 \left(\nabla\eta \times \hat{z}\right) \cdot \nabla\xi = A^1 \nabla\xi \cdot \nabla\eta \times \hat{z}. \tag{3.61}$$

We note that

$$\begin{aligned} \nabla\xi \cdot \nabla\eta \times \hat{z} &= \frac{\partial\xi}{\partial x}\frac{\partial\eta}{\partial y} - \frac{\partial\xi}{\partial y}\frac{\partial\eta}{\partial x} \\ &= \det\left(\boldsymbol{k}\right) \\ &= \frac{1}{J}, \end{aligned} \tag{3.62}$$

so we can write the left-hand side of Eqn. (3.61) as

$$A_1 g^{11} + A_2 g^{12} = \frac{A^1}{J}. \tag{3.63a}$$

Similarly,

$$A_1 g^{21} + A_2 g^{22} = \frac{A^2}{J}. \tag{3.63b}$$

Returning now to Eqn. (3.54) and using Eqns. (3.55) and (3.57) and the identity

$$\nabla \cdot (c\vec{v}) = \nabla c \cdot \vec{v} + c\nabla \cdot \vec{v}, \tag{3.64}$$

we can write the Laplacian operator as

$$\begin{aligned}
\frac{1}{f}\nabla \cdot f\nabla\Phi &= \nabla \cdot \vec{A} \\
&= \frac{1}{f}\nabla \cdot \left[fA^1(\nabla\eta \times \hat{z}) + fA^2(\hat{z} \times \nabla\xi)\right] \\
&= \frac{1}{f}\left[\nabla(fA^1) \cdot (\nabla\eta \times \hat{z}) + \nabla(fA^2) \cdot (\hat{z} \times \nabla\xi)\right].
\end{aligned}$$

Using $\nabla\xi \cdot (\nabla\xi \times \hat{z}) = 0$, Eqn. (3.65) can be written as

$$\frac{1}{f}\nabla \cdot f\nabla\Phi = \left[\frac{1}{f}\frac{\partial(fA^1)}{\partial\xi}\nabla\xi \cdot (\nabla\eta \times \hat{z}) + \frac{1}{f}\frac{\partial(fA^2)}{\partial\eta}\nabla\eta \cdot (\hat{z} \times \nabla\xi)\right]. \tag{3.65}$$

By Eqn. (3.62), we can rewrite Eqn. (3.65) as

$$\frac{1}{f}\nabla \cdot f\nabla\Phi = \frac{1}{fJ}\left(\frac{\partial(fA^1)}{\partial\xi} + \frac{\partial(fA^2)}{\partial\eta}\right) = -\frac{\rho}{\epsilon_0}. \tag{3.66}$$

Now converting the contravariant components of $\vec{A}$ to their covariant formulations using Eqns. (3.55) and (3.61), we can write the final form of the Poisson equation in curvilinear coordinates:

$$\frac{1}{fJ}\frac{\partial}{\partial\xi^\alpha}\left(fJg^{\alpha\beta}\frac{\partial\Phi}{\partial\xi^\beta}\right) = -\frac{\rho}{\epsilon_0}, \tag{3.67}$$

where $\rho = \rho^x$ is the *physical* charge density.

Chapter 4

Development of a 1d Logical Grid PIC Code

The first step in developing any new method is always to start as simply as possible. In our case, the simplest possible approach to building a nonuniform grid PIC code is to develop a 1d nonuniform grid code, and add increasing complexity as our understanding of the method grows. Furthermore, the notation and concepts of the nonuniform grid PIC method are much more easily understood in 1d.

4.1 Code Normalizations

It is often convenient to use normalized units in a computer code. By making wise normalizations, we introduce more convenient notation in which the dimensional parameters are lumped together into a smaller number of dimensionless parameters, thereby reducing the parameter space without any new physics being added or lost. As such, we now outline the procedure of converting the “real” macroparticle quantities into dimensionless parameters.

Parameters	Q, M, N_0, ϵ_0, L
Dynamical Variables	x, v, t, E^x, N

Table 4.1: List of the dimensional and independent parameters for a single species of macroparticles moving according to the 1d particle equations of motion.

In 1d, the physical grid equations of motion are given by Eqns. (2.18a) and (2.18b), where the electric field is obtained using Gauss' law,

$$\frac{dE^x}{dx} = \frac{NQ}{\epsilon_0} = \frac{\rho}{\epsilon_0}. \tag{4.1}$$

These equations contain the dimensional and independent parameters listed in Table 4.1. For simplicity, we outline the normalization procedure for a single species. Extension to other species is straightforward.

We begin by introducing the following normalizations on the physical plasma properties. Time scales are normalized with respect to a frequency, Ω, to be determined later:

$$\tilde{t} = \Omega\, t. \tag{4.2}$$

Length scales are normalized a characteristic length scale of the system,

$$\tilde{x} = \frac{x}{L}. \tag{4.3}$$

For the 2d annulus, we normalize with respect to the *outer* radius of the system, r_2. We are primarily concerned with two domains in this thesis: the annulus and the square grid. Using Eqns. (4.2) and (4.3), we can calculate normalized velocities (in 1d):

$$\tilde{v} = \frac{d\tilde{x}}{d\tilde{t}} = \frac{1}{L\,\Omega}\frac{dx}{dt} = \frac{v}{L\,\Omega}. \tag{4.4}$$

We define the normalized macroparticle density as

$$\tilde{N} = \frac{N}{N_0}, \tag{4.5}$$

where N_0 is the average density of the initial state, and normalize the electric fields as

$$\tilde{E} = \frac{E^x}{E_0}, \tag{4.6}$$

where E_0 is a normalization parameter to be determined.

Using these normalizations, Eqn. (2.18a) is

$$\frac{d\tilde{x}}{d\tilde{t}} = \tilde{v}, \tag{4.7}$$

Eqn. (2.18b) becomes

$$\frac{d\tilde{v}}{d\tilde{t}} = -\left(\frac{QE_0}{ML\Omega^2}\right)\tilde{E}, \tag{4.8}$$

and Eqn. (4.1) becomes

$$\frac{d\tilde{E}^x}{d\tilde{x}} = -\left(\frac{N_0QL}{\epsilon_0 E_0}\right)\tilde{E}. \tag{4.9}$$

We want to normalize such that the terms in parentheses in Eqns. (2.18b) and (4.9) are one, i.e.

$$\begin{aligned} \tfrac{eE_0}{ML\Omega^2} &= 1 \\ \tfrac{N_0QL}{\epsilon_0 E_0} &= 1. \end{aligned} \tag{4.10}$$

From the second term, we see $E_0 = \frac{N_0QL}{\epsilon_0}$. Plugging this value back into the first term and solving for Ω gives

$$\Omega^2 = \frac{N_0Q^2}{\epsilon_0 M} \equiv \omega_{pe}^2. \tag{4.11}$$

This means that (for electrons) we can set $|Q| = M = \epsilon_0 = 1$, such that we can rewrite the equations of motion of the normalized macroparticles as

$$\begin{aligned} \tfrac{d\tilde{x}}{d\tilde{t}} &= \tilde{v} \\ \tfrac{d\tilde{v}}{d\tilde{t}} &= -\tilde{E}^x \\ \tfrac{d\tilde{E}^x}{d\tilde{x}} &= -\tilde{N} \end{aligned} \tag{4.12}$$

such that $\omega_{pe}^2 = 1$. We find by direct substitution of the "real" macroparticle quantities defined in Eqn. (2.6) that the Eqns. 4.12 remain unchanged.

4.2 1d Governing Equations

In a system with a single free parameter in both space and velocity (1d1v), the logical grid Hamiltonian equations of motion, Eqns. (3.35) can be dramatically simplified into the form:

$$\dot{\xi} = \frac{P}{MJ^2}, \tag{4.13a}$$

and

$$\dot{P} = -\frac{P^2 J'}{MJ^3} + QE^{\xi}, \tag{4.13b}$$

where $J \equiv \frac{dx}{d\xi}$ and $J' \equiv \frac{dJ}{d\xi} = \frac{d^2x}{d\xi^2}$ are the Jacobian and its derivative, and $E^{\xi} \equiv -\frac{d\Phi}{d\xi}$ is the logical grid electric field. Here and in the rest of this work we have kept the normalized macroparticle charges and masses in the particle equations to show the most general form for species where Q and M are not necessarily one. Furthermore, the logical grid Poisson equation, Eqn. (3.67), simplifies to

$$\frac{1}{J}\frac{d}{d\xi}\left(\frac{1}{J}\frac{d\Phi}{d\xi}\right) = -\rho^x, \tag{4.14}$$

where again ρ^x is the charge density on the physical grid. However, since $\rho^x \, dx = \rho^{\xi} \, d\xi$, we can write

$$\frac{d}{d\xi}\left(\frac{1}{J}\frac{d\Phi}{d\xi}\right) = -\rho^{\xi}, \tag{4.15}$$

where ρ^{ξ} is the charge density on the logical grid as obtained by Eqn. (2.26). The ability to accumulate ρ^{ξ} on the logical grid is a very beneficial feature of our method, as location and weighting on the nonuniform physical grid can be quite difficult and requires special algorithms to minimize the amount of error introduced into the code (see for example Ref. [18]).

For our 1d code, we have set up a nonuniform grid given by

$$x(\xi) = \xi + \epsilon_{\text{grid}} \sin(2\pi\xi) \tag{4.16a}$$

such that for testing purposes we can analytically calculate the grid Jacobian and its derivative:

$$J = 1 + 2\pi\epsilon_{\text{grid}}\cos(2\pi\xi) \tag{4.16b}$$

$$J' = -4\pi^2\epsilon_{\text{grid}}\sin(2\pi\xi). \tag{4.16c}$$

This grid was chosen for its periodicity, such that the test cases outlined in the remainder of this chapter have seamlessly-periodic boundary conditions on both the fields and the grid. To vary the nonuniformity of the grid, we simply change ϵ_{grid}. To prevent the grid from folding, the maximum grid nonuniformity parameter we can choose is $\epsilon_{\text{grid}} = \frac{1}{2\pi} \approx 0.16$. For a uniform grid, we simply set $\epsilon_{\text{grid}} = 0$.

While this particular grid allows us to find simple analytic solutions for the Jacobian and its derivative, it is important to keep in mind that the idea behind our PIC method is that we can couple it to any kind of grid generation strategy. As such, we must in general obtain the grid derivatives by centered-differencing techniques. Since $x(\xi$ is defined on the vertices, the Jacobian is naturally defined on the cell-centers, and its derivative J' on the vertices.

4.2.1 Discretization of the Poisson Equation

Eqn. (4.15) can be discretized on the logical grid by first applying central differences to the inner term,

$$\left[\frac{1}{J}\frac{d\Phi}{d\xi}\right]_i = \frac{\Phi_{i+\frac{1}{2}} - \Phi_{i-\frac{1}{2}}}{J_i^v\Delta\xi}, \tag{4.17}$$

where $J_i^v \equiv \frac{J_{i+\frac{1}{2}}+J_{i-\frac{1}{2}}}{2}$ is the Jacobian at the vertex. Now applying central differences to the outer term of Eqn. (4.15) gives

$$\frac{\Phi_{i+\frac{3}{2}} - \Phi_{i+\frac{1}{2}}}{J_{i+1}^v} - \frac{\Phi_{i+\frac{1}{2}} - \Phi_{i-\frac{1}{2}}}{J_i^v} = -\Delta\xi^2\,\rho^{\xi}_{i+\frac{1}{2}}. \tag{4.18}$$

Notice that for a uniform grid ($\Delta\xi = \Delta x$), Eqn. (4.18) reduces to Eqn. (2.21). Eqn. (4.18) can easily be solved using a direct tridiagonal solver. However, since we have constructed the 1d code as a first stepping-block to the more complex 2d code where a direct tridiagonal solver would not work, we have chosen to solve the symmetric matrix generated by Eqn. (4.18) with a CG solver as in § 2.3.

Solvability Condition of Poisson Equation

The test cases presented in this thesis rely on the use of periodic boundary conditions on the fields. However, without at least one Dirichlet boundary condition, Poisson's equation has a 1d null space, i.e. it is negative semi-definite. It is easy to see that for periodic (or Neumann) boundary conditions, the Laplacian term $\nabla^2\Phi$ can be written as $\nabla^2\bar{\Phi}$, where $\bar{\Phi} = \Phi + C$ and C is a constant. Since $\nabla^2 C = 0$, this also satisfies the equation, meaning that there is not a unique solution to Poisson's equation. This will cause a lack of convergence in our CG solver if ρ is not in the range space of $\nabla^2\Phi$. We therefore ensure that ρ is in the range space of the Laplacian operator by integrating the charge density over the entire volume and renormalizing the background ion density such that

$$\int \rho^x \, dV = 0. \tag{4.19}$$

If no electrons have left the system and there is a neutralizing ion background, this should hold, but our renormalization procedure guarantees that exactly. Eqn. (4.19) is the solvability condition in the continuum. On the discrete grid, it is obtained using

$$\sum_i \rho^{\xi}_{i+\frac{1}{2}} = 0. \tag{4.20}$$

Assuming Φ satisfies Eqn. (4.18), this holds exactly for periodic or Neumann boundary conditions, because the sum telescopes.

Electric Field

Eqn. (4.13b) dictates that we use the logical grid electric field at the particle's position on the logical grid. The most obvious method to do this would seem to be by simply differencing Φ with respect to ξ,

$$E_i^\xi = -\frac{\Phi_{i+\frac{1}{2}} - \Phi_{i-\frac{1}{2}}}{\Delta\xi}, \tag{4.21}$$

where we have defined a staggered grid such that E^ξ exists on vertices and Φ and ρ^ξ exist on cell centers. To check the accuracy of our method, we set up a periodic system in which a single, stationary negatively charged particle in a uniform, neutralizing ion background (which is necessary to meet the solvability condition) is weighted to the grid, the fields are solved for, and the electric field E^ξ is interpolated to the particle position. From § 2.4.1 and Appendix B, we know that if the weighting and interpolation schemes are devised correctly, the self-force on a particle should be zero to computer precision. However, upon completing this calculation for the logical electric field, we find that there are actually non-zero self-fields associated with this method. We observe that these fields scale with 2^{nd}-order accuracy in Δx, but nevertheless are not machine precision zero.

We have therefore implemented a second, less direct method of obtaining the logical electric fields at the particle position, $\xi = \xi_p$, in which we solve for the physical electric field on the logical grid and interpolate this field to the particle position. As such, we calculate the physical electric fields on the vertices, formulated on the logical grid as

$$E_i^x = -\frac{\Phi_{i+\frac{1}{2}} - \Phi_{i-\frac{1}{2}}}{J_i^v \Delta\xi}, \tag{4.22}$$

and interpolate this to the particle's logical space position. Since $E^\xi = JE^x$, we then also interpolate the grid jacobian to the particle position and retroactively multiply $E^x(\xi_p)$ and $J(\xi_p)$ to obtain $E^\xi(\xi_p)$. This is slightly more computationally expensive than the direct interpolation of E^ξ approach described above, but repeating the test above reveals that this method leads to computer-precision zero self-fields at the particle position in 1d. Appendix **??** provides a clear explanation as to why this method leads to zero self-forces at the particle and the direct interpolation method does not.

4.2.2 Using Gauss' Law to Obtain Fields Directly

Unique to a 1d system, we can also obtain the electric fields directly from the charge density using Gauss' law (Eqn. (4.1)) as a method for comparison with our logical grid Poisson solver. Rewriting Eqn. (4.1) in terms of a derivative on the logical grid as

$$\frac{dE^x}{d\xi} = \rho^x J = \rho^\xi, \tag{4.23}$$

we can now discretize Eqn. (4.23) by central differences. Rearranging such that the physical electric field at vertex $i+1$ can be updated using the electric field at i and the charge density at the cell center $i+\frac{1}{2}$ by integrating over the logical domain gives

$$E^x_{i+1} = E^x_i + \Delta\xi \rho^\xi_{i+\frac{1}{2}}. \tag{4.24}$$

Since the electric field and charge density are on staggered grids, the result of this integration is 2nd order accurate in Δx. A constant of integration due to the boundary conditions on the potential also comes out of the integration of Eqn. (4.23) and must be accounted for to fix the electric field such that $\int E^x dx = 0$:

$$\Delta\Phi \equiv -\int E^x \, dx, \tag{4.25}$$

where we have assumed no external electric force has been applied. As such, Eqn. (4.24) can then be written

$$E^x_{i+1} = E^x_i + \Delta\xi \, \rho^\xi_{i+\frac{1}{2}} + \Delta\Phi. \tag{4.26}$$

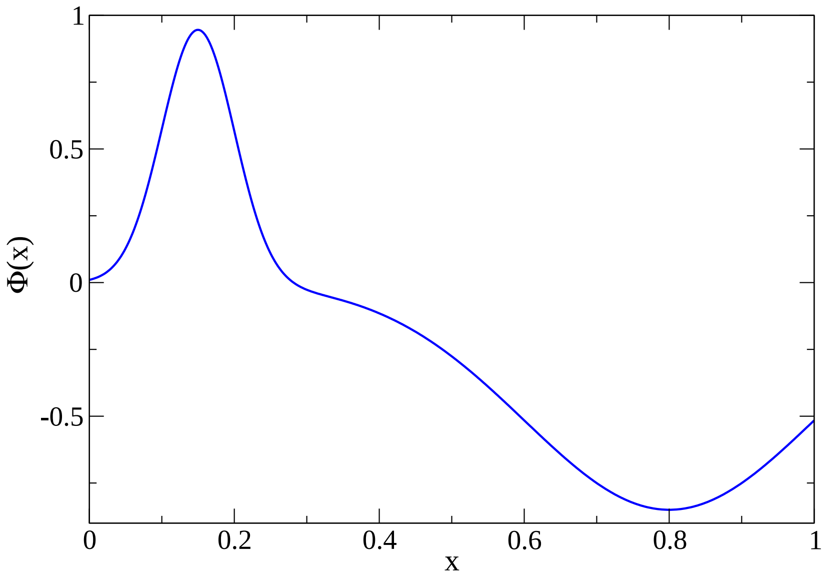
4.2.3 Validation of the Field Solvers using the Method of Manufactured Solutions

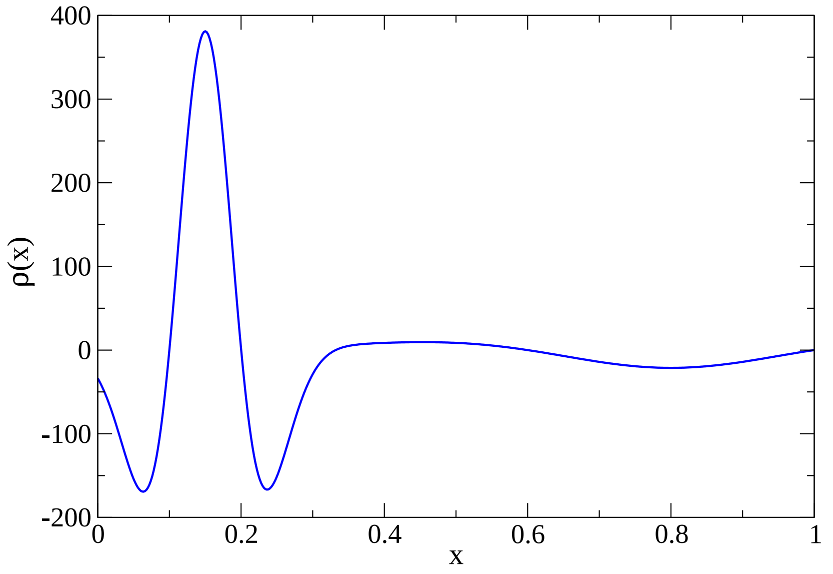
To compare the two methods of obtaining the electric field given by Eqns. (4.18) and (4.26), we have set up at test using the Method of Manufactured Solutions (MMS) [46]. MMS is an amazingly simple, yet powerful tool to construct solutions to PDE problems. By obtaining analytic solutions to problems, we are able to check that the error in the numerical solution converges to the analytic solution with 2^{nd} order accuracy. MMS is based upon choosing a function which satisfies a given set of boundary conditions *a priori*, then taking the required derivatives to find the source term. The source term is then inserted into the numerical system and the solution is compared to the exact solution. For example, with Poisson's equation, we simply choose a potential that satisfies a chosen set of boundary conditions, calculate the charge density, and use that density in our solver. We note here that the Poisson solver gives us Φ as an output, whereas the Gauss' law solver gives E^x. We can therefore compare both solvers by either differencing Φ obtained by the Poisson solver to obtain E^x or by integrating the electric field obtained by the Gauss' law solver to obtain Φ. We have decided to employ both methods as one more measure of comparison between the two methods.

For our 1d studies, we wish to construct a non-trivial potential with inhomogenous Dirichlet boundary conditions in an attempt to demonstrate the flexibility of our field solvers. As such, we have chosen a potential given by the sum of two Gaussians:

$$\Phi^{\mathrm{MMS}}(x) = A_1 \exp\left(-\frac{(x-B_1)^2}{2C_1^2}\right) - A_2 \exp\left(-\frac{(x-B_2)^2}{2C_2^2}\right). \tag{4.27}$$

Using the 1d Cartesian Poisson equation (Eqn. (2.4)), we can analytically calculate the charge density to be inserted into the solvers developed in § 4.2.1 and





(b)

Figure 4.1: 1d MMS potential as constructed by Eqn. (4.27) and the analytic 1d MMS charge density given by Eqn. (4.28).

N_ξ	$\epsilon_{\text{grid}} = 0$	$\epsilon_{\text{grid}} = 0.01875$	$\epsilon_{\text{grid}} = 0.0375$	$\epsilon_{\text{grid}} = 0.075$	$\epsilon_{\text{grid}} = 0.15$
25	2.57922E-2	2.93183E-2	3.33245E-2	4.17780E-2	5.70396E-2
50	6.64361E-3	7.57754E-3	8.65505E-3	1.10612E-2	1.66014E-2
100	1.67228E-3	1.90873E-3	2.18258E-3	2.79883E-3	4.24558E-3
200	4.18769E-4	4.78063E-4	5.46800E-4	7.01750E-4	1.06707E-3
400	1.04736E-4	1.19571E-4	1.36772E-4	1.75564E-4	2.67118E-4

Table 4.2: L_2-norm error between computational and analytic MMS potentials for different grid resolutions and grid nonuniformities (ϵ_{grid}) for the both the Poisson and Gauss' Law solvers. Errors between the two solvers are in the 10^{-15} range.

§ 4.2.2:

$$\rho^{\text{MMS}}(x) = \frac{A_1 \exp\left(-\frac{(x-B_1)^2}{2C_1^2}\right)(B_1 + C_1 - x)(x - B_1 + C_1)}{C_1^4} + \frac{A_2 \exp\left(-\frac{(x-B_2)^2}{2C_2^2}\right)(B_2 + C_2 - x)(x - B_2 + C_2)}{C_2^4}. \quad (4.28)$$

Furthermore, using Eqn. (2.5), we can also calculate the physical MMS electric field:

$$E^{\text{MMS}}(x) = \frac{A_1 \exp\left(-\frac{(x-B_1)^2}{2C_1^2}\right)(x - B_1)}{C_1^2} - \frac{A_2 \exp\left(-\frac{(x-B_2)^2}{2C_2^2}\right)(x - B_2)}{C_2^2}, \quad (4.29)$$

which can also be compared with the computational solutions of the Poisson and Gauss' Law solvers. For the following 1d MMS tests, we have employed the following constants:

$$\begin{aligned} A_1 &= 0.95 & A_2 &= -0.85 \\ B_1 &= 0.15 & B_2 &= 0.8 \\ C_1 &= 0.05 & C_2 &= 0.2. \end{aligned}$$

Figs. (4.1(a)) and (4.1(b)) show the MMS potential and charge density as given by Eqns. (4.27) and (4.28) using these constants.

Upon convergence, the numerical solution and the exact solution are compared,

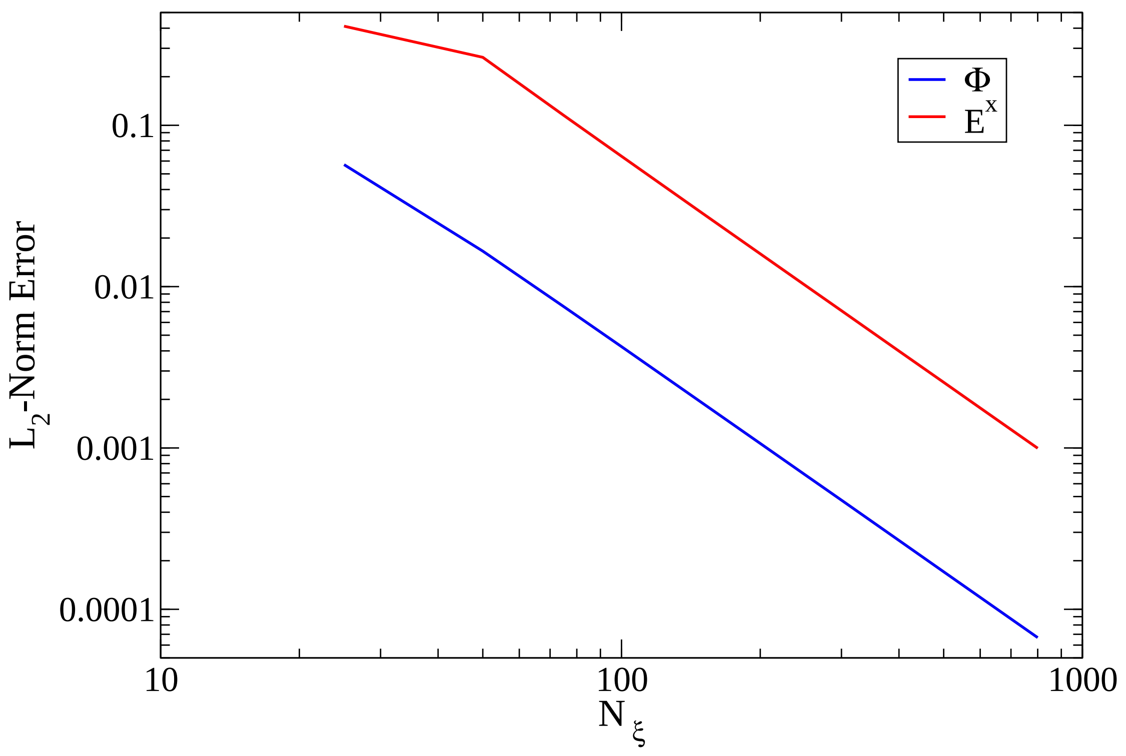


Figure 4.2: Scalings of L_2-norm of errors between computational and analytic solutions for potential and electric field. Here we are using the nonuniformity parameter $\epsilon_{grid} = 0.15$.

and the error between the two is calculated using the L_2-norm:

$$||f||_2 = \sqrt{\frac{\sum_{i=1}^{N_\xi} \left(f_i^{\text{num}} - f_i^{\text{MMS}}\right)^2}{N_\xi}}, \tag{4.30}$$

where f is either Φ or E^x. The data displayed in Table (4.2) is the error in Φ and was obtained using a grid given by Eqn. (4.16a) for a range of nonuniformities. Both the potential and electric field errors between the two solvers are indeed equal to approximately 10^{-15}, and as such the data displayed in Table (4.2) holds for either solver. We note here that while this data scales with 2^{nd}-order accuracy, the error coefficient increases as the grid nonuniformity increases. This is due to the fact that as ϵ_{grid} increases, the ratio of the length of the maximum grid cell size to the smallest becomes quite large, and accordingly the solution is less accurate in the region where the grid is coarse.

