Development of a Collisionless Plasma Kinetic Solver and an Investigation of One-Dimensional Plasma Waves and Instabilities

Daniel W. Crews

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science in Aeronautics & Astronautics

University of Washington

2018

Committee:

Uri Shumlak, Chair

Brian A. Nelson

Program Authorized to Offer Degree:
Aeronautics and Astronautics

University of Washington

Abstract

Development of a Collisionless Plasma Kinetic Solver and
an Investigation of One-Dimensional Plasma Waves and Instabilities

Daniel W. Crews

Chair of the Supervisory Committee:
Professor Uri Shumlak
Aeronautics and Astronautics

Several classic problems in collisionless plasma kinetics are treated by solving the one-dimensional multi-species Vlasov-Poisson system with a BGK collision operator. Calculations are done using a second order discontinuous Galerkin algorithm on a high resolution structured 1D1V grid, with speed and efficiency enabled by GPU parallelization. The numerical method and programming implementation are explained in detail with the intention of replication by interested parties. Elements of collisionless kinetic plasma theory are presented such as linearized Landau damping theory, electrostatic stability, mechanical resonance in a system with mean field interaction, and the phase mixing principle. Subjects of simulation and analysis include: linear and nonlinear Landau damping of electron plasma waves, plasma streaming instabilities, ion interaction with electron holes, and collisional interaction with Landau damping. General takeaways of electrostatic plasma instability are explored such as the ideas of collisionless heating and current drive.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

I sincerely thank my advisor, Professor Uri Shumlak, for his support and encouragement in pursuing this research, as well as Professor Brian Nelson for a keen eye in editing this document. I also appreciate and thank my fellow group members of the UW's Computational Plasma Dynamics laboratory for their enthusiasm for plasma physics and their friendship. That is, thanks to Iman Datta, Whitney Thomas, Andrew Ho, Aria Johansen, Sina Taheri, and Yu Takagaki. Additional thanks to Eric Meier for great discussions and insights into the physics of plasmas. Broad appreciation goes out to the Department of Aeronautics and Astronautics for their help and support throughout my time here at UW, and to South Puget Sound Community College for making it all happen.

DEDICATION

To my family and friends.

Chapter 1
INTRODUCTION

This thesis is composed of two parts, expositing first the development of a parallelized finite element code for solution of the Vlasov-Poisson system and second applying the program to the study of one-dimensional electrostatic plasma waves and instabilities. Within this introductory chapter is an intuitive development of the system of equations under study and an explanation of the importance of its study. Following this, the second chapter develops the numerical methods used for approximate solution of the Vlasov-Poisson system and is intended to be used as a tutorial for those interested in developing a similar code. Finally, the third chapter contains basic collisionless plasma kinetic theory, simulation of quintessential plasma instabilities and wave processes, and discussion of those essential principles of plasma physics implied by the simulation and theory.

1.1 Collective Interaction in Plasma Physics

The remarkable property which distinguishes the dynamics of plasma from those of a neutral fluid is the coherent behavior of the plasma's constituent particles on a microscopic level. This phenomenon, collective behavior, is a result of electromagnetic interactions between plasma particles and the bulk distribution and motion of charges. It is most important in the collisionless regime of plasma dynamics. A plasma's collective modes of motion, combined with microscopic collision physics, result in fluid motions like those of a gas or liquid. Yet when plasma parameters are right for strong collective interaction, nonequilibrium processes such as collisionless wave damping or one-dimensional wave turbulence occur which are unusual from the perspective of neutral fluid physics. This section provides a summary of the nature of collective behavior in collisionless and weakly collisional plasmas.

1.1.1 Collisionless plasma in nature and the laboratory

Plasma may be thought of as a statistical system (that is, of a great number) composed of freely moving charged particles [13]. For an electrostatic system, the associated electric potential $\phi(\vec{r})$ is determined through Poisson's equation,

$$\nabla^2\phi = -\frac{\rho_c}{\epsilon_0} \tag{1.1}$$

where the charge density $\rho_c = \sum_\alpha Z_\alpha n_\alpha$ is found through summation of the density n_α of charge Z_α for each species α (e.g. electrons, protons) within the plasma. The electric field is related to the potential by $\vec{E} = -\nabla\phi$.

The potential ϕ within a plasma is not given by a simple sum due to each charge in the plasma as if each were within a vacuum (such as that of an electron in free space, $\phi(\vec{r}) = -\frac{e}{4\pi\epsilon_0|r|}$). It is frequently shown in plasma physics that when a test charge is inserted into a plasma, its potential is shielded by the response of other charges. The shielded potential of a test electron is given by

$$\phi(r) = \frac{-e}{4\pi\epsilon_0 r}\exp\Big(-\frac{r}{\lambda_D}\Big). \tag{1.2}$$

Here the Debye length

$$\lambda_D = \sqrt{\frac{\epsilon_0 T_{e,eV}}{n_e e}} \tag{1.3}$$

is the potential shielding radius of individual plasma particles [5]. In other words, a charge at a distance greater than several λ_D from another does not directly experience acceleration due to the other's presence. The region of shielded potential surrounding an electron out to a Debye radius is referred to as its Debye sphere. Within an electron's Debye sphere, its Coulomb interaction with electrons is primarily binary or two-body in nature. Scattering results from this two-body interaction.

For example, consider an electron as it moves into the Debye sphere of another electron, as depicted in Fig. 1.1. The two-body Coulomb interaction between the two electrons scatters the incident particle. Typically the scattering angle is quite small. As an electron traverses

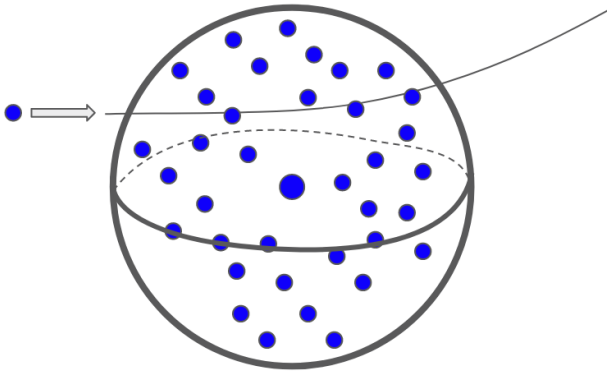


Figure 1.1: Cartoon of an electron penetrating the Debye sphere of another electron and scattering as a result of their Coulomb interaction. In a collisionless plasma, charges transit the Debye sphere of other charges without any change in trajectory.

many Debye spheres, the accumulation of these scattering events produces a large angle deviation from some original trajectory, which is termed a particle collision. The mean free path between large angle collisions is commonly estimated as

$$\lambda_{mfp} \sim \frac{\epsilon_0 T_{e,eV}^2}{\phi_e n_e \log(\Lambda)} \tag{1.4}$$

where $\phi_e = e^2/4\pi\epsilon_0$ are the constants from the Coulomb force law and $\Lambda = 9N_D$ is known as the plasma parameter, with $N_D = \frac{4}{3}n\pi\lambda_D^3$ the number of electrons in a sphere of Debye length radius [8].

Beyond the Debye sphere, the average potential of all screened electrons contributes to a mean field $\langle E \rangle$ which determines subsequent evolution of trajectories based on the state of the plasma as a whole. In other words, if the discreteness of particles is smoothed out by taking a spatial average over small volumes to yield an average charge density $\langle \rho_c \rangle$, the potential from solution of Eqn. 1.1 gives the mean field $\langle E \rangle$. The mean field is responsible for collective modes of plasma motion.

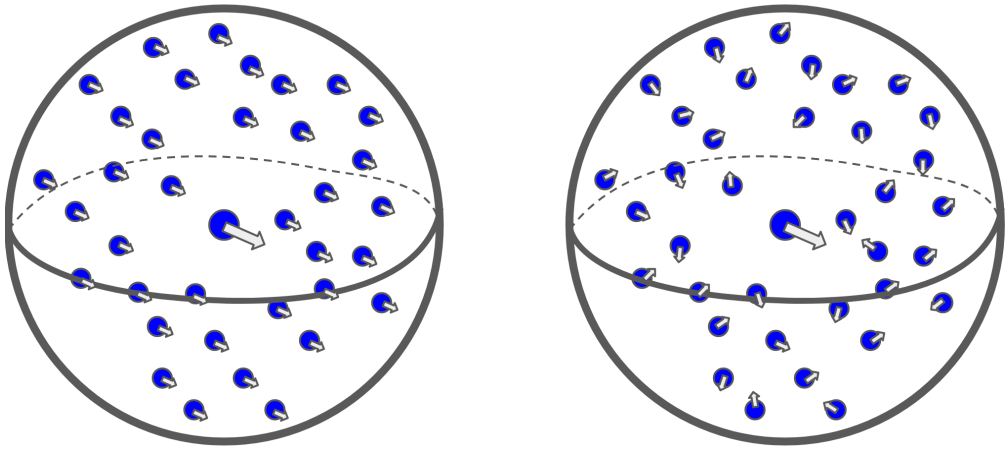


Figure 1.2: Comparison of electric field vectors experienced by electrons within a representative Debye sphere. The left cartoon depicts a collisionless plasma while the right depicts a highly collisional one.

In general, the electric field vector which accelerates charges may be split,

$$\vec{E} = \langle \vec{E} \rangle + \delta\vec{E}, \tag{1.5}$$

where $\delta\vec{E}$ represents microscopic fluctuations in the field due to discrete shielded electrons, which is responsible for scattering. The importance of the two fields may be determined through comparison of the Debye length and mean free path, given by Eqns. 1.3 and 1.4. For many plasmas, the Coulomb logarithm may be estimated as $\log\Lambda \sim 1 - 20$ [13]. Then comparing the length scales,

$$\frac{\lambda_{mfp}}{\lambda_D} \sim \frac{T_{e,eV}^{3/2}}{n_e^{1/2}}. \tag{1.6}$$

An ideal collisionless plasma ($\lambda_{mfp}/\lambda_D \to \infty$) is one in which charges transit the Debye spheres of other charges without any change in trajectory. In other words, the microscopic field $\delta\vec{E}$ in the splitting Eqn. 1.5 is neglected. Plasma is seen to become collisionless as the temperature becomes high or the plasma becomes more rarefied.

Figure 1.2 illustrates the difference in microscopic fields felt by an ideal collisionless plasma and a highly collisional one. The left image shows how in the collisionless limit,

each charge is accelerated in concert by the mean field $\langle E \rangle$ with no fluctuations $\delta\vec{E}$ due to neighboring charges. It's simple to see how this produces collective motion. The right image shows the randomizing effect of the fields due to discrete charges, where the mean field may be a small component of acceleration which is significant only on average over many collisions.

1.1.2 Phase space mechanics and the collisionless kinetic equation

Recall the mechanics of one-dimensional motion of a single particle of unit mass in the classical picture. The equation of motion may be written as the system of first order equations

$$\dot{x} = v \tag{1.7}$$

$$\dot{v} = F(x) \tag{1.8}$$

where x is the particle's position, v its velocity, and $F(x)$ the force vector acting upon it. The particle's state may be depicted geometrically as a succession of coordinates in the (x, v) plane, called the phase plane. Defining the coordinates as one vector $\vec{r} = [x, v]$, the equation of motion may be written as the first order equation $\dot{\vec{r}} = \vec{\mathcal{F}}$ where the flux, defined as $\vec{\mathcal{F}} = [v, F(x)]$, describes a vector field in phase space.

The flux vector field is analogous to the velocity field $\vec{u}$ of a fluid flow. Following the fluid analogy, the flux vector $\vec{\mathcal{F}}$ may be plotted in the phase plane to reveal the streamlines which a particle will follow if the flux is constant in time. This is known as the system's phase portrait. For example, if the coordinate x represents the angular deflection of the pendulum then the classic nonlinear pendulum has the force $F(x) = -\sin(x)$. Figure 1.3 depicts the phase portrait of the simple nonlinear oscillator.

Particles follow the streamlines of the phase portrait, thereby determining the system's evolution. The fluid analogy is now brought full circle: the motion of a continuous region of phase space under the action of the streamlines is a fluid deformation, termed a phase flow. Each point of the region represents the state of the same particle under different initial conditions. Figure 1.4 demonstrates the flow of phase fluid initially placed within the

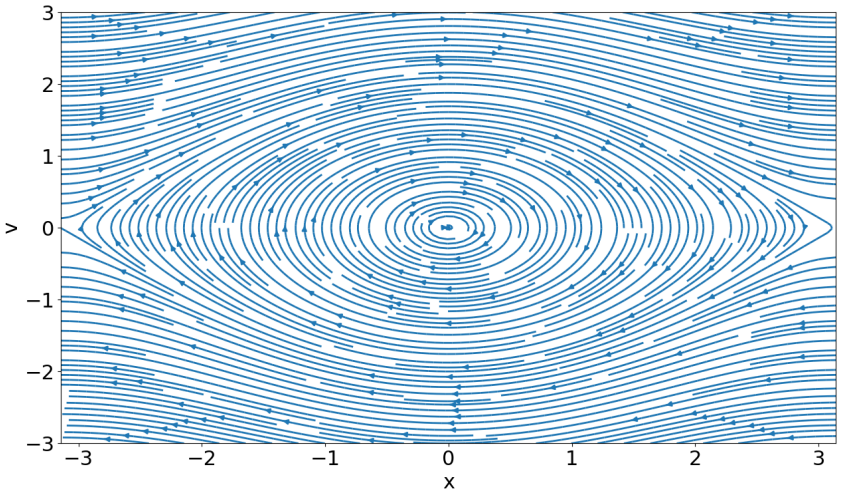


Figure 1.3: Streamlines of the flux function $\vec{\mathcal{F}} = [v, -\sin(x)]$ in the phase plane (x, v).

separatrix of Fig. 1.3. The phase flow is always analogous to that of an incompressible fluid because the flux divergence is zero, $\nabla \cdot [v, F(x)] = \frac{\partial v}{\partial x} + \frac{\partial F(x)}{\partial v} = 0$. A good reference for the mathematics of phase space mechanics is contained in [1].

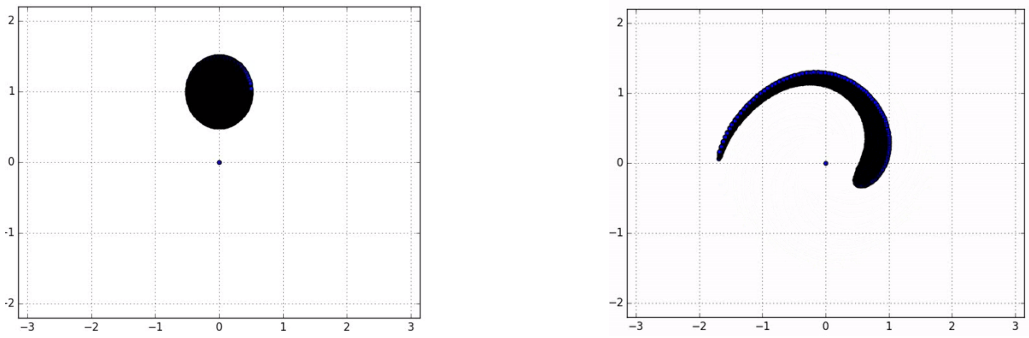
Suppose that the phase fluid density is given by a function $f(x, v, t)$, where t is the time parameter. This function may represent, for example, the probability that the particle satisfying the equation of motion, Eqns. 1.7 and 1.8, has a given initial condition (x_0, v_0) at time $t = t_0$. In this case the function $f(x, v, t)$ is called the single particle probability density. Because any instantiation of the particle cannot leave the phase plane, the probability density will be conserved. In other words, it satisfies a conservation law

$$\frac{\partial f(x, v, t)}{\partial t} = -\nabla \cdot (f(x, v, t)\vec{\mathcal{F}}). \tag{1.9}$$

Because of the flow's incompressibility, $\nabla \cdot (f\vec{\mathcal{F}}) = f(\nabla \cdot \vec{\mathcal{F}}) + \vec{\mathcal{F}} \cdot \nabla f = \vec{\mathcal{F}} \cdot \nabla f$. Then

$$\frac{\partial f}{\partial t} = -\Big[v, F(x)\Big] \cdot \Big[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial v}\Big] \tag{1.10}$$

$$\implies \frac{\partial f}{\partial t} + v\frac{\partial f}{\partial x} + F(x)\frac{\partial f}{\partial v} = 0, \tag{1.11}$$



(a) The initial placement of a droplet of phase fluid.

(b) At a later time in the phase flow, the droplet has spread out due to nonlinearity.

Figure 1.4: Illustration of phase flow for the nonlinear pendulum.

so that the probability density satisfies a simple advection equation in the phase space (x, v). An equation such as Eqn. 1.11 is known as a kinetic equation.

Liouville's theorem and the Vlasov-Poisson system

In the more general case of the mechanics of multiple interacting particles the situation becomes more complex. For example, with two interacting particles one must fix the initial state of one, say (x_0^0, x_0^0), and consider the probability that the other particle has any initial condition (x_0^1, v_0^1). Then the first particle's initial state is varied and probabilities assigned to the other particle again. In this way the joint probability density is a function of a higher dimensional phase space, i.e. $f(x^0, v^0, x^1, v^1, t)$, and satisfies a conservation law there.

The general result is known as Liouville's theorem, which states that for a system of N interacting particles, their joint probability density f in a $6N + 1$ dimensional phase space satisfies the conservation law

$$\frac{df}{dt} = 0, \tag{1.12}$$

which is one of the most important theorems in classical mechanics. Beginning from this theorem, it is possible to deduce Boltzmann's equation for the ensemble-averaged velocity

distribution function $f(\vec{x}, \vec{v}, t)$ of a gas or plasma,

$$\frac{\partial f}{\partial t} + \vec{v} \cdot \nabla_x f + \frac{1}{m} \vec{F} \cdot \nabla_v f = \mathcal{C}(f). \tag{1.13}$$

The term $\mathcal{C}(f)$ is in general an integral operator representing inter-particle correlations such as Coulomb scattering interactions. Rigorous derivations of the collision operator's form in various regimes are found in references such as [2].

A special case is when correlations are entirely neglected and only the electrostatic potential of the many-body system considered. The distribution function of each species α is found to satisfy Eqn. 1.13 with $\mathcal{C}(f_\alpha) = 0$ and the field potential determined through the zeroth moment of the probability densities of each species. Neglecting correlations is equivalent to the limit in which $\delta\vec{E} = 0$ in the splitting Eqn. 1.5. Using the distribution moment to determine the potential is called the mean field approximation. This result, known as the Vlasov-Poisson system of equations, is the focus of this thesis. For motion restricted to one spatial dimension, it has the form

$$\frac{\partial f_\alpha}{\partial t} + v\frac{\partial f_\alpha}{\partial x} - \frac{Z_\alpha}{m_\alpha}\frac{\partial \phi}{\partial x}\frac{\partial f_\alpha}{\partial v} = 0, \tag{1.14}$$

$$\frac{d^2\phi}{dx^2} = -\frac{1}{\epsilon_0}\sum_\alpha Z_\alpha \int_{-\infty}^{\infty} f_\alpha(x, v, t) dv. \tag{1.15}$$

The potential calculated in this way is said to be determined self-consistently. It has been said that the Vlasov equation is the most important expression in plasma physics, as its study provides insights into the nature of collective motions in plasma and provides a rigorous foundation for the plasma fluid equations and their limitations [17].

1.1.3 Wave-particle interaction and Landau resonance

The most striking difference between the dynamics of neutral fluids and collisionless plasmas is the appearance of non-equilibrium velocity distributions in plasma as a result of collective behavior. The collision operation in the Boltzmann equation, Eqn. 1.13, for neutral gas enforces strict local thermodynamic equilibrium (LTE). That is, the distribution function

$f(x,v)$ has Gaussian or Maxwellian statistics at each position x. In one dimension, a gas with number density $n(x)$, bulk velocity $u(x)$ and thermal velocity $v_{th}(x)$ has a Maxwellian distribution

$$f(x,v) = n(x)\sqrt{\frac{1}{2\pi v_{th}^2(x)}}\exp\Big(-\frac{(v-u(x))^2}{2v_{th}^2(x)}\Big). \tag{1.16}$$

The density, bulk velocity, and thermal velocity are in fact the normalization, mean value, and variance of the distribution. That is, they are successive moments of the probability distribution: $n(x) = \int_{-\infty}^{\infty} f(x,v,t)dv$, $u(x) = \frac{1}{n(x)}\int_{-\infty}^{\infty} vf(x,v,t)dv$, $v_{th}^2(x) = \frac{1}{n(x)}\int_{-\infty}^{\infty}(v-u(x))^2 f(x,v,t)dv$.

In contrast to the neutral fluid statistics, when there is a lack of collisions ($\mathcal{C}(f) = 0$) and a mean field, such as in the Vlasov-Poisson system, there is no driver for the velocity distribution to possess normal statistics. The mean field acts to deform the distribution function and over the course of dynamics a complex structure develops within the phase space. This complex structure is generated through wave-particle resonance, where waves with phase velocity $v_{ph} = \omega/k$ resonate with particles traveling at the same velocity, as the particles sense a wave with a stationary phase through the Doppler shift of the wave frequency. The process is also called Landau resonance in honor of Lev Landau's contributions to the theory of energy exchange between waves and particles.

For example, a planar monochromatic plasma wave will resonate with an initially Maxwellian distribution such as Eqn. 1.16, leading to a flattening of the tail around the phase velocity v_{ph}. The thermal width of the distribution increases as the wave energy is absorbed, so that the plasma is heated through its interaction with the wave. At the same time, the wave is damped as its energy is absorbed. The apparent dissipation of wave energy in a collisionless system is the famous Landau damping phenomenon, a subset of the general concept of Landau resonance. Nonlinear wave-particle resonance processes also produce undamped plasma waves in a collisionless plasma. Simulations of Landau damping and the wave-particle resonance are detailed in Chapter 3, and a good review of Landau damping is found in [19].

1.2 Collisionless Kinetic Modeling

Since the early days of the study of plasma it has been recognized that investigation of the kinetic equation is important because of the commonly low collision frequency compared to the frequency of plasma oscillations. This section provides a brief overview of the history of kinetic plasma modeling, as well as a comparison between different approaches to the solution of the kinetic equation.

1.2.1 Brief history of analysis

The collisionless kinetic equation has been used for the study of stellar dynamics since the early 20th century with the work of James Jeans [9]. In 1938, A.A. Vlasov applied the kinetic equation to "clouds" of electrons in a plasma, which interacted self-consistently through Poisson's equation and did not undergo collisions with one another. Following investigation of this model, L.D. Landau used Fourier-Laplace transform and a clever contour integral to show that the energy of Langmuir waves (or electron plasma waves) was damped in a collisionless plasma, and obtained the theoretical linear damping rate [14].

There was recognition in these early studies that the interaction of particles with waves would be important in a collisionless regime. In 1949 D. Bohm and E.P. Gross obtained the dispersion relation for simple Langmuir waves by neglecting the nonlinear effect of particle trapping [4]. After further study, in 1957 Bernstein, Greene, and Kruskal derived exact solutions to the fully nonlinear Vlasov-Poisson system wherein "essentially arbitrary traveling wave solutions can be constructed" based on the idea of wave-trapped particles within plasma waves inhibiting the Landau damping effect [12]. The idea is that Landau damping can saturate in the nonlinear regime. This helped to explain why large-amplitude Langmuir waves persist despite the collisionless damping phenomenon.

Besides damping of plasma waves, there was the question of wave emission due to plasma instability. It was discovered that if chunks of collisionless plasma traveling with different velocities occupied the same space then oscillations were excited. It turned out that this

instability was also due to a wave-particle resonance process, and a condition for instability depending on the shape of the distribution function for interpenetrating populations of plasma was given by O. Penrose in 1960 [18]. All of the above concepts are further developed with supporting examples in Chapter 3.

1.2.2 Modeling by phase space sampling: the particle-in-cell (PIC) method

Many important results were obtained by linearization in the fruitful early years of analysis of the collisionless kinetic equation. Because of the importance of nonlinear effects and the variety of collisionless plasma instabilities, computer simulation of plasma dynamics developed in the 1960's. The first method by which this was done was the particle-in-cell (PIC) method, which pushed particles representing clouds of plasma around the phase space, evolving their states by means of their equations of motion, and determining the electric field self-consistently through discretized solution of Poisson's equation. The classic reference for this method is found in [3].

The method led to breakthroughs in understanding of nonlinear processes in collisionless plasma, and it was relatively easy to extend the method to higher dimensions. However, the method suffers from error due to statistical noise, as it consists of a discrete sampling of the continuous distribution function. Such statistical noise can prevent the PIC method from converging to the Vlasov equation's solution, as the fine structure of phase space which develops in such solutions is erased by noise. Unrealistic physics can develop such as the decay of stable phase space structures.

1.2.3 Modeling by phase space discretization: the continuum-kinetic method

More accurate study of electrostatic plasma instabilities is enabled by using Eulerian methods to directly solve the system Eqns. 1.14 and 1.15, such as is done in computational fluid dynamics (CFD). The phase space is discretized and a functional approximation to the continuum solution of the Vlasov equation is obtained, rather than a statistical sampling. For this reason, the method is sometimes known as the continuum-kinetic approach. Early

methods used finite difference and finite volume methods to solve the Vlasov-Poisson system. Modern programs use higher order finite element methods, which also lend themselves well to rapidly developing parallel computing techniques.

The capability of such codes was initially low compared to those using the PIC method, particularly in high dimensional phase space. For example, solving the Vlasov-Poisson system in two dimensions requires discretization of a four-dimensional (2D2V) phase space in the continuum-kinetic approach, while only two dimensions need be discretized in the PIC method. For the same processing power, much higher resolution may be obtained in the PIC method. As parallel computing capability improves, the continuum kinetic approach is gaining popularity as a practical method, along with the need for solutions to the Vlasov-Poisson system with better convergence properties.

1.3 Vlasov1D1V: A Python Program for Fast Plasma Physics Calculations

This thesis describes development of an Eulerian Python code to solve the Vlasov-Poisson system using GPU parallelization for rapid calculations at high resolution. The program was developed to study one-dimensional waves and instabilities in collisionless plasma. Chapter 2 contains a detailed description of how the program works, while this section provides a brief overview of why such a program is useful and its notable features.

1.3.1 Utility of one-dimensional calculations in plasma physics

Plasma undergoes highly complex three-dimensional motions in almost all practical applications. For this reason, one-dimensional calculations such as the ones found in this thesis are most useful for the development of intuition for fundamental plasma processes. Analogy can be made to more complicated processes in magnetized plasma or for higher-dimensional problems, in particular for the notions of Landau damping, wave dispersion, and multiple-species effects in a collisionless plasma. Results in this thesis should be viewed with the perspective that actual plasma motions are three-dimensional and involve a spectrum of wave frequencies, wavevectors, and amplitudes.

1.3.2 Solving a coupled hyperbolic-elliptic system

The Vlasov-Poisson system of Eqns. 1.14 and 1.15 are classified in the theory of PDEs as hyperbolic and elliptic, respectively. As such, they generally require different methods for their solution, although there are finite element methods which provide a unitary solution framework. This thesis uses the Nodal Discontinuous Galerkin (NDG) finite element method for solution of the hyperbolic Vlasov equation and a finite difference method for solution of the elliptic Poisson equation. The NDG method is limited to the solution of hyperbolic equations.

The finite difference method is most simple when used on a grid with even node spacing. To use an NDG method of order higher than three, when nodal placement becomes uneven, it would be necessary to use a unitary method for solution of Poisson's equation. For future work, one such method is the Hybrid Discontinuous Galerkin (HDG) finite element method which is capable of solving elliptic equations by casting them as a system of first order equations.

1.3.3 Parallelization using CUDA in Python

Graphics processing units (GPUs) are now prolifically used in scientific computing applications. They contain thousands of processors, turning any desktop computer into a supercomputer compared to the machines of the past. Such devices enable high resolution solution to the Vlasov-Poisson system in a timely manner. Further, packages such as Numba for the Anaconda Python platform have been developed which significantly ease the implementation of CUDA code for Nvidia GPU parallelization. Simple GPU parallelization using Python is a focus of this thesis and use is explained in detail in Chapter 2.

Chapter 2

DEVELOPMENT OF A 1D1V KINETIC SOLVER

To investigate one-dimensional electrostatic motions in collisionless plasma a Python program was developed which implements an algorithm to solve the Vlasov-Poisson (VP) system in one spatial and one velocity dimension (1D1V). Finite difference, finite element, and Runge-Kutta methods were applied for this purpose. Within this chapter the numerical methods used to solve the component equations of the VP system, and their Python implementations, are described with the intention that a similar program could be created by a person interested in studying collisionless plasma physics through computer simulation.

Working with a Vlasov solver is beneficial for building intuition for collisionless plasma phenomena as it provides one with a tool for direct geometric experience with a seemingly abstract physics. For example, using this tool it becomes a simple matter to study the evolution of plasma velocity space instabilities or wave-particle interaction processes. It's hoped that the algorithms laid out in this chapter will assist future endeavors in the study of plasma physics by easing the difficulty in developing a tool for the visual study of plasma physics.

2.1 Program Overview

Vlasov1D1V is a parallelized two species continuum kinetic solver of the VP system with simple collision and particle ionization source models. By two species it is meant that the kinetic equations of the two typical plasma species, electrons and ions, are solved sequentially. Simulating both species permits study of the full dynamics of collisionless plasma, as well as cases of intermediate self-collisionality of either species. Having two species kinetically resolved can be of particular importance in the study of ion dynamics, as many analytical

results or prior studies have assumed strictly fluid electron behavior (the Boltzmann electron model).

The algorithms used in solution of the VP system consist of a nodal Discontinuous Galerkin finite element method for spatial solution of the Vlasov equation, a Runge-Kutta timestepping method for temporal evolution, and a finite difference method for solution of Poisson's equation. The mathematics behind these methods and their implementations are detailed in the relevant sections below. Details related to parallelization of the nodal finite element method with GPU on Python receieve their own sections as well.

2.1.1 Algorithm summary

The Vlasov1D1V program has an algorithmic structure similar to most computational fluid dynamics programs. For illustration, Fig. 2.1 presents a simple flow chart of the program's algorithmic progression. During an initialization phase, preliminary work is done such as loading the input parameters, setting up data arrays (the variables, grid, etc.), loading arrays for implementing numerical methods, determining the appropriate time step to take, and calculating the initial electric field given the specified initial condition. Following initialization the main loop begins, consisting of a timestep using the finite element method, moments taken using Gauss-Lobatto numerical quadrature, the Poission equation solved to determine the self-consistent electric field, and determination of an appropriate time increment Δt to preserve numerical stability. Data are saved every N timesteps as specified in the input file.

Many of these calculations benefit from parallelization. For example, the timestepping phase requires calculation of a matrix product at every point in the phase space domain, while the moment integrations consist of reduction operations along the velocity axis at each position x. Methods to ensure that these calculations are efficient and parallel are presented in the relevant sections.

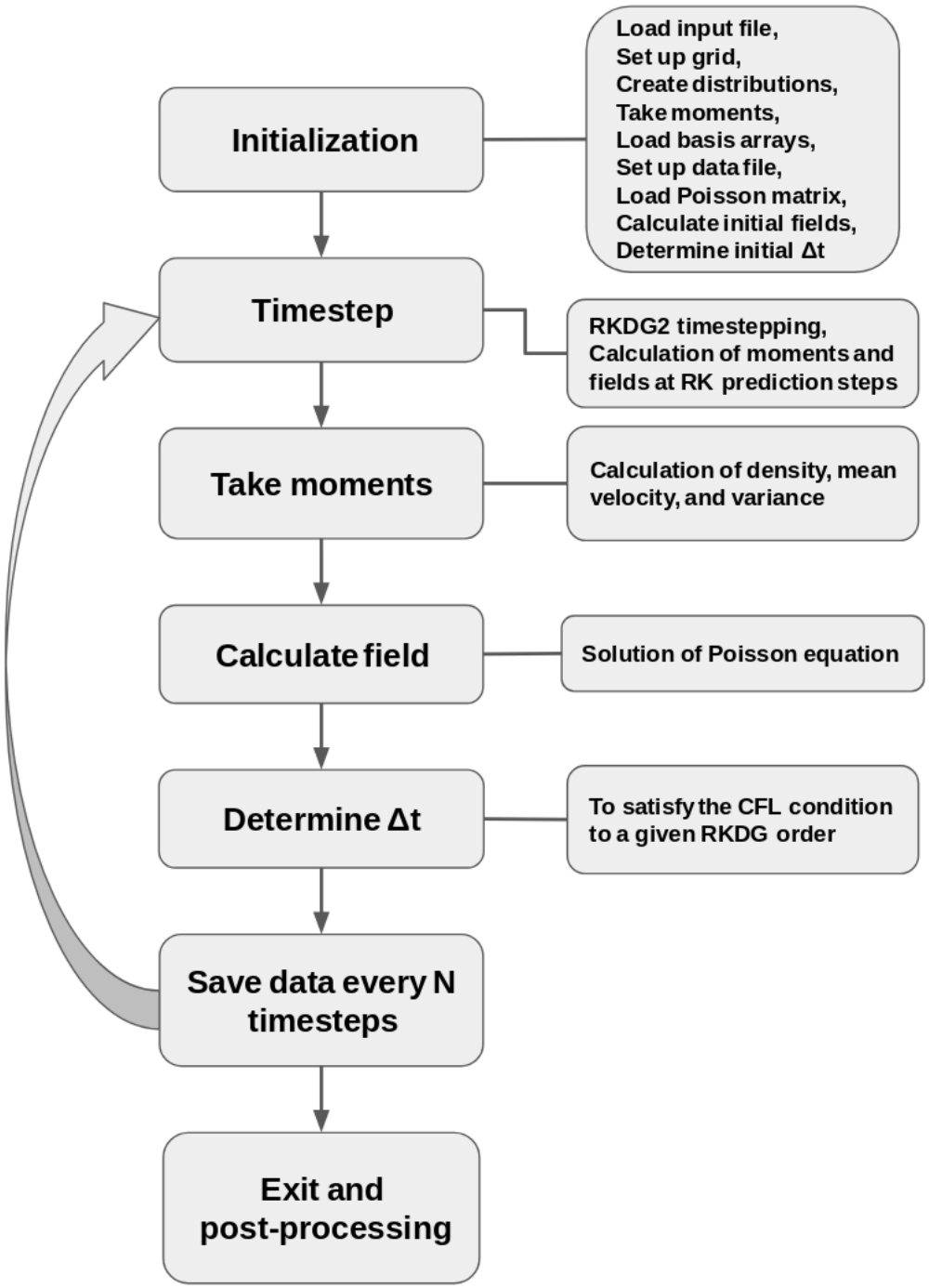


Figure 2.1: Flow chart of the Vlasov1D1V algorithm. Following program initialization, the main timestepping loop is entered which uses the Runge-Kutta Discontinuous Galerkin method for solution update, takes moments by Gauss-Lobatto quadrature, solves the Poisson equation by finite difference method, and saves data when appropriate. The program quits following a specified number of timesteps.

2.2 Nodal Discontinuous Galerkin Method for Hyperbolic Problems

Finite element methods for solving PDEs have been developed over decades into a broad class of methods for different kinds of problems. Many finite element methods enforce continuity of the approximate solution across all elements. This continuity leads to a necessary matrix inversion in order to obtain a global solution, which makes a parallel implementation of the solution method difficult. If the continuity requirement is relaxed such that a problem is solved locally within each element rather than one global problem being obtained, then parallelization of the method becomes possible.

This section presents the well-developed Nodal Discontinuous Galerkin (NDG) finite element method which has the strong advantage of requiring only calculations local to an element. A recommended wealth of information on the method is contained in Ref [10]. The local property is important because in GPU parallelization schemes, typically only data close by the point being updated can be accessed by memory within the solution kernel. The local character of NDG make parallelization simple to achieve, as shown later on.

2.2.1 Finite element spatial discretization on structured domains

The basic idea of the finite element method is that a continuous domain D defined by $D = \{(x, v)|x \in [a, b], v \in [c, d]\}$ for a problem involving a dependent variable $f(x, v, t)$ is divided into many small finite elements $D^\alpha = \{(x, v)|x \in [a^\alpha, b^\alpha], v \in [c^\alpha, d^\alpha]\}$ where an approximate solution $f^\alpha(x, v, t)$ is given within each element using some sort of polynomial representation. An illustration of the domain discretization is shown in Fig. 2.2.

Following domain discretization, choice of representation of f^α is necessary. There are two common choices in the finite element method for this choice. One is the *modal* representation where the approximate solution is represented in terms of time-dependent expansion coefficients. The other is a *nodal* representation where the time-dependent solution values at given spatial positions called nodes are combined with Lagrange polynomial interpolation functions.

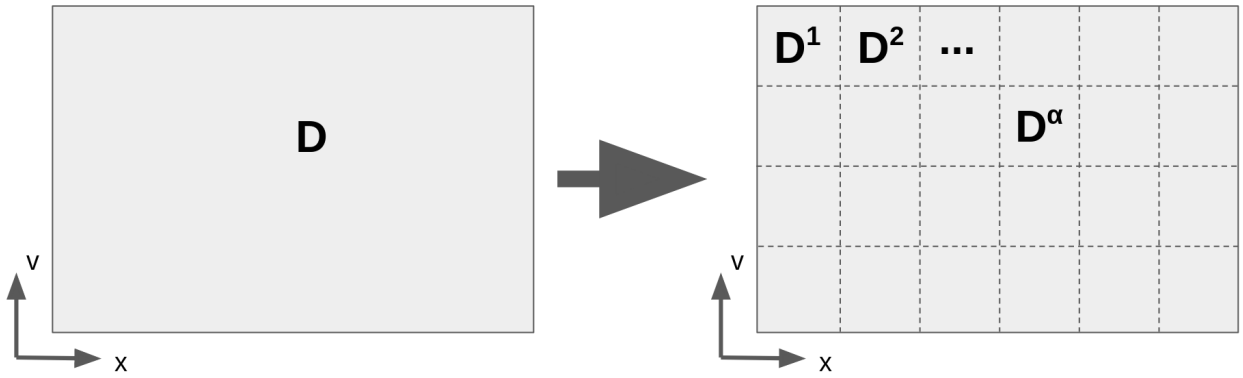


Figure 2.2: Division of a two-dimensional domain D into N_E nonoverlapping square elements D^α. Each element is identical so that the discretization is a structured one.

The program under discussion utilizes a nodal representation. One reason for this choice is that, as seen later, in order to calculate the surface integrals between elements typically it is required to know the approximate solution at the surface coordinates of an element. In a modal representation, all modal coefficients are necessary to specify the edge values, whereas in the nodal case, only one coefficient is required to obtain the edge value. This results in less memory usage when evaluating the surface integral terms.

In nodal DG, determination of the node locations is a problem considered to be constrained by the Lebesgue constant $\Lambda(r) = \max \sum_{i=1}^{N_p} \left|\ell_i(r)\right|$ of the interpolation polynomials is minimized [10]. It turns out that minimization of this constant for nodes on a line segment $x \in [-1, 1]$ is given by the set of Legendre-Gauss-Lobatto (LGL) quadrature points, or the N roots of the polynomial

$$g(x) = (1 - x^2)\widetilde{P}'_{N-1}(x). \tag{2.1}$$

These points are given for several orders by $N = 1$, $x_0 = 0$, $N = 2$, $x_0 = -1$, $x_1 = 1$, $N = 3$, $x_0 = -1$, $x_1 = 0$,$x_2 = 1$, and $N = 4$, $x_0 = -1$, $x_1 = \frac{-1}{\sqrt{5}}$, $x_2 = \frac{1}{\sqrt{5}}$, $x_3 = 1$, etc. Now to obtain nodal locations in a square element rather than a line segment, the node locations arranged in a square product maintain a minimal Lebesgue constant. The nodal coordinates

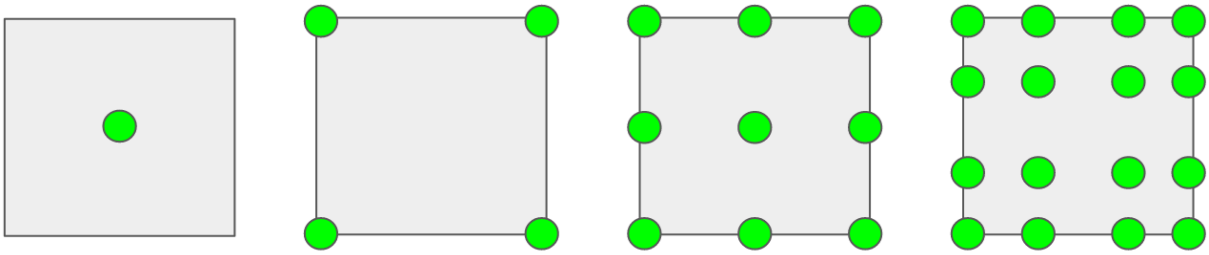


Figure 2.3: Node placement within square elements for first, second, third, and fourth order elements. Node positions along both axes follow from the Legendre-Gauss-Lobatto (LGL) quadrature points of the respective order.

for square elements for the listed orders are shown in Fig. 2.3. The Vlasov1D1V program uses third order elements with nine nodes per element.