We display the error between the numerical and MMS solutions as a function of the number of grid cells for both the potential and electric field in Fig. (4.2). We note that in the case of the lowest resolution electric field data, the scaling appears to be only 1^{st}-order accurate, becoming 2^{nd}-order accurate as the spatial resolution increases. The reason this apparent 1^{st}-order scaling appears in the electric fields but not in the potential is that, by calculating E^x, we are effectively "roughing" up the data with a derivative, and noise caused by under-resolution of the domain becomes more apparent. Likewise, the process of integrating the electric field to obtain Φ in the Gauss' law solver smoothes the noise inherited by a single integration of the charge density to obtain E^x with this method.

4.2.4 ML Mover Applied to 1d Non-Uniform Grid

If there is no external force applied to a particle moving across a periodic system, its physical velocity should remain a constant. However, in § 3.4.3, we found that our ML mover leads to an error of order Δt^2, and as such we know that the physical velocity will have some variation as a particle crosses a nonuniform grid.

To test the accuracy of our mover on the inertial force term in Eqn. (4.13b), we set up a system in which a single test particle with some initial physical velocity v_{x0} moves across a periodic 1d, nonuniform grid given by Eqn. (4.16a). To ensure charge neutrality, we assign a uniform ion background, the total charge of which is equivalent to the magnitude of the test particle charge. As this system is identical to that described in § 4.2.1, we know the self-forces on the particle are zero, and we are therefore justified to drop the field force term completely from the following tests in order to focus solely on the effects of the inertial term.

In our curvilinear coordinate PIC method, Eqn. (4.13b) dictates that we must also obtain the grid-based quantities $J(\xi_p)$ and $J'(\xi_p)$ at the particle positions. Fortunately, since we have chosen an analytic function for our grid, we can obtain these values by inserting the particle positions on the logical grid directly into Eqns. (4.16b) and (4.16c). As such, we can use these "analytic grid" values to eliminate all spatial scaling effects due to the interpolation of grid-based quantities from the problem to see the magnitude of the variation in the inertial force term as predicted in § 3.4.3, where we found that the inertial force term of Eqn. (4.13b) is not handled exactly, but instead gives a 2nd-order accurate approximation in Δt.

We note here that the analytic grids used here and in the rest of this thesis are employed solely as debugging tools. In the most general case in which the grid is obtained via some outside grid generator there is not a simple analytic expression which can be applied. All test cases in this and the following chapters which incorporate the full PIC code will therefore be conducted using an interpolated grid to describe more fully the properties of our method.

Using the initial conditions $N_\xi = 100$, $v_{x0} = 0.01$, and $\epsilon_{\text{grid}} = 0.15$, we can determine how small a timestep is necessary to meet the macroparticle Courant condition [10, 11],

$$v_{\max}\Delta t < \Delta x_{\min}. \tag{4.31}$$

Since Δx varies widely across the physical grid, we have decided to play it safe and assume that we must satisfy the most stringent requirement, with Δx_{min}. From Eqn. (4.16b), we can calculate the length of the smallest cell, $\Delta x_{\min} = \Delta\xi\,(1 - 2\pi\epsilon_{\text{grid}}) \approx 5.7 \times 10^{-4}$. Thus, by our initial conditions, Eqn. (4.31) requires a timestep $\Delta t < 5.7 \times 10^{-2}$. To be safe, we have set up a parameter scan over $\Delta t = 0.003125$, 0.00625, 0.0125, 0.025, and 0.05, to follow the physical velocity as a function of its logical grid position, as shown in Fig. 4.3. From Fig. 4.3, it is quite evident that the error in the present system scales as Δt^2. Further tests scanning over N_ξ show that in this particular test case there is no scaling with grid resolution, as is expected when the

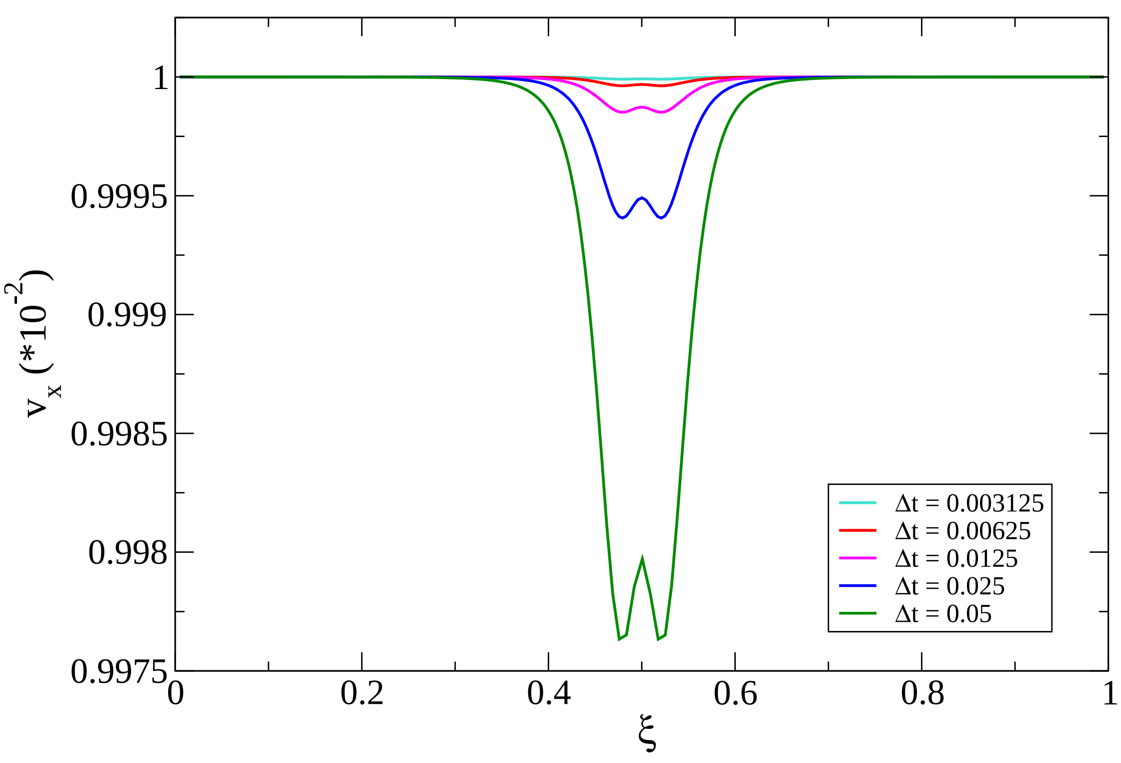


Figure 4.3: Comparison of particle velocity vs. logical position for various timesteps. Here $N_\xi = 100$, $\epsilon_{\text{grid}} = 0.15$ and we are using an analytic system with no grid interpolations.

grid values at the particle positions are given exactly by the analytic grid.

Finally, we wish to quantify the additional amount of error added in our system by grid interpolation rather than an analytic solution. In Fig. (4.4), we show a comparison of the effects of linear, quadratic, and cubic spline interpolation of the grid quantities to the particle position on the initial physical velocity as the particle moves across the grid. We have also plotted the analytic grid solution for reference. It is quite clear that the quadratic and cubic spline shape (interpolation) functions provide a much smoother result than the linear interpolation case. The difference between the quadratic and cubic splines interpolation cannot be seen at the resolution level shown in Fig. (4.4), revealing that the step from quadratic to cubic splines

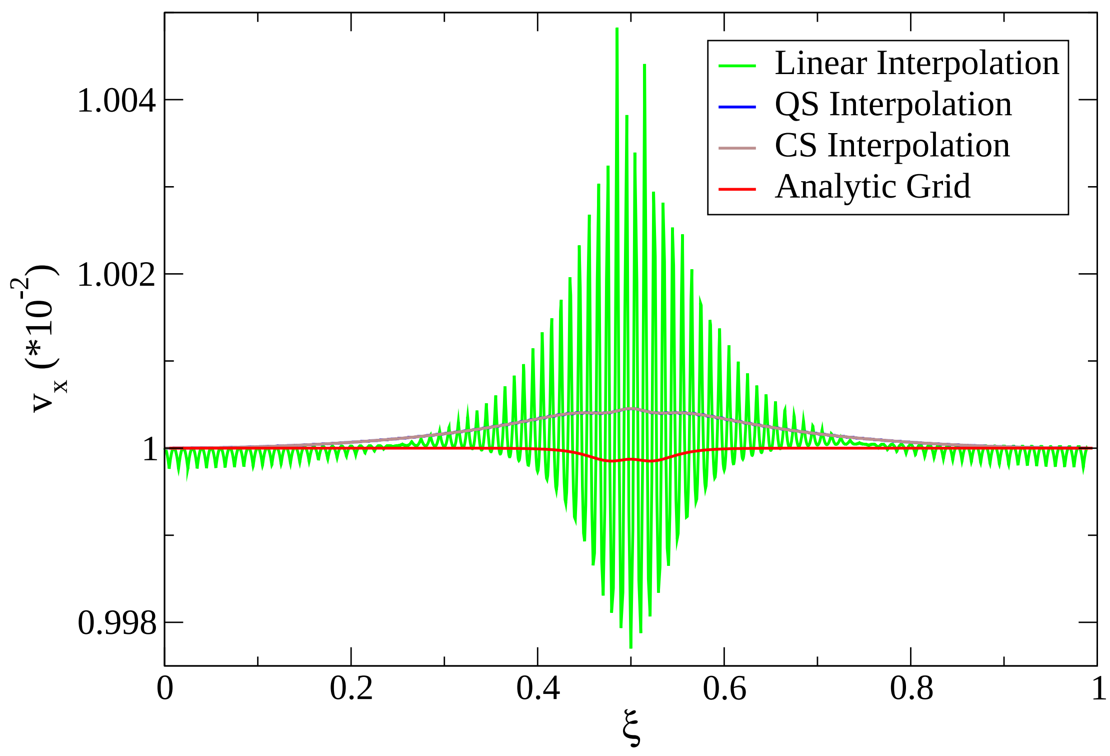
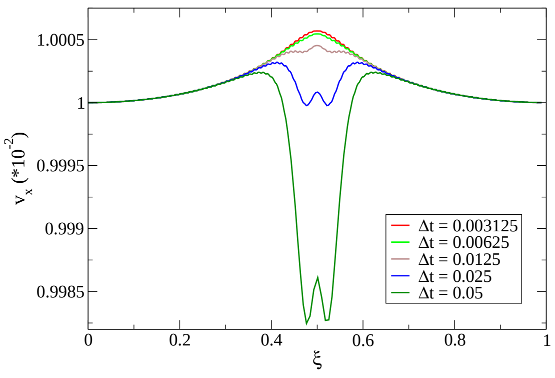
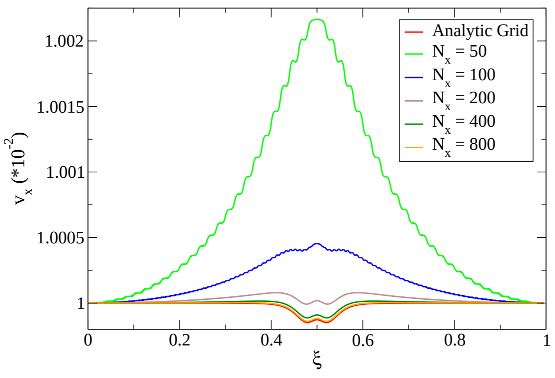


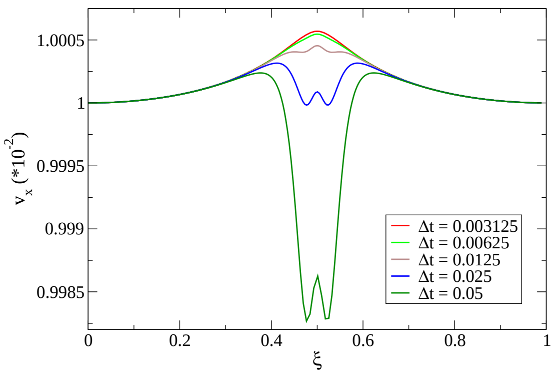
Figure 4.4: Comparison of particle velocity vs. logical position for linear, quadratic, and cubic b-spline interpolation, and the analytic grid solution with $N_\xi = 100$, $v_{x0} = 0.01$, $\epsilon_{grid} = 0.15$ and $\Delta t = 0.0125$.

results in a much smaller gain in smoothness than the linear to quadratic step.

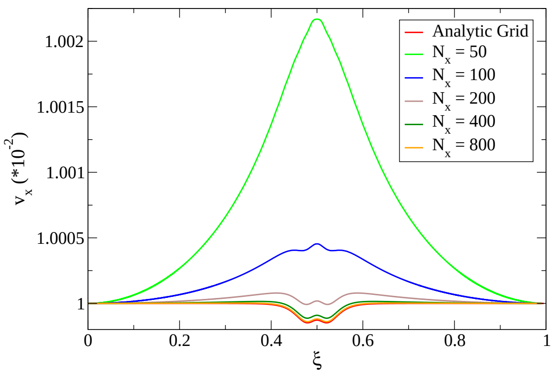
We have also performed a set of parameter scans over Δt, with N_ξ held constant and over N_ξ, with Δt held constant. Fig. (4.5)(a) shows that for a given N_ξ, as Δt gets smaller, the features of the system become more resolved with 2^{nd}-order accuracy in Δt, but are not moved closer to the analytic grid solution, which is plotted in Fig. (4.5)(b). This accuracy scaling is due to the Δt^2 scaling of the ML solver. Fig. (4.5)(b) shows that for a given Δt, as N_ξ is increased the solution moves ever closer to the analytic grid solution with 2^{nd}-order accuracy in N_ξ. This scaling is due to the interpolation of the grid. Note that the brown curves in Figs 4.5(a) and (c) are the same as the blue curves in Figs 4.5(b) and (d), respectively.







(c)



(d)

Figure 4.5: Single particle v_x vs ξ for different timesteps with fixed grid resolution $N_\xi = 100$ (a) and different grid resolutions with fixed timestep $\Delta t = 0.0125$ (b). Both cases use quadratic particle shapes for interpolation. The same runs were performed with cubic spline interpolation in (c) and (d).

4.3 Selected 1d Full PIC Benchmarking Cases

We have performed a vast array of tests on our code, but for the sake of brevity will detail only a few here. Several physics problems have become the standard tests for benchmarking a PIC code; among these are electrostatic plasma waves, two-stream instabilities, and Landau damping [10].

All tests described below are first performed on a uniform grid on $[0 : 1]$ in the physical space, such that any problems arising due to the non-uniform grid can be easily identified later and a baseline value can be compared to theory. We then perform the same test using a non-uniform grid given by Eqn. (4.16) to understand better the full effect of the grid on the problem at hand.

It is important to remember that we are trying to reproduce test cases designed to work well on the uniform grid, where smooth initial particle distributions can easily be achieved. Therefore, we must in general use both a higher number of grid points and particles per cell than is necessary to reproduce the same physics using standard uniform grid PIC techniques. We are not trying to show that our method performs better than the standard PIC codes often used on these problems, but rather trying to show that we can indeed reproduce the correct physics of these analytic problems to a high degree of accuracy before moving on to more difficult problems in more complex geometries, in which the nonuniform grid is adventageous.

Particle Initialization Techniques

For our cold plasma test cases, our PIC macroparticles are initialized for these tests using an extension of the well-known quiet start method [10]: we uniformly space particles throughout the (continuous) physical space, then map each particle to its logical position by inverting Eqn. (4.16a) using Newton iterations (the same inversion can be performed on any grid if $x(\xi)$ is given by a one-to-one map). Particle velocities are also initially generated in the physical space and transformed to to logical momentum P_ξ using Eqn. (3.29a), which in 1d can be written

$$P_\xi = J p_x. \tag{4.32}$$

While our method does not lead to an exact quiet start due to the fact that the number of particles per cell is not uniform across our nonuniform physical grid, it cancels enough particle noise to allow us to initialize a perturbation in the plasma system with a small amplitude.

In all the test cases described below, the electron-plasma component is followed numerically, whereas the ion component is assumed to be a uniform, static background on the physical grid such that $\rho^{\xi} = \rho^{\xi(i)} + \rho^{\xi(e)}$. As such, the logical grid ion charge density, $\rho^{\xi(i)}$, takes the form of the Jacobian on the logical grid,

$$\rho^{\xi(i)}(\xi) = J(\xi) = 1 + 2\pi\epsilon_{\text{grid}}\cos(2\pi\xi). \tag{4.33}$$

The electron charge density is accumulated using the weighting methods described in § 2.4:

$$\rho^{\xi(e)}_{i+\frac{1}{2}} = \sum_{p=1}^{N_p} QS(\xi_{i+\frac{1}{2}} - \xi_p). \tag{4.34}$$

Since these weighting methods are only approximations of the analytic electron charge density,

$$\hat{\rho}^{\xi(e)}(\xi) = -J(\xi) = -1 - 2\pi\epsilon_{\text{grid}}\cos(2\pi\xi), \tag{4.35}$$

we are automatically introducing some error into the system by adding the interpolated electron and exact ion components to obtain the total charge density. More specifically, the ion charge density component is *exactly* uniform on the physical grid, whereas the electron component has some slight variations from exact uniformity. Since this error is based on the interpolation properties and not the particle noise, it can be quantified by taking $N_p \to \infty$, such that Eqn. (4.34) becomes

$$\rho^{\xi(e)}(\xi') = \int_{-\infty}^{\infty} S(\xi - \xi')\hat{\rho}^{\xi(e)}(\xi')d\xi'. \tag{4.36}$$

Here, $\hat{\rho}^{\xi(e)}(\xi')$ is the exact electron density, equal in magnitude to the ion density, and $\rho^{\xi(e)}(\xi')$ is the electron density obtained by the weigthing scheme (in the limit that $N_p \to \infty$). We have written the limits of integration as over all space because $S(\xi - \xi')$ is localized by the definition of the particle shape function and we are ignoring the intracacies of the boundary conditions. Eqn. (4.36) is in the form of a convolution of the shape function and analytic charge density; thus we can rewrite it in the more convenient form

$$\rho^{\xi(e)}(\xi') = \int_{-\infty}^{\infty} S(\xi')\hat{\rho}^{\,\xi(e)}(\xi - \xi')\, d\xi'. \tag{4.37}$$

Applying a Taylor expansion to the exact electron density $\hat{\rho}^{\,\xi(e)}(\xi - \xi')$ around ξ allows us to write Eqn. (4.37) as

$$\rho^{\xi(e)}(\xi') = \int_{-\infty}^{\infty} S(\xi')\hat{\rho}^{\,\xi(e)}(\xi)\, d\xi' + \frac{1}{2}\int_{-\infty}^{\infty} S(\xi')\xi'^2\hat{\rho}^{\,\xi(e)''}(\xi)\, d\xi' \\ + \frac{1}{24}\int_{-\infty}^{\infty} S(\xi')\xi'^4\hat{\rho}^{\,\xi(e)''''}(\xi)\, d\xi', \tag{4.38}$$

where we have dropped the odd terms from the Taylor expansion because the particle shape function is an even function. Integrating each term over the interval in which the shape function is nonzero, e.g. $[-\Delta\xi : \Delta\xi]$ for linear weighting, and recalling from Eqn. (2.11) that the integral of the particle shape function is one, we can then carry out the integration of Eqn. (4.38). For linear weighting, we obtain

$$\rho^{\xi(e)}(\xi) = \hat{\rho}^{\,\xi(e)}(\xi) + \left(\frac{2\pi^3(\Delta\xi)^2\epsilon_{\text{grid}}}{3} - \frac{4\pi^5(\Delta\xi)^4\epsilon_{\text{grid}}}{45}\right)\hat{\rho}^{\,\xi(e)}(\xi); \tag{4.39a}$$

for quadratic splines it is

$$\rho^{\xi(e)}(\xi) = \hat{\rho}^{\,\xi(e)}(\xi) + \left(\pi^3(\Delta\xi)^2\epsilon_{\text{grid}} - \frac{13\pi^5(\Delta\xi)^4\epsilon_{\text{grid}}}{60}\right)\hat{\rho}^{\,\xi(e)}(\xi); \tag{4.39b}$$

and for cubic splines it is

$$\rho^{\xi(e)}(\xi) = \hat{\rho}^{\,\xi(e)} + \left(\frac{34\pi^3(\Delta\xi)^2\epsilon_{\text{grid}}}{45} - \frac{38\pi^5(\Delta\xi)^4\epsilon_{\text{grid}}}{45}\right)\hat{\rho}^{\,\xi(e)}(\xi). \tag{4.39c}$$

Using Eqns. (4.39), we can then calculate the amount of error introduced into the system by summing the electron charge density (obtained by weighting) and ion charge density (exact). For linear interpolation with $N_\xi = 100$ and $\epsilon_{\text{grid}} = 0.15$, the magnitude of the $O(\Delta\xi)^2$ correction term is $|\delta\rho| \approx 3.1 \times 10^{-4}$, meaning that even with an infinite number of particles, we would have a small error introduced into our system by the charge accumulation process. We can correct for this error by simply adding the higher-order correction terms in Eqns. (4.39) to the background ion charge density, assuring that the charge density is uniform on the physical grid to $O(\Delta\xi)^4$ in the limit of $N_p \to \infty$. To be clear, the effect of this difference in physical charge density leads to *inaccuracies*, it is completely separated from the *noise* effects due to having a finite number of particles within the system.

Since we are constrained to a finite number of particles in our system and the particle induced noise levels in a PIC code scale proportional to $\frac{1}{\sqrt{N_{\rm ppc}}}$ [10], we see that a highly nonuniform grid is capable of producing large levels of noise in the system, as the smallest cells contain many fewer computational particles than the largest. For example, with $\epsilon_{\rm grid} = 0.15$, the ratio of the largest cell to the average cell is $1 + 0.15 \times 2\pi \approx 1.94$ and the smallest is $1 - 0.15 \times 2\pi \approx 0.057$. If we are using an average of 100 particles per cell, the largest cells contain almost 200 particles, whereas the smallest have approximately 5! This can lead to large amounts of noise in the initial charge density where the grid spacing is small. This noise is then translated back to the particles through the field solve, and thus large amounts of noise can easily make their way through the system as time marches forward, leading to, for example, a loss of coherent wave structure.

An alternate method for this initialization process would be to initialize nonuniform macroparticle charges and masses based upon the size of the cell they are initially located in. These particles could then initially be placed uniformly on the *logical* grid such that each cell contains a uniform initial charge density. However, as time marches forward, this method introduces several problems of its own, namely that as the particles from larger cells move into the smaller cells, their statistical weight overwhelms that of several small particles and introduces numerical noise to the system. We note here that there have been several particle control methods devised to fix this type of problems [24, 25, 27, 28], but they are outside of the scope of this thesis.

4.3.1 Cold Plasma Oscillations

If the electrons in a cold plasma are displaced from their equilibrium positions in the presence of a uniform, static ion background, electric fields will be built up such that the electrons are pulled back towards their initial positions. However, due to their inertia, the electrons will overshoot this position and oscillate around their equilibrium positions at the electron plasma frequency, ω_{pe}.

Since this oscillation is on the electron timescale, it is justified to assume the ions are fixed. The electrons are perturbed from their initial physical equilibrium positions using

$$x_{\text{pert}} = x_0 + \epsilon_{\text{pert}} \sin 2\pi x_0, \tag{4.40}$$

where ϵ_{pert} is a small parameter to control the magnitude of the perturbation on the $k = 2\pi$ mode. The electrons are then allowed to oscillate around their equilibrium positions. As the code progresses in time, we record the electrostatic field energy of the system,

$$< E^2 >= \frac{1}{2}\int (E^x)^2 dx = \frac{1}{2}\int (E^x)^2 J d\xi. \tag{4.41}$$

The slope of the electrostatic field energy curve in time can be related to the growth rate, a measure of the imaginary frequency component of E^x, by

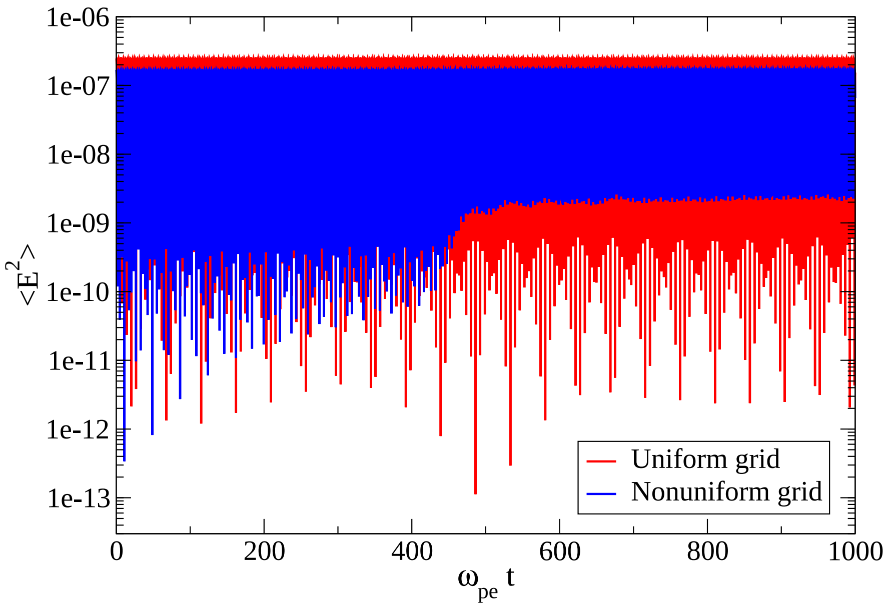
$$\gamma = \frac{\ln \frac{<E^2>_2}{<E^2>_1}}{2(t_2 - t_1)}. \tag{4.42}$$

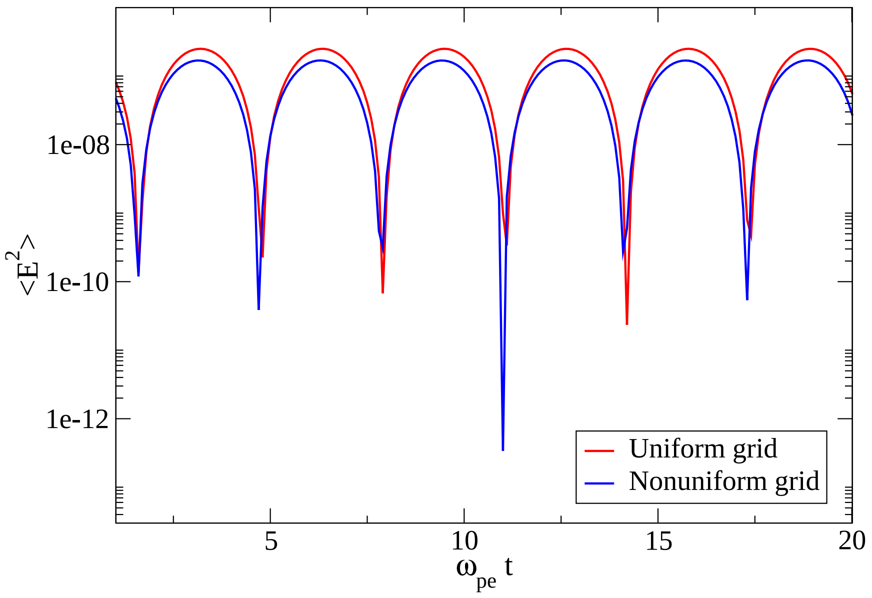
Here $< E^2 >_1$ is the value of the field energy of the system at time t_1 and $< E^2 >_2$ corresponds to time t_2.

For a cold initial distribution the growth rate should be exactly zero, but due to numerical heating [10] it is nearly impossible to eliminate all growth from the system even for a uniform grid PIC code. This test case is a particularly stringent test of our method, as any noise in the system can excite oscillations in the system due to grid aliasing [10, 11, 47, 48] which can compete with the primary wave we are trying to initialize and follow. This can lead to mode interference which looks like growth and decay within the system on very long timescales.

There are several methods of dealing with this noise. The simplest of these consists of increasing the resolution in Δx or Δt, and increasing the average number of particles per cell, $\bar{N}_{\text{ppc}}$. We can also increase the initial perturbation on the equilibrium particle positions such that the magnitude of the 2π-mode we are trying to induce is much larger than the noise due to the fluctuations varying on the grid scale. We have performed a large number of parameter scans over Δx, Δt, and $\bar{N}_{\text{ppc}}$ to determine the regimes in which our code is able to perform well. Although we have determined in § 4.2.4 that interpolation of the grid using higher-order shape functions leads to smoother interpretation of the inertial force term in Eqn. (4.13b), we find that in this case the *grid* interpolation order has little effect on the overall noise in the system. Rather, the interpolation and weighting method used on the *fields* appears to be the most important factor for preserving a coherent electrostatic oscillation. This is to be expected, as the noise within the system lies entirely in the $\rho^\xi \to \Phi \to E^x$ process. There is no noise induced into the system by interpolation of the grid quantities to the particle positions.

Fig. (4.6a) shows a comparison of the evolution of a cold electrostatic plasma wave generated by a uniform and non-uniform grid over a long timescale. It can easily be seen that the growth rate is practically zero in both cases, but that the noise levels are different in the two cases. Fig. (4.6b) is a zoomed snapshot of the long run, which shows that the plasma oscillations occur with a period of 6.29 for both uniform and nonuniform grids, which is consistent with our normalization such that $\omega_{pe} = 1$. Here we are using $N_\xi = 256$, $\bar{N}_{\text{ppc}} = 100$ and $\Delta t = 0.01$, with $\epsilon_{\text{grid}} = 0.15$





(b)

Figure 4.6: Comparison of cold plasma oscillation field energies for uniform and non-uniform grids for (a) a long run and (b) a zoomed section of the run at early time showing the oscillation period of 2π for both grids. Here we have used cubic-spline (S_3) particle shape functions with $N_\xi = 256$, $\bar{N}_{\text{ppc}} = 100$, $\Delta t = 0.01$, for both cases and $\epsilon_{\text{grid}} = 0.15$ for the nonuniform grid.

and a cubic spline particle shape. We have also performed several tests with higher values of $\bar{N}_{\text{ppc}}$, and the net effect is a longer run time before loss of the coherent oscillation with higher numbers of particles per cell, as is expected.

It should be noted here that while we have been forced to use very high resolution in both Δx and Δt, we are in fact simulating this problem on a grid in which the length of the largest cell is ≈ 35 times that of the smallest. For less nonuniform grids, much less resolution is required. In fact, for cases with a largest to smallest cell size ratio of ≈ 2 ($\epsilon_{\text{grid}} = 0.05$), we can run to time $\omega_{pe}t = 10^4$ without losing the coherent wave structure using the same parameters as we have been using for the uniform grid case, $N_\xi = 64$, $\bar{N}_{\text{ppc}} = 100$ and $\Delta t = 0.1$.

4.3.2 Cold Electron-Electron Two-Stream Instability

A system consisting of two counter-propagating cold beams of particles with velocities $\pm v_0$ and density $\frac{n_0}{2}$ per beam can give rise to what is known as the two-stream instability. A density perturbation on one beam causes bunching of the particles in the second beam. This bunching of particles produces electric fields which then cause bunching in the first beam. This leads to the characteristic bunching of the beams in phase-space and an exponential growth rate. In the case of two cold electron beams, it is permissable to model the ion species (of density n_0) as an immobile background, as the interaction of the beams happens on the electron timescale. Simple linear theory for a cold electron-electron two-stream reveals that a quartic dispersion relation with four independent solutions exists:

$$\omega = \pm\sqrt{k^2 v_0^2 + \frac{\omega_{pe}^2}{2} \pm \frac{\omega_{pe}}{2}\sqrt{8k^2 v_0^2 + \omega_{pe}^2}}. \tag{4.43}$$

Fig. (4.7) shows a comparison of the growth rate of both the uniform and non-

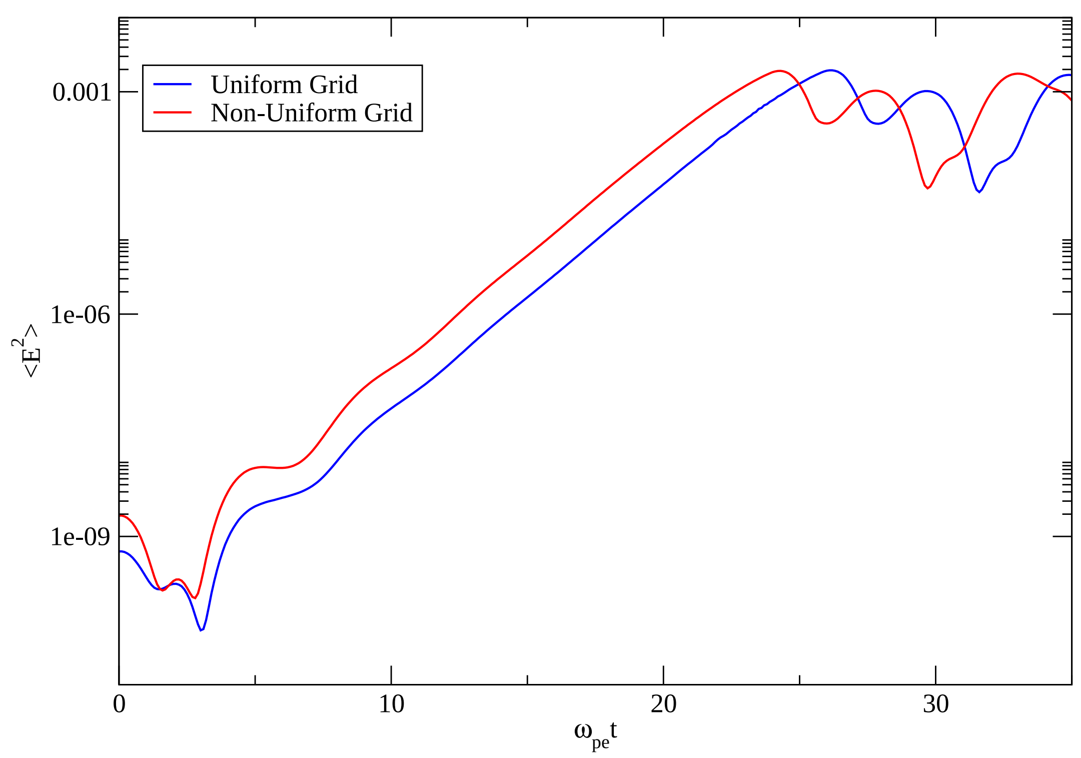


Figure 4.7: Comparison of cold electron-electron two-stream instability growth rates for uniform and non-uniform grids. Note the difference in field energies at $\omega_{pe}t = 0$ due to particle noise from the smallest cells.

uniform grid cases, showing again that, for a well-resolved case, there is no difference between the uniform grid solution and that of our method on a nonuniform grid with $\epsilon_{grid} = 0.15$. Here we have used $N_\xi = 128$, $\bar{N}_{\rm ppc} = 200$ (per beam), $v_0^x = \pm 0.1$ and $\Delta t = 0.01$ for both the uniform and nonuniform grid. With these initial values, we have $kv_0^x \approx 0.63 < \omega_{pe}$. The uniform grid gives a growth rate $\gamma_{\rm un} = 0.3516$, whereas the non-uniform grid case shown here gives $\gamma_{\rm non} = 0.3518$, which compare extremely well with the theory prediction in Eqn. (4.43) of $\gamma_{\rm theory} = 0.3515$. It is interesting to note that as the fidelity of our simulations decreases as we increase $\epsilon_{\rm grid}$, the initial noise in the system due to the nonuniform grid becomes higher and the period of linear growth becomes shorter (but retains the correct growth rate). The problem saturates at the same electrostatic energy regardless of the case studied. From our

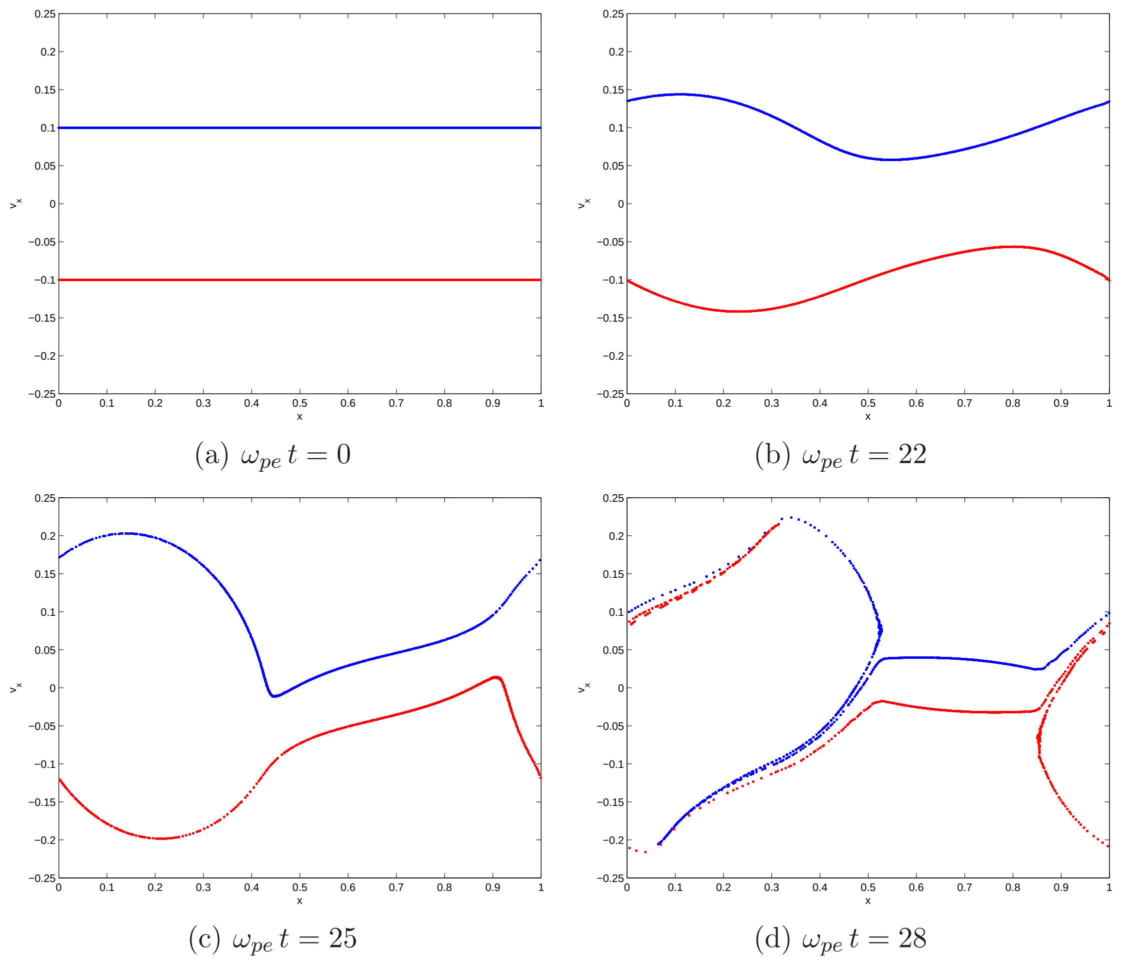

(a) $\omega_{pe}\, t = 0$ (b) $\omega_{pe}\, t = 22$ (c) $\omega_{pe}\, t = 25$ (d) $\omega_{pe}\, t = 28$

Figure 4.8: Phase-space evolution of the electron-electron two-stream instability.

studies, we conclude that a smaller initial perturbation on the beams simply leads to a time-delay in the formation of the instability, as the susequent evolution of the instability is the same for all cases, even in the nonlinear phase! I guess sometimes the laws of physics just work.

Fig. (4.8) gives the reader an example of the phase-space evolution of the two-stream instability on the nonuniform grid as in Fig. (4.7). In this example, we have used used the same parameters as before. In this particular case we have found that fidelity in the time step becomes the most important parameter, as when the particle

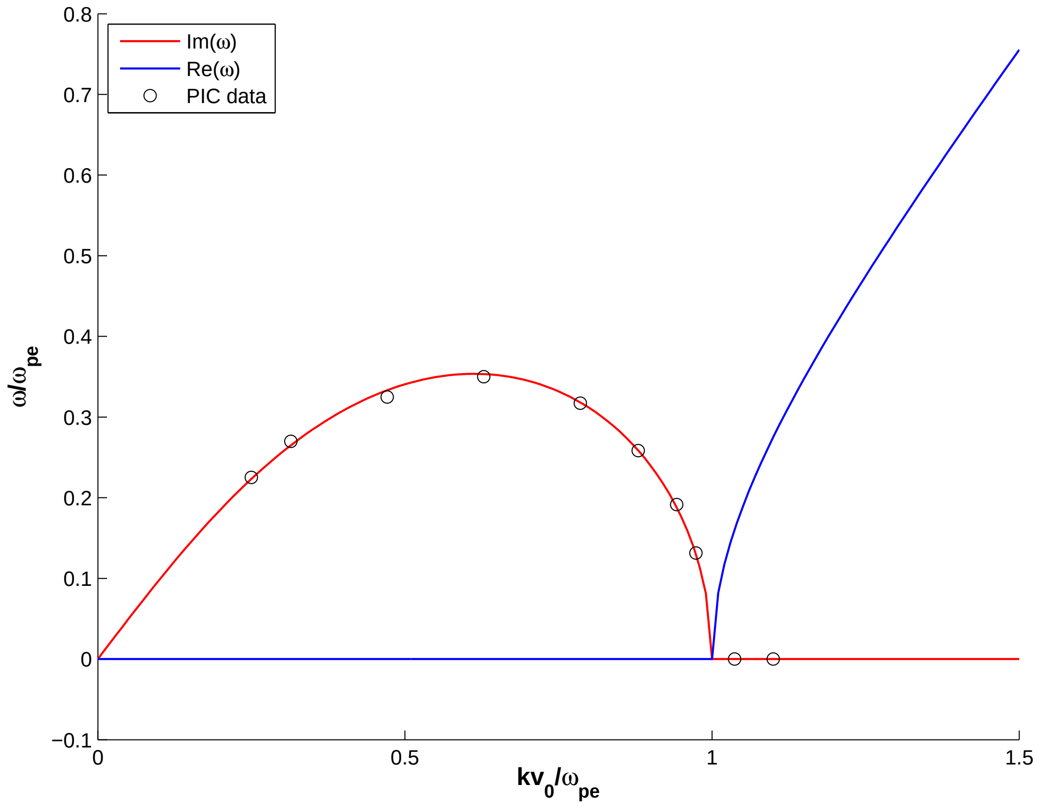


Figure 4.9: Plot of the electron-electron two-stream instability dispersion relation (Eqn. (4.43)) overlayed with data points obtained from our PIC code using $\epsilon_{grid} = 0.15$, $N_\xi = 128$, $\bar{N}_{\text{ppc}} = 200$ (per beam), and $\Delta t = 0.005$.

phase-space trajectories become distorted in time as in Fig. (4.8c), the velocities of the fastest particles can become 2-3 times higher than the initial beam velocity.

Finally, we have run the two-stream instability test for a large number of cases to test the linear dispersion relation given by Eqn. (4.43). Fig. (4.9) shows the comparison between the theoretical prediction (lines) and our data for several cases (circles). As can be seen, the agreement with theory is quite good for all cases.

v_{th}	ω_{PIC}	ω_{theory}	% error
0.0	1.00081	1.00000	0.08
0.005	1.00081	1.00148	0.07
0.01	1.00644	1.00590	0.05
0.015	1.01660	1.01324	0.33
0.02	1.02601	1.02341	0.25
0.025	1.02887	1.03635	0.72

Table 4.3: Table of PIC results for Langmuir waves compared with Eqn. (4.44) using $\epsilon_{\text{grid}} = 0.15$, $N_\xi = 128$, $\bar{N}_{\text{ppc}} = 200$, and $\Delta t = 0.01$.

4.3.3 Langmuir Waves

A Maxwellian thermal spread with temperature T_e is added to the electrons described in § 4.3.1 by choosing random velocities according to a Maxwellian distribution at each position. The resulting oscillation frequency should be related to the electron plasma frequency by the Bohm-Gross dispersion relation [49]:

$$\omega^2 = \omega_{pe}^2 + 3k^2 v_{th}^2. \tag{4.44}$$

Again the ions are assumed to be a uniform immobile background of density n_0. Eqn. (4.44) describes a purely real plasma wave, therefore adding some thermal velocity to the electrons should produce oscillations quite similar to those shown in Fig. (4.6), but with an increase in ω^2 proportional to square of the thermal velocity. However, further analysis reveals that there is an imaginary component associated with Eqn. (4.44) due to Landau damping (§ 4.3.4). We therefore limit our Langmuir wave comparisons for this section to low thermal velocities, at which the Landau damping is negligible.

Fig. (4.10) shows a comparison of the oscillation periods for several thermal velocities, using $\epsilon_{\text{grid}} = 0.15$, $N_\xi = 128$, and $\bar{N}_{\text{ppc}} = 200$ with $\Delta t = 0.01$. It is quite evident that as the thermal velocity is increased, the oscillation period decreases (or conversely, the oscillation frequency increases). For the largest case, $v_{th} = 0.025$, the

(

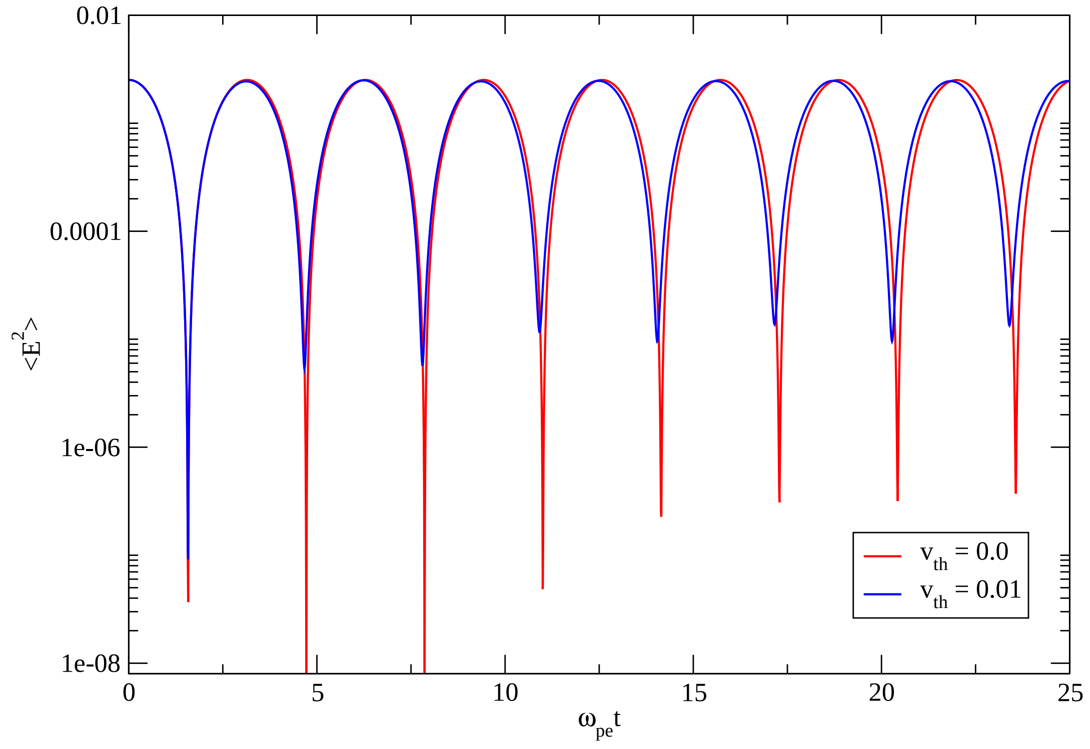


Figure 4.10: Comparison of Langmuir wave periods for different thermal velocities using a non-uniform grid with $\epsilon_{\text{grid}} = 0.15$, $N_\xi = 128$, $\bar{N}_{\text{ppc}} = 200$, and $\Delta t = 0.01$.

effects of Landau damping become noticeable (see next section). Table (4.3) shows a comparison of our results with those given by Eqn. (4.44). As can be seen, the real oscillation frequency results are quite accurate, even for the $v_{th} = 0.025$ case. Interestingly, from a computational standpoint, the addition of a thermal velocity adds "noise" to the system (since for no thermal velocity we applied a quiet start technique, thereby eliminating the noise from the initial conditions) and as such, the problem becomes easier to simulate with our nonuniform grid as the thermal velocity increases and dominates over the noise in the system due to the grid effects.

v_{th}	γ_{PIC}	γ_{theory}	% error
0.06	-0.0732	-0.0774	5.426
0.08	-0.1453	-0.1521	4.471
0.10	-0.1504	-0.1589	5.349
0.12	-0.1311	-0.1354	3.176

Table 4.4: Table of PIC results for Landau damped Langmuir waves compared with Eqn. (4.45) using $N_\xi = 256$, $\bar{N}_{\text{ppc}} = 1000$, and $\epsilon_{\text{grid}} = 0.075$.

4.3.4 Landau Damping

Landau damping [2] is a characteristic of collisionless plasmas in which waves are damped due to phase mixing in phase space. It leads to the energy exchange between the plasma waves, which travel at a phase velocity $v_{\text{ph}} = \frac{\omega}{k}$, and the plasma particles that travel at velocities very close to this phase velocity. Particles having velocities slightly lower than v_{ph} are accelerated by the wave electric field. Conversely, particles moving slightly faster than v_{ph} are decelerated by the wave electric field. The net effect is to remove energy from the wave and transfer it to the particles.