2.2.2 Obtaining nodal basis functions from modal ones

A suitable choice of modal basis function is easy to make. A family of orthogonal polynomials will give a simple method, one which will involve products of the form $M^{-1}S$ where M is a diagonal matrix. Yet to use a nodal method, a means of obtaining nodal basis functions for a given set of nodal locations in multiple dimensions is required. These means are illustrated in this section by demonstrating the relationship between the modal and nodal representations. Staying in one dimension for the time being, first note that the modal representation is written

$$f^{\alpha}(x,t) = \sum_{n=1}^{N_p} \hat{f}_n^{\alpha}(t)\psi_n(x) \tag{2.2}$$

up to a desired polynomial order N_p for a given set of modal polynomial basis functions ψ_n and weighting coefficients $\hat{f}_n^{\alpha}(t)$. On the other hand, the nodal representation is given by

$$f^{\alpha}(x,t) = \sum_{i=1}^{N_p} f^{\alpha}(x_i^{\alpha},t)\ell_i(x). \tag{2.3}$$

Recall that the interpolating Lagrange polynomials $\ell_i(x)$ of a set of points x_i have the property that $\ell_i(x_i) = 1$ at the point x_i and $\ell_j(x_i) = 0$ for $i \neq j$. Although in one dimension they are given by

$$\ell_j = \prod_{i=0, i\neq j}^{N} \frac{x - x_i}{x_j - x_i}, \tag{2.4}$$

the general form of interpolation polynomials in higher dimensions is not known. Yet due to the uniqueness of polynomial representation, they may be found for a specified set of nodes through the following construction procedure. Evaluating the approximations $f^\alpha(x,t)$ at the nodal locations x_i yields the expression

$$f^\alpha(x_i^\alpha, t) = \sum_{n=1}^{N_p} \hat{f}_n^\alpha(t)\psi_n(x_i) \tag{2.5}$$

because $\ell_j(x_i) = 0$ for $i \neq j$ and $\ell_i(x_i) = 1$. The right hand side has the form of a matrix-vector product,

$$\sum_{n=1}^{N_p} \hat{f}_n^\alpha(t)\psi_n(x_i) = V\vec{\hat{f}} \tag{2.6}$$

where $V_{ij} = \psi_j(x_i)$ is a matrix of Vandermonde type and $\vec{\hat{f}} = \hat{f}_n$ is the vector of expansion coefficients. Substituting this result into the relating equation

$$f^\alpha(x,t) = \sum_{n=1}^{N_p} \hat{f}_n^\alpha(t)\psi_n(x) = \sum_{i=1}^{N_p} f^\alpha(x_i^\alpha, t)\ell_i(x), \tag{2.7}$$

one obtains

$$\sum_{n=1}^{N_p} \hat{f}_n^\alpha(t)\psi_n(x) = \sum_{n=1}^{N_p} \hat{f}_n^\alpha(t) \sum_{i=1}^{N_p} \psi_n(x_i)\ell_i(x). \tag{2.8}$$

As the polynomial representation of f^α must be unique, we may equate the terms multiplying each expansion coefficient, giving

$$\psi_n(x) = \sum_{i}^{N_p} \psi_n(x_i)\ell_i(x) \tag{2.9}$$

which is equivalent to the matrix-vector equation,

$$\vec{\psi}(x) = V^T\vec{\ell}(x). \tag{2.10}$$

Therefore the Lagrange interpolating polynomials may be determined in terms of the transpose inverse of the Vandermonde matrix. This result holds for higher dimensional interpolating polynomials as well, so that given a set of nodes in the plane $\vec{x}_i$, for example, the interpolating polynomials are given by

$$\vec{\ell}(\vec{x}) = (V^T)^{-1}\vec{\psi}(\vec{x}) \tag{2.11}$$

in terms of the Vandermonde matrix $V_{ij} = \psi_j(\vec{x}_i)$.

Example 1: Calculation of nodal basis for second order elements

Consider the problem of obtaining the interpolating polynomial basis for the second order element of Fig. 2.3, given an isoparametric element with coordinates (ξ_0, ξ_1) and nodal coordinates $\vec{r}_0 = (0,0), \quad \vec{r}_1 = (1,0), \quad \vec{r}_2 = (0,1), \quad \vec{r}_3 = (1,1)$. First a modal basis is chosen. A good choice is the set of Legendre polynomials, a family of orthogonal polynomials, which gives a basis set of

$$\psi_0(\xi_0, \xi_1) = 1, \qquad \psi_1(\xi_0, \xi_1) = \xi_1 \tag{2.12}$$

$$\psi_2(\xi_0, \xi_1) = \xi_0, \qquad \psi_3(\xi_0, \xi_1) = \xi_0\xi_1. \tag{2.13}$$

In order to obtain the *nodal* basis functions, we construct the element's Vandermonde matrix,

$$V_{ij} = \psi_j(\vec{r}_i) = \begin{bmatrix} \psi_0(0,0) & \psi_1(0,0) & \psi_2(0,0) & \psi_3(0,0) \\ \psi_0(1,0) & \psi_1(1,0) & \psi_2(1,0) & \psi_3(1,0) \\ \psi_0(0,1) & \psi_1(0,1) & \psi_2(0,1) & \psi_3(0,1) \\ \psi_0(1,1) & \psi_1(1,1) & \psi_2(1,1) & \psi_3(1,1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}. \tag{2.14}$$

The nodal basis is calculated through use of Eqn. 2.11. The formula $\phi_j(\vec{r}) = V_{ij}^{-T}\psi_i(\vec{r})$ is applied to obtain

$$\vec{\phi} = \begin{bmatrix} \phi_0(\xi_0, \xi_1) \\ \phi_1(\xi_0, \xi_1) \\ \phi_2(\xi_0, \xi_1) \\ \phi_3(\xi_0, \xi_1) \end{bmatrix} = \begin{bmatrix} 1 & -1 & -1 & 1 \\ 0 & 0 & 1 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ \xi_1 \\ \xi_0 \\ \xi_0\xi_1 \end{bmatrix} = \begin{bmatrix} 1 - \xi_0 - \xi_1 + \xi_0\xi_1 \\ \xi_0 - \xi_0\xi_1 \\ \xi_1 - \xi_0\xi_1 \\ \xi_0\xi_1. \end{bmatrix} \tag{2.15}$$

These two-dimensional interpolating polynomials are 1 at their own node and are 0 at other nodes. Calculation of the nodal basis functions for higher order elements is more cumbersome to do by hand. Omitting the calculation, third order elements as in Fig. 2.3 have the nine-member nodal basis set,

$$\begin{bmatrix} \phi_0(\xi_0,\xi_1) \\ \phi_1(\xi_0,\xi_1) \\ \phi_2(\xi_0,\xi_1) \\ \phi_3(\xi_0,\xi_1) \\ \phi_4(\xi_0,\xi_1) \\ \phi_5(\xi_0,\xi_1) \\ \phi_6(\xi_0,\xi_1) \\ \phi_7(\xi_0,\xi_1) \\ \phi_8(\xi_0,\xi_1) \end{bmatrix} = \begin{bmatrix} 4\xi_0^2\xi_1^2 - 6\xi_0^2\xi_1 - 6\xi_0\xi_1^2 + 2\xi_0^2 + 2\xi_1^2 + 9\xi_0\xi_1 - 3\xi_0 - 3\xi_1 + 1 \\ \xi_0(-8\xi_0\xi_1^2 + 8\xi_1^2 + 12\xi_0\xi_1 - 12\xi_1 - 4\xi_0 + 4) \\ \xi_0(4\xi_0\xi_1^2 - 2\xi_1^2 + 6\xi_0\xi_1 + 2\xi_0 + 3\xi_1 - 1) \\ \xi_1(-8\xi_0^2\xi_1 + 8\xi_0^2 + 12\xi_0\xi_1 - 12\xi_0 - 4\xi_1 + 4) \\ 16\xi_0\xi_1(\xi_0\xi_1 - \xi_0 - \xi_1 + 1) \\ \xi_0\xi_1(-8\xi_0\xi_1 + 8\xi_0 + 4\xi_1 - 4) \\ \xi_1(4\xi_1\xi_0^2 - 2\xi_0^2 - 6\xi_0\xi_1 + 2\xi_1 + 3\xi_0 - 1) \\ \xi_0\xi_1(-8\xi_0\xi_1 + 4\xi_0 + 8\xi_1 - 4) \\ \xi_0\xi_1(4\xi_0\xi_1 - 2\xi_0 - 2\xi_1 + 1) \end{bmatrix}. \tag{2.16}$$

The program Vlasov1D1V utilizes third-order nodal elements with the above basis set used as the support for the approximate solution in the nodal representation, as in Eqn. 2.3. Contour plots of this polynomial basis are shown in Fig. 2.4.

2.2.3 The semi-discrete equation and computation of the basis arrays

The Vlasov equation has the form of a hyperbolic PDE in two dimensions. For an evolving scalar variable $f(\vec{x},t)$, given a flux function $\vec{F}(f) = \{vf, \frac{q}{m}Ef\}$ and general source function s, a hyperbolic PDE may be written within the finite element α as

$$\partial_t f + \partial_{x_i} F_i = s \tag{2.17}$$

where the Einstein summation notation is used for summation over repeated indices. Dividing the domain into a number N_E of nonoverlapping finite elements α, the approximate solution f^α will satisfy

$$\partial_t f^\alpha + \partial_{x_i} F_i^\alpha = s^\alpha \tag{2.18}$$

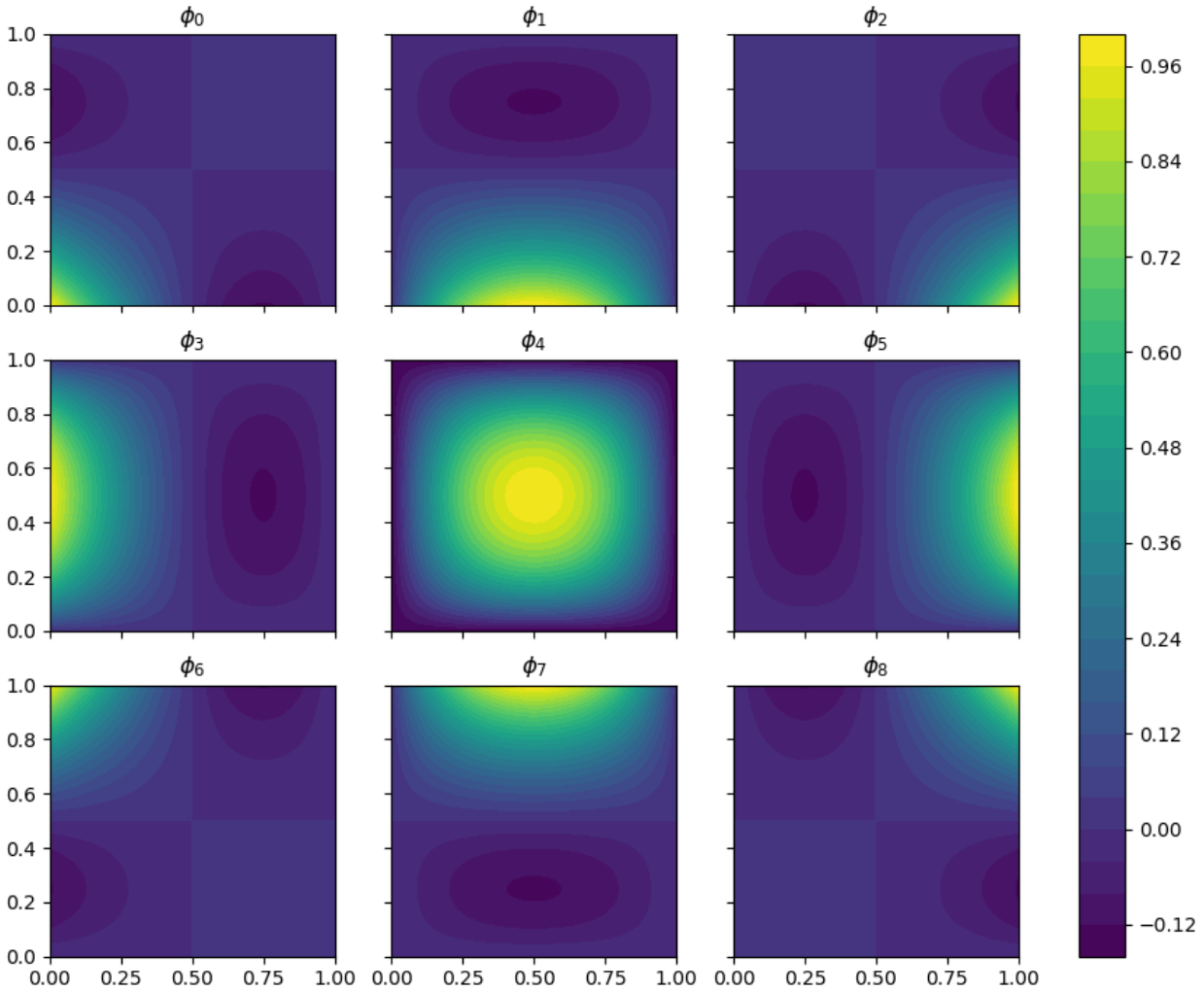


Figure 2.4: Contour plots of the nodal basis functions for a third order square element. Note that the basis functions are 1 at the node location and 0 at other nodes. The interpolating functions between nodes are quadratic, giving fourth order polynomials in two variables ξ_0, ξ_1. The approximate solution $f = \sum_{i=1}^{N_p} f(\xi_i^\alpha)\phi_i(\xi_0, \xi_1)$ is a superposition of these basis functions ϕ_i with weights given by the solution value at the given node.

within each element α. The derivation which follows results in a form which is simple to understand computationally and uses notation found within Sean Miller's PhD dissertation [16]. To keep the calculations standard for each potentially different shaped element, the method will be formulated in a standard coordinate space called the isoparametric space. This is a cartesian space with coordinates ξ_i. We can write the gradient ∂_{x_i} within the element in terms of the Jacobian of the transformation from the isoparametric to the actual element,

$$\partial_t f^\alpha + \partial_{x_i}\xi_j \partial_{\xi_j} F_i^\alpha = s^\alpha. \tag{2.19}$$

Let the Jacobian be denoted $J_{ij} = \partial_{x_i}\xi_j$. Now the conserved variable may be expanded onto a set of nodal basis functions $\phi_i(\vec{\xi})$, so that $f^\alpha(\vec{\xi}, t) = f(x_j^\alpha, t)\phi_j(\vec{\xi})$ where x_j^α are the element's nodal coordinates. This gives

$$\phi_k(\vec{\xi})\partial_t f(x_k^\alpha, t) + J_{ij}\partial_{\xi_j} F_i^\alpha = s^\alpha \tag{2.20}$$

as the resulting PDE. Given that the exact solution satisfies $\partial_t f + \partial_{x_i} F_i - s = 0$, the residual function after approximation is

$$R^\alpha = \phi_k(\vec{\xi})\partial_t f(x_k^\alpha, t) + J_{ij}\partial_{\xi_j} F_i^\alpha - s^\alpha. \tag{2.21}$$

The requirement for the residual to be minimal is that it be orthogonal to a complete space of test functions [10],

$$\int_{D_\alpha} R^\alpha \phi_l dV = 0 \tag{2.22}$$

where the integral is over the isoparametric element D^α and the test functions have been chosen to be the expansion basis ϕ_n. This is called the Galerkin choice of residual weighting function, which is what makes the finite element method a Galerkin method. Substituting the residual, the term $\partial_t f(x_k^\alpha, t)$ may be factored out of the integral over the reference element as it is not a function of the coordinate but rather of the nodal locations,

$$\Big[\int_{D^\alpha} \phi_l\phi_k dV\Big]\partial_t f(x_k^\alpha, t) + J_{ij}\Big[\int_{D^\alpha} \phi_l \partial_{\xi_j} F_i^\alpha dV\Big] - \int_{D^\alpha} \phi_l s^\alpha dV = 0. \tag{2.23}$$

Now use the inner product notation $\langle fg \rangle \equiv \int_{D^\alpha} f(\vec{x})g(\vec{x})dV$ to simplify how the integrals are written, and solve for the time derivative to obtain an evolution equation for the solution value at the nodal locations,

$$\partial_t f(x_k^\alpha, t) = -J_{ij}\langle \phi_l \phi_k \rangle^{-1} \langle \phi_l \partial_{\xi_j} F_i^k \rangle + \langle \phi_l \phi_k \rangle^{-1} \langle \phi_l s^\alpha \rangle. \tag{2.24}$$

In principle the time evolution of the approximation may now be obtained. However, solution of Eqn. 2.24 cannot yet be done because there is no way to evaluate the flux integral $\langle \phi_l \partial_{\xi_j} F_i^\alpha \rangle$ without boundary information at the element surface. There must be a mechanism to exchange information between elements or the solution cannot be global. To resolve this dilemma, the flux integral is integrated by parts to yield

$$\langle \phi_l \partial_{\xi_j} F_i^\alpha \rangle = \langle \partial_{\xi_j} (\phi_l F_i^\alpha) \rangle - \langle F_i^\alpha \partial_{\xi_j} \phi_l \rangle \tag{2.25}$$

$$= \langle \phi_l \widetilde{F}_i^{\alpha\gamma} \rangle_{\partial D^{\alpha\gamma}} - \langle F_i^\alpha \partial_{\xi_j} \phi_l \rangle \tag{2.26}$$

following application of the divergence theorem on the first term to yield a surface integral, where the index γ represents the faces of the element. Now both integrals contain the flux, a known quantity, rather than the flux gradient. Also, the surface integral may be used to exchange information of the solution across elements. The flux term $\widetilde{F}_i^\alpha$ is called the *numerical flux*. The form of the numerical flux, or in other words the method of information exchange across element interfaces, is free to be chosen and is explained in a later section. The second term containing the analytically evaluated flux F_i^α is known as the *internal flux* because it is evaluated within the element, as it is part of a volume integral.

The flux and source functions are now expanded onto the nodal basis,

$$F_i^\alpha = F_i(x_n^\alpha, t)\phi_n \tag{2.27}$$

$$\widetilde{F}_i^\alpha = \widetilde{F}_i(x_m^\alpha, t)\phi_m \tag{2.28}$$

$$s^\alpha = s(x_p^\alpha, t)\phi_p \tag{2.29}$$

so that the integrals involving these terms may be written as

$$\langle F_i^\alpha \partial_{\xi_j} \phi_l \rangle = F_i(x_n^\alpha, t) \langle \phi_n \partial_{\xi_j} \phi_l \rangle \tag{2.30}$$

$$\langle \phi_l \widetilde{F}_i^{\alpha\gamma} \rangle_{\partial D^{\alpha\gamma}} = \widetilde{F}_i(x_m^{\alpha\gamma}, t) \langle \phi_l \phi_m \rangle_{\partial D^{\alpha\gamma}} \tag{2.31}$$

$$\langle \phi_l s^\alpha \rangle = s(x_p^\alpha, t) \langle \phi_l \phi_p \rangle. \tag{2.32}$$

Upon substitution of the expansions Eqns. 2.30, 2.31 and 2.32 into Eqn. 2.24, the nodal discontinuous Galerkin *semi-discrete equation* is obtained,

$$\partial_t f(x_k^\alpha, t) = J_{ij} F_i(x_n^\alpha, t) \langle \phi_l \phi_k \rangle^{-1} \langle \phi_n \partial_{\xi_j} \phi_l \rangle - J_{ij} \widetilde{F}_i(x_m^{\alpha\gamma}, t) \langle \phi_l \phi_k \rangle^{-1} \langle \phi_l \phi_m \rangle_{\partial D^{\alpha\gamma}} + I_{kp} s(x_p^\alpha, t) \tag{2.33}$$

with I_{kp} the identity matrix. The equation is so-called because spatial derivatives are fully discretized but temporal ones are not. The choice of time integrator is then free. Computation of the time derivative estimate in the discontinuous Galerkin method at some node of an element amounts to computation of a series of matrix products, provided that the elements of the basis integrals are known. Defining these *basis arrays*,

$$\Upsilon_{jkn} = \langle \phi_l \phi_k \rangle^{-1} \langle \phi_n \partial_{\xi_j} \phi_l \rangle \tag{2.34}$$

$$\Xi_{\gamma km} = \langle \phi_l \phi_k \rangle^{-1} \langle \phi_l \phi_m \rangle_{\partial D^{\alpha\gamma}}, \tag{2.35}$$

a more compact form of Eqn. 2.33 may be written as

$$\partial_t f(x_k^\alpha, t) = J_{ij} \Upsilon_{jkn} F_i(x_n^\alpha, t) - J_{ij} \Xi_{\gamma km} \widetilde{F}_i(x_m^{\alpha\gamma}, t) + s(x_k^\alpha, t). \tag{2.36}$$

Equation 2.36 is the form of the nodal DG method used in Vlasov1D1V. A list of indices used and their ranges follows for clarity;

- α - element index within domain,
- k - node within element, not summed over,
- i - dimension index (e.g. Vlasov1D1V has two dimensions),
- j - dimension index (same range as i),

- n - node within element, summed over,
- γ - element face index (e.g. squares have four faces),
- m - node within element face, summed over,
- p - node within element, summed over.

Example 2: Calculation of the basis arrays for second order elements

This example is an extension of Example 1 to obtain the basis arrays for calculation of the semi-discrete equation, Eqn. 2.36. Suppose it is desired to calculate the basis arrays Υ_{jkn} and $\Xi_{\gamma km}$ for second order elements, as in Fig. 2.3. Both arrays require the inverse of the symmetric matrix $M = \langle \phi_l \phi_k \rangle = \int_0^1 \int_0^1 d\xi_0 d\xi_1 \phi_l \phi_k$, known as the *mass matrix*. Using the four-member basis set in Eqn. 2.15, the ten integrations (for a 4x4 symmetric matrix) are carried out to obtain

$$M = \frac{1}{36} \begin{bmatrix} 4 & 2 & 2 & 1 \\ 2 & 4 & 1 & 2 \\ 2 & 1 & 4 & 2 \\ 1 & 2 & 2 & 4 \end{bmatrix} \quad \Longrightarrow \quad M^{-1} = \begin{bmatrix} 16 & -8 & -8 & 4 \\ -8 & 16 & 4 & -8 \\ -8 & 4 & 16 & -8 \\ 4 & -8 & -8 & 16 \end{bmatrix}. \tag{2.37}$$

First the internal flux array Υ_{jkn} will be calculated, a 2x4x4 array as there are two dimensions and four nodes in the element. In order to do so, the *advection matrix* $S = \langle \phi_n \partial_{\xi_j} \phi_l \rangle$ must be determined. This is a third-order object with 4x2x4 elements. Before integrations may be carried out, the nodal basis gradient is computed as

$$\partial_{\xi_0}\phi_0 = \xi_1 - 1, \quad \partial_{\xi_1}\phi_0 = \xi_0 - 1, \quad \partial_{\xi_0}\phi_1 = 1 - \xi_1, \quad \partial_{\xi_1\phi_1} = -\xi_0, \tag{2.38}$$

$$\partial_{\xi_0}\phi_2 = -\xi_1, \quad \partial_{\xi_1}\phi_2 = 1 - \xi_0, \quad \partial_{\xi_0}\phi_3 = \xi_1, \quad \partial_{\xi_1}\phi_0 = \xi_0. \tag{2.39}$$

The component integrations of the advection matrix are then performed to obtain its two 4x4 component matrices,

$$\langle \phi_n \partial_{\xi_0} \phi_l \rangle = \frac{1}{12} \begin{bmatrix} -2 & 2 & -1 & 1 \\ -2 & 2 & -1 & 1 \\ -1 & 1 & -2 & 2 \\ -1 & 1 & -2 & 2 \end{bmatrix}, \quad \langle \phi_n \partial_{\xi_1} \phi_l \rangle = \frac{1}{12} \begin{bmatrix} -2 & -1 & 2 & 1 \\ -1 & -2 & 1 & 2 \\ -2 & -1 & 2 & 1 \\ -1 & -2 & 1 & 2 \end{bmatrix}. \tag{2.40}$$

The basis array $\Upsilon_{jkn} = M^{-1}_{lk} S_{njl}$ is then found by computing its two component matrices separately,

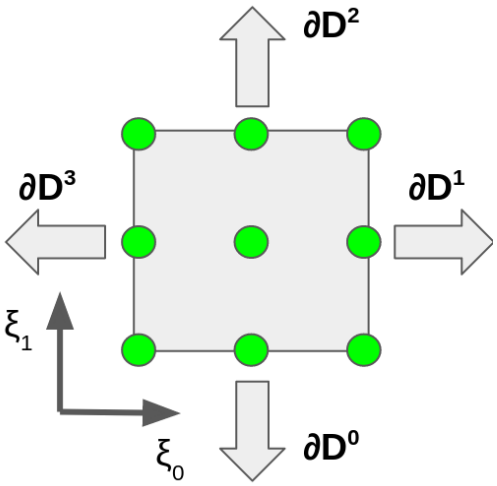
$$\Upsilon_{0kn} = M^{-1}_{lk} S_{n0l} = \frac{1}{12} \begin{bmatrix} 16 & -8 & -8 & 4 \\ -8 & 16 & 4 & -8 \\ -8 & 4 & 16 & -8 \\ 4 & -8 & -8 & 16 \end{bmatrix} \begin{bmatrix} -2 & 2 & -1 & 1 \\ -2 & 2 & -1 & 1 \\ -1 & 1 & -2 & 2 \\ -1 & 1 & -2 & 2 \end{bmatrix} = \begin{bmatrix} -3 & -3 & 0 & 0 \\ 3 & 3 & 0 & 0 \\ 0 & 0 & -3 & -3 \\ 0 & 0 & 3 & 3 \end{bmatrix}. \tag{2.41}$$

Similarly,

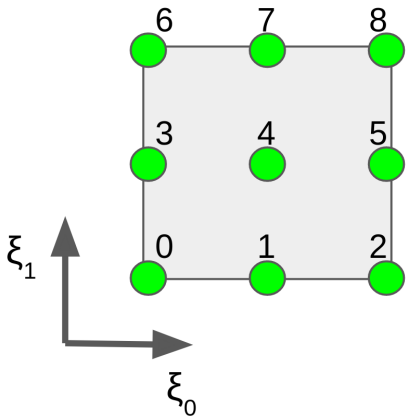
$$\Upsilon_{1kn} = M^{-1}_{lk} S_{n1l} = \frac{1}{12} \begin{bmatrix} 16 & -8 & -8 & 4 \\ -8 & 16 & 4 & -8 \\ -8 & 4 & 16 & -8 \\ 4 & -8 & -8 & 16 \end{bmatrix} \begin{bmatrix} -2 & -1 & 2 & 1 \\ -1 & -2 & 1 & 2 \\ -2 & -1 & 2 & 1 \\ -1 & -2 & 1 & 2 \end{bmatrix} = \begin{bmatrix} -3 & 0 & -3 & 0 \\ 0 & -3 & 0 & -3 \\ 3 & 0 & 3 & 0 \\ 0 & 3 & 0 & 3 \end{bmatrix}. \tag{2.42}$$

These are the entries of the internal flux array. Now the numerical flux array $\Xi_{\gamma km}$ is to be calculated, a 4x4x2 array as there are four faces, four nodes, and two nodes per element face. The array $\langle \phi_l \phi_m \rangle_{\partial D^{\alpha\gamma}}$ must be calculated, which is a sort of element face mass matrix consisting of four 4x2 matrices. Label the element faces as shown in Fig. 2.5a.

Only those nodal basis functions which are nonzero on a face need be evaluated. These



(a) The four faces of the isoparametric element are labelled 0–3 beginning down and proceeding counter-clockwise.



(b) Node numbering convention for the isoparametric third-order square element. Enumeration begins at the origin.

Figure 2.5: Element index conventions for face and node numbering.

are,

$$\partial D^0: \quad \xi_1 = 0, \xi_0 \in [0,1], \quad \phi_0 = 1 - \xi_0, \phi_1 = \xi_0 \tag{2.43}$$

$$\partial D^1: \quad \xi_0 = 1, \xi_1 \in [0,1], \quad \phi_1 = 1 - \xi_1, \phi_3 = \xi_1 \tag{2.44}$$

$$\partial D^2: \quad \xi_1 = 1, \xi_0 \in [0,1], \quad \phi_2 = 1 - \xi_0, \phi_3 = \xi_0 \tag{2.45}$$

$$\partial D^3: \quad \xi_0 = 0, \xi_1 \in [0,1], \quad \phi_0 = 1 - \xi_1, \phi_2 = \xi_1. \tag{2.46}$$

Then the eight integrations for each of the four 4x2 matrices of the element face mass matrix are determined,

$$\langle \phi_l \phi_m \rangle_{\partial D^0} = \frac{1}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad \langle \phi_l \phi_m \rangle_{\partial D^1} = \frac{1}{6} \begin{bmatrix} 0 & 0 \\ 2 & 1 \\ 0 & 0 \\ 1 & 2 \end{bmatrix}, \tag{2.47}$$

$$\langle \phi_l \phi_m \rangle_{\partial D^2} = \frac{1}{6} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 2 \\ 2 & 1 \end{bmatrix}, \quad \langle \phi_l \phi_m \rangle_{\partial D^3} = \frac{1}{6} \begin{bmatrix} 1 & 2 \\ 0 & 0 \\ 2 & 1 \\ 0 & 0 \end{bmatrix}. \tag{2.48}$$

Then the four component matrices of the $\Xi_{\gamma km} = \langle \phi_l \phi_k \rangle^{-1} \langle \phi_l \phi_m \rangle_{\partial D^{\alpha\gamma}}$ basis array may be calculated. For the first array,

$$\Xi_{0km} = M_{lk}^{-1} \langle \phi_l \phi_m \rangle_{\partial D^0} = \frac{1}{6} \begin{bmatrix} 16 & -8 & -8 & 4 \\ -8 & 16 & 4 & -8 \\ -8 & 4 & 16 & -8 \\ 4 & -8 & -8 & 16 \end{bmatrix} \begin{bmatrix} 2 & 1 \\ 1 & 2 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 4 & 0 \\ 0 & 4 \\ -2 & 0 \\ 0 & -2 \end{bmatrix}. \tag{2.49}$$

In a similar fashion the other components are calculated. In summary, the numerical flux

array for second order square elements is given by the following,

$$\Xi_{0km} = \begin{bmatrix} 4 & 0 \\ 0 & 4 \\ -2 & 0 \\ 0 & -2 \end{bmatrix}, \quad \Xi_{1km} = \begin{bmatrix} -2 & 0 \\ 4 & 0 \\ 0 & -2 \\ 0 & 4 \end{bmatrix}, \tag{2.50}$$

$$\Xi_{2km} = \begin{bmatrix} 0 & -2 \\ -2 & 0 \\ 0 & 4 \\ 4 & 0 \end{bmatrix}, \quad \Xi_{3km} = \begin{bmatrix} 0 & 4 \\ 0 & -2 \\ 4 & 0 \\ -2 & 0 \end{bmatrix}. \tag{2.51}$$

The code Vlasov1D1V uses third-order elements. The calculation to obtain them is more cumbersome and is omitted. The result is that the internal flux array consists of the matrices,

$$\Upsilon_{0kn} = \begin{bmatrix} -6 & -4 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2.5 & 0 & -2.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ -4 & 4 & 6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -6 & -4 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2.5 & 0 & -2.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & -4 & 4 & 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -6 & -4 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2.5 & 0 & -2.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & -4 & 4 & 6 \end{bmatrix}, \tag{2.52}$$

$$\Upsilon_{1kn} = \begin{bmatrix} -6 & 0 & 0 & -4 & 0 & 0 & 4 & 0 & 0 \\ 0 & -6 & 0 & 0 & -4 & 0 & 0 & 4 & 0 \\ 0 & 0 & -6 & 0 & 0 & -4 & 0 & 0 & 4 \\ 2.5 & 0 & 0 & 0 & 0 & 0 & -2.5 & 0 & 0 \\ 0 & 2.5 & 0 & 0 & 0 & 0 & 0 & -2.5 & 0 \\ 0 & 0 & 2.5 & 0 & 0 & 0 & 0 & 0 & -2.5 \\ -4 & 0 & 0 & 4 & 0 & 0 & 6 & 0 & 0 \\ 0 & -4 & 0 & 0 & 4 & 0 & 0 & 6 & 0 \\ 0 & 0 & -4 & 0 & 0 & 4 & 0 & 0 & 6 \end{bmatrix}, \tag{2.53}$$

while the numerical flux array has the entries

$$\Xi_{0km} = \begin{bmatrix} 9 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 9 \\ -1.5 & 0 & 0 \\ 0 & -1.5 & 0 \\ 0 & 0 & -1.5 \\ 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{bmatrix}, \quad \Xi_{1km} = \begin{bmatrix} 3 & 0 & 0 \\ -1.5 & 0 & 0 \\ 9 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & -1.5 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 3 \\ 0 & 0 & -1.5 \\ 0 & 0 & 9 \end{bmatrix}, \tag{2.54}$$

$$\Xi_{2km} = \begin{bmatrix} 0 & 0 & 3 \\ 0 & 3 & 0 \\ 3 & 0 & 0 \\ 0 & 0 & -1.5 \\ 0 & -1.5 & 0 \\ -1.5 & 0 & 0 \\ 0 & 0 & 9 \\ 0 & 9 & 0 \\ 9 & 0 & 0 \end{bmatrix}, \quad \Xi_{3km} = \begin{bmatrix} 0 & 0 & 9 \\ 0 & 0 & -1.5 \\ 0 & 0 & 3 \\ 0 & 9 & 0 \\ 0 & -1.5 & 0 \\ 0 & 3 & 0 \\ 9 & 0 & 0 \\ -1.5 & 0 & 0 \\ 3 & 0 & 0 \end{bmatrix}. \tag{2.55}$$

2.2.4 Evaluation of the semi-discrete equation: upwind numerical fluxes

In the DG method, information is passed between elements using surface integrals which in discrete form appear as the matrix product $J_{ij}\Xi_{\gamma km}\widetilde{F}_i(x_m^{\alpha\gamma}, t)$. During derivation of the numerical flux array $\Xi_{\gamma km}$, the question of how to determine the actual numerical flux $\widetilde{F}_i(x_m^{\alpha\gamma}, t)$ was put off for the moment. In general the evaluation of numerical fluxes for nonlinear hyperbolic equations in a manner which conserves the evolved variable f while balancing artifically introduced diffusion can be tricky. An excellent reference for numerical fluxes is given in [15]. Fortunately, at any given moment of time t the Vlasov equation

$$\partial_t f(x, v, t) + v\partial_x f(x, v, t) + E(x)\partial_v f(x, v, t) = 0 \tag{2.56}$$

appears to be a linear hyperbolic PDE, although subsequent evolution has a nonlinear character because the field $E(x)$ depends on the distribution $f(x, v)$ through the Poisson equation. Treating the Vlasov equation as a linear hyperbolic equation allows simple *upwind fluxes* to be used for evaluation of numerical flux $\widetilde{F}_i(x_m^{\alpha\gamma}, t)$ at element surface nodes $x_m^{\alpha\gamma}$.

The idea behind upwind numerical flux is simple. Here the wind in "upwind" refers to the direction of the flux F_i. Any face γ of an element α has a surface normal $\hat{n}$. For example, with the convention adopted in Fig. 2.5a, face 1 has surface normal $\hat{n})_1 = \{1, 0\}$. The numerical flux which must be evaluated on face γ is $F_i \cdot \hat{n}_\gamma$, which for face 1 is the

x-directed flux. Now the three possible cases of $F_i \cdot \hat{n}_\gamma$ are:

- $F_i = 0$ (no wind),
- $F_i \cdot \hat{n}_\gamma > 0$ (wind directed out of element),
- $F_i \cdot \hat{n}_\gamma < 0$ (wind directed into element).

The first case is trivial and the flux need not be evaluated as it does not contribute to the matrix product. The second and third cases are treated using the upwinding principle. If the wind is directed out of the element, the value $f(x_m^{\alpha\gamma})$ within the element is upwind of the flux direction and so the flux using this value $F(f(x_m^{\alpha\gamma}))$ should be chosen as the numerical flux $\widetilde{F}$. Now if the wind is directed into the element, then the value $f(x_m^{\beta\gamma})$ should be used to evaluate the numerical flux, where β is the neighbor of element α in the direction of the flux. For example, look at Fig. 2.6 which considers the x-directed numerical flux at node 8 of an element. In the first case, the upwind flux is evaluated using the node within the element α, while in the second case it must be evaluated using the nodal value of its neighboring element $\alpha + 1$.

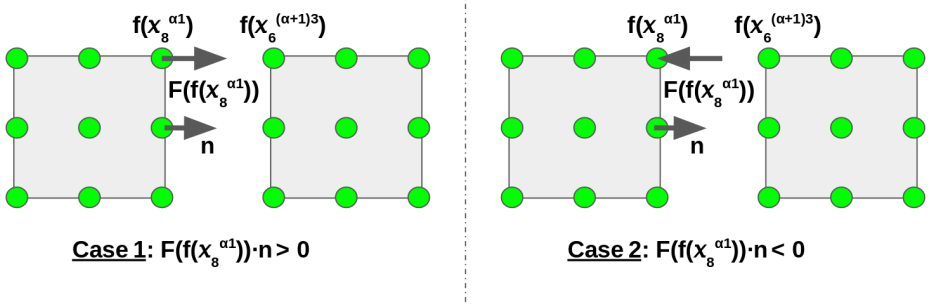


Figure 2.6: The two cases of x-directed numerical flux for an example node located at the top-most corner of an element. The first case results in the numerical flux being chosen local to the element α. The second case requires the numerical flux to be chosen using the value at the neighboring element $\alpha + 1$, in the direction of the wind.

2.2.5 Sparseness of the basis arrays and calculation optimization

As the number of nodes within a square element scales as $N = (k+1)^2$ with the polynomial degree k, the number of computations required to compute the matrix products of Eqn. 2.36 per element becomes rather large. However as the element order increases, so too does the sparseness of the basis arrays Υ_{jkn}, $\Xi_{\gamma km}$. For the calculations be optimized, the basis arrays may be stored as sparse third order arrays and the matrix products computed sparsely so that all calculations involving a multiplication by zero are skipped. To form a sparse third order array, it is helpful to think of the array as a collection of sparse matrices. Then four helper arrays are required. These are vectors of,

- nonZeros, the number of nonzero entries per matrix, used to keep track of which component matrix one is in,
- rowIndex, the row index of the nonzero entry within its component matrix,
- colIndex, the column index of the nonzero entry,
- values, the nonzero value of the basis array.

These arrays are formed for each of the basis arrays and are used to simplify calculation of the semi-discrete equation. For low-order elements, the additional memory usage of indexing into these helper arrays may outweigh the simplification of calculations and prevent any speed-up. However for higher order elements, or certainly for problems involving higher dimensional elements, usage of sparse basis arrays can provide speed-up of several times over calculation involving all elements of the basis.

Python code example 1: sparse matrix product

An example of how the sparse basis may be used to calculate the internal flux matrix product $J_{ij}\Upsilon_{jkn}F_i(x_n^\alpha, t)$ is now given. For a square element, the Jacobian is diagonal, given by $J_{ij} = \begin{bmatrix} 1/\Delta x & 0 \\ 0 & 1/\Delta v \end{bmatrix}$. The first parts of the below calculation reflect this. The code supposes

that we are evaluating the semi-discrete equation RHS, called dfdt, using the analytical flux formula $F_i = \{v, E\}$, at a third-order element located at indices i, j on a rectangular grid. At this element, the internal flux is calculated using two simple for loops, as shown.

```
##### Internal Flux
    ### Contributions from x-directed fluxes
    for l0 in range(NonZeroUp[0]):
        dfdt[i, j, RowIdxUp[l0]] += (1.0/delta_x)*f[i, j, ColIdxUp[l0]]*
                                                    ValsUp[l0]*
                                                    v[j, ColIdxUp[l0]//3]
    ### Contributions from v-directed fluxes
    for l1 in range(NonZeroUp[0], NonZeroUp[1]):
        dfdt[i, j, RowIdxUp[l1]] += (1.0/delta_v)*f[i, j, ColIdxUp[l1]]*
                                                    ValsUp[l1]*
                                                    E[i, ColIdxUp[l1]%3]
```

For third-order square elements, the number of products required to form the sum of the matrix product is reduced by a factor of two by using sparse basis arrays. Here the velocity v and acceleration E arrays are held as Nvx3 and Nxx3 arrays respectively, as they are one-dimensional entities rather than defined on the phase space. The floor division // and modulus % operators are used to properly index into these one-dimensional arrays while using an index with range greater than 3.

2.2.6 Explicit timestepping with the Runge-Kutta Discontinuous Galerkin method

The Runge-Kutta (RK) method is an explicit discretization scheme which obtains approximate solutions of systems of ODEs $\frac{d}{dt}f(t) = F(f(t))$ such as the semi-discrete equation given by Eqn. 2.36. The simplest such RK method is simply the explicit forward Euler scheme. For high-order accurate solutions, the RK method uses multiple steps to adjust the approximate slope. Here is summarized the algorithm obtained when the Discontinuous Galerkin method

is combined with the Runge-Kutta method for solution of a hyperbolic PDE, a procedure known as an RKDG method. A good review of this solution method is presented in [6].

In the solution of hyperbolic partial differential equations, care must be taken around the timestep when the RK temporal approximation method is combined with a method of spatial discretization. Consider a linear hyperbolic PDE in two variables such as

$$\partial_t f(x,v,t) + v\partial_x f(x,v,t) + E\partial_v f(x,v,t) = 0. \tag{2.57}$$

Here the quantities v and E are called the wave speeds. After application of the DG discretization, the semi-discrete equation is obtained as an ODE in time. For an explicit approximation, the solution's stability is limited by the timestep in a relation known as the Courant-Friedrichs-Lewy (CFL) condition. For given node spacings Δx and Δv, the CFL condition is given as

$$\max\left\{|v|\frac{\Delta t}{\Delta x}, |E|\frac{\Delta t}{\Delta v}\right\} \leq \text{CFL}. \tag{2.58}$$

The timestep is then chosen such that this CFL condition is satisfied. If the DG discretization uses k-degree polynomials (or $k+1$ nodes per dimension) together with a $k+1$ stage RK method, then a practical formula for the limiting CFL number for use in Eqn. 2.58 is given by

$$\text{CFL} = \frac{1}{2k+1}. \tag{2.59}$$

The RK3 method used in the present program

For example, the program Vlasov1D1V uses elements with 3 nodes per spatial direction. This gives $k = 2$ for a timestep limiting condition of CFL $= 1/5$. For this stability-preserving condition to be valid, it is necessary to use a 3-stage RK method. Suppose that the semi-discrete equation is written in the form $\frac{d}{dt}f(t) = F(f(t))$ and that the solution at timestep i is given as f^i, and that it is desired to obtain the solution at timestep $i+1$. Then the

three-stage RK method (RK3) may be written as

$$f^{(1)} = f^i + \Delta t F(f^i) \tag{2.60}$$

$$f^{(2)} = f^i + \frac{\Delta t}{2}\Big(\frac{F(f^i) + F(f^{(1)})}{2}\Big) \tag{2.61}$$

$$f^{i+1} = f^i + \Delta t \frac{F(f^i) + F(f^{(1)}) + 4F(f^{(2)})}{6}, \tag{2.62}$$

which in practical terms may be interpreted as a full prediction step of Δt followed by a half-step of $\Delta t/2$ using the average slope of the given and prediction steps. Finally a weighted full step using the given data and two prediction steps is used to take a timestep to order $(\Delta t)^3$ accuracy.

It is important to recognize some critical details in implementation of this method. First, the method requires evaluating the semi-discrete equation three times per timestep. As well, it is necessary to take moments of the distribution function and obtain an updated electric field E by solving Poisson's equation at each stage of the RK method, because the field used as the wavespeed E must be self-consistent with the predicted distributions $f^{(1)}, f^{(2)}$.

2.2.7 Design of a parallelized kernel for timestepping

In the DG method, elements need only communicate with their immediate neighbors in order to evaluate the semi-discrete equation for timestepping. Therefore the calculation is easily parallelized. These days it is fairly simple to write Python code which utilizes Nvidia's CUDA GPU computing architecture using the Numba Python package, a part of the Anaconda development effort.

The basic idea of CUDA is that the arrays necessary for carrying out a certain computation are sent from the CPU (the "host") to the GPU (the "device"), becoming a device array. The device arrays are partitioned into sections ("blocks"), and a certain number of worker cores ("threads") are assigned to each section. There must be minimal memory access between sections while the calculation is carried out. Once finished, the modified device array is returned to the host and the algorithm may continue. The greatest limitation here is that

the device memory is far more limited than host memory, so that when working with very large data sets the process is slowed by necessarily greater communication with the host.