In a thermal collisionless plasma, the plasma particles are distributed according to a Maxwellian distribution function. As such, there are more slow electrons than fast ones, leading to a net surplus of particles taking energy from the wave than providing energy, and the wave is damped according to [49]:

$$\omega_i = -\sqrt{\frac{\pi}{8}}\frac{\omega_{pe}}{|k^3\lambda_{De}^3|}\exp\left[-\left(\frac{1}{2k^2\lambda_{De}^2}+\frac{3}{2}\right)\right]. \tag{4.45}$$

Table (4.4) shows a comparison of the PIC results with the theory presented in Eqn. (4.45). As can be seen, the PIC data consistently damps at a slightly lower rate than that predicted by theory, even for higher resolution runs with $N_\xi = 256$, $\bar{N}_{\text{ppc}} = 1000$, and $\epsilon_{\text{grid}} = 0.075$. However, since Landau damping is due to only a

(

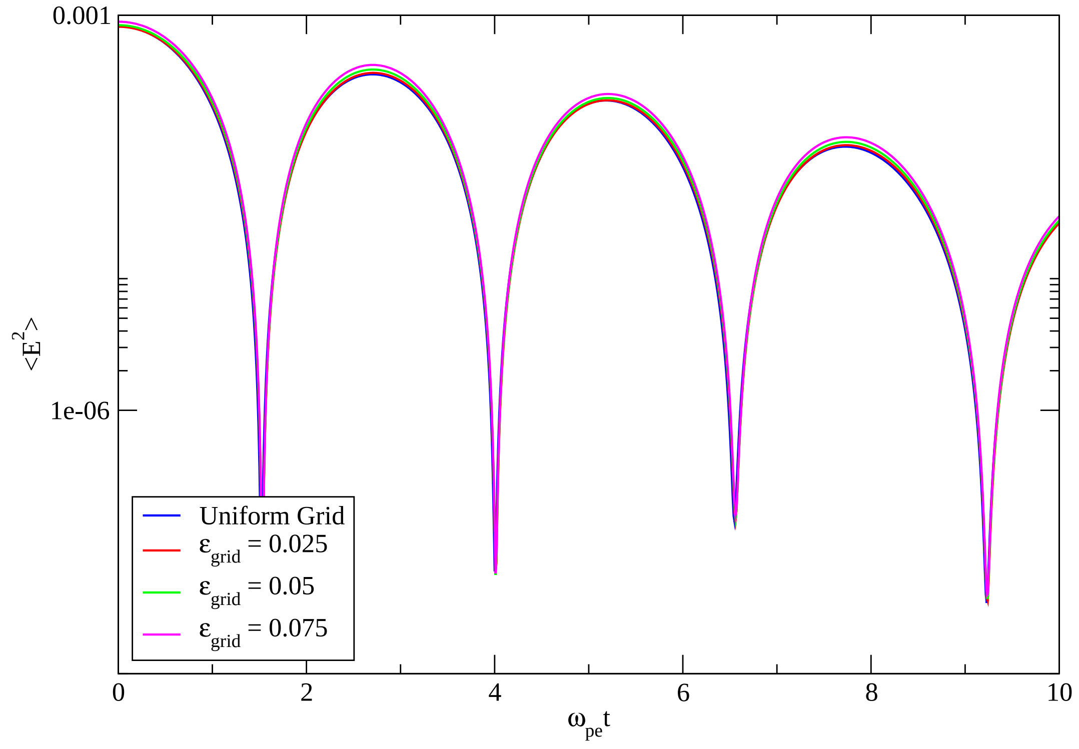


Figure 4.11: Comparison of Landau damping on uniform and nonuniform grids. Here $L = 1$, $v_{th} = 0.08$, and an initial perturbation of $\epsilon_{\text{pert}} = 0.05$ is used.

small set of resonant particles, even uniform grid codes have trouble reproducing Landau damping rates accurately.

As is shown in Fig. (4.11), the uniform and nonuniform grids give nearly identical damping rates for a case where $v_{th} = 0.08$ with an initial perturbation on the particle positions of $\epsilon_{\text{pert}} = 0.05$. It is impressive that even a highly nonuniform case of $\epsilon_{grid} = 0.075$ gives results which are very close to those of the uniform grid case. Here, the initial noise in the system due to the large grid nonuniformity is small compared to the noise due to the thermal motions of the particles.

Chapter 5

Development of a 2D Logical Grid PIC Code

5.1 2d Governing Equations

In two spatial dimensions, the logical grid particle equations of motion (Eqns. (3.35)) can be written as

$$\begin{aligned}\dot{\xi} &= \tfrac{1}{m}\left(g^{11}P_\xi + g^{12}P_\eta\right), \\ \dot{\eta} &= \tfrac{1}{m}\left(g^{12}P_\xi + g^{22}P_\eta\right),\end{aligned} \tag{5.1a}$$

and

$$\begin{aligned}\dot{P}_\xi &= \tfrac{-1}{2m}\left(P_\xi^2\tfrac{\partial g^{11}}{\partial \xi} + 2P_\xi P_\eta \tfrac{\partial g^{12}}{\partial \xi} + P_\eta^2 \tfrac{\partial g^{22}}{\partial \xi}\right) - \tfrac{\partial V(\vec{\xi}\,)}{\partial \xi}, \\ \dot{P}_\eta &= \tfrac{-1}{2m}\left(P_\xi^2\tfrac{\partial g^{11}}{\partial \eta} + 2P_\xi P_\eta \tfrac{\partial g^{12}}{\partial \eta} + P_\eta^2 \tfrac{\partial g^{22}}{\partial \eta}\right) - \tfrac{\partial V(\vec{\xi}\,)}{\partial \eta}.\end{aligned} \tag{5.1b}$$

Note that we have written Eqns. (5.1) in terms of the contravariant metric tensor, $g^{\mu\nu} = g^{\mu\nu}(\vec{x}\,)$, which is easily obtained from the covariant metric tensor, $g_{\mu\nu}(\vec{\xi}\,)$, obtained during the grid generation procedure outlined in § 3.1 using Eqn. (A.10).

The third momentum component, P_z, is generated at $t = 0$ and held as a constant as the simulation progresses. This term contributes to the simulation through its inclusion in the effective potential term, $V(\vec{\xi})$. In azimuthal symmetry, $V_{\mathrm{azi}}(\vec{\xi}) = \frac{p_z^2}{2m} + q\Phi(\vec{\xi})$, such that the ignorable-direction momentum does not contribute to the momentum update equations. However, for an axisymmetric problem, $V_{\mathrm{axi}}(\vec{\xi}) = \frac{p_\phi^2}{2mx(\vec{\xi})^2} + q\Phi(\vec{\xi})$, such that its derivative is

$$\begin{aligned} \frac{\partial V(\vec{\xi})}{\partial \xi} &= -\frac{j_{11}p_\phi^2}{mx^3} - qE_\xi \\ \frac{\partial V(\vec{\xi})}{\partial \eta} &= -\frac{j_{12}p_\phi^2}{mx^3} - qE_\eta. \end{aligned} \tag{5.2}$$

Thus, in the axisymmetric case we must also interpolate the j_{11} and j_{12} components of the Jacobi matrix and the x-coordinate of the particle's physical space position to the particle position on the logical grid.

In two dimensions the logical grid Poisson equation, Eqn. (3.67), takes the form

$$\frac{\partial}{\partial \xi}\left(D^{11}\frac{\partial \Phi}{\partial \xi} + D^{12}\frac{\partial \Phi}{\partial \eta}\right) + \frac{\partial}{\partial \eta}\left(D^{12}\frac{\partial \Phi}{\partial \xi} + D^{22}\frac{\partial \Phi}{\partial \eta}\right) = -f\rho^\xi, \tag{5.3}$$

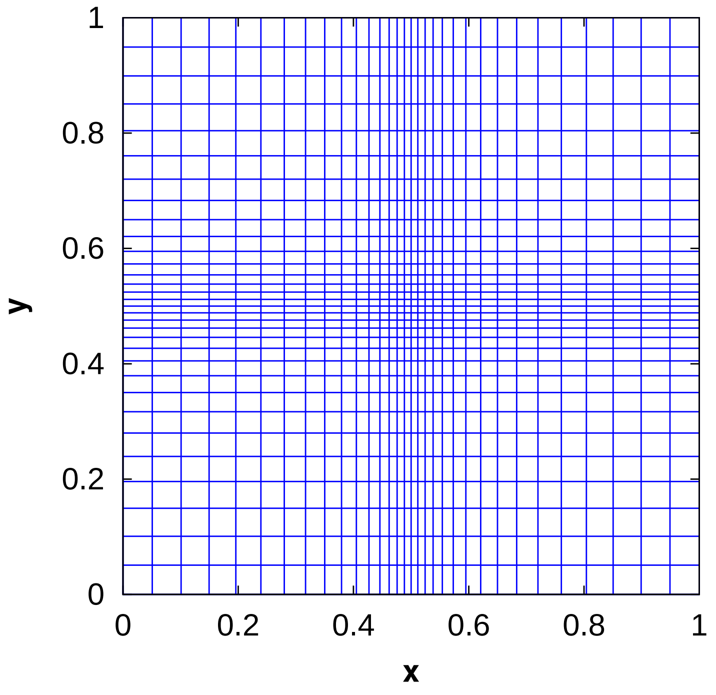
where we have defined

$$D^{\mu\nu} \equiv fJg^{\mu\nu}. \tag{5.4}$$

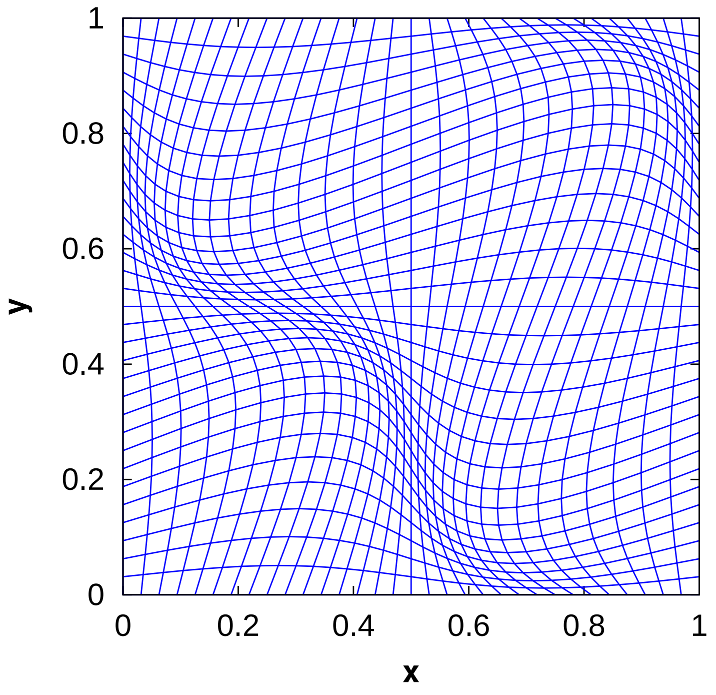
For our 2d code, we have a wide range of grid choices on which to perform our test cases. For the sake of brevity, we outline only two here. In addition to the annulus grid generated numerically in § 3.2, we define a doubly periodic, orthogonal grid using:

$$\begin{aligned} x &= x_{\mathrm{min}} + (x_{\mathrm{max}} - x_{\mathrm{min}})(\xi + \epsilon_{\mathrm{grid}} \sin 2\pi\xi) \\ y &= y_{\mathrm{min}} + (y_{\mathrm{max}} - y_{\mathrm{min}})(\eta + \epsilon_{\mathrm{grid}} \sin 2\pi\eta). \end{aligned} \tag{5.5a}$$

Here x_{min}, x_{max}, y_{min}, and y_{max} are constants to scale boundaries of the physical grid to form a rectangle of arbitrary size. We have also designed a doubly-periodic,



(a)



(b)

Figure 5.1: Grids used for validation of the 2d code given by Eqns. (5.5a) and (5.5b). For this figure, we have used $N_\xi = N_\eta = 32$ and $\epsilon_{\text{grid}} = 0.1$ so that the structure of the grid can be more clearly seen.

nonorthogonal grid given by

$$\begin{aligned} x &= x_{\min} + (x_{\max} - x_{\min})(\xi + \epsilon_{\text{grid}} \sin 2\pi\xi \sin 2\pi\eta) \\ y &= y_{\min} + (y_{\max} - y_{\min})(\eta + \epsilon_{\text{grid}} \sin 2\pi\xi \sin 2\pi\eta). \end{aligned} \tag{5.5b}$$

to test the effects of non-zero cross terms (g^{12}) in our code. For simplicity, we have constrained the grid nonuniformity parameter ϵ_{grid} in Eqns. (5.5a) and (5.5b) to be the same in each dimension. Notice that both Eqns. (5.5a) and (5.5b) require that $\epsilon_{\text{grid}} < \frac{1}{2\pi}$ so that the grid does not fold.

5.1.1 Conservative Discretization of Poisson Equation

Eqn. (5.3) is comprised of a set of co-directed derivatives proportional to the orthogonal metric tensor components (D^{11} and D^{22}) and a set of cross-directed derivatives for the nonorthogonal metric tensor components (D^{12}), i.e.

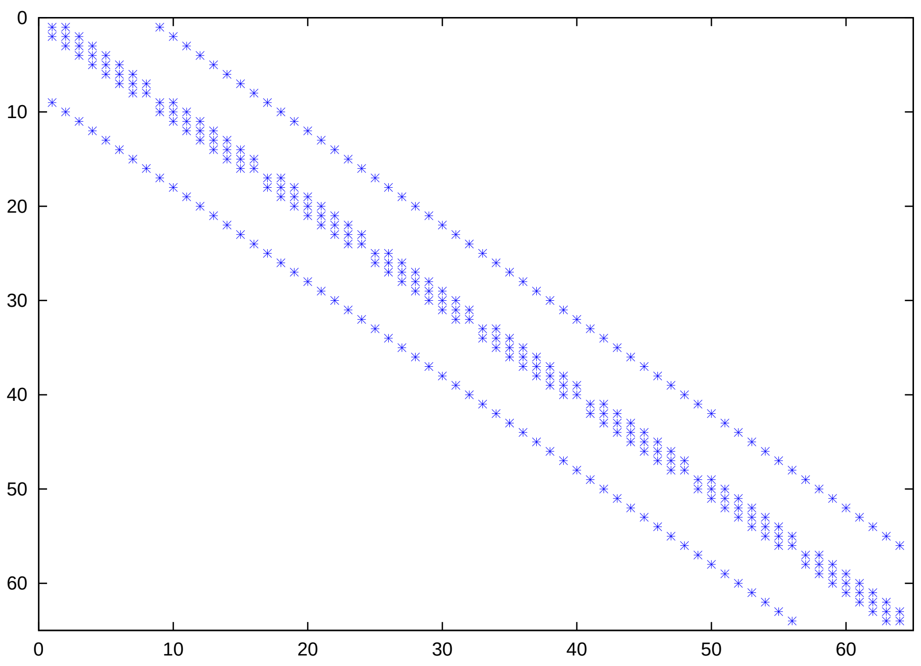
$$\nabla^2\Phi = (\nabla^2\Phi)^{\text{co}} + (\nabla^2\Phi)^{\text{cross}}. \tag{5.6}$$

We can discretize the co-directed terms of Eqn. (5.3),

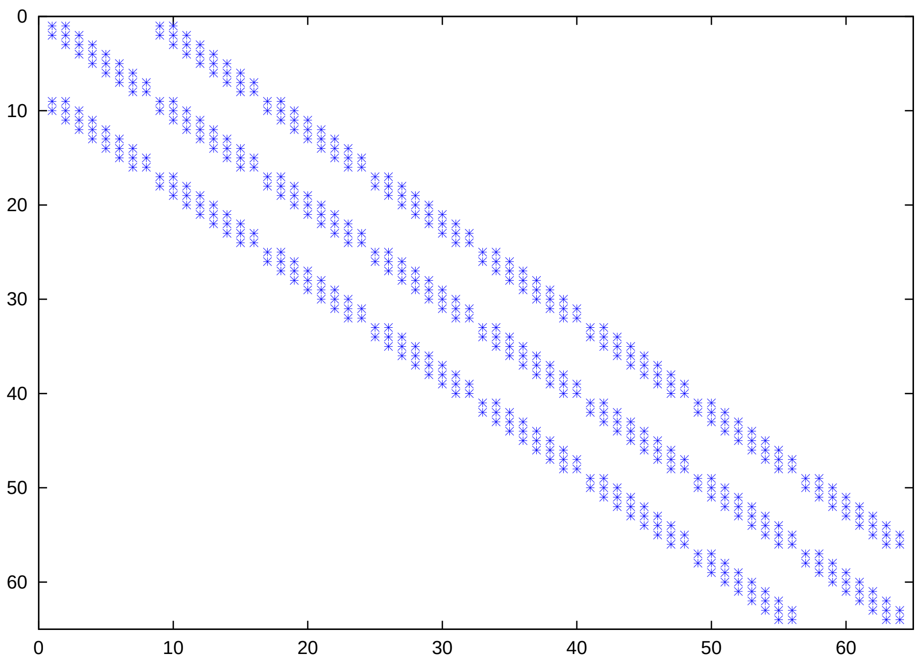
$$(\nabla^2\Phi)^{\text{co}} = \frac{\partial}{\partial\xi}\left(D^{11}\frac{\partial\Phi}{\partial\xi}\right) + \frac{\partial}{\partial\eta}\left(D^{22}\frac{\partial\Phi}{\partial\eta}\right), \tag{5.7}$$

using an approach similar to the method used in § 4.2.1. We use two-point centered differences to first discretize the interior derivatives, then apply another centered difference on the outer derivative term such that Eqn. (5.7) becomes

$$\begin{aligned} &(\nabla^2\Phi)^{\text{co}}_{i+\frac{1}{2},j+\frac{1}{2}} = \\ &\frac{D^{11}_{i+1,j+\frac{1}{2}}\left(\Phi_{i+\frac{3}{2},j+\frac{1}{2}} - \Phi_{i+\frac{1}{2},j+\frac{1}{2}}\right) - D^{11}_{i,j+\frac{1}{2}}\left(\Phi_{i+\frac{1}{2},j+\frac{1}{2}} - \Phi_{i-\frac{1}{2},j+\frac{1}{2}}\right)}{\Delta\xi^2} + \\ &\qquad \frac{D^{22}_{i+\frac{1}{2},j+1}\left(\Phi_{i+\frac{1}{2},j+\frac{3}{2}} - \Phi_{i+\frac{1}{2},j+\frac{1}{2}}\right) - D^{22}_{i+\frac{1}{2},j}\left(\Phi_{i+\frac{1}{2},j+\frac{1}{2}} - \Phi_{i+\frac{1}{2},j-\frac{1}{2}}\right)}{\Delta\eta^2}. \end{aligned} \tag{5.8}$$



(a)



(b)

Figure 5.2: Structure of the Laplacian operator for (a) orthogonal and (b) nonorthogonal grids. Here the markers represent nonzero terms in the Laplacian matrix for Dirichlet field boundary conditions.

Since the D^{11} and D^{22} terms are naturally defined at cell centers, they must be averaged to the correct cell face locations. By our discretization, the D^{11} terms must be defined on the vertical faces of grid cells, whereas the D^{22} terms are required to be defined along the horizontal faces. For a uniform grid with $x, y \in [0:1]$, the D^{11} and D^{22} terms become unity (assuming the geometry factor, $f = 1$) and Eqn. (5.8) reduces to the standard 5-point discretization used in Cartesian PIC codes. The structure of the Laplacian operator matrix resulting from the co-directed terms is shown in Fig. 5.2(a), where we have marked nonzero terms with an x. Here we have used Dirchlet boundary conditions during the matrix formation.

The discretization of the cross-derivative terms

$$(\nabla^2\Phi)^{\text{cross}} = \frac{\partial}{\partial\xi}\left(D^{12}\frac{\partial\Phi}{\partial\eta}\right) + \frac{\partial}{\partial\eta}\left(D^{12}\frac{\partial\Phi}{\partial\xi}\right) \tag{5.9}$$

requires slightly more care. The correct method for discretizing Eqn. (5.9) is to apply a 4-point centered difference on each derivative. For example,

$$\left(\frac{\partial\phi}{\partial\xi}\right)_{i,j} = \frac{\Phi_{i+\frac{1}{2},j+\frac{1}{2}} - \Phi_{i-\frac{1}{2},j+\frac{1}{2}} + \Phi_{i+\frac{1}{2},j-\frac{1}{2}} - \Phi_{i-\frac{1}{2},j-\frac{1}{2}}}{2\Delta\xi} \tag{5.10}$$

We also calculate $\left(\frac{\partial\phi}{\partial\xi}\right)_{i+1,j}, \left(\frac{\partial\phi}{\partial\xi}\right)_{i+1,j+1}$, and $\left(\frac{\partial\phi}{\partial\xi}\right)_{i,j+1}$ such that we can then take the η-derivatives:

$$\begin{aligned}\frac{\partial}{\partial\eta}\left(D^{12}\frac{\partial\Phi}{\partial\xi}\right) &= \frac{1}{2\Delta\eta}\left[D^{12}_{i+1,j+1}\left(\frac{\partial\phi}{\partial\xi}\right)_{i+1,j+1} - D^{12}_{i+1,j}\left(\frac{\partial\phi}{\partial\xi}\right)_{i+1,j} + \right.\\ &\qquad \left. D^{12}_{i,j+1}\left(\frac{\partial\phi}{\partial\xi}\right)_{i,j+1} - D^{12}_{i,j}\left(\frac{\partial\phi}{\partial\xi}\right)_{i,j}\right].\end{aligned}$$

Here, the D^{12} terms are defined on the vertices. Combining the contributions from both terms in Eqn. (5.9) and simplifying, we have

$$\begin{aligned}(\nabla^2\Phi)^{\text{cross}} &= \frac{1}{4\,\Delta\xi\,\Delta\eta}\left[D^{12}_{i+1,j+1}\left(\Phi_{i+\frac{3}{2},j+\frac{3}{2}} - \Phi_{i+\frac{1}{2},j+\frac{1}{2}}\right) - \right.\\ &\quad D^{12}_{i,j}\left(\Phi_{i+\frac{1}{2},j+\frac{1}{2}} - \Phi_{i-\frac{1}{2},j-\frac{1}{2}}\right) + D^{12}_{i+1,j}\left(\Phi_{i+\frac{1}{2},j+\frac{1}{2}} - \Phi_{i+\frac{3}{2},j-\frac{1}{2}}\right) + \\ &\quad \left. D^{12}_{i,j+1}\left(\Phi_{i-\frac{1}{2},j+\frac{3}{2}} - \Phi_{i+\frac{1}{2},j+\frac{1}{2}}\right)\right].\end{aligned} \tag{5.11}$$

We can then construct the final, discretized form of Eqn. (5.3) by combining Eqns. (5.8) and (5.11):

$$\begin{aligned} f\rho^{\xi} \; = \; & \frac{D^{11}_{i+1,j+\frac{1}{2}}\left(\Phi_{i+\frac{3}{2},j+\frac{1}{2}} - \Phi_{i+\frac{1}{2},j+\frac{1}{2}}\right) - D^{11}_{i,j+\frac{1}{2}}\left(\Phi_{i+\frac{1}{2},j+\frac{1}{2}} - \Phi_{i-\frac{1}{2},j+\frac{1}{2}}\right)}{\Delta\xi^2} + \\ & \frac{D^{22}_{i+\frac{1}{2},j+1}\left(\Phi_{i+\frac{1}{2},j+\frac{3}{2}} - \Phi_{i+\frac{1}{2},j+\frac{1}{2}}\right) - D^{22}_{i+\frac{1}{2},j}\left(\Phi_{i+\frac{1}{2},j+\frac{1}{2}} - \Phi_{i+\frac{1}{2},j-\frac{1}{2}}\right)}{\Delta\eta^2} + \\ & \frac{1}{4\,\Delta\xi\,\Delta\eta}\left[D^{12}_{i+1,j+1}\left(\Phi_{i+\frac{3}{2},j+\frac{3}{2}} - \Phi_{i+\frac{1}{2},j+\frac{1}{2}}\right) - D^{12}_{i,j}\left(\Phi_{i+\frac{1}{2},j+\frac{1}{2}} - \Phi_{i-\frac{1}{2},j-\frac{1}{2}}\right) + \right. \\ & \left. D^{12}_{i+1,j}\left(\Phi_{i+\frac{1}{2},j+\frac{1}{2}} - \Phi_{i+\frac{3}{2},j-\frac{1}{2}}\right) + D^{12}_{i,j+1}\left(\Phi_{i-\frac{1}{2},j+\frac{3}{2}} - \Phi_{i+\frac{1}{2},j+\frac{1}{2}}\right)\right]. \end{aligned} \quad (5.12)$$

The structure of the Laplacian operator matrix generated by this full 9-point stencil applied to the nonorthogonal grid (Eqn. (5.5b)) is shown in Fig. 5.2(b). We have again assumed Dirichlet boundary conditions.

We note here that this same discretization can be obtained by the minimization of a variational principle chosen for Eqn. (5.3) [50]. The variational principle approach guarantees that, for properly applied boundary conditions, the matrix formed by the application of the ∇^2 operator is symmetric. Since we have obtained the same result, we know our matrix will also be symmetric. This property is important since we have chosen to use a CG field solver; for while CG requires a symmetric matrix to converge to the *correct* solution, it does not in fact require a symmetric matrix to *converge* in general!

5.1.2 Validation of 2d Poisson Solver using MMMS

The MMS technique is again utilized to validate our 2d Poisson solver. Since we are capable of simulating a wide range of physical domains with our 2d code, we have decided to run MMS tests appropriate for both a unitary, square grid, as well as for the case in which an annular grid is numerically generated using the techniques of

N_ξ	$\epsilon_{\text{grid}} = 0$	$\epsilon_{\text{grid}} = 0.025$	$\epsilon_{\text{grid}} = 0.05$	$\epsilon_{\text{grid}} = 0.075$	$\epsilon_{\text{grid}} = 0.15$
16	1.72090E-3	8.29163E-4	6.88949E-4	2.42544E-3	9.06392E-3
32	4.14875E-4	1.97193E-4	1.69813E-4	5.95391E-4	2.21714E-3
64	1.02009E-4	4.83358E-5	4.20043E-5	1.47031E-4	5.47060E-4
128	2.52982E-5	1.19786E-5	1.04332E-5	3.65026E-5	1.35787E-4
256	6.29958E-6	2.98229E-6	2.59901E-6	9.09201E-6	3.38199E-5

Table 5.1: L_2-norm error between computational and analytic MMS potentials for different grid resolutions and grid nonuniformities (ϵ_{grid}) for a 2d orthogonal grid given by Eqn. (5.5a). For these tests, we have chosen $N_\xi = N_\eta$.

§ 3.1. While we have run several different MMS cases, we present only one for each geometry here.

Unitary Square Grid

For the square grid, we have chosen an MMS potential given by

$$\phi_{\text{MMS}} = \sin(2\pi x)\sin(2\pi y), \tag{5.13}$$

such that, assuming Cartesian geometry, the MMS charge density is

$$\rho_{\text{MMS}} = 8\pi^2 \sin(2\pi x)\sin(2\pi y). \tag{5.14}$$

For this particular choice of ϕ_{MMS}, we test our Poisson solver with periodic boundary conditions on all boundaries. We have chosen to display this particular MMS test because we use doubly-periodic field boundary conditions for the PIC benchmarking tests presented in § 5.3.

As is seen in Table 5.1, the L_2-norm of the error in $(\phi - \phi_{\text{MMS}})$ as given by Eqn. (4.30) scales with 2^{nd}-order accuracy as expected, even for the $\epsilon_{\text{grid}} = 0.15$ case, in which the ratio of the *area* of the largest to the smallest cells is $\frac{J_{max}}{J_{min}} = \left(\frac{1+2\pi\epsilon_{\text{grid}}}{1-2\pi\epsilon_{\text{grid}}}\right)^2 \approx 1140!$

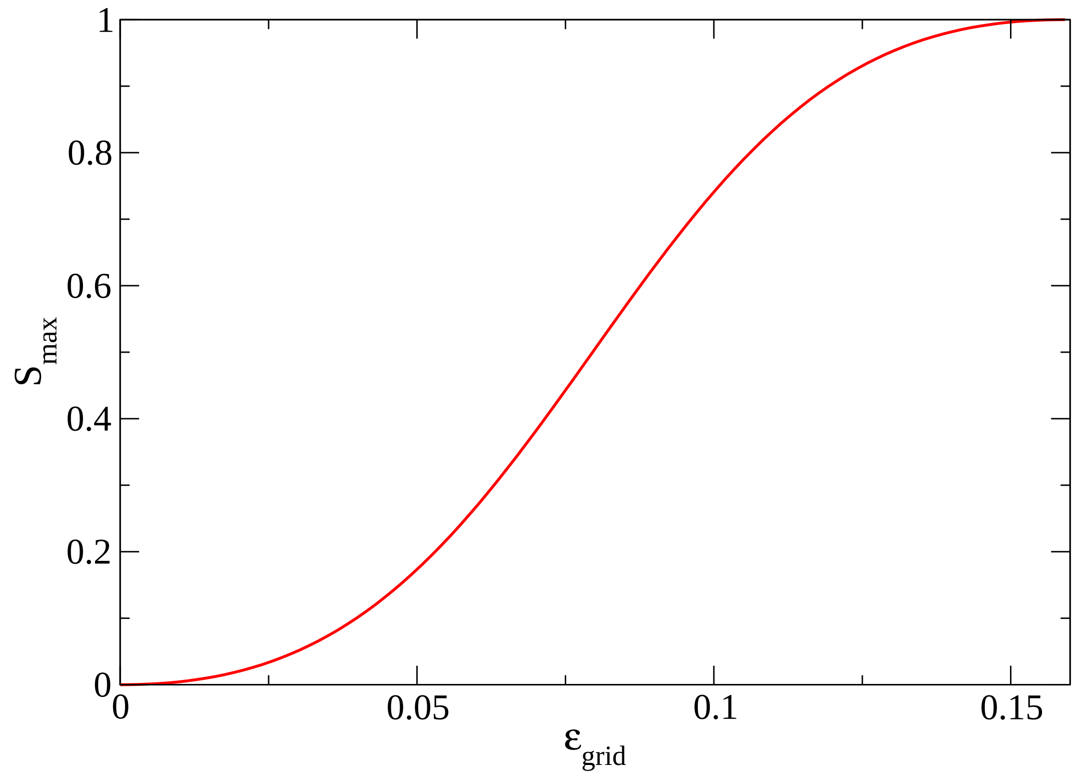


Figure 5.3: Graph of the maximum skewness parameter S as defined in Eqn. (3.16) as a function of ϵ_{grid} for the grid given by Eqn. (5.5b).

Unique to the nonorthogonal grid, we must now also consider the amount of "skewness" of the grid as a major factor in the difficulty associated with solving the curvilinear Poisson equation (Eqn. (5.3)). For these tests, we have used the maximum skewness parameter, S_{max}, as defined in Eqns. (3.16) and (3.17) to characterize the nonorthogonality of the grid defined by Eqn. (5.5b) for each particular value of ϵ_{grid} used.

Fig. 5.3 shows S_{max} as a function of ϵ_{grid} for the grid given by Eqn. (5.5b). For $\epsilon_{\text{grid}} \geq 0.125$, S_{max} is very nearly unity, and thus the grid is almost singular. As noted in § 5.1, this grid folds at $\epsilon_{\text{grid}} = \frac{1}{2\pi} \approx 0.16$. By the definition of the curvilinear

N_ξ	$\epsilon_{\text{grid}} = 0.025$	$\epsilon_{\text{grid}} = 0.05$	$\epsilon_{\text{grid}} = 0.075$	$\epsilon_{\text{grid}} = 0.15$
16	2.08019E-3	3.27156E-3	5.57519E-3	8.99093E-3
32	5.07340E-4	8.20596E-4	1.44430E-3	2.34622E-3
64	1.25103E-4	2.03916E-4	3.62134E-4	5.90944E-4
128	3.10476E-5	5.07074E-5	9.02599E-5	1.47505E-4
256	7.73260E-6	1.26353E-5	2.25042E-5	3.67914E-5

Table 5.2: L_2-norm error between computational and analytic MMS potentials for different grid resolutions and grid nonuniformities (ϵ_{grid}) for a 2d nonorthogonal grid given by Eqn. (5.5b). For these tests, we have chosen $N_\xi = N_\eta$.

Poisson equation (Eqn. (5.3)), we see that as $S_{\max} \to 1$, roundoff errors become large and the field solver will not easily converge, if at all.

We have performed our MMS tests on the nonorthogonal grid using the same MMS potential as was used for the orthogonal square grid. Table 5.2 shows that the L_2-norm of the error scales with 2^{nd}-order accuracy in Δx as expected. We note here that the ratio of maximum cell area to minimum for the grid defined in Eqn. (5.5b) scales according to $\frac{J_{max}}{J_{min}} = \frac{1+2\pi\epsilon_{\text{grid}}}{1-2\pi\epsilon_{\text{grid}}}$, meaning that the ratio of the largest cell area to the smallest is much smaller than that of the orthogonal, nonuniform grid case, but the skewness of the grid more than accounts for the difference.

Annular Grid

We have also designed an MMS test to validate our Poisson solver for the annular grid generated in § 3.2. Since we are able to simulate half the annulus and apply symmetry conditions, we must choose our boundary conditions carefully. We wish to be able to apply either Dirichlet or Neumann boundary conditions along the inner and outer boundaries $r = r_1, r_2$, and to satisfy the symmetry of the problem we apply homogenous Neumann boundary conditions along $\theta = 0, \pi$. We have therefore constructed the following potential:

$$\Phi\left(r,\theta\right) = 1 - r^3 + \left(r - r_1\right)\left(r_2 - r\right)\cos\left(\theta\right), \tag{5.15}$$

such that the boundary conditions become

$$\begin{aligned}
\Phi\left(r_1\right) &= 1 - r_1^3 && (5.16)\\
\Phi\left(r_2\right) &= 1 - r_2^3 && (5.17)\\
\frac{\partial\Phi}{\partial\theta}\Big|_{\theta=0} &= -\left(r - r_1\right)\left(r_2 - r\right)\sin\left(0\right) = 0 && (5.18)\\
\frac{\partial\Phi}{\partial\theta}\Big|_{\theta=\pi} &= -\left(r - r_1\right)\left(r_2 - r\right)\sin\left(\pi\right) = 0. && (5.19)
\end{aligned}$$

Here we have used polar coordinates to express Eqn. (5.15) in the physical space, as it more naturally aligns with the annular grid case than the Cartesian notation we have been using up until this point.

As our Poisson solver is designed solve multiple geometries in the physical space, we have decided apply the MMS test to two cases: an azimuthally symmetric system and an axisymmetric system. In the context of dusty plasmas, the azimuthally symmetric case can be thought of as a system in cylindrical $\left(\hat{r}_c, \hat{\theta}, \hat{z}\right)$ coordinates where the $\hat{z}$-direction is ignored, creating an infinite cylindrical dust particle of radius r_1. The physical Laplacian in this geometry is given by

$$\nabla^2\Phi\left(r_c,\theta\right) = \frac{1}{r_c}\frac{\partial}{\partial r_c}\left(r_c\frac{\partial\Phi}{\partial r_c}\right) + \frac{1}{r_c^2}\frac{\partial^2\Phi}{\partial\theta^2}. \tag{5.20}$$

Inserting Eq. (5.15) into Eq. (5.20) gives

$$\rho\left(r_c,\theta\right) = 9\,r_c + \left(3 - \frac{r_1 r_2}{r_c^2}\right)\cos\left(\theta\right), \tag{5.21}$$

which is then multiplied by the Jacobian and inserted into the Poisson solver with the proper boundary conditions.

Likewise, the axisymmetric system can be thought of as being in spherical $\left(\hat{r}_s, \hat{\theta}, \hat{\phi}\right)$ coordinates in the physical space, where the $\hat{\phi}$ direction is ignored. Thus, an outer

(a) Cylindrical Laplacian

$\boldsymbol{N}$	$\boldsymbol{R=5}$	$\boldsymbol{R=10}$	$\boldsymbol{R=20}$
16	4.86684E-3	8.78720E-3	1.35298E-2
32	1.18612E-3	2.15682E-3	3.35229E-3
64	2.92357E-4	5.32567E-4	8.29718E-4
128	7.25471E-5	1.32213E-4	2.06105E-4
256	1.80677E-5	3.29310E-5	5.13433E-5

(b) Spherical Laplacian

$\boldsymbol{N}$	$\boldsymbol{R=5}$	$\boldsymbol{R=10}$	$\boldsymbol{R=20}$
16	3.84324E-3	6.41377E-3	9.23903E-003
32	9.43887E-4	1.60080E-3	2.35531E-003
64	2.33099E-4	3.96923E-4	5.87121E-004
128	5.78702E-5	9.86410E-5	1.46102E-004
256	1.44142E-5	2.45755E-5	3.64121E-005

Table 5.3: L_2-norm error between computational and analytic MMS potentials for different grid resolutions and ratios $R \equiv \frac{r_2}{r_1}$ for the annular grids generated in § 3.1 for both Cartesian (a) and cylindrical (b) solvers. For these tests, we have chosen $N_r = N_\theta$

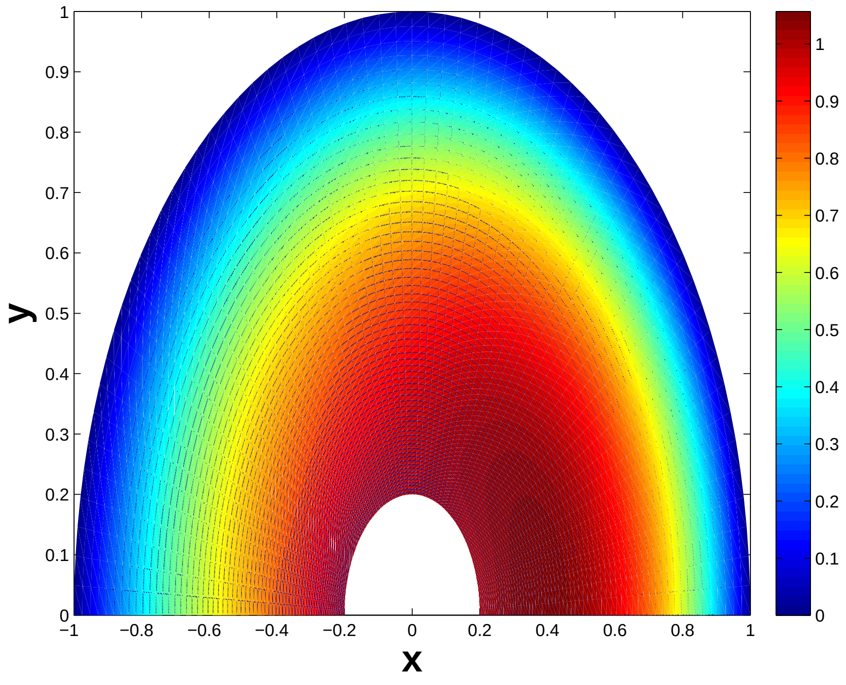
sphere of radius r_2 is created in which an inner sphere of radius r_1 has been removed. The corresponding physical space Laplacian for this geometry is given by

$$\nabla^2\Phi\left(r_s,\theta\right) = \frac{1}{r_s^2}\frac{\partial}{\partial r_s}\left(r_s^2\frac{\partial\Phi}{\partial r_s}\right) + \frac{1}{r_s^2\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\left(\theta\right)\frac{\partial\Phi}{\partial\theta}\right). \tag{5.22}$$

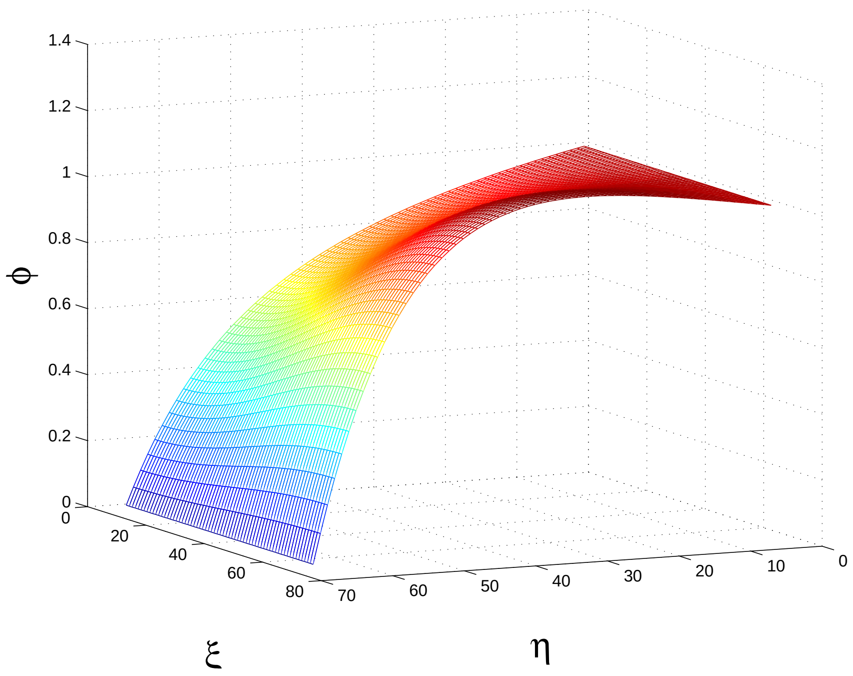
Inserting Eq. (5.15) into Eq. (5.22) gives

$$\rho\left(r_s,\theta\right) = 12\,r_s + \left(4 - \frac{2r_1r_2}{r_s^2}\right)\cos\left(\theta\right). \tag{5.23}$$

The L_2-norm of the error for both cylindrical and spherical solvers is shown in Table (5.3). Here $R = \frac{r_2}{r_1}$ is the ratio of the outer boundary to the inner boundary and $\boldsymbol{N}$ is the number of uniformly spaced grid points in logical domain. Fig. (5.4) displays the chosen MMS potential on both the physical and logical domains as obtained by the logical grid Poisson solver. Notice that the different values of ρ inserted in the solver for the different coordinate systems should (and do) return the same potential on the grid.



(a)



(b)

Figure 5.4: Two-dimensional MMS potential plots as obtained computationally using Eqns. (5.21) and (5.23) in physical (a) and logical (b) space.

5.1.3 Electric Fields

In § 4.2.1, we determined that the more accurate method of obtaining the logical electric fields at the particle position was by interpolating the physical electric fields and Jacobi matrix components to the particle position, then multiplying them to calculate the logical fields at the particle. We follow the same approach in 2d, but in order to extrapolate the electric fields onto ghost cells (required for interpolating the fields to the particles), we must first calculate the logical electric fields on the grid. In this way, we are able to determine the normal and tangential components of the electric fields along arbitrary grid boundaries.

The logical electric fields are calculated using

$$E^{\xi}_{i,j} = -\frac{\Phi_{i+\frac{1}{2},j+\frac{1}{2}} - \Phi_{i-\frac{1}{2},j+\frac{1}{2}} + \Phi_{i+\frac{1}{2},j-\frac{1}{2}} - \Phi_{i-\frac{1}{2},j-\frac{1}{2}}}{2\Delta\xi} \tag{5.24a}$$

and

$$E^{\eta}_{i,j} = -\frac{\Phi_{i+\frac{1}{2},j+\frac{1}{2}} - \Phi_{i+\frac{1}{2},j-\frac{1}{2}} + \Phi_{i-\frac{1}{2},j+\frac{1}{2}} - \Phi_{i-\frac{1}{2},j-\frac{1}{2}}}{2\Delta\eta}. \tag{5.24b}$$

We then apply the proper extrapolation techniques such that the overall 2^{nd} order accuracy of the system is upheld and calculate the physical electric fields on the vertices using

$$E^{x}_{i,j} = \frac{1}{J^{v}_{i,j}}\left(j^{v}_{22,i,j}E^{\xi}_{i,j} - j^{v}_{21,i,j}E^{\eta}_{i,j}\right) \tag{5.25a}$$

and

$$E^{y}_{i,j} = \frac{1}{J^{v}_{i,j}}\left(j^{v}_{11,i,j}E^{\eta}_{i,j} - j^{v}_{12,i,j}E^{\xi}_{i,j}\right), \tag{5.25b}$$

where we have transformed from the inverse Jacobian matrix, $k^{\mu\nu}$ to the Jacobian matrix. Here, the superscript v on the components of the Jacobi matrix and its determinant signify that they are calculated on the vertices using a four-point average from their natural cell-centered locations.

Logical Electric Fields at a Particle

To test the self-forces on the particle in 2d, we have set up a system in which a single particle is at rest on a doubly-periodic grid with a neutralizing background such that we can apply the 2d extension of the method developed for 1d in § 4.2.1 by interpolating both of the physical electric fields and required grid derivatives to the grid positions and retroactively multiplying them to obtain the logical fields at the particle position. As shown in Appendix **??**, the method utilized in the 1d code will only work for 1d, and we expect to have some small self-force in 2d. Below we attempt to quantify these forces.

In 2d, the logical electric fields are obtained from the physical fields using

$$\frac{\partial \Phi}{\partial \xi^{\mu}} = \frac{\partial x^{\nu}}{\partial \xi^{\mu}} \frac{\partial \Phi}{\partial x^{\nu}} \tag{5.26}$$

such that

$$\begin{aligned} E^{\xi} &= j_{11}\, E^{x} + j_{21}\, E^{y} \\ E^{\eta} &= j_{12}\, E^{x} + j_{22}\, E^{y}. \end{aligned} \tag{5.27}$$

We have run a number of tests to determine the fields at the particle. As a check, we first checked the fields on a fully uniform grid. The electric fields at the particle are zero in this case. However, as soon as we allow one of the grid dimensions to become nonuniform, the fields at the particle in the nonuniform dimension are no longer zero, even for $\epsilon_{\text{grid}} = 10^{-4}$. We find that the fields at the particle position scale with 2^{nd}-order accuracy in Δx, and the magnitude of the field at the particle in the nonuniform dimension is dependent upon the magnitude of the grid nonuniformity parameter, ϵ_{grid}. For $\epsilon_{\text{grid}} = 0.15$ in x and holding y to be uniform ($\epsilon_{\text{grid}} = 0$), $E^{\xi}(\xi_p, \eta_p)$ is $\approx 10^{-5}$, whereas $E^{\eta}(\xi_p, \eta_p)$ is zero to machine precision. For $\epsilon_{\text{grid}} = 0.1$, $E^{\xi}(\xi_p, \eta_p)$ is $\approx 10^{-7}$. For both of these tests, we have used $N_{\xi} = N_{\eta} = 32$.

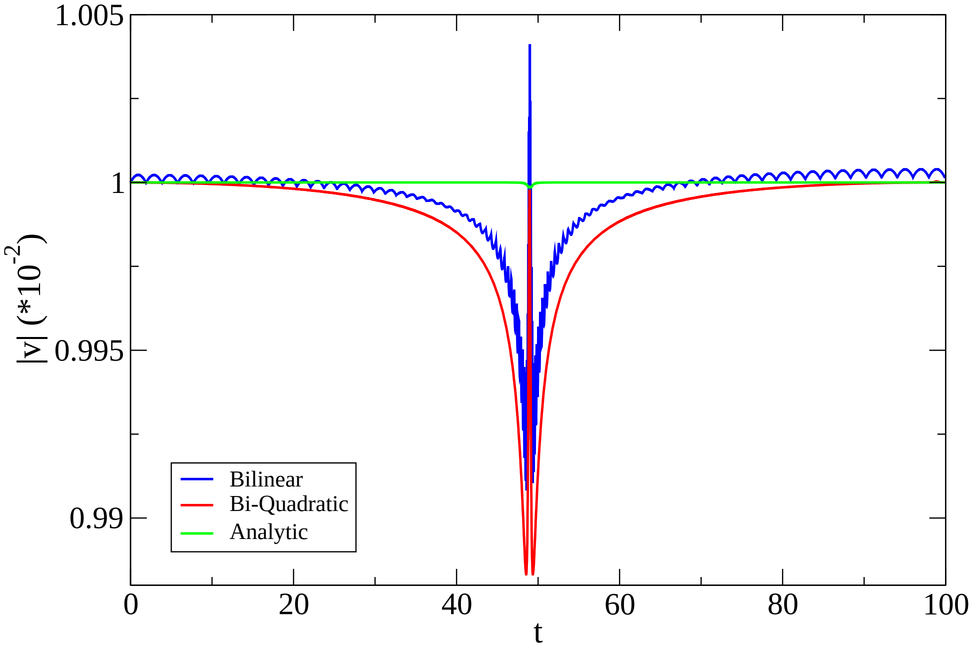
We have also tried several other methods of calculating the fields at the particle position, including direct interpolation of the logical fields and calculation of the fields on a colocated grid. All methods tested scale with 2^{nd} order accuracy in Δx, but the method detailed above (the extension of the method of § 4.2.1 to 2d) consistently provides the smallest fields at the particle position. However, as will be seen in § 5.2.1, the effects of the non-exact treatment of the inertial forces are larger in magnitude than these errors.

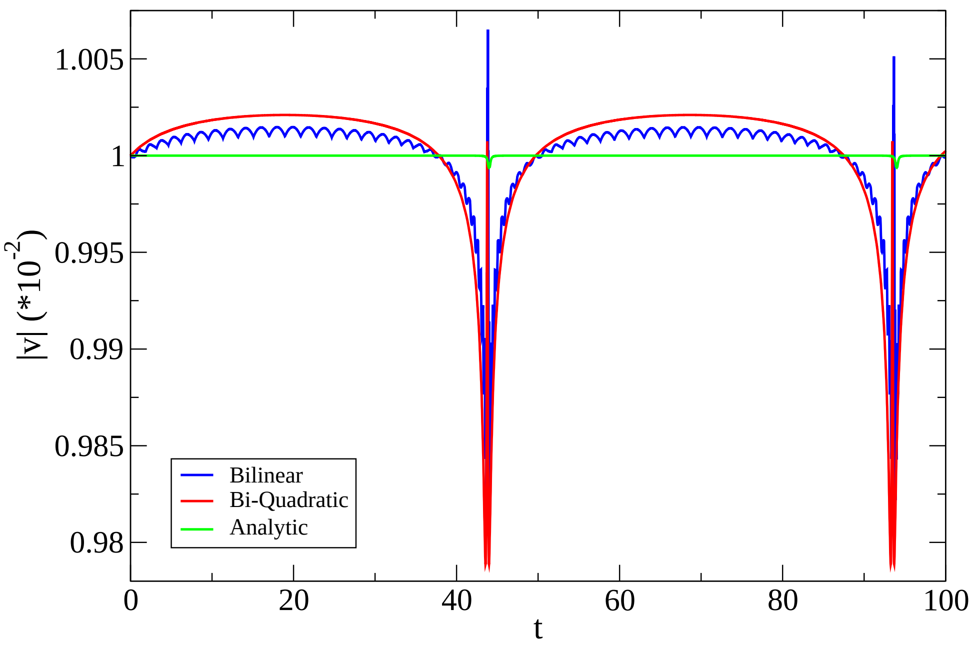
5.2 Integration of 2d Particle Equations of Motion

5.2.1 ML Mover Applied to 2d Non-Uniform Grid

In this section, we examine the effects of the inertial forces on a particle moving in a neutralizing background with doubly periodic background in two degrees of freedom by repeating the single particle tests performed in § 4.2.4. To ensure we get the full effects of the 2d grid, we run a single particle across the system at a 45° angle. In § 5.1.3, we found that the electric field at the particle is not zero. However, for the purposes of these tests we want to isolate the effects of the inertial force terms in Eqns. (5.1a), and as such set the field force term to zero.

While we have performed scaling tests in both $\Delta\xi$ and Δt as in § 4.2.4, we only show the effects of the particle shape function used for interpolation and weighting here. Fig. 5.5(a) shows the effects of the different shape functions for a nonuniform, orthogonal grid on the unit square (Eqn. (5.5a)). The analytic grid solution, in which the grid quantities $g^{\mu\nu}$ and $\frac{\partial g^{\mu\nu}}{\partial \xi^{\gamma}}$ are calculated at the particle position analytically (i.e. without interpolation), is shown for comparison. The analytic grid, while appearing as a constant in both of these figures, does actually have an approximately 0.1% variation in the velocity as the particle crosses the grid (in both cases). In these tests, we use $N_\xi = N_\eta = 100$ spatial cells, $\Delta t = 0.0125$, and $\epsilon_{\text{grid}} = 0.15$, and an initial velocity vector of $\vec{v} = [0.01, 0.01, 0]$. Particles are allowed to cross the





(b)

Figure 5.5: Single particle initially at $\vec{x} = [0.01, 0.01]$ with initial velocity given by $\vec{v} = [0.01, 0.01, 0]$ moving across a 2d nonuniform (a) orthogonal and (b) nonorthogonal grid in a unit square, given by Eqns. (5.5a) and (5.5b), respectively. Both cases use $N_\xi = N_\eta = 100$ spatial cells, $\Delta t = 0.0125$, and $\epsilon_{\text{grid}} = 0.15$.