Python code example 2: CUDA kernel for the semi-discrete equation

Here an example is given of the code structure required to perform GPU calculations using Python. The instance used for illustration is calculation of the semi-discrete equation, Eqn. 2.36, in the special case of periodic boundary conditions. Modifications required for adding a particle source or collision operator are omitted for clarity.

Two functions are required; first one for set up, copying arrays to the device and establishing the blocked grid and threads per block. Another function, the kernel, is executed on the device to perform the calculation. The arrays required on the device for calculation of Eqn. 2.36 are: the distribution function f, the velocity coordinate v, the acceleration field E, the basis arrays Ξ and Υ. The elements of the Jacobian Δx and Δv may be passed as scalar values. The velocity coordinate and basis arrays may be passed to the device at the beginning of the program, but the other arrays must be passed every timestep within the set-up function. Device arrays are denoted with a "d_" prefix. The number of blocks are then specified using a standard formula in terms of the number of threads per block.

```
from numba import cuda
TPB = 16

def semiDiscreteDGEqn_thirdOrder(f, d_v, E, delta_x, delta_v,
                                 dNonZeroXi, dRowIdxXi, dColIdxXi, dValsXi,
                                 dNonZeroUp, dRowIdxUp, dColIdxUp, dValsUp,
                                 dFaceNodesX, dFaceNodesV):
    nx = f.shape[0] # number of elements in x-direction
    nv = f.shape[1] # number of elements in v-direction
    N = 9 # nodes per element

    # Send input arrays to device
    d_f = cuda.to_device(f)
```

```
    d_E = cuda.to_device(E)

    # Create output array on device
    d_dfdt = cuda.device_array((nx,nv,N))

    # Grid parameters
    blockSize = (TPB, TPB)
    numBlocksX = (nx + blockSize[0] - 1)//blockSize[0]
    numBlocksV = (nv + blockSize[1] - 1)//blockSize[1]
    numBlocks = (numBlocksX, numBlocksV)

    # Call kernel
    DGEqn_kernel[numBlocks, blockSize](d_f, d_v, d_E, delta_x, delta_v,
                                       dNonZeroXi, dRowIdxXi,
                                       dColIdxXi, dValsXi,
                                       dNonZeroUp, dRowIdxUp,
                                       dColIdxUp, dValsUp,
                                       dFaceNodesX, dFaceNodesV, d_dfdt)

    # Return output array to host
    return d_dfdt.copy_to_host()
```

The kernel function is then structured to carry out the semi-discrete equation calculation. The function must be given a "@cuda.jit" decorator to declare it to be compiled by Numba as CUDA code.

```
@cuda.jit
def DGEqn_kernel(d_f, d_v, d_E, delta_x, delta_v,
                 dNonZeroXi, dRowIdxXi, dColIdxXi, dValsXi,
                 dNonZeroUp, dRowIdxUp, dColIdxUp, dValsUp,
                 dFaceNodesX, dFaceNodesV, d_dfdt):
    # Grid coordinates
    i,j = cuda.grid(2)
```

```
# Remain within bounds of this block
if (i > d_f.shape[0] or j > d_f.shape[1]):
    return
# Zero entry for this block in case of junk memory
for k in range(9):
    d_dfdt[i,j,k] = 0
# Skip calculation within ghost elements
if (i == 0 or i == (d_f.shape[0]-1) or j == 0 or j == (d_f.shape[1]-1)):
    return

###### Numerical Flux Calculation
# Face 0: V Negative Facing
for k0 in range(dNonZeroXi[0]):
    # Determine sign of flux at face
    sgn = math.copysign(1.0, d_E[i, dColIdxXi[k0]])
    # Calculate dfdt // if sgn = -1.0, use local value
    # if sgn = 1.0, use neighbor
    d_dfdt[i, j, dRowIdxXi[k0]] += -0.5/delta_v*dValsXi[k0]*
        d_E[i, dColIdxXi[k0]]*(
        (sgn - 1.0)*d_f[i, j, dFaceNodesV[1, dColIdxXi[k0]]] -
        (sgn + 1.0)*d_f[i, j-1, dFaceNodesV[0, 2-dColIdxXi[k0]]] )

# Face 1: X Positive Facing
for k1 in range(dNonZeroXi[0], dNonZeroXi[1]):
    # Determine sign of flux at face
    sgn = math.copysign(1.0, d_v[j, dColIdxXi[k1]])
    # Calculate dfdt // if sgn = 1.0, use local value
    # if sgn = -1.0, use neighbor
    d_dfdt[i, j, dRowIdxXi[k1]] += -0.5/delta_x*dValsXi[k1]*
        d_v[j, dColIdxXi[k1]]*(
        (sgn + 1.0)*d_f[i, j, dFaceNodesX[0, dColIdxXi[k1]]] -
        (sgn - 1.0)*d_f[i+1, j, dFaceNodesX[1, 2-dColIdxXi[k1]]] )
```

```
# Face 2: V Positive Facing
for k2 in range(dNonZeroXi[1], dNonZeroXi[2]):
    # Determine sign of flux at face
    sgn = math.copysign(1.0, d_E[i, dColIdxXi[k2]])
    # Calculate dfdt // if sgn = 1.0, use local value
    # if sgn = 1.0, use neighbor
    d_dfdt[i, j, dRowIdxXi[k2]] += -0.5/delta_v*dValsXi[k2]*
        d_E[i, dColIdxXi[k2]]*(
        (sgn + 1.0)*d_f[i, j, dFaceNodesV[0, dColIdxXi[k2]]] -
        (sgn - 1.0)*d_f[i, j+1, dFaceNodesV[1, 2-dColIdxXi[k2]]] )

# Face 3: X Negative Facing
for k3 in range(dNonZeroXi[2], dNonZeroXi[3]):
    # Determine sign of flux at face
    sgn = math.copysign(1.0, d_v[j, dColIdxXi[k3]])
    # Calculate dfdt // if sgn = -1.0, use local value
    # if sgn = 1.0, use neighbor
    d_dfdt[i, j, dRowIdxXi[k3]] += -0.5/delta_x*dValsXi[k3]*
        d_v[j, dColIdxXi[k3]]*(
        (sgn - 1.0)*d_f[i, j, dFaceNodesX[1, dColIdxXi[k3]]] -
        (sgn + 1.0)*d_f[i-1, j, dFaceNodesX[0, 2-dColIdxXi[k3]]] )

###### Internal flux calculation
### Contribution from x-directed flux
for k4 in range(dNonZeroUp[0]):
    d_dfdt[i, j, dRowIdxUp[k4]] += (1.0/delta_x)*d_f[i, j, dColIdxUp[k4]]*
                                   dValsUp[k4]*d_v[j, dColIdxUp[k4]//3]
for k5 in range(dNonZeroUp[0], dNonZeroUp[1]):
    d_dfdt[i, j, dRowIdxUp[k5]] += (1.0/delta_v)*d_f[i, j, dColIdxUp[k5]]*
                                   dValsUp[k5]*d_E[i, dColIdxUp[k5]%3]
```

The numerical flux array utilizes the sparse basis calculation method. For implementation of the upwind numerical flux method, the "copysign" function is used to determine the sign

of the flux. If the flux is directed out of the element, then the flux is evaluated using the element's nodal value. If directed inward, then the neighbor's neighbor's overlapping value is to be used.

For example, if s is the sign of the flux (± 1) at the element face, then the formula $\frac{1}{2}\Big[(s+1)f_i - (s-1)f_{i+1}\Big]$ chooses the correct nodal value for positively directed element faces, and $\frac{1}{2}\Big[(s-1)f_i - (s+1)f_{i-1}\Big]$ for negatively directed element faces. This method is more computationally efficient than using a conditional statement in the kernel, which leads to instruction branching upon compilation and should be avoided. A formulation in terms of $\vec{F}\cdot\hat{n}$, where $\hat{n}$ is the face surface normal, is possible but not necessary in practice.

The nodes corresponding to the various faces are given by face node arrays for each dimension. These are 2xN arrays where N is the number of nodes per face. The first entry is a vector of face nodes in the positive direction, and the second face nodes in the negative direction. Consulting Fig. 2.5b, the FaceNodesX array is given by $\Big[[2,5,8],[6,3,0]\Big]$ and FaceNodesV by $\Big[[8,7,6],[0,1,2]\Big]$. The direction of evaluation of the face nodes must match the convention chosen when the numerical flux basis array Ξ was calculated. Further, the face nodes considered for a numerical flux calculation must overlap, which is accomplished with the $2-\text{idx}$ indexing in the neighboring face node array.

When the kernel calculation is completed, the array "d_dfdt" is implicitly returned to the set-up function and subsequently returned to the host. These functions are called at every RK stage of the timestepping routine.

2.3 Hydrodynamic Boundary Conditions in Phase Space

The Vlasov equation describes an incompressible flow of probability fluid (whose density is given by the distribution function f) in a phase space of some dimension, depending on the degrees of freedom of the given problem. As such, boundary conditions of the phase space domain are hydrodynamic in character. The difference between the Vlasov flow and standard incompressible hydrodynamics is that the Vlasov flux is given by the phase space coordinate v and the local mean field $E(x)$ rather than by a velocity field. That is, $\vec{F} = \{v, E(x)\}$.

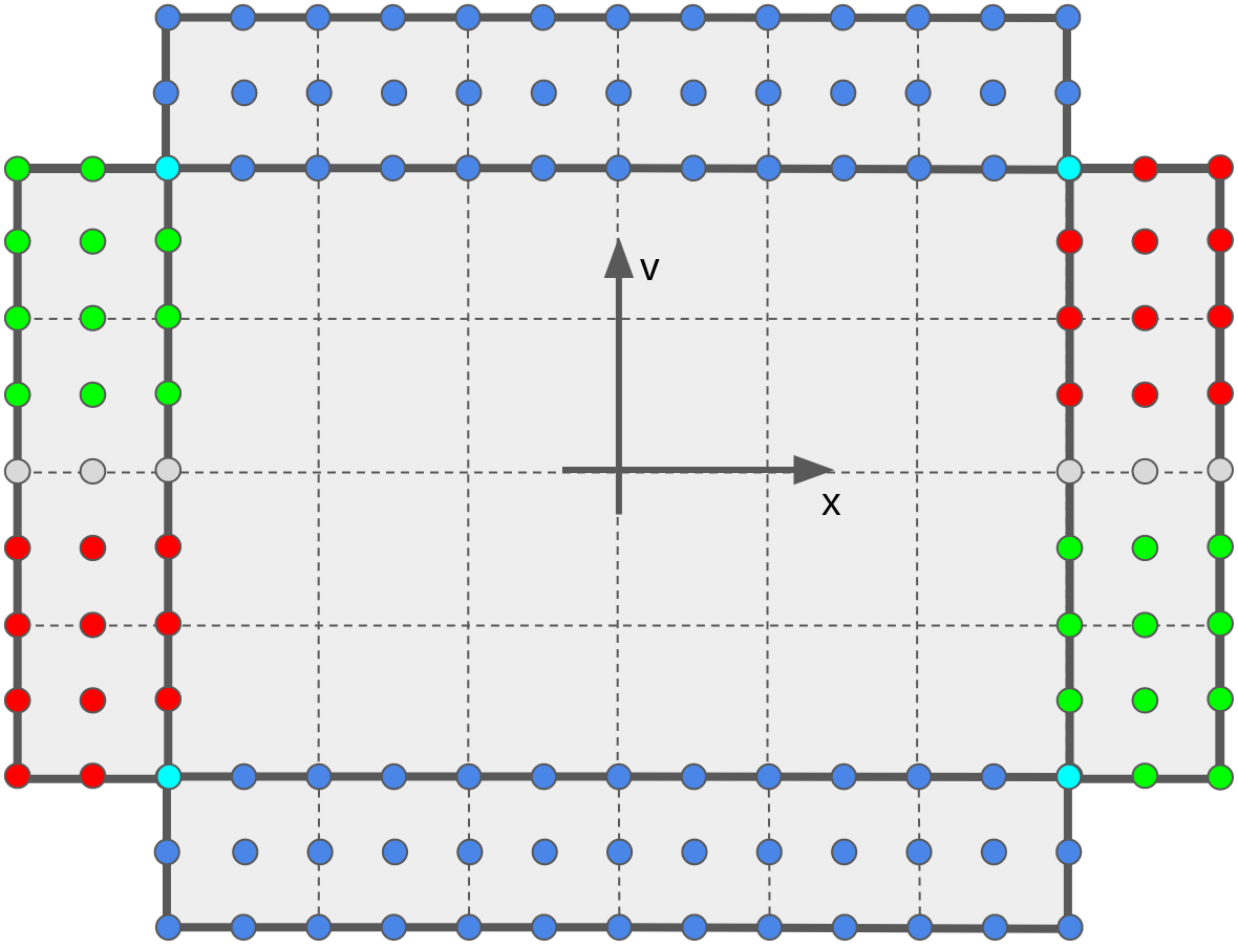


Figure 2.7: A 1D1V phase space domain with a layer of third order ghost elements on its boundary. The nodes within the ghost elements are color-coded according to their significance for the implementation of boundary conditions. Red corresponds to an outflow boundary ($\vec{v}\cdot\hat{n} > 0$) while green corresponds to an inflow boundary ($\vec{v}\cdot\hat{n} < 0$). Gray boundaries for which $\vec{v} = 0$ are ambiguous and need not be set. Velocity boundaries are colored in dark blue. Overlapping nodes for physical and velocity ghost elements are colored teal, and the color of their neighboring nodes within their element should be mentally filled-in.

In the finite element method, boundary conditions are typically handled by a layer of ghost elements at the domain boundaries. Figure 2.7 depicts a discretized phase space with ghost layers at the space-oriented (physical) and velocity-oriented (velocity) boundaries. The following sections describe how to set proper boundary conditions for each of these cases.

2.3.1 Conditions at domain boundary space-oriented element faces

The phase space boundaries corresponding to space-oriented faces are the most important ones to get right for a Vlasov solver to work properly. There are two important cases. Either $\vec{v} \cdot \hat{n} > 0$ or $\vec{v} \cdot \hat{n} < 0$. The first case is referred to in computational fluid dynamics as *outflow*, as the flux is oriented out of the domain. The second case is referred to as *inflow* as something is to be entering the domain at this area. Because the flux v is also the velocity coordinate, the left boundary will have outflow for $v < 0$ and inflow where $v > 0$, and on the right boundary the opposite. In Fig. 2.7, outflow nodes are color-coded red and inflow green.

Inflow and outflow boundary conditions

Outflow boundaries are somewhat simpler than inflow ones. The values set in the ghost nodes here may be considered as "junk" and ignored, as they can never enter the domain (they are downwind of the flux). There are some cases where what leaves the domain is important and should be noted. In the special case of using an x-dimension symmetry-plane boundary, where what leaves the domain is also what is entering the domain, then the outflow flux should be mapped to the inflow nodes. Also, if one is studying a sheath problem with some degree of secondary emission, then the outflow should be noted so that some specified fraction may re-enter the domain.

On the other hand, the values set in the ghost nodes of an inflow boundary are critical. The upwind numerical flux of the neighboring domain element will grab these ghost values. Typically then one will set within the ghost layer a Maxwellian distribution $f(v)$ with a specified density, temperature, and mean velocity in order to satisfy the desired physics (or

some superposition of Maxwellians). One may also set a time-dependent boundary condition here, with the density $n(t)$ varying sinusoidally, for example. Examples of common boundary conditions are provided below.

Periodic boundary condtions

Oftentimes it is desirable to use a periodic physical domain, so that the ends of a domain are connected. This replicates an endlessly periodic motion in one or more dimensions with a period set by the domain length. It is primarily useful in studying problems involving periodic plasma waves. When a layer of ghost elements are used as in Fig. 2.7, setting periodic boundary conditions simply involves mapping the left column of ghost elements to the right-most column of domain elements, and vis-versa by mapping the right column of ghost elements to the left-most column of domain elements. Then when applying the DG method to domain elements, the value taken from the ghost element for evaluation of the numerical flux is that of the periodic neighbor on the other side of the domain.

2.3.2 Conditions at domain boundary velocity-oriented element faces

The acceleration field $E(x)$ changes throughout the course of a problem, so the boundary condition to set at velocity-oriented faces is not as straight-forward as at element faces. In the kinetic model, the velocity boundary is supposed to extend all the way to $\pm\infty$, which is apparently not possible in computational models. The primary concern with imposition of boundaries to velocity space is that probability density may not be conserved at these edges, as what density leaves due to a local outflow will be lost irretrievably.

The natural solution to this problem is to impose "no-flux" conditions at the velocity boundaries. The result is that while density is conserved, it also then reflects from the edge as if from a solid wall. Certainly this is not a realistic boundary condition. On the other hand, a simple solution is to note that if the velocity boundary is carried out far enough from the bulk of the distribution, the value in the boundary nodes will be exponentially small.

Then it is safe enough to keep an exponentially small value in the ghost elements here and to treat losses through the boundary as inconsequential for most problems.

2.3.3 Examples of boundary conditions for common problems

A 1D1V kinetic program is capable of treating a number of one-dimensional problems in plasma physics through application of various sorts of boundary conditions. Besides the common usage of periodic boundaries, there exist a number of other domain configurations. Here some examples are given of boundary conditions for a few classes of problems.

- Perfectly absorbing wall boundaries. In this case the distribution in the ghost elements is set as

$$f(x_L, v > 0) = 0, \qquad f(x_R, v < 0) = 0 \tag{2.63}$$

  where x_L and x_R refer to the left and right boundary extents, respectively. A Dirichlet condition should be set for the electric potential at both walls (with the same value or different ones).

- Counter-streaming electron beams between absorbing walls set at fixed potentials. This boundary condition matches some early experiments to study the two stream instability. Maxwellian distributions at opposite drift velocities should be set at either boundary,

$$f(x_L, v > 0) = n_0 \sqrt{\frac{1}{2\pi v_{th}^2}} \exp\left(\frac{-(v - v_0)^2}{2v_{th}^2}\right) \tag{2.64}$$

$$f(x_R, v < 0) = n_0 \sqrt{\frac{1}{2\pi v_{th}^2}} \exp\left(\frac{-(v + v_0)^2}{2v_{th}^2}\right) \tag{2.65}$$

  where n_0 is the desired beam density, v_{th} the beam thermal velocity, and v_0 the beam velocity. Setting Dirichlet conditions on the electric potential at either wall then sets up a problem of counter-streaming electron beams between perfectly absorbing electrodes. The initial condition within the domain should be chosen in a way compatible with this picture.

2.4 The BGK Collision Operator

In many problems it is desirable to include augmentations to the collisionless Vlasov equation to simulate more rich physics. Additions to the equation may take the form of sources and sinks in phase space (that is, terms a in Eqn. 2.17), and may include terms to model collisional relaxation of distributions. A very useful linear collisional relaxation model is known as the BGK operator. This section describes how the additional collision physics are implemented in the Vlasov1D1V program.

2.4.1 Implementation of the Bhatnagar-Gross-Krook (BGK) collision operator

Sophisticated collision operators augment the equation with terms involving derivatives of velocity, making the kinetic equation elliptic and more difficult to solve than the hyperbolic Vlasov equation. Yet distributions tend towards a Maxwellian, or normal, distribution as a result of randomizing particle collisions. Therefore the simplest model for collisional relaxation is the simple linear model,

$$S(x,v) = \frac{f_M(x,v) - f(x,v)}{\tau} \tag{2.66}$$

where $f_M(x,v)$ is the Maxwellian function at the local position x constructed using the density, mean velocity, and variance of the local distribution, and τ is an appropriately chosen relaxation time. That is,

$$f_M(x,v) = n(x)\sqrt{\frac{1}{2\pi v_{th}^2(x)}}\exp\Big(-\frac{(v-\langle v\rangle(x))^2}{2v_{th}^2(x)}\Big). \tag{2.67}$$

The approximation Eqn. 2.66 is named the Bhatnagar-Gross-Krook (BGK) operator. In the absence of spatial nonuniformity, the Vlasov equation with BGK operator appears as

$$\frac{df(v,t)}{dt} = \frac{f_M(v) - f(v,t)}{\tau} \tag{2.68}$$

with the solution

$$f(v,t) = f_m(v) + (f_0 - f_M)\exp\Big(-\frac{t}{\tau}\Big) \tag{2.69}$$

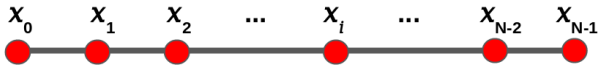


Figure 2.8: Labeling of evenly-spaced grid points in one dimension for solution of the discrete Poisson equation with the method of finite differences.

where $f_0 = f(v, 0)$ is the initial condition. The BGK operator leads the distribution function to relax with an e-folding time τ.

2.5 Finite-Difference Method for Poisson's Equation

Gauss's law $\nabla \cdot \vec{E} = \epsilon_0^{-1}\rho_c$ is a differential equation relating the electric field $\vec{E}$ to the distribution of charge density $\rho_c(\vec{x})$. In electrostatics the electric potential is defined as $\vec{E} = -\nabla\phi$, so that Poisson's equation $\nabla^2\phi = -\epsilon_0^{-1}\rho_c$ is obtained. This section considers solution of the one-dimensional Poisson equation on a grid of evenly spaced points x_i with N points, as in Fig. 2.8, through the method of finite differences.