Update	$\frac{t_{update}}{t_{\Delta t}}$
Implicit Position	67.6
Charge Accumulation	25.0
Field Solve	2.2
Explicit and Implicit Momentum + Explicit Position	5.2

Table 5.4: Comparison of the average time required for each update during a PIC cycle for a cold 2-stream instability test on a non-uniform grid. Times are normalized with respect to the total time for the current timestep.

grid boundaries to check that our doubly-periodic particle boundary conditions are working correctly.

5.2.2 Hybrid Particle Sort

In PIC simulations, the particle push and accumulate are generally the most computationally expensive portions of the code in terms of both time and memory. As the computational particles of the PIC code move about the mesh and are stored in a global array, adjacent particles in the array at a given time are in general located randomly on the mesh. Even arrays that have been initially sorted eventually decay to random positioning as the simulations advance in time [51]. As a direct result, memory accesses necessary for the push and accumulation steps thrash the memory cache heavily, stymying the simulation performance. Several authors have shown that sorting the particles within the global arrays by their physical positions helps to mitigate this cache-thrashing, leading to much more efficient particle handling, and therefore faster runtimes. [51–56]

As shown in Appendix D, our ML mover requires two passes through the particle array: first to update the positions implicitly, after which the fields are solved. The second pass through the particle array uses the fields defined at $t + \frac{\Delta t}{2}$ to update the momentum (both explicitly and implicitly), then the positions are updated explicitly.

This formulation is required for the ML integrator to retain its' 2^{nd}-order accuracy in $\Delta\xi$ and Δt.

As shown in Table 5.4, the implicit position update step alone accounts for approximately 70% of the of the total code run time, dependent on the ratio of the total number of particles and N_ξ. The reason for this large amount of run time is that, for the 2d implicit position update, the ξ and η position updates must be solved simultaneously by iterations, and each iteration moves the particle positions slightly. This means that all three contravariant metric tensor components must be interpolated to the particle position for every iteration step. We note here that the implicit momentum update also requires iterations, but since we do not move the particle during the momentum update, the field and grid quantities are only interpolated once.

The charge density is accumulated after the implicit position update step, as our ML integrator allows us to solve the field equations a only once per timestep. The charge accumulation process accounts for 25% of the code run time, whereas the field solve uses $\approx 2\%$. As such, we have implemented a hybrid particle-sort routine similar to that of Bowers [53] to reduce the cache-thrashing inherent in the memory-intensive particle pushing and charge accumulation steps of our PIC code. For our particle sort, we are able to perform efficiently an out-of-place sorting algorithm without any extra passes through the particle list. The particle list is periodically sorted by location on the logical mesh simultaneously with the push and accumulation steps, which already generate most of the data necessary for the sort, leading to gains of approximately $30-35\%$ over the unsorted code run times.

5.3 Selected 2d Full PIC Benchmarking Cases

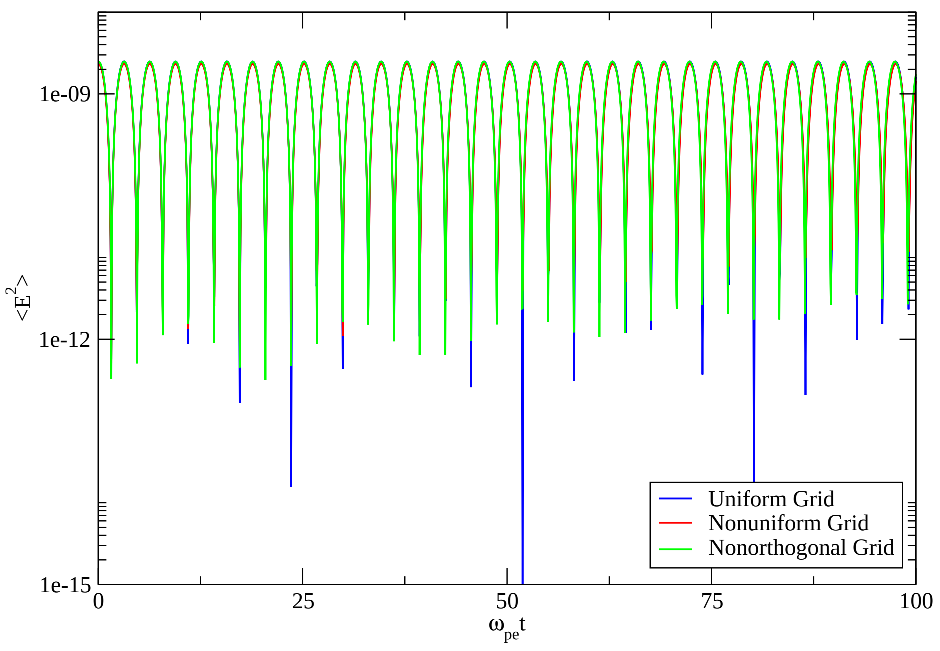
We have performed several benchmarking tests to validate the full 2d curvilinear coordinate PIC code. Presented below are several cases, all of which use physical grids with $x, y \in [-\pi : \pi]$. We have chosen these boundaries such that the boundary mapping to logical grid is tested more thoroughly, i.e. the physical grid boundaries are no longer the same as the logical grid boundaries. We utilize the same particle initialization procedures as outlined in § 4.3 for the tests presented here. For all tests presented below, we use a nonuniformity parameter of $\epsilon_{\text{grid}} = 0.1$ (for both orthogonal and nonorthogonal grids), as this leads to a ratio of area of the largest cell to smallest of ≈ 20 for the nonuniform orthogonal grid (Eqn. (5.5a)) and ≈ 4.4 for the nonorthogonal grid (Eqn. (5.5b)). The maximum skewness parameter $S_{\max}$ is ≈ 0.75 for the nonorthogonal grid with this nonuniformity parameter.

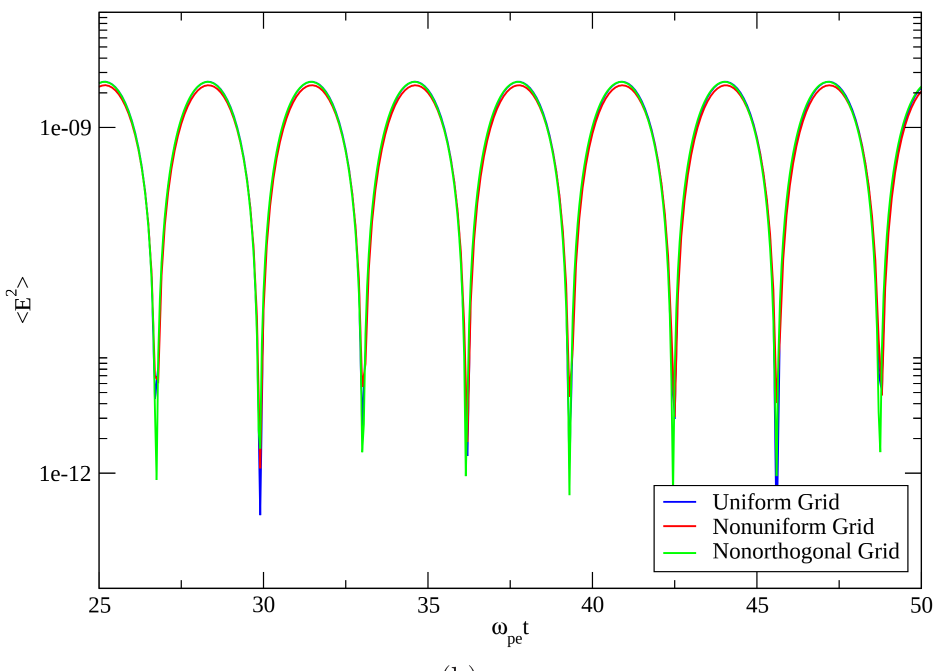
5.3.1 Cold Plasma Oscillations on a Square Physical Domain

Fig. 5.9 shows a comparison of the evolution of the electrostatic field energy, defined in 2d as

$$< E^2 >= \frac{1}{2} \int \left[(E^x)^2 + (E^y)^2 \right] dxdy, \tag{5.28}$$

for a cold plasma oscillation on the uniform, nonuniform but orthogonal, and nonorthogonal grids. For these tests, we initialize a perturbaion in the system by pertubing the particle positions with respect to the uniform neutralizing background. We define the quantity ϵ_{pert} to be the size of this initial perturbation on the particle positions. For these tests, the initial particle perturbation is directed at a 45° angle across the grid to check the effects of the interpolation in multiple dimensions on the data produced. In Fig. 5.9, we show both a long-time evolution and a zoomed section of the field energy from these same runs. We have used a perturbation of $\vec{\epsilon}_{\text{pert}} = (7.07e^{-5}, 7.07e^{-5})$,





(b)

Figure 5.6: Comparison of cold plasma oscillation field energies using a uniform grid and the nonuniform grids given by Eqns. (5.5a) and (5.5b) for (a) a long run and (b) a zoomed section of the run showing an oscillation period of 2π for all three grids. Here we have used quadratic (S_2) particle shape functions with $N_\xi = N_\eta = 128$, $\bar{N}_{\text{ppc}} = 225$, $\Delta t = 0.025$, and $\epsilon_{\text{grid}} = 0.1$ (for the nonuniform case).

such that the magnitude of the perturbation is $|\vec{\epsilon}_{\text{pert}}| = 1 \times 10^{-4}$, $N_\xi = N_\eta = 128$, $\bar{N}_{\text{ppc}} = 225$, and $\Delta t = 0.025$. No measurable growth in the electrostatic field energy was observed during the course of these runs. We note that the freqency of the plasma oscillations observed in Fig. 5.9 is very nearly 2π (as is expected) for all three grid choices. Further testing has revealed that the period of the plasma oscillations converges to the value of 2π with 2^{nd}-order accuracy as expected.

5.3.2 Cold Electron-Electron Two-Stream Instability

As our physical grid is $x, y \in [-\pi : \pi]$, the wave vector $\vec{k}$ is given by $\vec{k} = [1, 1]$, such that, to generate a two stream instability at a 45° degree angle across the grid, we use the effective wave vector, $k^{45^\circ} = \sqrt{2} \approx 1.414$. From Fig. 4.9, we know that the maximum growth rate for the two stream instability occurs at $|\vec{k} \cdot \vec{v}_0| \approx 0.63$. As such, we have chosen an initial velocity of $\vec{v}^{\vec{x}} = [0.314, 0.314]$.

Fig. 5.8 shows the time evolution of the electrostatic field energy for a uniform grid compared with the nonuniform grids given by Eqns. (5.5a) and (5.5b). Here we have used quadratic particle shape functions with $N_\xi = N_\eta = 128$, $\bar{N}_{\text{ppc}} = 225$ per species, $\Delta t = 0.025$ for all cases, again with $\epsilon_{\text{grid}} = 0.1$ for the nonuniform orthogonal and nonorthogonal cases. The same cases have been performed using bilinear particle shape functions, revealing identical results. For all cases, the growthrate is ≈ 0.35, which matches the theoretical curve shown in Fig. 4.9 quite well.

5.3.3 Landau Damping

In this section we study the effects of Landau damping with our 2d code, on a unit square physical grid. Since the our thermal distribution is taken to be Maxwellian, the particle velocities are isotropic in x, y, and as such we have set up a case in

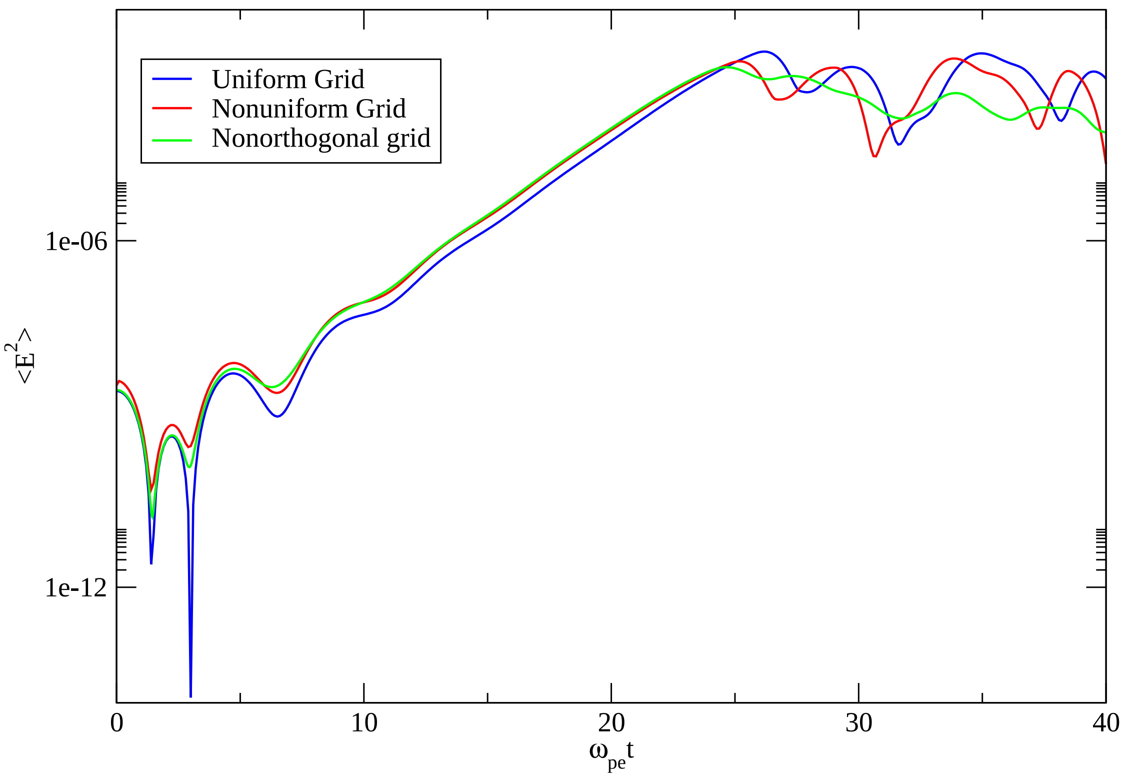


Figure 5.7: Comparison of cold electron-electron two-stream instability growth rates for uniform and non-uniform grids. Higher initial noise levels due in the nonuniform grid case provide a larger initial perturbation in the system, leading to the differences seen between the two curves. Here we have used quadratic particle shape functions with $N_\xi = N_\eta = 128$, $\bar{N}_{\text{ppc}} = 225$ per species, $\Delta t = 0.025$ for all cases, with $\epsilon_{\text{grid}} = 0.1$ for the nonuniform orthogonal and nonorthogonal cases.

which the initial perturbation is only in x for simplicity. Fig. 5.8 shows the time evolution of the Landau damping on our electrostatic field energy for the uniform, nonuniform orthogonal and nonorthogonal square grids. Here we have used $N_\xi = N_\eta = 128$, $\bar{N}_{\text{ppc}} = 400$ per species, $\Delta t = 0.025$, and $v_{th} = 0.07$ for all cases. Note that the nonorthogonal grid more closely follows the damping rate of the uniform grid than does the nonuniform, orthogonal grid. The real oscillation frequency agrees very well with the theoretical values given by Eqn. (4.44), where we have taken $\omega_{pe} = 1$ by the normalizations described in § 4.1 for all three grids. The damping

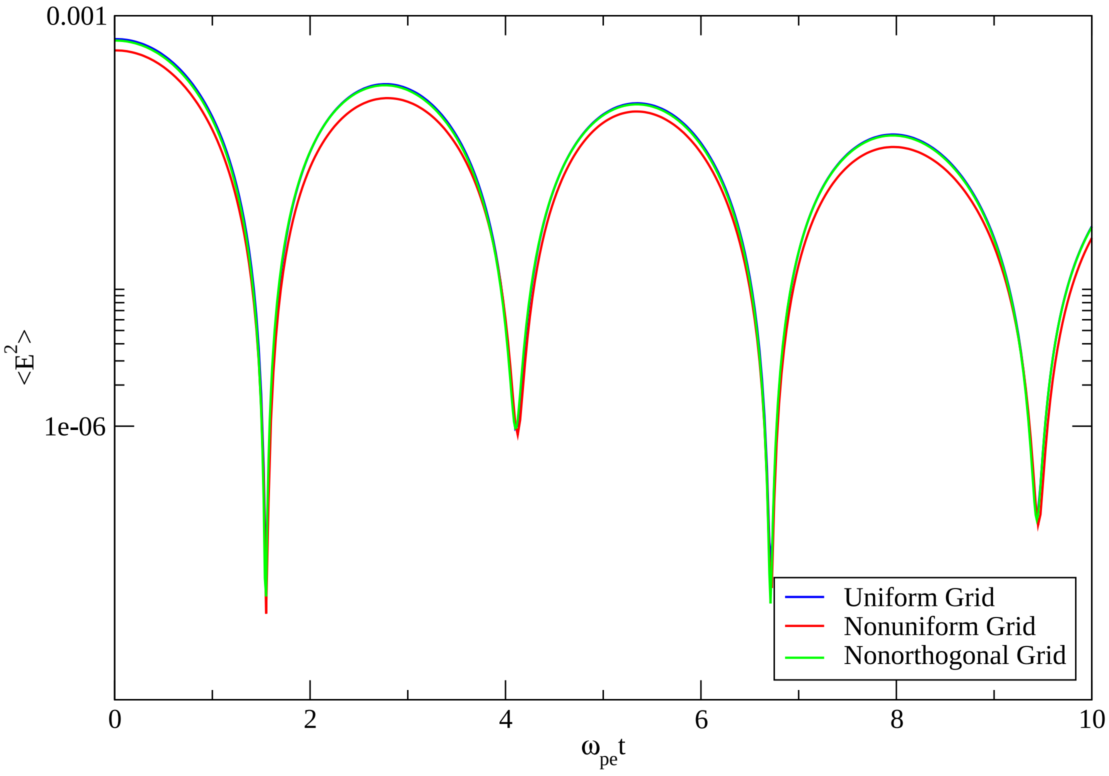


Figure 5.8: Comparison of Landau damping rates for a uniform grid with those obtained using the nonuniform, orthogonal and nonorthogonal square grids given by Eqns. (5.5a) and (5.5b). Here we have used $N_\xi = N_\eta = 128$, $\bar{N}_{\text{ppc}} = 400$ per species, $\Delta t = 0.025$, $v_{th} = 0.07$, and quadratic particle shape functions for all cases, with $\epsilon_{\text{grid}} = 0.1$ for the nonuniform orthogonal and nonorthogonal cases.

rate across the simulation time agrees to within 3% of the theoretical value of 0.124 as given by Eqn. (4.45) for these parameters for the uniform and nonorthogonal grids, whereas the damping rate on nonuniform, orthogonal grid agrees to within 5%. The lower damping rate observed for the nonuniform orthogonal grid case is due to the nonuniformity of the grid. As stated in § 5.3, the ratio of the largest to the smallest cell areas for this grid, in which we have used the nonuniformity parameter $\epsilon_{\text{grid}} = 0.1$, is ≈ 20. To check this hypothesis, we have also run cases in which we have used $\epsilon_{\text{grid}} = 0.06$ such that the largest to smallest cell area ratio is ≈ 4.88. These cases give the same damping rates as the uniform grid case shown above.

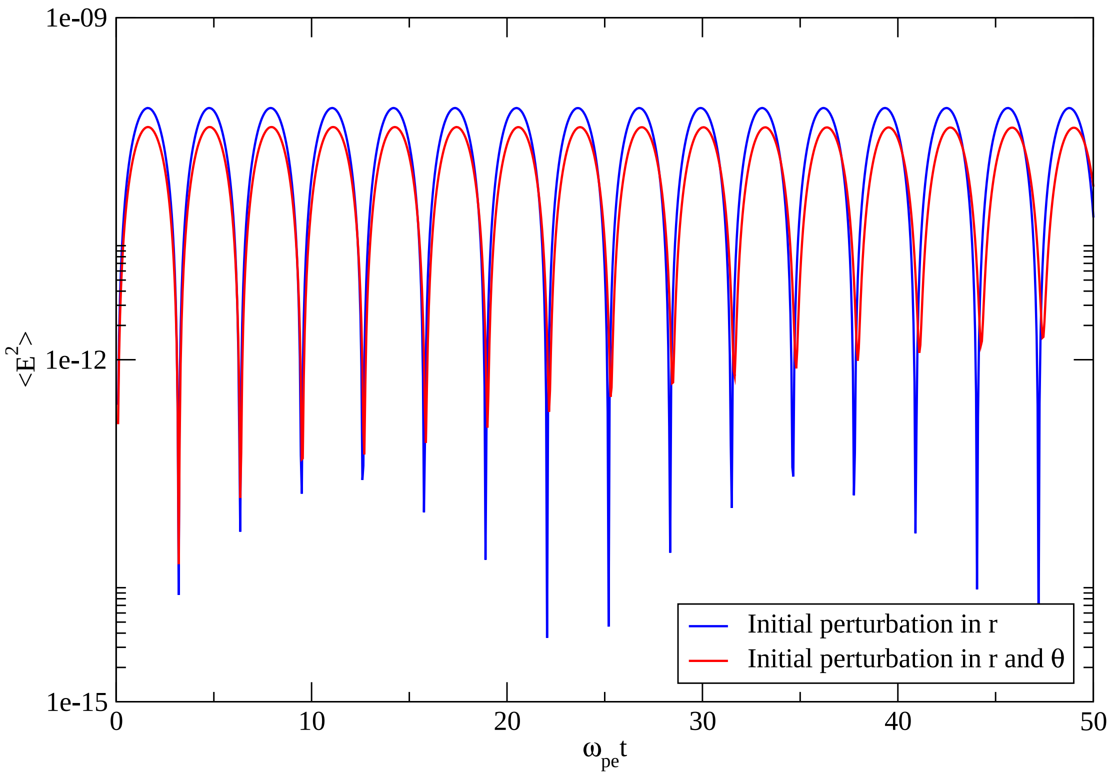


Figure 5.9: Comparison of cold plasma oscillation field energies on an annular physical grid using an initial perturbation in r only and a combination of r and θ initial perturbation showing an oscillation period very close to 2π for both cases. Here we have used quadratic (S_2) particle shape functions with $N_\xi = N_\eta = 64$, $\bar{N}_{\text{ppc}} = 400$, and $\Delta t = 0.05$.

As a final test of our entire method, we have set up a cold plasma oscillation on the concentric annulus grid. As an initial test, we perturbed the initial potential radially using

$$\tilde{\phi} = \epsilon_{\text{pert}} \cos\left(\pi\left(\frac{r - r_1}{r_2 - r_1}\right)\right), \tag{5.29}$$

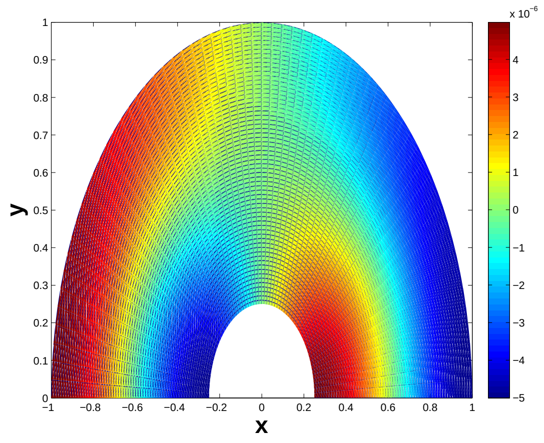
where we have used $\epsilon_{\text{pert}} = 10^{-4}$, $r_1 = 0.25$, and $r_2 = 1.0$. This initial perturbation satisfies homogeneous Neumann boundary conditions along the entire boundary of the system. With our input parameters, the ratio of the area of the largest grid cell to the smallest is 16. This is a very simple test of the system, and is analogous to perturbing our system in y only on a rectangular grid. The time evolution of the electrostatic field energy for this test is shown in the blue curve of Fig. 5.9 for a 64×64 physical grid with $\Delta t = 0.1$, $\bar{N}_{\mathrm{ppc}} = 225$, and quadratic spline shape functions. The period of oscillation of the field energy for this set of initial conditions is ≈ 6.3 for this case, and scales as $\Delta\xi^2$, meaning that the frequency is indeed unity.

As a more challenging case, we then perturbed the initial potential using

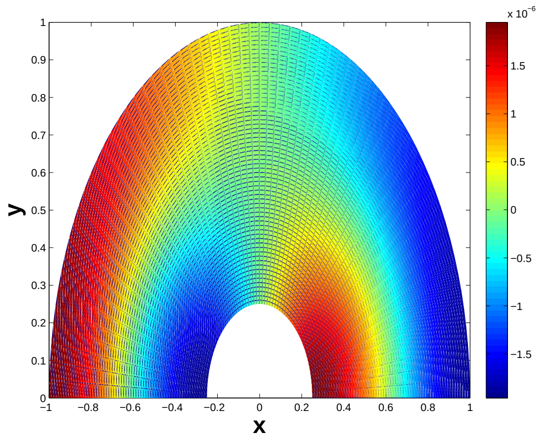
$$\tilde{\phi} = \epsilon_{\mathrm{pert}} \cos\left(\pi\left(\frac{r - r_1}{r_2 - r_1}\right)\right)\cos(\theta), \tag{5.30}$$

where we have again used $\epsilon_{\mathrm{pert}} = 10^{-4}$, $r_1 = 0.25$, and $r_2 = 1.0$. The red curve in Fig. 5.9 shows the time evolution of the electrostatic field energy for a 64×64 physical grid with $\Delta t = 0.05$, $\bar{N}_{\mathrm{ppc}} = 400$, and quadratic spline shape functions. We have used more resolution in this particular case in order to more fully resolve the complicated features of our initial potential. Again, the period of oscillation is ≈ 6.3 (the difference between the two curves is approximately 0.085%); thus the plasma frequency is unity.

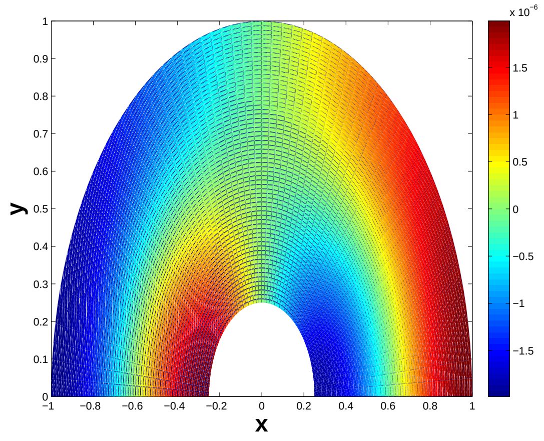
Fig. 5.10 shows the time evolution of one period of the potential on the physical grid using the initial perturbation as given by Eqn. (5.30) for the cold electrostatic plasma oscillation test described above. Figs. 5.10(a) and (b) are from one half of the plasma period and Figs. 5.10(c) and (d) are from the next. Notice the exact reversal of the potentials between the two halves of the plasma period.



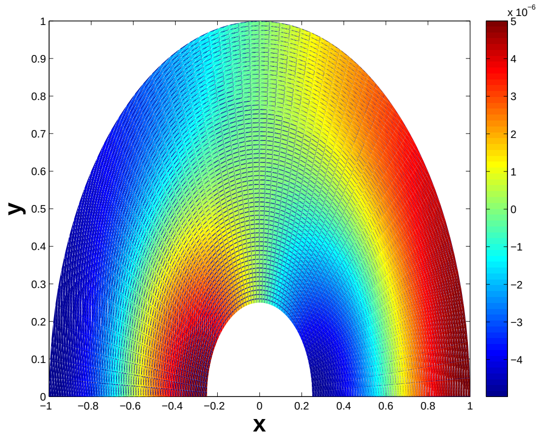
(a) $\omega_{pe}\, t = 1.6$



(b) $\omega_{pe}\, t = 2.8$



(c) $\omega_{pe}\, t = 3.6$



(d) $\omega_{pe}\, t = 4.8$

Figure 5.10: Snapshots of one period of the evolution the potential of a cold plasma oscillation on the annular physical grid. Here we have perturbed the inital potential using Eqn. (5.30).

Chapter 6

Conclusions and Future Work

The PIC method has become one of the most widely used methods for the kinetic simulation of plasmas over the course of the past 50 years. However, most "production level" PIC codes used today are still confined to using the standard, decades-old uniform physical grid methods to simulate systems in complex physical geometries. In this thesis we set out to study the feasability of developing a new approach to the PIC method, in which the key components of the PIC method–the mover, field solve, charge accumulation, and field interpolation–are carried out on a uniform, unit square logical grid mapped to an arbitrary grid on an arbitrary physical domain. For simplicity, our studies were limited to an electrostatic code and 2d.

An elliptic grid generation technique known as Winslow's method [30] was used to generate an initial boundary-fitted grid by solving a set of coupled elliptic equations using a nonlinear Newton-Krylov solver. The generated grids are then mapped onto the logical grid through the use of the mapping $\vec{x}(\vec{\xi})$ and its inverse, $\vec{\xi}(\vec{x})$, such that the PIC components can be run on the logical grid using these mappings. We have shown examples of Winslow's method as applied to a concentric annulus, for which we know the analytic solution by conformal mapping theory, as a method of checking the accuracy of our numerical solution of the coupled equations. We have also applied the method to more complicated systems such as an eccentric annulus grid, domains to represent multiple circular objects, and to elliptic objects within a circular outer boundary.

We have derived the logical grid macroparticle equations of motion based on a canonical transformation of Hamilton's equations from the physical domain to the logical. We have shown that, by our choice of canonical transformation, the initially separable physical grid Hamiltonian is transformed into a nonseparable system. The momentum component of the particle equations of motion is composed of two terms. The first is an inertial force term, which accounts for the curvature of the physical grid through the derivatives of the metric tensors on the logical grid. The second is a term proportional to the electric field which accounts for the effects of the mean fields on the particle's trajectory.

Integration of the logical grid particle equations of motion by standard "naive" leapfrog techniques would result in the loss of the Hamiltonian property inherent in our system of equations. As such, we have extended the semi-implicit modified leapfrog integrator, which was originally developed for a single degree of freedom to higher dimensions to update the logical grid particle positions and momenta in time. This method was shown here to be symplectic for a system of arbitrary dimension and 2^{nd}-order accurate in Δt if a time-centered formulation is utilized. Being semi-implicit, we are required to solve a half-timestep each of the particle positions and momenta implicitly. We have chosen to use Newton iterations for the efficiency of its convergence. The implicit position update and charge accumulation steps of the ACC-PIC cycle were shown to account for more than 90% of the total simulation run time, and as such we have implemented a hybrid particle sort method based on Bowers' [53] algorithm to manage more efficiently the particle list and reduce cache-thrashing caused by the memory-intensive push and accumulation stages of the PIC cycle. This sorting routine was shown to increase the overall performance of the code by approximately $30 - 35\%$ with respect to unsorted code run times.

In order to obtain the electrostatic fields on the logical grid, we have constructed a generalized curvilinear coordinate formulation of Poisson's equation. This equation is discretized conservatively on the logical grid using a staggered mesh. Field boundary conditions are applied in such a way as to produce a symmetric operator matrix which we then solve using a Jacobi-preconditioned conjugate gradient solver. We have performed validation tests on our Poisson solver using the method of manufactured solutions with both orthogonal and nonorthogonal grids.

By our formulation of the curvilinear coordinate Poisson equation, we are required to supply the logical grid charge density as a source term. As such, we accumulate the charge from the particles directly onto the logical grid using standard particle shape functions rather than the more complicated, weighted shape functions which must be used if the charge is accumulated on a nonuniform physical grid. In this way, the symmetric particle shape functions on the logical grid can be thought of as automatically weighted in the physical space by the components of the Jacobi matrix. Furthermore, the particle equations of motion require that the derivative of the electrostatic potential on the logical grid be obtained at the particle positions for the update of the particle momentum. These logical electric fields are interpolated to the particle positions on the logical grid using the symmetric particle shape functions which have been slightly modified from the standard shape functions used in the charge accumulation process in order to account for our choice of a staggered mesh.

We have devoted much time and effort to the study of the errors introduced into the system by the inertial and field force terms in the momentum update equations. Since our modified leapfrog integrator is not an *exact* integrator, we expect to have a small amount of error introduced into nonuniform grid systems by the inertial force term. We have found that this error is indeed small and scales with the expected

2^{nd}-order accuracy in Δt. On a uniform grid, it can easily be shown that if the same particle shape functions are used for the charge accumulation and field interpolation steps of the PIC code, the force on a single particle in a neutralizing background with periodic boundary conditions will have zero self-forces on itself. This seemingly simple proof can no longer be done for a nonuniform grid, but we have developed a method for obtaining the fields at the particle position which leads to zero self forces in 1d. The analagous method to the 1d field interpolation method has been applied to the 2d code, but does not result in zero self forces. However, our method did result in 2^{nd}-order accuracy in Δx, and the resulting fields are several orders of magnitude smaller than the errors due to the integration of the inertial forces.

The individual PIC components were then coupled to form a complete ACC-PIC code, first in 1d as a proof of principle, and then extended into two dimensions. We have validated the accuracy of both codes against a standard, uniform physical grid PIC code for several standard physics tests, including electrostatic plasma oscillations, Langmuir waves, linear and nonlinear two-stream instabilities, and Landau damping in both 1d and 2d for nonuniform, orthogonal (in 1d and 2d) and nonorthogonal (in 2d) square domains. For simplicity, we have chosen to initialize our all of our macroparticles with uniform charges and masses. As such, we initially uniformly distribute our particles across the physical domain in order to keep the initial charge density as close to uniform as possible. This means that for a nonuniform grid, the smaller grid cells tend to contain fewer particles than do the larger cells. Fluctuations in the number of particles per cell in the smallest cells leads to localized noise within the system, which was seen to lead to loss of coherence in the electrostatic plasma oscillations after very long time intervals. As expected, our method applied to a nonuniform physical domain generates more noise than the standard uniform physical grid PIC method for our chosen initial conditions. However, by increasing the resolution of our nonuniform grid simulations we are able to reproduce the theoretical values for each test to high accuracy.

Finally, we have used the concentric annulus grid as generated using Winslow's method to show that our ACC-PIC method generalizes to the boundary fitted geometry of that domain. Tests were performed in which an electrostatic wave was initialized within the annular domain. We have shown that for multiple initial perturbations within this domain, the oscillation period is indeed 2π as predicted by our code normalizations.

6.1 Future Work

There is much work to be done to extend the methods developed and tested in this thesis to the full production level ACC-PIC method discussed in the Chapter 1. Logical extensions of our work include the extension to 3d and the inclusion of electromagnetic effects. Furthermore, parallelization techniques should be utilized to handle more efficiently the particle push and charge accumulation stages of our method.

In Chapter 1, we alluded to the fact that other key components must be developed in parallel with the development of the ACC-PIC method presented in this thesis. These include a particle control method capable of managing the number of particles per cell throughout the grid and a robust moving mesh adaptation method. Both of these objectives are major undertakings, each worthy of dissertations as stand-alone methods.

Once fully developed, our method can be coupled to a moving mesh algorithm. This moving mesh, ACC-PIC coupled method would be limited by the same particle noise we have noted in this thesis, however, and therefore work must also be done to couple our method to the particle control method mentioned above, particularly because the ACC methods should be more efficient than standard PIC techniques only if the grid is quite nonuniform. This method must first be proven capable of operating on the logical domain with a static, nonuniform physical grid before we can allow the physical grid to change in time. In short, the road to our ultimate goal of a fully adaptive ACC-PIC code is sure to be a challenging, but worthwhile journey for the evolution of the PIC method.

Appendices

Appendix A

Curvilinear Coordinates: General Relations and Background

This Appendix presents the reader with a coherent overview of the relations between Cartesian and curvilinear coordinates which form the basis for this thesis.

Let the values $x^\alpha, \alpha = 1, \cdots, n$ be the Cartesian coordinates of the vector $\vec{x}$. The coordinate transformation $\vec{x}(\vec{\xi}\,)$ defines a set of curvilinear coordinates $\xi^\alpha, \cdots, \xi^n$ in the domain X^n. We define the Jacobi matrix of this transformation

$$j_{\alpha\beta}(\vec{\xi}\,) \equiv \left(\frac{\partial x^\alpha}{\partial \xi^\beta}\right), \quad \alpha, \beta = 1, \cdots, n, \tag{A.1}$$

and its Jacobian

$$J(\vec{\xi}\,) \equiv \det(\boldsymbol{j})\,. \tag{A.2}$$

Conversely, we can also think of this transformation as a mapping of $\vec{\xi}$ onto $\vec{x}$, $\vec{\xi}(\vec{x}\,)$. Defining the inverse of the matrix $j_{\alpha\beta}$ as

$$k^{\alpha\beta}(\vec{x}\,) \equiv \left(\frac{\partial \xi^\alpha}{\partial x^\beta}\right), \quad \alpha, \beta = 1, \cdots, n, \tag{A.3}$$

we can write its Jacobian

$$K(\vec{x}) \equiv \det(\boldsymbol{k}) = \frac{1}{J}. \tag{A.4}$$

Simple linear algebra tells us that

$$j_{\alpha\beta}k^{\beta\gamma} = \frac{\partial x^\alpha}{\partial \xi^\beta}\frac{\partial \xi^\beta}{\partial x^\gamma} \equiv \delta_\alpha^\gamma, \tag{A.5}$$

therefore $\boldsymbol{jk} = \boldsymbol{I}$.

Given a Euclidean metric on the physical space, we have

$$dx^\gamma dx^\gamma = \frac{\partial x^\gamma}{\partial \xi^\alpha}\frac{\partial x^\gamma}{\partial \xi^\beta} d\xi^\alpha d\xi^\beta = g_{\alpha\beta} d\xi^\alpha d\xi^\beta, \tag{A.6}$$

and the covariant metric tensor is defined as

$$g_{\alpha\beta}(\vec{\xi}\,) \equiv \frac{\partial x^\gamma}{\partial \xi^\alpha}\frac{\partial x^\gamma}{\partial \xi^\beta}, \quad \alpha, \beta, \gamma = 1, \cdots, n. \tag{A.7}$$

From Eq. (A.1), we see $g_{\alpha\beta} = j_{\gamma\alpha}j_{\gamma\beta}$, which is simply $\boldsymbol{g}_{\text{cov}} = \boldsymbol{j}^T\boldsymbol{j}$. Likewise, the contravariant metric tensor is based upon the inverse Jacobian matrix, $\boldsymbol{k}$:

$$g^{\alpha\beta}(\vec{x}\,) \equiv \frac{\partial \xi^\alpha}{\partial x^\gamma}\frac{\partial \xi^\beta}{\partial x^\gamma}, \quad \alpha, \beta, \gamma = 1, \cdots, n, \tag{A.8}$$

thus $\boldsymbol{g}^{\text{contra}} = \boldsymbol{k}\boldsymbol{k}^T$.

Finally, by multiplying the covariant and contravariant metric tensors and applying the identity $\boldsymbol{jk} = \boldsymbol{I}$, we can prove that the covariant and contravariant metric tensors are in fact inverses of each other:

$$\boldsymbol{g}_{\text{cov}}\boldsymbol{g}^{\text{contra}} = \boldsymbol{j}^T\boldsymbol{j}\boldsymbol{k}\boldsymbol{k}^T = \boldsymbol{j}^T\boldsymbol{k}^T = (\boldsymbol{kj})^T = \boldsymbol{I}^T = \boldsymbol{I}. \tag{A.9}$$

In two dimensions, we can easily convert from the covariant to the contravariant metric tensor using the following equation:

$$g^{\alpha\beta} = (-1)^{\alpha+\beta}\,\frac{g_{3-\alpha,3-\beta}}{g_{\text{cov}}}, \quad \alpha, \beta = 1, 2, \tag{A.10}$$

where $g_{\text{cov}} = J^2$ is the determinant of the covaraint metric tensor, and $g_{\text{cov}} = \frac{1}{g^{\text{contra}}}$, where $g^{\text{contra}} = K^2$ is the determinant of the contravariant metric tensor. Likewise, we can shift from the contravariant to the covariant metric tensor using

$$g_{\alpha\beta} = (-1)^{\alpha+\beta}\, g_{\text{cov}} g^{3-\alpha,3-\beta}, \quad \alpha, \beta = 1, 2. \tag{A.11}$$

Appendix B

Macroparticle Interpolation and Weighting on a Staggered Mesh

In a method utilizing spatial differencing such as PIC, it is often more convenient to define grid-based quantities on a uniform, *staggered* mesh. In § 2.4.1, we showed that it is imperative that the same shape functions be used in weighting the particles to the grid as for interpolation from the grid to the particles (on a uniform grid). However, on a staggered mesh, we are weighting particles from the contiuum to the cell-centers, whereas the electric fields are interpolated from the vertices to the continuum. This means that without some corrections to our weighting function for the staggered grid, we will have self-forces on the macroparticles. As a reminder, in arbitrary curvilinear coordinates, total momentum conservation is dependent on two terms: inertial forces and field forces. The proof constructed below deals solely with the field force terms. We specialize to 1d for clarity.

B.1 Extending Momentum Conservation to a Uniform, Staggered Grid

To extend the proof of § 2.4.1 to a uniform staggered grid, we must first define our grid layout (assuming a 1d grid for simplicity). As in § 2.4.1, we assume that ρ and Φ exist at the cell-centers and will be denoted by $i \pm \frac{1}{2}$, whereas the electric field is placed at the vertices, as it is obtained from Eqn. (2.5) by a centered-differencing of the potential:

$$\frac{\Phi_{i-\frac{1}{2}} - \Phi_{i+\frac{1}{2}}}{\Delta x} = E_i. \tag{B.1}$$

Starting with Eqn. (2.30) and defining the vertex-valued density as $\rho_i^v \equiv \frac{\rho_{i+\frac{1}{2}}+\rho_{i-\frac{1}{2}}}{2}$, we can write

$$\frac{dP}{dt} = \Delta x \sum_i E_i \left(\frac{\rho_{i+\frac{1}{2}} + \rho_{i-\frac{1}{2}}}{2} \right). \tag{B.2}$$

From a direct implementation of Gauss' law on the staggered grid, we can find the electric field on the vertices due to the charge density on the cell-centers by

$$E_{i+1} = E_i + \Delta x \, \rho_{i+\frac{1}{2}}, \tag{B.3}$$

therefore we can rewrite Eqn. (B.2) as

$$\frac{dP}{dt} = \sum_i E_i \left(\frac{E_{i+1} - E_{i-1}}{2} \right), \tag{B.4}$$

which is identical to Eqn. (2.30) but with the electric field now defined on vertices instead of cell-centers. This sum telescopes for appropriate boundary conditions, leading to exact momentum conservation in a Cartesian system.

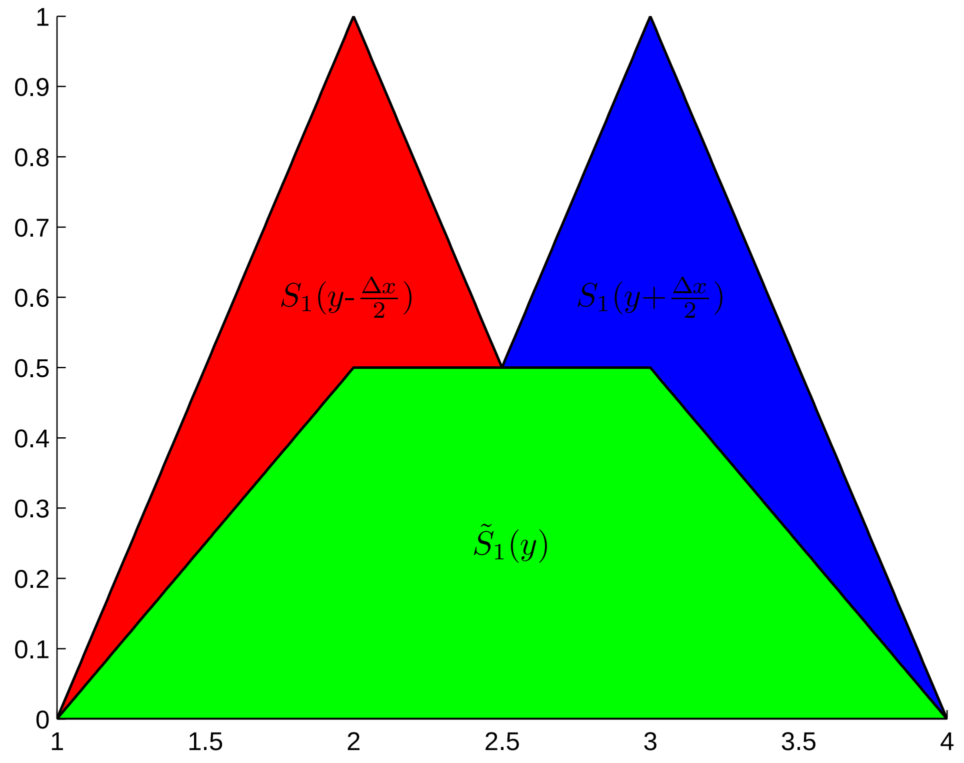


Figure B.1: The shape function $\tilde{S}_1(y)$ for linear interpolation of vertex-defined quantities on a staggered grid in 1d. $\tilde{S}_1(y)$ is half the sum of the cell-centered shape functions $S_1\left(y - \frac{\Delta x}{2}\right)$ and $S_1\left(y + \frac{\Delta x}{2}\right)$.

B.2 A Straight-Forward Approach to Linear Interpolation by Substitution

Here we present an application of the staggered-grid momentum conservation methods detailed in § B.1 to find the electric field at a particle using linear interpolation. To do this, we create a new particle shape $\tilde{S}_1$ such that the charge accumulation step can retain its normal weighting scheme. The algorithm based on values of the electric field at the cell-centers $E^c_{i+\frac{1}{2}}$ to be linearly interpolated to the particle position is given by

$$E(x_p) = \frac{E^c_{i-\frac{1}{2}}(1-x_l) + E^c_{i+\frac{1}{2}}(1+x_l)}{2}. \tag{B.5}$$

The quantity $x_l = x_l(x_p)$ is the local particle position on the interval $[x_{i-\frac{1}{2}} : x_{i+\frac{1}{2}}]$ mapped linearly onto $[-1:1]$.

If we choose to substitute the cell-centered electric fields using their vertex-based counterparts, $E^c_{i-\frac{1}{2}} = \frac{E_i + E_{i-1}}{2}$ and $E^c_{i+\frac{1}{2}} = \frac{E_i + E_{i+1}}{2}$, we obtain

$$E(x_p) = \frac{(E_{i-1} + E_i)(1-x_l) + (E_i + E_{i+1})(1+x_l)}{4}. \tag{B.6}$$

Simplifying, we have

$$E(x_p) = \frac{E_{i-1}(1-x_l) + 2E_i + E_{i+1}(1+x_l)}{4}, \tag{B.7}$$

which gives us a three-point linear interpolation as shown in Fig. B.1. In this case, the particle is located on a grid vertex, and the resulting coefficients for the vertex-based electric fields are $\left(\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\right)$.

B.3 Extension to Quadratic Spline Interpolation

We can extend the method utilized in § B.2 to account for a smoother particle shape by using quadratic splines. Assuming that we now locate our particles within the computational interval $[x_i : x_{i+1}]$ and map them linearly onto $[-0.5 : 0.5]$, the algorithm for a colocated electric field to be interpolated to the particle position via quadratic splines is given by

$$E(x_p) = \frac{1}{2}E_{i-\frac{1}{2}}\left(\frac{1}{2} - x_l\right)^2 + E_{i+\frac{1}{2}}\left(\frac{3}{4} - x_l^2\right) + \frac{1}{2}E_{i+\frac{3}{2}}\left(\frac{1}{2} + x_l\right)^2. \tag{B.8}$$

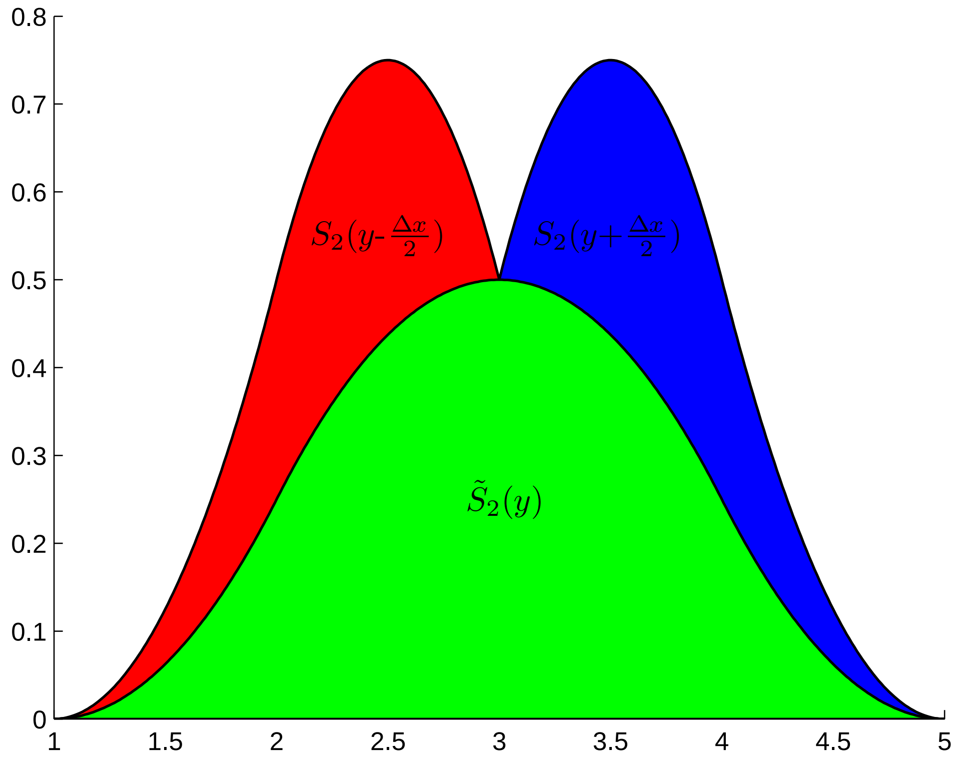


Figure B.2: The shape function $\tilde{S}_2(y)$ for quadratic spline interpolation of vertex-defined quantities on a staggered grid in 1d.

Again replacing the cell-centered electric fields with their vertex-based counterparts, expanding the coefficient terms, and simplifying gives

$$\begin{aligned} E(x_p) &= \frac{1}{4}\left[E_{i-1}\left(\frac{1}{2}-x_l\right)^2 + E_i\left(\frac{7}{4}-x_l-x_l^2\right) + \right. \\ &\qquad \left. E_{i+1}\left(\frac{7}{4}+x_l-x_l^2\right) + E_{i+2}\left(\frac{1}{2}+x_l\right)^2\right]. \end{aligned} \tag{B.9}$$

Again we see that the shape function stencil for $\tilde{S}_2(y)$ is wider than the original shape functions S_2 by Δx, leading to a smoother particle shape. Also notice that both 1stand 2$^{\text{nd}}$-order terms are now included for the two middle grid points. Fig. B.2 shows the final shape of $\tilde{S}_2$ in relation to the two original shape functions S_2.

Appendix C

Self-forces on Nonuniform Grid

In § 2.4.1 and Appendix B.1 of this thesis, we showed that for a distribution of particles in a neutralizing background on a periodic grid, the total momentum of the system is conserved. On a 1d grid, we can also show that for a single particle in a uniform neutralizing background, the self-forces at the particle position are zero if implemented correctly.

C.1 Uniform physical grid

The force on a single particle is given by

$$F^x(x_p) = q \sum_i E_i^x S(x_p - x_i). \tag{C.1}$$

Since

$$\rho_i^x = qS(x_i - x_p) - \rho_n^x, \tag{C.2}$$

where ρ_n^x is the uniform neutralizing background (a constant on the physical grid), we can rewrite Eqn. C.1 as

$$\begin{aligned} F^x(x_p) &= \sum_i E_i \left(\frac{\rho^x_{i+\frac{1}{2}} + \rho^x_{i-\frac{1}{2}}}{2} + \rho^p_n \right) && \text{(C.3)} \\ &= \sum_i E^x_i \left(\frac{\rho^x_{i+\frac{1}{2}} + \rho^x_{i-\frac{1}{2}}}{2} \right) + \rho^x_n \sum_i E_i, && \text{(C.4)} \end{aligned}$$

where we have assumed a staggered grid. Here we have used the symmetry of the particle shape function, $S_{pi} = S_{ip}$. (Since ρ_n^x is a constant on the uniform physical grid, it doesn't matter where we define it.)