Consider the problem of solving the elliptic equation

$$\frac{d^2\phi(x)}{dx^2} = f(x) \tag{2.70}$$

where the source function $f(x)$ is defined on the grid. The second order central difference approximation of $d^2\phi(x)/dx^2$ at the points x_i is given by

$$\frac{d^2\phi(x)}{dx^2} = \frac{\phi_{i-1} - 2\phi_i + \phi_{i+1}}{\Delta x^2} \tag{2.71}$$

with ϕ_i the solution value at point i, and $\Delta x = x_i - x_{i-1}$ the grid spacing. Substituting into Eqn. 2.70, the equation

$$\phi_{i-1} - 2\phi_i + \phi_{i+1} = \Delta x^2 f(x_i) \tag{2.72}$$

is obtained. This defines a system of linear equations for the unknowns ϕ_i with given data $f(x_i)$, represented by $A\vec{\phi} = \vec{S}$ where A is a tridiagonal matrix with diagonal -2 and upper/lower diagonals 1, known as the Poisson matrix, and $\vec{S} = \Delta x^2\vec{f}$ is the right-hand side

source vector. However the system of equations is not closed without boundary information for the points $i = 0$ and $i = N - 1$. Common boundary conditions imposed for the one-dimensional Poisson equation are Dirichlet (fixed values of ϕ), Neumann (fixed values of $\frac{d\phi}{dx}$), mixed Dirichlet-Neumann (fixed ϕ on one boundary and fixed $\frac{d\phi}{dx}$ on the other), and periodic (where ϕ_0 neighbors ϕ_{N-1}).

2.5.1 Dirichlet boundary conditions

Fixed values of potential on either end of a domain apply for example to electrodes charged to certain potentials. Given Dirichlet information on the boundary such as $\phi_0 = A$ and $\phi_{N-1} = B$, the Poisson matrix may be determined by truncating the boundary values from the vector of unknowns, so that $\vec{\phi} = \{\phi_1, \phi_2, ...\phi_{N-3}, \phi_{N-2}\}$, and moving the terms ϕ_0, ϕ_{N-1} in the first and last finite difference equations to the right-hand side. For example in the first finite difference equation,

$$\phi_0 - 2\phi_1 + \phi_2 = \Delta x^2 f(x_1) \quad \Longrightarrow \; -2\phi_1 + \phi_2 = \Delta x^2 f(x_1) - \phi_0. \tag{2.73}$$

A fully tridiagonal matrix results from this operation on the first and last finite difference equations, so that the Poisson matrix A and source vector S are given by

$$A = \begin{bmatrix} -2 & 1 & 0 & \dots & 0 & 0 & 0 \\ 1 & -2 & 1 & \dots & 0 & 0 & 0 \\ 0 & 1 & -2 & \dots & 0 & 0 & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & 0 & 0 & \dots & -2 & 1 & 0 \\ 0 & 0 & 0 & \dots & 1 & -2 & 1 \\ 0 & 0 & 0 & \dots & 0 & 1 & -2 \end{bmatrix}, \quad \vec{S} = \begin{bmatrix} \Delta x^2 f(x_1) - \phi_0 \\ \Delta x^2 f(x_2) \\ \vdots \\ \Delta x^2 f(x_{N-3}) \\ \Delta x^2 f(x_{N-2}) - \phi_{N-1} \end{bmatrix}. \tag{2.74}$$

The solution potential $\vec{\phi}$ is then determined through $\vec{\phi} = A^{-1}\vec{f}$. The given boundary conditions ϕ_0 and ϕ_{N-1} are appended to $\vec{\phi}$ to complete the solution.

2.5.2 Neumann boundary conditions

In contrast to the approach with Dirichlet boundary conditions, when given Neumann conditions such as $\phi'(x_0) \equiv \phi'_0 = A$, $\phi'(x_{N-1}) \equiv \phi'_{N-1} = B$ the boundary values do not need to be truncated from the solution vector. Instead, dummy grid points $i = -1$ and $i = N$ are used to construct second order central-difference approximations about the points $i = 0$ and $i = N - 1$. The dummy points are then algebraically eliminated from the system. For example the Poisson finite difference equation about the point $i = 0$ is

$$\phi_{-1} - 2\phi_0 + \phi_1 = \Delta x^2 f(x_0) \tag{2.75}$$

and the central difference approximation of ϕ' about $x = 0$ is given by

$$\frac{\phi_1 - \phi_{-1}}{2\Delta x} = \phi'_0 \quad \Longrightarrow \quad \phi_{-1} = \phi_1 - 2\Delta x \phi'_0. \tag{2.76}$$

Eliminating the dummy point, one obtains

$$-\phi_0 + \phi_1 = \frac{1}{2}\Delta x^2 f(x_0) + \Delta x \phi'_0. \tag{2.77}$$

A similar result is obtained for the finite difference equation at $i = N - 1$. A tridiagonal Poisson matrix A is again obtained. It is given, along with the source vector S, by

$$A = \begin{bmatrix} -1 & 1 & 0 & \dots & 0 & 0 & 0 \\ 1 & -2 & 1 & \dots & 0 & 0 & 0 \\ 0 & 1 & -2 & \dots & 0 & 0 & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & 0 & 0 & \dots & -2 & 1 & 0 \\ 0 & 0 & 0 & \dots & 1 & -2 & 1 \\ 0 & 0 & 0 & \dots & 0 & 1 & -1 \end{bmatrix}, \quad \vec{S} = \begin{bmatrix} \frac{1}{2}\Delta x^2 f(x_0) + \Delta x \phi_0 \\ \Delta x^2 f(x_1) \\ \vdots \\ \Delta x^2 f(x_{N-2}) \\ \frac{1}{2}\Delta x^2 f(x_{N-1}) + \Delta x \phi_{N-1} \end{bmatrix}. \tag{2.78}$$

The pure Neumann condition discrete Poisson problem gives a singular, and therefore non-invertible, matrix A. The equation $A\vec{\phi} = \vec{S}$ cannot be solved through matrix inversion. The crux of the issue is that with boundary conditions on the derivative, there is no reference

potential. This means that there are an infinite number of possible solutions for $\vec{\phi}$, each separated by a constant offset. Methods to solve the singular equation are given in a later section.

2.5.3 Mixed Dirichlet/Neumann boundary conditions

When Dirichlet conditions are given on one side of a domain and Neumann on the other, the problem treatment proceeds identically to the cases treated in the above sections on the respective sides. Supposing that Neumann conditions are given on the left side ($x = x_0$) and Dirichlet on the right side ($x = x_{N-1}$), the Poisson matrix and source vector accurate to second order are

$$A = \begin{bmatrix} -1 & 1 & 0 & \dots & 0 & 0 & 0 \\ 1 & -2 & 1 & \dots & 0 & 0 & 0 \\ 0 & 1 & -2 & \dots & 0 & 0 & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & 0 & 0 & \dots & -2 & 1 & 0 \\ 0 & 0 & 0 & \dots & 1 & -2 & 1 \\ 0 & 0 & 0 & \dots & 0 & 1 & -2 \end{bmatrix}, \quad \vec{S} = \begin{bmatrix} \frac{1}{2}\Delta x^2 f(x_0) + \Delta x \phi_0 \\ \Delta x^2 f(x_1) \\ \vdots \\ \Delta x^2 f(x_{N-3}) \\ \Delta x^2 f(x_{N-2}) - \phi_{N-1} \end{bmatrix}, \tag{2.79}$$

where the Dirichlet condition at $x = x_{N-1}$ must be appended to the solution vector $\vec{\phi}$. Matrix inversion of A may be used to obtain the solution, which needs be done only once in the course of the Vlasov1D1V solution algorithm.

2.5.4 Periodic boundary conditions

In periodic boundary conditions the domain becomes topologically a torus so that x_0 neighbors x_{N-1}. In this case, a simple substitution of ϕ_{N-1} for ϕ_{-1} and ϕ_0 for ϕ_N yields the

periodic Poisson matrix A with source vector $\vec{S}$,

$$A = \begin{bmatrix} -2 & 1 & 0 & \dots & 0 & 0 & 1 \\ 1 & -2 & 1 & \dots & 0 & 0 & 0 \\ 0 & 1 & -2 & \dots & 0 & 0 & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & 0 & 0 & \dots & -2 & 1 & 0 \\ 0 & 0 & 0 & \dots & 1 & -2 & 1 \\ 1 & 0 & 0 & \dots & 0 & 1 & -2 \end{bmatrix}, \quad \vec{S} = \begin{bmatrix} \Delta x^2 f(x_0) \\ \Delta x^2 f(x_1) \\ \vdots \\ \Delta x^2 f(x_{N-2}) \\ \Delta x^2 f(x_{N-1}) \end{bmatrix}. \tag{2.80}$$

While this problem is the simplest to derive, the matrix A is again singular.

2.5.5 Solution of singular Poisson matrices

It was seen in the cases of Neumann and periodic boundary conditions that the finite difference matrices were singular. Further it was noted that the problem is a lack of reference value for the potential. There exist some methods which create a fixed point for the potential, a sort-of interior Dirichlet boundary condition. A simpler method is described here which augments the system with an additional variable and equation by fixing the mean value of the potential. If $\Phi = \sum_{i=0}^{N-1} \phi_i$, the mean potential $\phi(x)$, is used as an additional variable and

$$\Phi = \sum_{i=0}^{N-1} \phi_i = 0 \tag{2.81}$$

used as an additional equation, then an augmented non-singular system may be created. A reference value is given as the potential $\phi(x)$ must either be identically zero or cross zero in order to have a zero mean. A row of ones is added to the matrix with the mean condition. In order to keep the matrix square and invertible, a column of ones is also added, which does not augment the equations as the variable Φ is identically zero. The augmented system may be written

$$\begin{bmatrix} A & 1_{N\text{x}1} \\ 1_{1\text{x}N} & 0 \end{bmatrix} \begin{bmatrix} \vec{\phi} \\ \Phi \end{bmatrix} = \begin{bmatrix} \vec{S} \\ 0 \end{bmatrix}. \tag{2.82}$$

2.6 Calculation of Moments

In solution of the Vlasov-Poisson system, the electric field $-\nabla\phi$ must always be self-consistent with the distribution function $f(x,v)$. Consistency is maintained through Poisson's equation, which requires that the zeroth moment $n_\alpha(x) = \int_{-\infty}^{\infty} f_\alpha(x,v)dv$ be calculated for both species. Further, in order that the BGK operator be applied, the first (mean velocity $\langle v \rangle$) and second (thermal velocity v_{th}) distribution moments must also be calculated as part of the solution algorithm. Since the distribution data is given as a grid of nodal values, the moment integrations may actually be calculated exactly using the method of Gauss-Lobatto numerical quadrature.

2.6.1 Gauss-Lobatto quadrature

Gaussian quadrature is a method of numerical integration whereby definite integrals are approximated by summation of sampled points v_i with corresponding weights w_i,

$$\int_{-1}^{1} f(v)dv = \sum_{i=1}^{n} w_i f(v_i). \tag{2.83}$$

If the weights are appropriately chosen, the scheme often produces exact integrations of polynomials. When the sample locations v_i correspond to the nodal locations described as in Fig. 2.3, the method is referred to as a Gauss-Lobatto quadrature. A table online should be consulted for quadrature weights for different orders. For third order elements with isoparametric nodal positions at -1, 0, and 1, the weights are respectively $\frac{1}{3}$, $\frac{4}{3}$, and $\frac{1}{3}$.

To calculate the moments with discontinuous elements, the integral

$$n(x) = \int_{-\infty}^{\infty} f(x,v)dv \approx \int_{v_{min}}^{v_{max}} f(x,v)dv, \tag{2.84}$$

is evaluated over the square elements. As the solution is continuous only within each element, the inner integral is split over each element in the velocity axes,

$$\int_{v_{min}}^{v_{max}} f(x,v)dv = \sum_{i=1}^{N_v} \int_{v_{min,i}}^{v_{max,i}} f(x,v)dv. \tag{2.85}$$

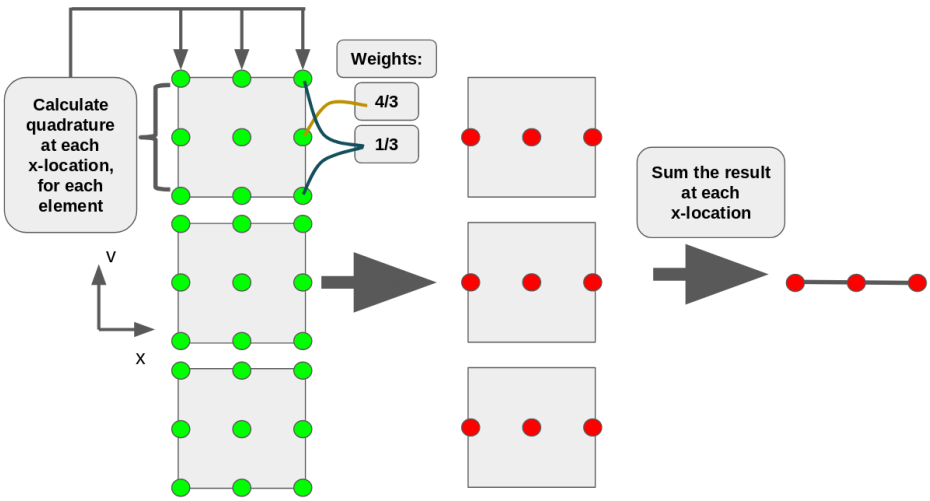


Figure 2.9: Graphic depicting implementation of numerical integration across one dimension of a grid of third-order discontinuous elements. The moment taken can be to any order, in which case the value $f(v_i)$ chosen should simply correspond to that moment $(v_i f(v_i))$ for example.

Now each of these integrals is evaluated with a three-point quadrature and the result summed. Figure 2.9 depicts this procedure for an array of third-order elements. It must be noted that when transforming from the interval $[v_{min,i}, v_{max,i}]$ to $[-1, 1]$ for an integral over the $i'th$ element, the result is

$$\int_{v_{min,i}}^{v_{max,i}} f(x,v)dv = \frac{v_{max,i} - v_{min,i}}{2} \int_{-1}^{1} f(x, \frac{v_{max,i} - v_{min,i}}{2} v + \frac{v_{max,i} + v_{min,i}}{2})dv. \tag{2.86}$$

Then the quadrature may be applied with a multiplicative modifier $\frac{\Delta v}{2}$. Using a three-point quadrature rule, the integration may be carried out as

$$\int_{v_{min,i}}^{v_{max,i}} f(x,v)dv = \frac{\Delta v}{6}\Big(f(x,v_{min,i}) + 4f(x,v_{mid,i}) + f(x,v_{max,i})\Big) \tag{2.87}$$

so that upon summing the result at each x-station, the density is given by

$$n(x) = \frac{\Delta v}{6} \sum_{i=1}^{N_v} \Big(f(x,v_{min,i}) + 4f(x,v_{mid,i}) + f(x,v_{max,i})\Big) \tag{2.88}$$

The first and second moments are normalized to the distribution density, and the variance v_{th} is calculated with respect to the distribution mean. Their formula are then given by

$$\langle v\rangle(x) = \frac{1}{n(x)}\int_{-\infty}^{\infty} v f(x,v)dv \tag{2.89}$$

$$v_{th}^2(x) = \frac{1}{n(x)}\int_{-\infty}^{\infty} (v-\langle v\rangle(x))^2 f(x,v)dv \tag{2.90}$$

so that the quadrature rules give, using the above procedure,

$$\langle v\rangle(x) = \frac{\Delta v}{6n(x)}\sum_{i=1}^{N_v}\Big(f(x,v_{min,i})v_{min,i} + 4f(x,v_{mid,i})v_{mid,i} + f(x,v_{max,i})v_{max,i}\Big) \tag{2.91}$$

$$v_{th}^2(x) = \frac{\Delta v}{6n(x)}\sum_{i=1}^{N_v}\Big(f(x,v_{min,i})(v_{min,i}-\langle v\rangle(x))^2 + 4f(x,v_{mid,i})(v_{mid,i}-\langle v\rangle(x))^2 \tag{2.92}$$

$$+ f(x,v_{max,i})(v_{max,i}-\langle v\rangle(x))^2\Big). \tag{2.93}$$

The species temperature may be determined from the thermal velocity as $kT = mv_{th}^2$.

2.6.2 Steps to ensure an efficient calculation

The run time for subroutines of a finite element algorithm should be kept as short as possible, so parallelization of the moment calculations is important for computational efficiency. The integration within each element can easily be blocked out and run in CUDA. The result is an array sized as the grid size by the element order, such as in step 2 of Fig. 2.9. Summation of the array at each x position is tricky to do as information in neighboring blocks is required. This kind of parallel operation is termed a reduction. In Python the integrations may be performed using CUDA and the reduction done using vectorized operations from the Numpy package.

Python code example 3: efficient moment calculation

This example describes parallelized calculation of the first moment, mean velocity $\langle v\rangle$. As in the code example 2 on calculation of the semi-discrete equation, here a set-up function is required to send the arrays required for moment calculation to the GPU, call the kernel

function, and reduce the arrays using a vectorized summation function. First presenting the kernel function, we implement Eqn. 2.91 in a jit-decorated function. Here "d_f" is the distribution array, "d_n" is the density $n(x)$, "d_v" is the velocity coordinate array, "d_u" is the mean velocity target array, and "dv"=Δv is the element width in velocity.

```
@cuda.jit
def firstMomentKernel(d_f, d_v, d_n, d_u, dv):
    i,j = cuda.grid(2)

    if (i > d_f.shape[0] or j > d_f.shape[1]):
        return

    # Integrate using gauss-lobatto quadrature
    d_u[i,j,0] = (dv/(6.0*d_n[i,0]))*(d_f[i,j,0]*d_v[j,0] + d_f[i,j,6]*d_v[j,2]
                                      + 4.0*d_f[i,j,3]*d_v[j,1])
    d_u[i,j,1] = (dv/(6.0*d_n[i,1]))*(d_f[i,j,1]*d_v[j,0] + d_f[i,j,7]*d_v[j,2]
                                      + 4.0*d_f[i,j,4]*d_v[j,1])
    d_u[i,j,2] = (dv/(6.0*d_n[i,2]))*(d_f[i,j,2]*d_v[j,0] + d_f[i,j,8]*d_v[j,2]
                                      + 4.0*d_f[i,j,5]*d_v[j,1])
```

The helper function copies arrays to the device, creates the device arrays, sets up the blocks of the grid, calls the kernel function, and reduces the phase space grid to a 1D grid (i.e. the summation step of Eqn. 2.91). Reduction is done through Numpy's "sum" function.

```
def firstMoment(f, v, n, order):
    # Grid dimensions
    nx = f.shape[0]
    nv = f.shape[1]

    # Send input arrays to device
    d_f = cuda.to_device(f)
    d_v = cuda.to_device(v)
    d_n = cuda.to_device(n)
```

```
# Create output array on device
d_u = cuda.device_array((nx,nv,order))

# Grid parameters
blockSize = (TPB, TPB)
numBlocksX = (nx + blockSize[0] - 1)//blockSize[0]
numBlocksV = (nv + blockSize[1] - 1)//blockSize[1]
numBlocks = (numBlocksX, numBlocksV)

# Element width
dv = v[0,2] - v[0,0]

### Calculate the first moment
# Element-wise
firstMomentKernel[numBlocks, blockSize](d_f, d_v, d_n, d_u, dv)
# Return to host
u = np.empty(shape=d_u.shape, dtype=d_u.dtype)
d_u.copy_to_host(u)
u_return = np.zeros((nx,order))
# Reduce to 1D
for i in range(nx):
    u_return[i,0] = np.sum(u[i,:,0])
    u_return[i,1] = np.sum(u[i,:,1])
    u_return[i,2] = np.sum(u[i,:,2])

return u_return
```

2.7 Data Output Using HDF5

It is good practice to save data along with a file listing the input parameters. Data are saved for repeated post-processing runs, while a saved input file allows reproduction by yourself or other programs. A powerful method for saving many arrays in one dataset is called Hierarchical Data Format 5, or HDF5. Arrays may be saved in a dataset of essentially

arbitrary dimension within string-labelled "slabs". The Python package which allows reading and writing to HDF5 files is called h5py. This section concisely describes how to create an HDF5 data file from Python, save data to it, and load it later for post-processing.

2.7.1 Python code example 4: formatting data file using h5py

To save data, the data file must be created and formatted. This is done by specifying a dataset within the file with a string label, the size of the array, and the type of the data to be saved. Dynamic sizing of the data file is possible, but it is much simple to specify the size as the number of times data will be saved.

```
import h5py

# Set h5 filename
data_name = 'test_file.hdf5'
# Create file
hdf5_data = h5py.File(data_name, "w") # "w" to open for write
# Specify dataset with label, size, and data type
# arguments are: label as a string
#                dimension, (X size, V size, Nodes, Timesteps)
#                data type, "f" for floating point
# Multiple datasets may be specified of different sizes;
# First here is distribution and second is electric potential
dset_pdf = hdf5_data.create_dataset("pdf", (Nx, Nv, N, Nt//Nsave), dtype="f")
dset_phi = hdf5_data.create_dataset("phi", (Nx, order, Nt//Nsave), dtype = "f")
# Make some data here, ex: f0 as initial distribution, phi0 as initial potential
# Save into the file
dset_pdf[:, :, :, 0] = f0
dset_phi[:, :, 0] = phi0
```

This example demonstrates how to set the filename, create the file, specify multiple datasets of different sizes, and save data into the file.

2.7.2 Python code example 5: reading the HDF5 data file

The data within the HDF5 file is labelled by the key string provided when the dataset was specified. Somewhat surprisingly, it is a little more difficult to read data from the HDF5 file than it is to write to it. After loading the data file, the label strings are stored as a list of keys which should be printed out and inspected in order to identify the order that the arrays were saved in. Then the arrays may be extracted using the key data.

```
import h5py

# Load file
hdf5_data = h5py.File('test_file.hdf5', "r+") # r+ to open for reading
# obtain list of keys
key_list = list(hdf5_data.keys())
# Print and inspect list of keys
print(key_list)
# Obtain key data from the list
key_data_pdf = key_list[0]
key_data_phi = key_list[1]
# Read data
data_pdf = hdf5_data[key_data_pdf].value
data_phi = hdf5_data[key_data_phi].value
```

This concludes how to read the data file. The syntax to retrieve the data arrays using a list of key objects is valid as of time of writing. Official h5py documentation should be consulted for possible changes to the syntax for future versions.