By Gauss' law (Eqn. (4.1)),

$$\rho^x_{i+\frac{1}{2}} = \frac{E_{i+1} - E_i}{\Delta x} \tag{C.5}$$

we can rewrite Eqn. C.3 as

$$F^x(x_p) = \frac{1}{2\Delta x} \sum_i E_i \left(E_{i+1} - E_{i-1} \right) + \rho^x_n \sum_i E_i. \tag{C.6}$$

The first term in Eqn. C.6 telescopes as was shown in § 2.4.1, and is therefore zero. The second term is also zero since, because of our periodic boundary conditions, the discrete form of $\int E\, dx = 0$, namely $\sum_i E_i = 0$, holds.

C.2 Nonuniform physical grid

C.2.1 Direct interpolation of the logical electric fields

On a nonuniform grid, we can follow the same procedures to interpolate the logical electric fields (in 1d), $E^\xi = -\frac{\partial \Phi}{\partial \xi}$, to the particle position on the physical grid. For a single particle on the logical grid, we have

$$F^{\xi}(\xi_p) = q\sum_i E_i^{\xi} S(\xi_i - \xi_p). \tag{C.7}$$

On the logical grid,

$$\rho^{\xi}_{i+\frac{1}{2}} = qS(\xi_i - \xi_p) - \rho^{\xi}_{n,i+\frac{1}{2}}, \tag{C.8}$$

where $\rho^{\xi}_{n,i+\frac{1}{2}}$ is the neutralizing background charge density at the cell center $i+\frac{1}{2}$. Since we are on the logical grid this background density is not a constant (the charge density on the *physical* grid is a constant). As such, we rewrite Eqn. C.7 as

$$F^{\xi}(\xi_p) = \sum_i E_i^{\xi}\left(\frac{\rho^{\xi}_{i+\frac{1}{2}} + \rho^{\xi}_{i-\frac{1}{2}}}{2}\right) + \sum_i E_i^{\xi}\left(\frac{\rho^{\xi}_{n,i+\frac{1}{2}} + \rho^{\xi}_{n,i-\frac{1}{2}}}{2}\right). \tag{C.9}$$

Since $\rho^{\xi}_{i+\frac{1}{2}} = \rho^{x}_{i+\frac{1}{2}} J_{i+\frac{1}{2}}$, we can write

$$\begin{aligned} F^{\xi}(\xi_p) &= \frac{1}{2\Delta\xi}\sum_i E_i^{\xi}\left(E^x_{i+1} - E^x_{i-1}\right) + \frac{\rho^x_n}{2}\sum_i E_i^{\xi}\left(J_{i+\frac{1}{2}} - J_{i-\frac{1}{2}}\right) \\ &= \frac{1}{2\Delta\xi}\sum_i E_i^{\xi}\left(J^v_{i+1}E^{\xi}_{i+1} - J^v_{i-1}E^{\xi}_{i-1}\right) + \rho^x_n \sum_i E_i^{\xi}\, J_i^v \end{aligned} \tag{C.10}$$

where J_i^v is the Jacobian defined at the vertex i. If we now look at these terms, we can easily see that *neither* is zero, thereby explaining the nonzero logical electric field at the particles observed in § 4.2.1.

C.2.2 Interpolation of E^x and J to Obtain E^{ξ}

If instead we first interpolate the physical electric fields on the logical grid and the Jacobian to the particle position, then multiply, there are no self-forces at the particle. In the following we examine why this occurs. Starting with

$$F^{x}(\xi_p) = q\sum_i E_i^{x} S(\xi_p - \xi_i) \tag{C.11}$$

and using Eqn. (C.8), we can then rewrite Eqn. C.11 as

$$\begin{aligned} F^x(\xi_p) &= \frac{1}{2}\sum_i E_i^x \left(\rho^\xi_{i+\frac{1}{2}} + \rho^\xi_{i-\frac{1}{2}}\right) + \frac{1}{2}\sum_i E_i^x \left(\rho^\xi_{n,i+\frac{1}{2}} + \rho^\xi_{n,i-\frac{1}{2}}\right) \\ &= \frac{1}{2\Delta\xi}\sum_i E_i^x \left(E_{i+1}^x - E_{i-1}^x\right) + \frac{\rho_n^x}{2}\sum_i E_i^x\, J_i^v. \end{aligned} \tag{C.12}$$

The first term telescopes in the same fashion as in the uniform physical grid case and is therefore zero. In the second term of Eqn. (C.12), we are able to simply pull the Jacobian off the charge density term and apply it to the electric field terms. Therefore, since $E_i^x\, J_i^v = E_i^\xi$ and $\int E^\xi\, d\xi = \int E^x\, dx = 0$, the second term is also zero. Therefore, after interpolating the Jacobian to the particle positions and multiplying by the physical force on the particle, we have $F^\xi(\xi_p) = 0$ as well. There are therefore no self forces at the particle position using this method. We note here that while the Jacobian must also be interpolated to the particle positions here and in our mover, the method used for this interpolation *does not* have to be the same as that used for the charge weighting/field interpolation steps.

Interestingly, the method outlined here happens to be a "trick" that only works in 1d. If we apply this same method to the 2d case, we can no longer simply pull the Jacobian off the neutralizing background density term and apply it to the electric field term, as the electric field is now a 2d tensor (E^ξ, E^η), requiring individual components of the 2d Jacobi matrix rather than the entire determinant.

Appendix D

2d ML Mover Pseudocode

In this Appendix, we provide pseudocode for the 2d ML integrator applied to the logical grid particle equations of motion.

Algorithm 1 Unsorted ML Integration of 2d Logical Grid Equations of Motion

```
for $i = 1, N_{\text{tot}}$ do
  if (particle is alive) then
    Interpolate grid properties to particle position
    Make an initial guess for implicit position update:
    call explicit_position_update
    Update positions implicitly:
    call implicit_position_update
    Check particle boundary conditions
    if (particle is alive) then
      Update $\rho^{\xi}$
    end if
  end if
end for
Update boundary conditions on $\rho^{\xi}$
Update fields
for $i = 1, N_{\text{tot}}$ do
  if (particle is alive) then
    Interpolate grid properties to particle position
    Interpolate fields to particle position
    Calculate logical electric fields:
    $E_{\xi} = j_{11}E_x + j_{12}E_y$; $E_{\eta} = j_{21}E_x + j_{22}E_y$
    Make an initial guess for implicit momentum update:
    call explicit_momentum_update
    Update momentum implicitly:
    call implicit_momentum_update
    Update momentum explicitly:
    call explicit_momentum_update
    Update position explicitly:
    call explicit_position_update
    Check particle boundary conditions
  end if
end for
```

Algorithm 2 Explicit Position Update Subroutine

```
Input: Grid quantities $g^{11}(\xi_p,\eta_p), g^{12}(\xi_p,\eta_p), g^{22}(\xi_p,\eta_p)$
$\xi'_p = \xi_p + \frac{\Delta t}{2M_p}\left(g^{11}(\xi_p,\eta_p)\,P_p^{\xi} + g^{12}(\xi_p,\eta_p)\,P_p^{\eta}\right)$
$\eta'_p = \eta_p + \frac{\Delta t}{2M_p}\left(g^{12}(\xi_p,\eta_p)\,P_p^{\xi} + g^{22}(\xi_p,\eta_p)\,P_p^{\eta}\right)$
```

Algorithm 3 Implicit Position Update Subroutine

```
Input: Initial position guess $\xi^{(0)}, \eta^{(0)}$
2d Newton iterations:
while (not converged) do
    Interpolate grid properties to particle position:
    $g^{11}(\xi_p,\eta_p), g^{12}(\xi_p,\eta_p), g^{22}(\xi_p,\eta_p)$
    $\frac{\partial g^{11}}{\partial \xi}(\xi_p,\eta_p), \frac{\partial g^{11}}{\partial \eta}(\xi_p,\eta_p), \frac{\partial g^{12}}{\partial \xi}(\xi_p,\eta_p), \frac{\partial g^{12}}{\partial \eta}(\xi_p,\eta_p), \frac{\partial g^{22}}{\partial \xi}(\xi_p,\eta_p), \frac{\partial g^{22}}{\partial \eta}(\xi_p,\eta_p)$
    Set up nonlinear residual
    $f_1(\xi',\eta') = \xi' - \xi - \frac{\Delta t}{2M_p}\left(g^{11}(\xi'_p,\eta'_p)\,P_p^{\xi} + g^{12}(\xi'_p,\eta'_p)\,P_p^{\eta}\right)$
    $f_2(\xi',\eta') = \eta' - \eta - \frac{\Delta t}{2M_p}\left(g^{12}(\xi'_p,\eta'_p)\,P_p^{\xi} + g^{22}(\xi'_p,\eta'_p)\,P_p^{\eta}\right)$
    Calculate Jacobi matrix components from residual:
    $a = \frac{\partial f_1}{\partial \xi'}$    $b = \frac{\partial f_1}{\partial \eta'}$
    $c = \frac{\partial f_2}{\partial \xi'}$    $d = \frac{\partial f_2}{\partial \eta'}$
    Invert:
    $\delta_{\xi}^{(n)} = -\frac{df_1 - bf_2}{ad - bc}$
    $\delta_{\eta}^{(n)} = -\frac{af_2 - cf_1}{ad - bc}$
    Update positions for current iteration:
    $\xi^{(n)} = \xi^{(n-1)} + \delta^{\xi\,(n)}$
    $\eta^{(n)} = \eta^{(n-1)} + \delta^{\eta\,(n)}$
end while
```

Algorithm 4 Explicit Momentum Update Subroutine

```
Input: Grid quantities $\frac{\partial g^{11}}{\partial \xi}(\xi_p,\eta_p), \frac{\partial g^{11}}{\partial \eta}(\xi_p,\eta_p), \frac{\partial g^{12}}{\partial \xi}(\xi_p,\eta_p), \frac{\partial g^{12}}{\partial \eta}(\xi_p,\eta_p),$
       $\frac{\partial g^{22}}{\partial \xi}(\xi_p,\eta_p), \frac{\partial g^{22}}{\partial \eta}(\xi_p,\eta_p)$
Input: Field quantities $E^{\xi}(\xi_p,\eta_p), E^{\eta}(\xi_p,\eta_p)$
if (Spherical Poisson Solve) then
    Input: Grid quantities $j_{11}(\xi_p,\eta_p), j_{12}(\xi_p,\eta_p), x(\xi_p,\eta_p)$
    $V^{\xi}_{\text{eff}} = \frac{\Delta t\, j_{11}(\xi_p,\eta_p)\, P^{z\,2}_p}{M_p\, x(\xi_p,\eta_p)^3}$
    $V^{\eta}_{\text{eff}} = \frac{\Delta t\, j_{12}(\xi_p,\eta_p)\, P^{z\,2}_p}{M_p\, x(\xi_p,\eta_p)^3}$
else
    $V^{\xi}_{\text{eff}} = 0.$
    $V^{\eta}_{\text{eff}} = 0.$
end if
$P^{\xi'}_p = P^{\xi}_p - \frac{\Delta t^2}{2m}\left(P^{\xi\,2}_p \frac{\partial g^{11}}{\partial \xi}(\xi_p,\eta_p) + 2P^{\xi}_p P^{\eta}_p \frac{\partial g^{12}}{\partial \xi}(\xi_p,\eta_p) + P^{\eta\,2}_p \frac{\partial g^{22}}{\partial \xi}(\xi_p,\eta_p)\right) + V^{\xi}_{\text{eff}} + \Delta t\, Q_p\, E^{\xi}(\xi_p,\eta_p)$
$P^{\eta'}_p = P^{\eta}_p - \frac{\Delta t^2}{2m}\left(P^{\xi\,2}_p \frac{\partial g^{11}}{\partial \eta}(\xi_p,\eta_p) + 2P^{\xi}_p P^{\eta}_p \frac{\partial g^{12}}{\partial \eta}(\xi_p,\eta_p) + P^{\eta\,2}_p \frac{\partial g^{22}}{\partial \eta}(\xi_p,\eta_p)\right) + V^{\eta}_{\text{eff}} + \Delta t\, Q_p\, E^{\eta}(\xi_p,\eta_p)$
```

Algorithm 5 Implicit Momentum Update Subroutine

```
Input: Initial momentum guess $P^{\xi\,(0)}, P^{\eta\,(0)}$
Input: Grid quantities $\frac{\partial g^{11}}{\partial \xi}(\xi_p,\eta_p)$, $\frac{\partial g^{11}}{\partial \eta}(\xi_p,\eta_p)$, $\frac{\partial g^{12}}{\partial \xi}(\xi_p,\eta_p)$, $\frac{\partial g^{12}}{\partial \eta}(\xi_p,\eta_p)$
       $\frac{\partial g^{22}}{\partial \xi}(\xi_p,\eta_p)$, $\frac{\partial g^{22}}{\partial \eta}(\xi_p,\eta_p)$
Input: Field quantities $E^{\xi}(\xi_p,\eta_p), E^{\eta}(\xi_p,\eta_p)$
if (Spherical Poisson Solve) then
    Input: Grid quantities $j_{11}(\xi_p,\eta_p), j_{12}(\xi_p,\eta_p), x(\xi_p,\eta_p)$
    $V^{\xi}_{\text{eff}} = \frac{\Delta t\, j_{11}(\xi_p,\eta_p)\, P^{z\,2}_p}{M_p\, x(\xi_p,\eta_p)^3}$
    $V^{\eta}_{\text{eff}} = \frac{\Delta t\, j_{12}(\xi_p,\eta_p)\, P^{z\,2}_p}{M_p\, x(\xi_p,\eta_p)^3}$
else
    $V^{\xi}_{\text{eff}} = 0.$
    $V^{\eta}_{\text{eff}} = 0.$
end if


2d Newton iterations:
while (not converged) do
    Set up nonlinear residual
    $f_1(P^{\xi'}, P^{\eta'}) = P^{\xi'}_p - P^{\xi}_p + \frac{\Delta t^2}{2m}\left(P^{\xi\,2}_p \frac{\partial g^{11}}{\partial \xi}(\xi_p,\eta_p) + 2P^{\xi}_p P^{\eta}_p \frac{\partial g^{12}}{\partial \xi}(\xi_p,\eta_p) + P^{\eta\,2}_p \frac{\partial g^{22}}{\partial \xi}(\xi_p,\eta_p)\right) -$
    $V^{\xi}_{\text{eff}} - \Delta t\, Q_p\, E^{\xi}(\xi_p,\eta_p)$
    $f_2(P^{\xi'}, P^{\eta'}) = P^{\eta'}_p - P^{\eta}_p + \frac{\Delta t^2}{2m}\left(P^{\xi\,2}_p \frac{\partial g^{11}}{\partial \eta}(\xi_p,\eta_p) + 2P^{\xi}_p P^{\eta}_p \frac{\partial g^{12}}{\partial \eta}(\xi_p,\eta_p) + P^{\eta\,2}_p \frac{\partial g^{22}}{\partial \eta}(\xi_p,\eta_p)\right) -$
    $V^{\eta}_{\text{eff}} - \Delta t\, Q_p\, E^{\eta}(\xi_p,\eta_p)$
    Calculate Jacobi matrix components from nonlinear residual:
    $a = \frac{\partial f_1}{\partial \xi'}$    $b = \frac{\partial f_1}{\partial \eta'}$
    $c = \frac{\partial f_2}{\partial \xi'}$    $d = \frac{\partial f_2}{\partial \eta'}$
    Invert:
    $\delta^{(n)}_{P^{\xi}} = -\frac{df_1 - bf_2}{ad-bc}$
    $\delta^{(n)}_{P^{\eta}} = -\frac{af_2 - cf_1}{ad-bc}$
    Update positions for current iteration:
    $P^{\xi\,(n)} = P^{\xi\,(n-1)} + \delta^{(n)}_{P^{\xi}}$
    $P^{\eta\,(n)} = P^{\eta\,(n-1)} + \delta^{(n)}_{P^{\eta}}$
end while
```

References

[1] R. J. Goldston and P. H. Rutherford, *Introduction to Plasma Physics*. Institute of Physics Publishing, Philadelphia, 2003.

[2] F. F. Chen, *Introduction to Plasma Physics and Controlled Fusion*. Plenum Press, New York, 1984.

[3] D. R. Nicholson, *Introduction to Plasma Theory*. Kreiger Publishing Company, 1992.

[4] D. Nunn, "Vlasov-hybrid simulation–An efficient and stable algorithm for the numerical simulation of collision-free plasma," *Transport Theory and Statistical Physics*, vol. 34, p. 151, 2005.

[5] M. R. Feix and P. Bertrand, "A universal model: The Vlasov equation," *Transport Theory and Statistical Physics*, vol. 34, p. 7, 2005.

[6] A. J. Christlieb, R. Krasny, J. P. Verboncoeur, J. W. Emhoff, and I. D. Boyd, "Grid-free plasma simulation techniques," *IEEE Transactions on Plasma Science*, vol. 34, no. 2, p. 149, 2006.

[7] F. H. Harlow, "A machine calculation method for hydrodynamic problems," Tech. Rep. LAMS-1956, Lawrence Alamos Scientific Laboratory, 1955.

[8] O. Buneman, "Dissipation of currents in ionized media," *Physical Review*, vol. 115, p. 503, 1959.

[9] J. M. Dawson, "One-dimensional plasma model," *Physics of Fluids*, vol. 5, p. 445, 1962.

[10] C. K. Birdsall and A. B. Langdon, *Plasma Physics via Computer Simulation*. Adam Hilger, New York, 1991.

[11] R. W. Hockney and J. W. Eastwood, *Computer Simulation using Particles*. McGraw Hill Company, New York, 1981.

[12] M. E. Jones, "Electromagnetic PIC codes with body-fitted coordinates," 12th International Conference on the Numerical Simulation of Plasmas, 1987. (San Francisco, CA), Talk IM3.

[13] T. Westermann, "Electromagnetic Particle-in-Cell simulations of the self-magnetically insulated B_θ-diode," *Nuclear Instruments and Methods in Physics Research*, vol. A281, p. 253, 1989.

[14] C.-D. Munz, R. Schneider, E. Sonnendrücker, E. Stein, U. Voss, and T. Westermann, "A finite-volume Particle-in-Cell method for the numerical treatment of Maxwell-Lorentz equations on boundary-fitted meshes," *International Journal for Numerical Methods in Engineering*, vol. 44, p. 461, 1999.

[15] J. W. Eastwood, W. Arter, N. J. Brealey, and R. W. Hockney, "Body fitted electromagnetic PIC software for use on parallel computers," *Computer Physics Communications*, vol. 87, p. 155, 1995.

[16] T. Westermann, "Particle-in-Cell simulations with moving boundaries – Adaptive mesh generation," *Journal of Computational Physics*, vol. 114, p. 161, 1994.

[17] G. Lapenta, "Automatic adaptive multi-dimensional Particle-in-Cell," *Advanced Methods for Space Simulations*, p. 61, 2007.

[18] D. Seldner and T. Westermann, "Algorithms for interpolation and localization in irregular 2d meshes," *Journal of Computational Physics*, vol. 79, p. 1, 1988.

[19] T. Westermann, "Localization schemes in 2d boundary-fitted grids," *Journal of Computational Physics*, vol. 101, p. 307, 1992.

[20] K. J. Bowers, B. J. Albright, L. Yin, B. Bergen, and T. J. T. Kwan, "Ultra-high performance three-dimensional electromagnetic relativistic kinetic plasma simulation," *Physics of Plasmas*, vol. 15, p. 055703, 2008.

[21] G. Lapenta, F. Iinoya, and J. U. Brackbill, "Particle-in-Cell simulation of glow discharges in complex geometries," *IEEE Transactions on Plasma Science*, vol. 23, no. 4, p. 769, 1995.

[22] J. L. Vay, P. Colella, P. McCorquodale, B. V. Straalen, A. Friedman, and D. P. Grote, "Mesh refinement for Particle-in-Cell plasma simulations: Applications to and benefits for heavy ion fusion," *Laser and Particle Beams*, vol. 20, p. 569, 2002.

[23] J.-L. Vay, P. Colella, J. W. Kwan, P. McCorquodale, D. B. Serafini, A. Friedman, D. P. Grote, G. Westenskow, J.-C. Adam, A. Héron, and I. Haber, "Application of adaptive mesh refinement to Particle-in-Cell simulations of plasmas and beams," *Physics of Plasmas*, vol. 11, p. 2928, 2004.

[24] G. Lapenta and J. U. Brackbill, "Dynamic and selective control of the number of particles in kinetic plasma simulations," *Journal of Computational Physics*, vol. 115, p. 213, 1994.

[25] G. Lapenta, "Particle rezoning for multidimensional kinetic particle-in-cell simulations," *Journal of Computational Physics*, vol. 181, p. 317, 2002.

[26] D. del Castillo-Negrete, D. A. Spong, and S. P. Hirshman, "Proper orthogonal decomposition methods for noise reduction in particle-based transport calculations," *Physics of Plasmas*, vol. 15, p. 092308, 2008.

[27] F. Assous, T. P. Dulimbert, and J. Segré, "A new method for coalescing particles in pic codes," *Journal of Computational Physics*, vol. 187, p. 550, 2003.

[28] D. R. Welch, T. C. Genoni, R. E. Clark, and D. V. Rose, "Adaptive particle management in a paricle-in-cell code," *Journal of Computational Physics*, vol. 227, p. 143, 2007.

[29] G. L. Delzanno, L. Chacón, J. M. Finn, Y. Chung, and G. Lapenta, "An optimal robust equidistribution method for two-dimensional grid generation based on Monge-Kantorovich optimization," *Journal of Computational Physics*, vol. 227, p. 9841, 2008.

[30] A. M. Winslow, "Numerical solution of the quasilinear Poisson equation in a nonuniform triangle mesh," *Journal of Computational Physics*, vol. 2, p. 149, 1967.

[31] J. W. Eastwood, "Particle simulation methods in plasma physics," *Computer Physics Communications*, vol. 43, p. 89, 1986.

[32] H. Okuda and C. K. Birdsall, "Collisions in a plasma of finite-size particles," *Physics of Fluids*, vol. 13, p. 2123, 1970.

[33] A. B. Langdon and C. K. Birdsall, "Theory of plasma simulation using finite-size particles," *Physics of Fluids*, vol. 13, p. 2115, 1970.

[34] J. M. Dawson, "Particle simulation of plasmas," *Reviews of Modern Physics*, vol. 55, p. 403, 1983.

[35] G. Lapenta, "Description of the PIC method and its use." `https://perswww.kuleuven.be/~u0052182/teaching.html`.

[36] E. Cormier-Michel, B. A. Shadwick, C. G. R. Geddes, E. Esarey, C. B. Schroeder, and W. P. Leemans, "Unphysical kinetic effects in particle-in-cell modeling of laser wakefield accelerators," *Physical Review E*, vol. 78, p. 016404, 2008.

[37] J. J. Monaghan, "Extrapolating b-splines for interpolation," *Journal of Computational Physics*, vol. 60, p. 253, 1985.

[38] H. X. Vu and J. U. Brackbill, "CELEST1D: An implicit, fully-kinetic model for low-frequency, electromagnetic plasma simulation," *Computer Physics Communications*, vol. 69, p. 253, 1992.

[39] G. F. Carrier and C. E. Pearson, *Partial Differential Equations: Theory and Technique.* Academic Press, Boston, 1988.

[40] J. Shewchuck, "An introduction to the conjugate gradient method without the agonizing pain," Tech. Rep. CMU-CS-TR-94-125, Carnegie Mellon University, 1994. `http://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.ps`.

[41] A. M. Winslow, "Adaptive mesh zoning by the equipotential method," Tech. Rep. UCID-19062, Lawrence Livermore National Laboratory, 1981.

[42] V. D. Liseikin, *Grid Generation Methods.* Springer-Verlag, Berlin, Heidelberg, New York, 1999.

[43] R. C. Buck, *Advanced Calculus.* McGraw-Hill Book Company, New York, 1965.

[44] H. Goldstein, *Classical Mechanics.* Addison-Wesley, Massachusetts, 1965. p. 239-245.

[45] J. M. Finn and L. Chacón, "Volume preserving integrators for solenoidal fields on a grid," *Physics of Plasmas*, vol. 12, p. 054503, 2005.

[46] P. J. Roache, "Code verification by the method of manufactured solutions," *Journal of Fluids Engineering*, vol. 127, p. 4, 2002.

[47] A. B. Langdon, "Effects of the spatial grid in simulation plasmas," *Journal of Computational Physics*, vol. 6, p. 247, 1970.

[48] H. Okuda, "Nonphysical noises and instabilities in plasma simulation due to a spatial grid," *Journal of Computational Physics*, vol. 10, p. 475, 1972.

[49] N. A. Krall and A. W. Trivelpiece, *Principles of Plasma Physics*. San Francisco Press, San Francisco, 1986.

[50] L. Chacon. Private communication, June 2010.

[51] D. Tskhakaya and R. Schneider, "Optimization of PIC codes by improved memory management," *Journal of Computational Physics*, vol. 225, p. 829, 2007.

[52] V. K. Decyk, S. R. Karmesin, A. deBoer, and P. C. Liewer, "Optimization of Particle-in-Cell codes on reduced instruction set computer processors," *Computers in Physics*, vol. 10, p. 290, 1996.

[53] K. J. Bowers, "Accelerating a Particle-in-Cell simulation using a hybrid counting sort," *Journal of Computational Physics*, vol. 173, p. 393, 2001.

[54] T. MacFarland, H. M. P. Couchman, F. R. Pearce, and J. Pichlmeier, "A new parallel P^3M code for very large-scale cosmological simulations," *New Astronomy*, vol. 3, p. 687, 1998.

[55] R. J. Thacker and H. M. P. Couchman, "A parallel adaptive P^3M code with hierarchical particle reordering," *Computer Physics Communications*, vol. 174, p. 540, 2006.

[56] D. V. Anderson and D. E. Shumaker, "Hybrid Ordered Particle Simulation (HOPS) code for plasma modelling on vector-serial, vector-parallel, and massively parallel computers," *Computer Physics Communications*, vol. 87, p. 16, 1995.