2.8 Physics Considerations: Two-Species Vlasov-Poisson Normalization

A normalization must be chosen for the Vlasov-Poisson system in order for the results to reflect relevant physics. The Computational Plasma Dynamics laboratory at University of Washington uses a scheme which normalizes all equations to the fundamental proton scales [23]. The idea is that a plasma state is chosen, and the subsequent plasma parameters such

as density, proton plasma frequency, proton skin depth, etc. are used for reference quantities.

2.8.1 Proton-centered normalization

The plasma state is chosen in terms of a triplet of values which fixes a basis state. In this thesis, the basis values are a reference plasma density n_0, temperature T_0 (in electronvolts), and length L_0. The length scale L_0 is chosen to be the Debye length λ_D, which is fixed by the plasma density and temperature. The reference time is then fixed as the Debye length transit time τ by an average thermal proton, and the reference velocity v_0 fixed as the proton thermal velocity. With normalized variables given tildes,

$$\begin{aligned} f_\alpha &= \widetilde{f}_\alpha n_0 & L &= \lambda_D = \sqrt{\frac{\epsilon_0 T_0}{n_0 e}} & v_0 &= v_{th} = \sqrt{\frac{eT_0}{m_p}} & \tau &= \frac{L}{v_0} \\ t &= \widetilde{t}\tau & x &= \widetilde{x}L & q_\alpha &= Z_\alpha e & m_\alpha &= A_\alpha m_p \\ \omega_p &= \sqrt{\frac{e^2 n_0}{\epsilon_0 m_p}} & \delta_p &= \frac{c}{\omega_p} & \nu_\alpha &= \widetilde{\nu}_\alpha \nu_p & E_0 &= \frac{T_0}{\delta_p} \end{aligned} \tag{2.94}$$

represents the set of important quantities used in the Vlasov-Poisson normalization. Under this scheme, the normalized kinetic equations for each species α (with applied BGK operator) are

$$\frac{\partial \widetilde{f}_\alpha}{\partial \widetilde{t}} + \widetilde{v}\frac{\partial \widetilde{f}_\alpha}{\partial \widetilde{x}} + \Big(\frac{L}{\delta_p}\Big)\frac{Z_\alpha}{A_\alpha}\widetilde{E}\frac{\partial \widetilde{f}_\alpha}{\partial \widetilde{v}} = (\nu_p\tau)\widetilde{\nu}_\alpha(\widetilde{f}_\alpha^M - \widetilde{f}_\alpha), \tag{2.95}$$

with $E = -\frac{d\phi}{dx}$ and the normalized Poisson equation given by

$$-\frac{1}{(\omega_p\tau)^2}\Big(\frac{L}{\delta_p}\Big)\frac{d^2\widetilde{\phi}}{dx^2} = \sum_\alpha Z_\alpha \int_{-\infty}^{\infty} \widetilde{f}_\alpha(\widetilde{v})d\widetilde{v}. \tag{2.96}$$

By setting the reference length to the Debye length λ_D it's simple to choose the number of Debye lengths within the domain. Another important quantity which must be chosen is the electron-ion temperature ratio T_e/T_i, which allows one to set the electron thermal velocity. Since the maximum velocity in the electron grid usually sets the timestep restriction by the CFL condition, as it is the region of maximum velocity flux, it's desirable to keep $T_e/T_i \sim 1$ for most problems.

2.8.2 Choosing a basis triplet

The important quantities in the normalized equations are the weightings on the acceleration of each species and the weighting of the charge density in Poisson's equation. They are not independent of the choice of reference state. To be precise, the proton acceleration weight is $\frac{\lambda_D}{\delta_p}$, the electron weight is $\frac{-1}{A_e}\frac{\lambda_D}{\delta_p}$, and the weighting on the charge density is $-(\omega_p\tau)^2\frac{\delta_p}{\lambda_D}$.

Now, the skin depth δ_p is always larger than the Debye length λ_D, so that the proton weighting $\frac{\lambda_D}{\delta_p} < 1$, and the electron mass ratio is $A_e = \frac{1}{1836}$ so that the electron weighting is often greater than one and always larger than the proton acceleration coefficient. However, the magnitude of the acceleration weighting is $\frac{\lambda_D}{\delta_p} \sim \frac{v_0}{c} \sim \sqrt{T}$, so that higher temperature increases the acceleration coefficient.

On the other hand, the charge density coefficient goes like $(\omega_p\tau)^2\frac{\delta_p}{\lambda_D} \sim \frac{1}{\sqrt{T}}$ as the normalized proton plasma frequency is unity in the chosen normalization scheme. Then increasing the temperature tends to decrease the electric potential from a given separation in charge. Since the field depends on gradients in the potential and the typical scale is $\sim \lambda_D$, $E \sim \frac{\Delta\phi}{\lambda_D} \sim \frac{T^{-1/2}}{\sqrt{T/n}} \sim \frac{n^{1/2}}{T}$. Combining this estimate with the acceleration coefficient shows that the accleration magnitude scales like $\sqrt{\frac{n}{T}} \sim \lambda_D^{-1}$. This can be used to fine tune the plasma response if one is having trouble with plasma being accelerated to very high velocity and exiting the velocity space in a given problem.

2.8.3 Determination of the BGK collision frequency

The formula for estimation of the plasma collision frequencies are standard and presented in most text books. The general principle is that the collision frequency of a species α varies like $\nu_\alpha \sim \frac{n}{T_\alpha^{3/2}}$. For proton-electron plasma, the electron-electron and ion-ion Coulomb collision frequencies may be estimated by [8]

$$\nu_e \sim (5 \times 10^{-11}) n T_e^{-3/2}, \tag{2.97}$$

$$\nu_i \sim (1 \times 10^{-12}) n T_i^{-3/2}. \tag{2.98}$$

In the normalization used, the reference proton collision frequency can then be estimated as

$$\nu_p \sim (1 \times 10^{-12}) n_0 T_0^{-3/2} \tag{2.99}$$

and therefore the local normalized collision frequencies for use in the BGK operator should be determined by the expressions

$$\widetilde{\nu}_p \sim \widetilde{n}_p \widetilde{T}_p^{-3/2} \tag{2.100}$$

where n_p and T_p are both normalized quantities (with values around 1), and

$$\widetilde{\nu}_e \sim 50 \widetilde{n}_e \widetilde{T}_e^{-3/2}. \tag{2.101}$$

It's most accurate to apply these formula locally. For common choices of basis triplet the reference time τ is very small, so that $(\nu_p \tau)\widetilde{\nu}_\alpha$ can be a very small quantity for the ions. The basis triplet should be adjusted for larger reference times to investigate problems with varying degrees of collisionality in physically accurate problems. Of course the collision frequency for either species may also be set arbitrarily depending on what is desired.

2.8.4 Effect of the proton-electron mass ratio A_e

The physical value of the proton-electron mass ratio is $A_e = \frac{m_e}{m_p} \sim \frac{1}{1836}$. An important consequence of this mass ratio is that the electron thermal velocity $v_{th,e}$ is approximately 42.8 times greater than the ion thermal velocity. As mentioned earlier, this results in a severely limited timestep from the perspective of proton dynamics. If it is desirable to study the dynamics of ions for many proton plasma periods, then this timestep limitation can be alleviated by artifically decreasing the proton-electron mass ratio. This should be avoided in general as it produces unphysical results, but does better allow the study of ion-acoustic wave problems, for example, while maintaining electron dynamics. All results presented in this thesis are done with the physical value of the mass ratio, $A_e = 1836$.

2.9 Program Validation

Implementation of a physics code should be validated through replication of correct results for a problem with well-known theoretical results. Perhaps the simplest test case for a 1D1V collisionless plasma kinetic program is linear Landau damping of electron plasma waves. Through contour integration of the Fourier-Laplace transformed linearized Vlasov-Poisson system, the frequency of a monochromatic plane wave $\exp\Big(i(\omega t - kx)\Big)$ with $\omega \sim \omega_e$ is approximately [8]

$$\omega(k) = \omega_e - i\frac{1}{2}\Big(\frac{\pi}{2}\Big)^{1/2}\omega_e(k\lambda_D)^{-3}\exp\Big(-\frac{1}{2}(k\lambda_D)^{-2}\Big). \tag{2.102}$$

Inserting this frequency into the plane wave expression, a term $\exp\Big(-\gamma t\Big) = \exp\Big(-\gamma\tau\widetilde{t}\Big)$ appears giving a damping decrement

$$\gamma\tau = \frac{1}{2}\Big(\frac{\pi}{2}\Big)^{1/2}\omega_e\tau(k\lambda_D)^{-3}\exp\Big(-\frac{1}{2}(k\lambda_D)^{-2}\Big). \tag{2.103}$$

Figure 2.10 demonstrates this function normalized to the electron plasma frequency $\omega_e\tau$. A more in-depth discussion of Landau damping of Langmuir waves is contained in Chapter 3.

2.9.1 Simulation result

For validation, the Vlasov1D1V program was run with periodic boundary conditions with a domain length and perturbation wavelength of $10\lambda_D$. The maximum velocity extent was set to six electron thermal velocities. As the code is intended to be a two-species continuum-kinetic program, the normalization was kept to that described in the above section with a physical proton-electron mass ratio and electron-ion temperature ratio of $T_e/T_i = 1.0$. That is, the electron plasma frequency was $\omega_e\tau = 42.8$ and thermal velocity $v_{th,e} = 42.8$. The BGK collision frequency was set to $\widetilde{\nu}_e = 0$ so as not to interfere with the collisionless damping process. The basis state was chosen with $n_0 = 10^{12}[\mathrm{m}^{-3}]$ and $T_0 = 10^4[\mathrm{eV}]$.

The electron distribution was initialized with a density of $n_e = (1.0 + \alpha\cos(\frac{2\pi}{10\lambda_D}x))$ with a perturbation of $\alpha = 0.01$ applied to obtain a linear result. The program was run with

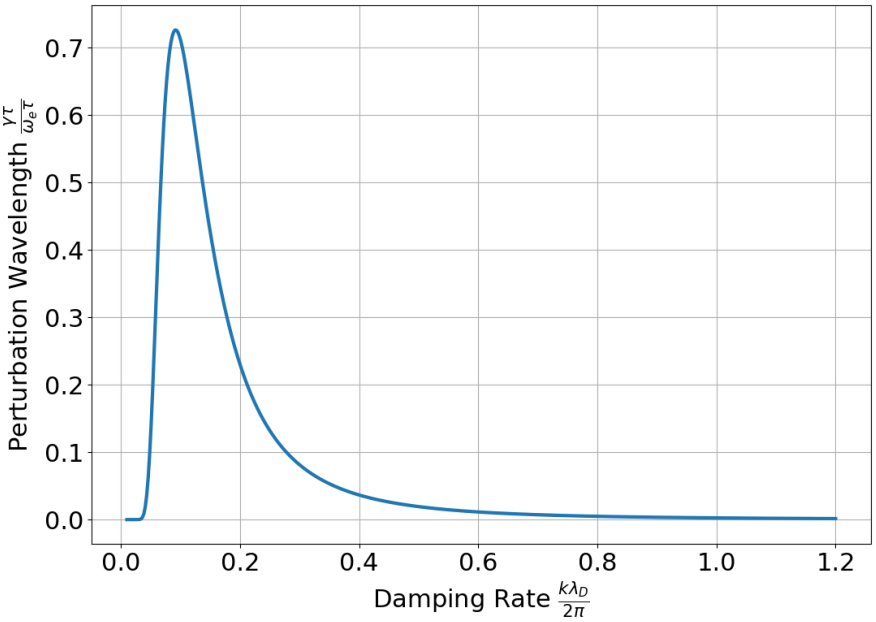


Figure 2.10: Linear Landau damping decrement as a function of wavenumber. The maximum damping rate is at a wavelength of approximately $11\lambda_D$.

128x128 element phase space resolution, with third-order RKDG discretization and a CFL number of CFL $= \frac{0.95}{5} = 0.19$ (close to the stability ceiling). Figure 2.11 shows the integrated electric field energy of the Langmuir wave in time while Fig. 2.12 shows on an (x, t) diagram how the wave amplitude decreases in time. The logarithmic slope between field energy peaks gives the simulation's damping decrement to be $\gamma\tau \approx 31$. Evaluation of Eqn. 2.103 with $\frac{k\lambda_D}{2\pi} = 0.1$ gives $\gamma\tau \approx 30.5$. Since Eqn. 2.103 is an approximate formula, these numbers are in reasonable agreement.

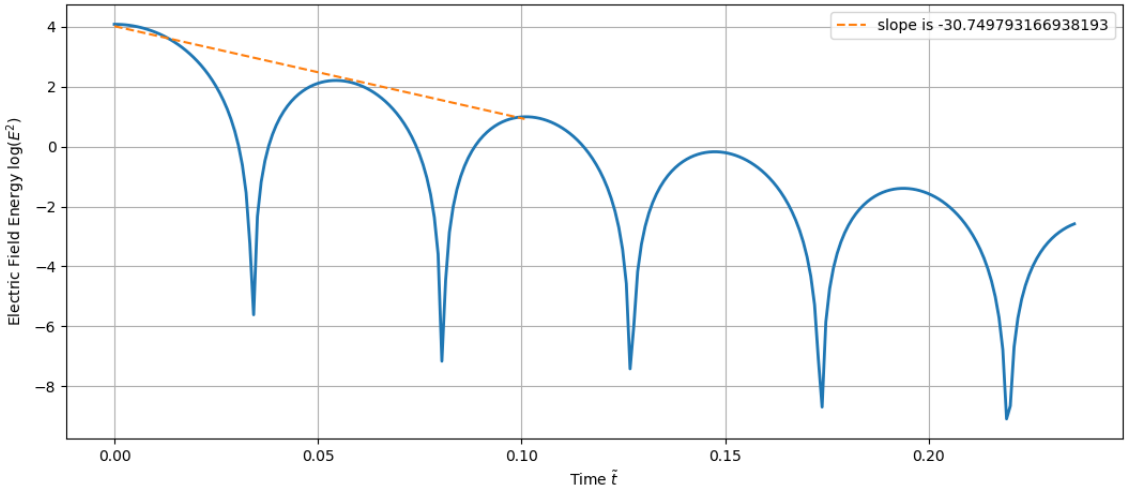


Figure 2.11: Simulated decay of Langmuir wave energy due to collisionless Landau damping for program validation.

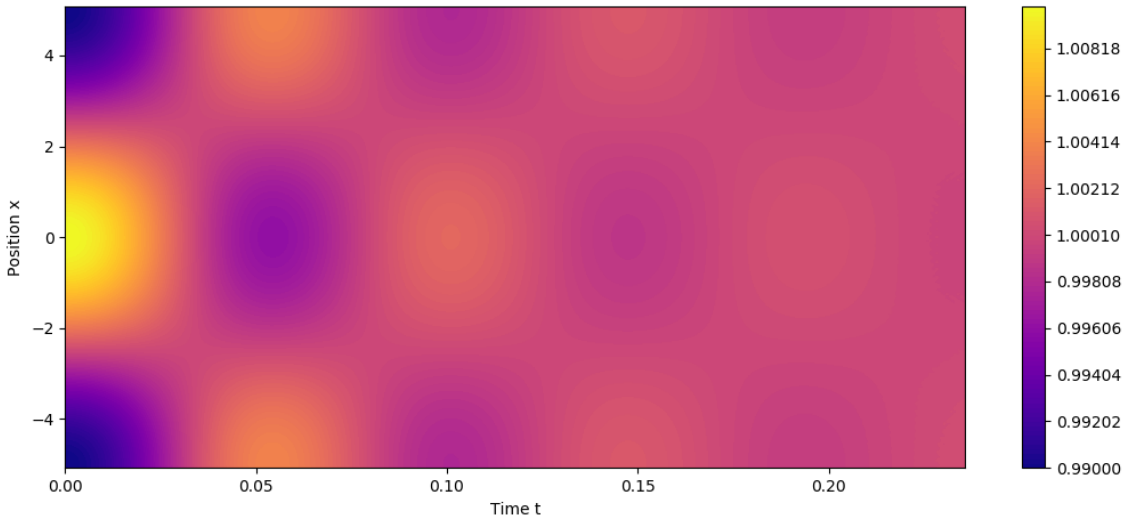


Figure 2.12: Landau damping of a electron density perturbation of wavelength $\lambda = 10\lambda_D$. Decay of density peaks shown in an (x, t) diagram.

Chapter 3

EXPLORING COLLISIONLESS PLASMA PHYSICS THROUGH SIMULATION

This chapter contains simulations of classic problems in collisionless plasma physics using the one-dimensional Vlasov-Poisson solver described in the previous chapter. After a review of theoretical elements of the physics to be explored, the simulation results are presented. The intention behind the problems presented is to build intuition for important concepts in the physics of collisionless plasma such as linear and nonlinear Landau damping, resonant excitation of plasma waves, two-species coupling in plasma waves, and plasma streaming instabilities.

3.1 A Survey of Analytical Methods and Intuitive Principles

As suggested in Chapter 1, a great deal of analytical work has been done on the foundations of collisionless plasma physics. This section will review those results necessary in order to understand the simulation results to be presented in the following sections. The review consists of first a discussion of collisionless wave damping or absorption followed by principles of wave emission as a consequence of kinetic plasma instability. Results are a condensed form of the presentation in [17], one of the best introductory texts for plasma kinetic theory.

3.1.1 Obtaining dispersion relations for waves in hot plasma

A dispersion relation is a functional relationship $\omega(\vec{k})$ between the frequency ω and wavevector $\vec{k}$ of a wave. It is of fundamental importance to the propagation of a wave-packet consisting of a spectrum of spatial frequencies k with various amplitudes, as it may be shown that the phase velocity of wave components goes as $v_{ph}(k) = \frac{\omega}{k}$ and the group velocity of

the wave packet like $v_g(k) = \frac{\partial \omega}{\partial k}$. Nonlinear dispersion relations mean that wave-packets are modulated and spread out (disperse). Plasma is in general a dispersive medium for waves.

The dispersion relation of small-amplitude electrostatic waves in a plasma may be determined through linearization of the Vlasov-Poisson system. Considering the ion population to be static, the electron distribution may be expanded to first order $f_e = f_0 + f_1$ and the electric field response taken as first order $E = E_1$. Linearization of the electron Vlasov equation, Eqn. 1.14, gives

$$\frac{\partial f_1}{\partial t} + v\frac{\partial f_1}{\partial x} - \frac{e}{m_e}E_1\frac{\partial f_0}{\partial v} = 0 \tag{3.1}$$

so that with a trigonometric form of the perturbation, $f_1 \sim \widetilde{f_1}\exp\Big(-i(\omega t - kx)\Big)$, one may solve to find the perturbed distribution amplitude to be

$$\widetilde{f_1} = \frac{ie}{m_e(\omega - kv)}E_1\frac{\partial f_0}{\partial v}. \tag{3.2}$$

Linearization of Gauss's law around an unperturbed plasma density n_0 gives

$$\frac{dE_1}{dx} = \frac{e}{\epsilon_0}(n_i - n_e) = -\frac{e}{\epsilon_0}n_1 = -\frac{e}{\epsilon_0}\int_{-\infty}^{\infty} f_1 dv \tag{3.3}$$

so that with $E_1 \sim \widetilde{E}_1\exp\Big(-i(\omega t - kx)\Big)$, elimination of E_1 using the form of f_1 yields the implicit dispersion relation

$$1 + \frac{\omega_e^2}{k^2}\int_{-\infty}^{\infty}\frac{1}{n_0}\frac{\partial f_0}{\partial v}\frac{1}{(\frac{\omega}{k} - v)}dv = 0. \tag{3.4}$$

Note that Eqn. 3.4 has a singular integrand for $v = \frac{\omega}{k}$ in a form known as a *resonant denominator*, which may be evaluated by means of contour integration. Its physical interpretation is that a resonance occurs between the wave with phase velocity $v_{ph} = \frac{\omega}{k}$ and particles traveling at that velocity. These resonant particles experience a constant amplitude field E because they are moving in phase with the wave. A more general result including ion motion is presented in [17].

The resonant portion of the distribution $f(v)$ may be ignored by expanding the denominator to second order about $v = 0$. Supposing the unperturbed distribution f_0 to be

Maxwellian, the expanded Eqn. 3.4 may be integrated to obtain the linear dispersion relation for electron plasma waves,

$$\omega^2 = \omega_e^2 + 3k^2 v_{th,e}^2 = \omega_e^2(1 + 3(k\lambda_D)^2). \tag{3.5}$$

Equation 3.5 is known variably as the Langmuir wave dispersion relation, or the Bohm-Gross dispersion relation after the authors who first obtained it in 1949 [4].

3.1.2 Landau damping and the phase mixing paradigm

The true dispersion relation of electron plasma waves must consider the resonant particles for which $v = \frac{\omega}{k}$. This is done rigorously in textbooks and results in an imaginary component to the dispersion relation Eqn. 3.5. An approximate expression for this imaginary part in the case of a Maxwellian distribution f_0 was given in Eqn. 2.103. As wave energy damps the plasma temperature increases, or in other words the distribution acquires a larger thermal spread v_{th}.

Of real interest here is the physical mechanism of Landau damping. An important first notion is that of *phase mixing*. A particle's position in the phase plane is often referred to as its phase. Consider a collisionless system with no mean field, and suppose that a slug of fluid is placed in the phase space. As time evolves, the slug appears to rapidly diffuse as viewed in the physical space, as its distribution rapidly falls out of phase with itself. The phase space picture of simple phase mixing is illustrated in Fig. 3.1. Density gradients are naturally smoothed out due to filamentation in phase space.

The electric field couples waves to the Vlasov phase fluid so that the simple phase mixing picture is inadequate. The effect of wave-particle resonance is that, due to the Doppler shift of the wave frequency as seen by particles with a velocity v, those particles slightly faster than the phase velocity v_{ph} give up energy to the wave while those slightly slower absorb wave energy. If the slope $\frac{\partial f_0}{\partial v} < 0$ near v_{ph}, as it is for a Maxwellian f_0 for a wave with $v_{ph} > 0$, then wave energy is lost and the wave is damped. The effect of phase mixing is that these wave resonant particles then carry their energy away from the wave [19]. By this

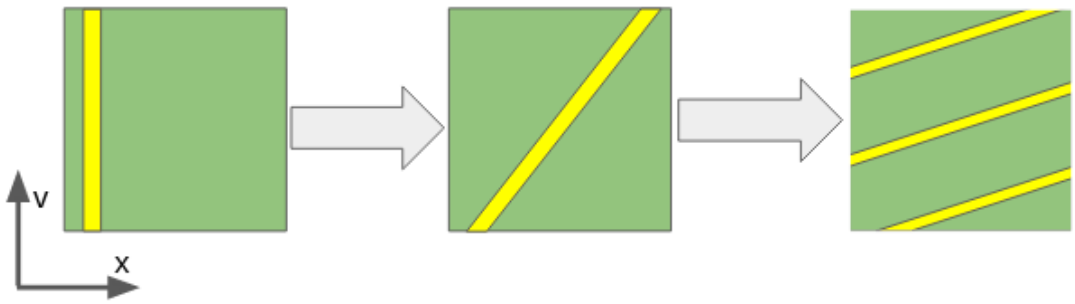


Figure 3.1: Illustration of phase mixing where a lump of phase fluid undergoes filamentation, falling out of phase with itself until it is spread evenly over the phase space.

means the wave is Landau damped. Recalling the results of Chapter 1, it is necessary that the phase portrait streamlines of the wave be "open", leading away from the wave, for the wave-resonant particles to phase mix the wave energy and damp the plasma wave.

3.1.3 Nonlinear Landau damping and Bernstein-Greene-Kruskal (BGK) modes

The supposition that wave-resonant particles exit the plasma wave following their interaction is true for linear (small-amplitude) waves. In the nonlinear regime, the change in trajectory of wave-resonant particles must be taken into consideration. The electric potential of an electron plasma wave acts as a potential well for electrons with energy low enough to be trapped within the well, so that resonant electrons begin oscillating within the wave potential.

Trapped particles are not able to exit the wave, so that their perturbed energies cannot phase mix into the plasma away from the wave. In this way, collisionless wave damping is arrested by a nonlinear effect in a class of phenomena usually referred to as nonlinear Landau damping. Particle trapping *stabilizes* plasma waves in a collisionless plasma, although the energy necessary to set the particles in motion does damp the waves somewhat. Regions of trapped particles within a plasma wave are characterized by phase space structures with a clear separatrix, filling out a region qualitatively similar to that of Fig. 1.3, so that trapped

particles phase mix with each other within a plasma wave but not with the exterior phase fluid.

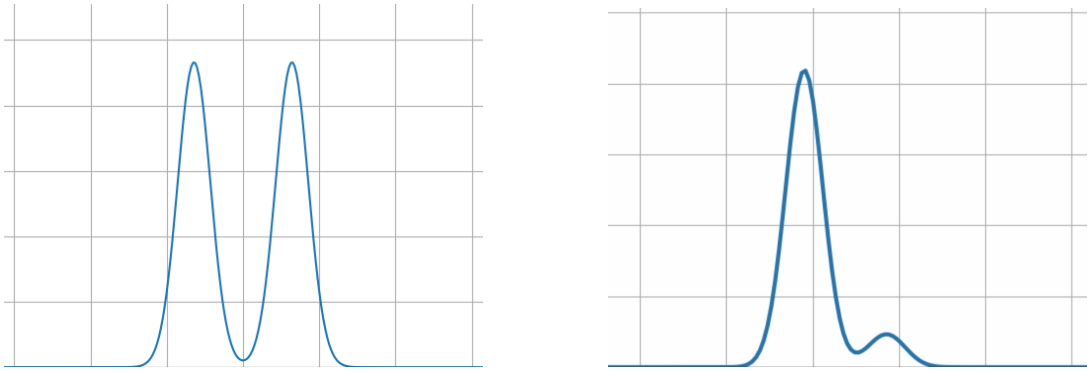
3.1.4 The Penrose stability criterion

The previous sections concerned the mechanism of collisionless absorption of wave energy by a thermal plasma, a process referred to as Landau damping. Plasma waves may also be emitted collisionlessly in the reverse process of Landau damping. Known as Landau resonance, the process occurs as a consequence of the instability of certain distributions. The criterion for marginal stability of a given distribution $f(x, v, t)$ was derived by Oliver Penrose in 1960, hence the result is called the Penrose stability criterion. Discussed in detail in references such as [17], the criterion states that spatially-averaged distributions which are too "lumpy" will excite plasma waves through wave-particle resonance. Emission saturates when the lumps (local maxima) are smoothed out.

Consider a spatially averaged distribution function, $\langle f \rangle(v)$. Dropping the average angle brackets for simplicity, the condition for linear instability is given by

$$\int_{-\infty}^{\infty} dv \Big[\frac{f(v) - f(v_{min})}{(v - v_{min})^2} \Big] > 0 \tag{3.6}$$

where v_{min} is the location of a local minima between two local maxima. For example, consider the distributions in Figs. 3.2a and 3.2b. In both cases the distribution is unstable under the condition Eqn. 3.6. Plasma waves will be excited at phase velocities ω/k where the lump distribution has a positive slope. Physically, lumps represent a thermal distribution of plasma which has drifted into another thermal distribution, and their superposition is seen. Collisionless plasma has the capacity to interpenetrate itself, with different thermal populations coexisting. The physical meaning of the Penrose criterion is that a form of dynamic frition exists between interpenetrating thermal populations which excites plasma oscillations as a means of exhausting the free energy present in the drifting particles.



(a) Two-stream unstable distribution. (b) Bump-on-tail unstable distribution.

Figure 3.2: Examples of spatially-averaged distributions unstable under the Penrose criterion. Both distributions are not in a state of thermodynamic equilibrium, where $f(v)$ has Maxwellian statistics.

3.1.5 Quasi-linear diffusion theory

The evolution of instability predicted by the Penrose criterion has linear and nonlinear regimes just like Landau damping. In the linear case, a useful picture for understanding the instability's evolution is provided by a framework called quasilinear diffusion, which casts the evolution of the spatially-averaged distribution as a diffusion equation, [8]

$$\frac{\partial \langle f \rangle}{\partial t} = \frac{\partial}{\partial v}\Big(D(v)\frac{\partial \langle f \rangle}{\partial v}\Big). \tag{3.7}$$

The form of the diffusivity $D(v)$ is not necessary for the purpose of this thesis. The diffusion analogy works best with a distribution such as Fig. 3.2b. Here the dip between the two local maxima is filled in by plasma wave emission at a rate well approximated by a diffusion equation until the distribution $\langle f \rangle(v)$ satisfies $\frac{\partial f}{\partial v} \leq 0$ for $v > v_{max}$ of the distribution peak. The flat, stabilized portion of the distribution represents in fact a projection of a BGK-like mode of trapped particles within the spectrum of waves emitted during evolution of the plasma instability. This idea is explored further in the simulation of the bump-on-tail instability section.

3.2 Landau Damping of One-Dimensional Electrostatic Plasma Waves

The collisionless damping of plasma waves is an important problem in plasma kinetic theory. This section presents the results of simulations conducted using the Vlasov1D1V program on the Landau damping of electrostatic plasma waves. Simulations of linear Landau damping are presented with an emphasis on the responsibility of damping due to phase mixing of resonant particles. Nonlinear Landau damping is then discussed by viewing the development of phase space structures.

3.2.1 Linear damping of long and short wavelength Langmuir waves

As suggested by the linear Landau damping decrement in Fig. 2.10, there is little damping of long-wavelength Langmuir waves. The reason for this can be understood in terms of wave-particle resonance. Whereas $v_{ph} = \frac{\omega}{k}$ becomes large for $k \ll 1$, the distribution slope $\frac{\partial f_0}{\partial v} \to 0$ for $v \to 0$. There are few particles to exchange energy with the wave and subsequently phase mix. Phase mixing still occurs for particles which exchange very little energy with the waves.

The structure within phase space due to phase mixing is more apparent for larger amplitude Langmuir waves, so now consider the problem of the Program Validation section with the perturbation amplitude α increased to $\alpha = 0.1$. The wave energy is damped in much the same way as that problem. Of interest is the structure of the phase space, shown in Fig. 3.3, after the wave energy has been damped through several periods of plasma oscillation. The filamented structure is characteristic of phase mixing.

It should be noted that it is only the resonant particles at the wave phase velocity ω/k which are reponsible for the wave damping. Figure 3.4 shows phase mixing in the distribution structure of a Langmuir wave of wavelength $\lambda = 50\lambda_D$. However, the high phase velocity at the lower perturbation wavenumber k leads to exponentially small damping from an absence of resonant particles as seen in Fig. 3.5.

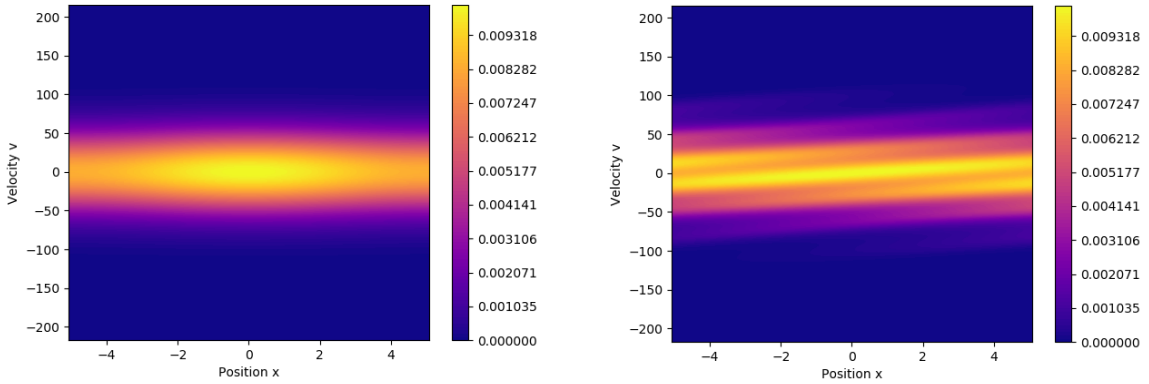


Figure 3.3: Demonstration of the phase-mixed structure of the electron distribution function following Landau damping of a Langmuir wave of wavelength $\lambda = 10\lambda_D$.

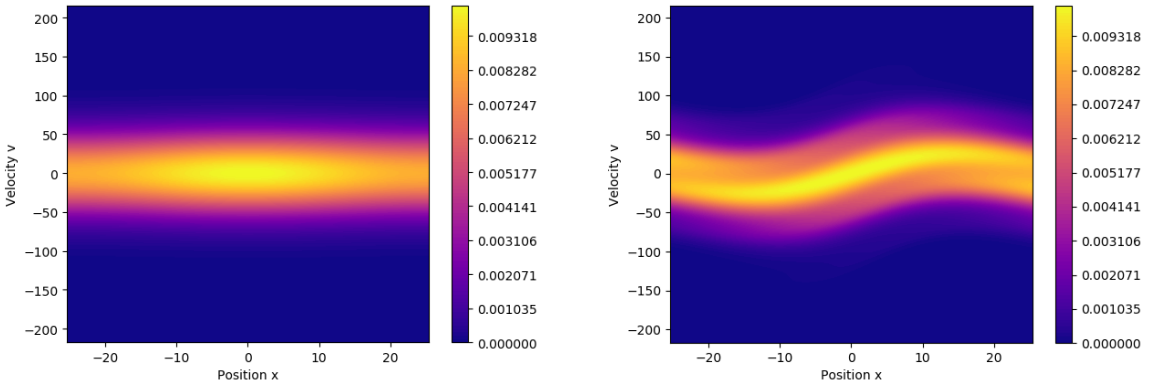


Figure 3.4: Development of phase space structure of the electron distribution function for a Langmuir wave of wavelength $\lambda = 50\lambda_D$ with weak Landau damping.

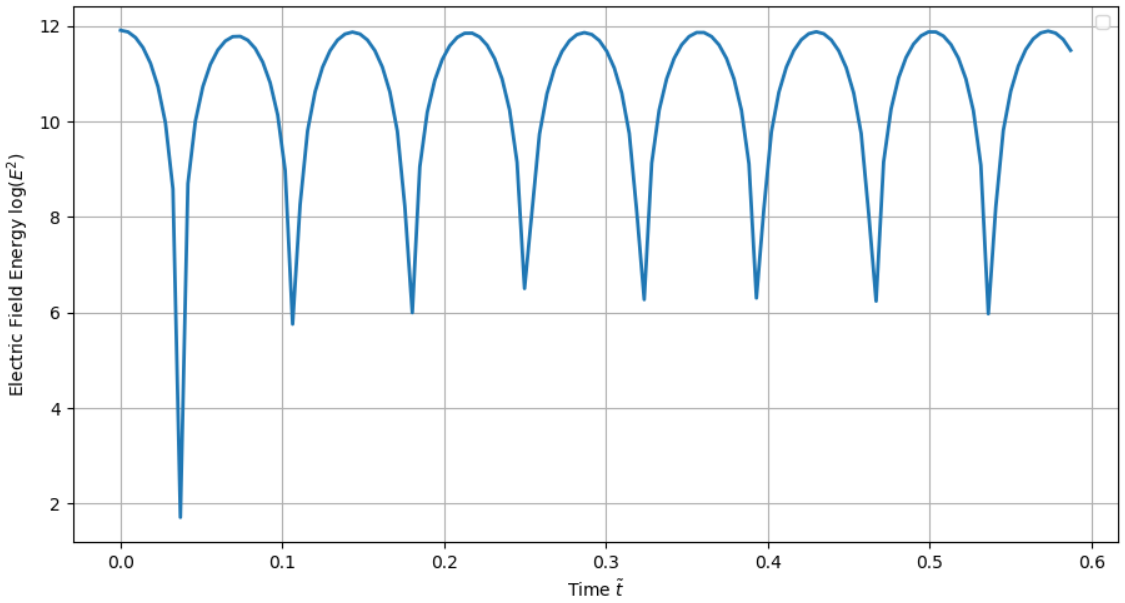


Figure 3.5: A Langmuir wave of wavelength $\lambda = 50\lambda_D$ experiences exponentially small Landau damping.

3.2.2 Nonlinear Landau damping and evolution of Langmuir waves

As discussed in section 3.1, nonlinear physics prevelant in the Landau damping process can lead to a reversal of Landau damping when a significant portion of wave-resonant particles become trapped in the plasma wave. Unable to carry their energy beyond the wave, they continue to exchange energy with the wave while phase-mixing within the separatrix of the plasma wave. This can be demonstrated clearly by increasing the perturbation amplitude α into a strongly nonlinear regime, such as $\alpha = 0.5$, although such a strong wave is not really realistic. With a wavelength of $\lambda = 15\lambda_D$ to make the phase velocity of the wave more obvious, Fig. 3.6 depicts the development of wave-trapped particles in the nonlinear Landau damping regime. Two counter-propagating waves are evident, with wave-trapped particles forming characteristic eye-shaped structures centered around the wave phase velocity ω/k. Figure 3.7 shows how the significant particle-trapping in the nonlinear regime leads to a reversal of the Landau damping of wave energy.

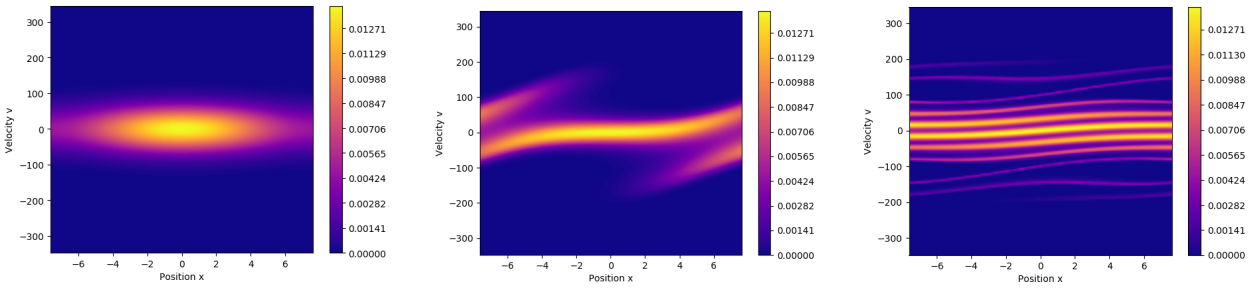


Figure 3.6: The development of BGK modes becomes evident in the Landau damping of strong perturbations, in the form of oval-shaped phase space structures outside of the bulk distribution. Here $\alpha = 0.5$ and the Langmuir waves are given a wavelength of $\lambda = 15\lambda_D$.

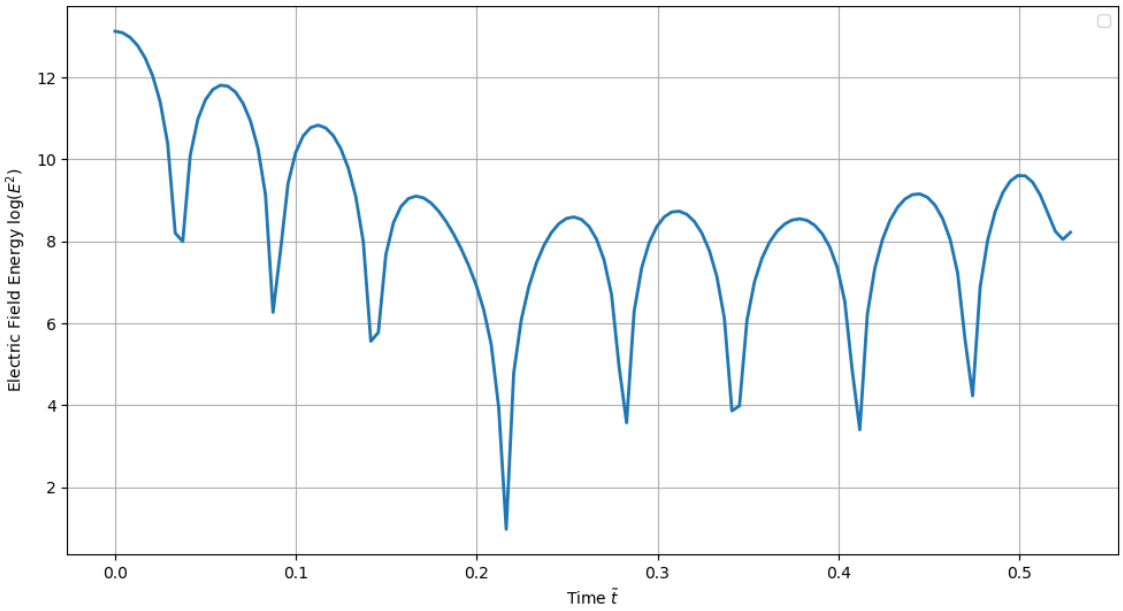


Figure 3.7: The development of particle trapping in the nonlinear regime of Landau damping halts the damping process.

3.3 Landau Resonant Excitation of Plasma Waves

This section presents the evolution of a collisionless plasma due to kinetic instability in the structure of its velocity space, as per the Penrose criterion. Two interesting problems are investigated primarily. First the bump-on-tail instability is studied with focus on the wave spectrum generated by the instabiity and its long time evolution. The second problem is that of the famous electron two-stream instability, the evolution of which produces a plasma structure of some interest known as an electron hole. Two-species effects in the two-stream are studied by including the response of a population of ions at the resultant velocity of the electron hole.

3.3.1 Theory and simulation of electron bump-on-tail instability

The instability resulting from a thermal bump on the tail of a Maxwellian electron distribution was of interest in the early study of plasma physics, as such distributions commonly arise. An example is when populations of electrons energized by some event in plasma stream along field lines, or when high energy electrons carry a current between electrodes in a plasma. The theory of plasma wave turbulence that develops as a result of this instability is developed in [21]. A summary of this theory follows.

As suggested by the Penrose criterion, a spectrum of waves are excited with wavenumbers between the limiting velocities of the unstable region. If v_1 and v_2 are the minimum and maximum velocities of the final stable flat region, then waves are excited in a spectrum of wavenumbers spanning $k_{min} = \frac{\omega_p}{v_2}$, $k_{max} = \frac{\omega_p}{v_1}$. Particles belonging to the unstable lump begin to be trapped in the wave and form BGK-like modes. Because of the dispersion of these different components of the unstable wave spectra, as suggested by Eqn. 3.5, the waves overlap, triggering the onset of a period of wave turbulence called Langmuir turbulence. Provided that the Langmuir waves are not too strong, the effect of wave turbulence is to lead to the emergence of just a few undamped plasma waves [7].

The instability was investigated for domain lengths of $32\lambda_D$ and $128\lambda_D$. To set up the

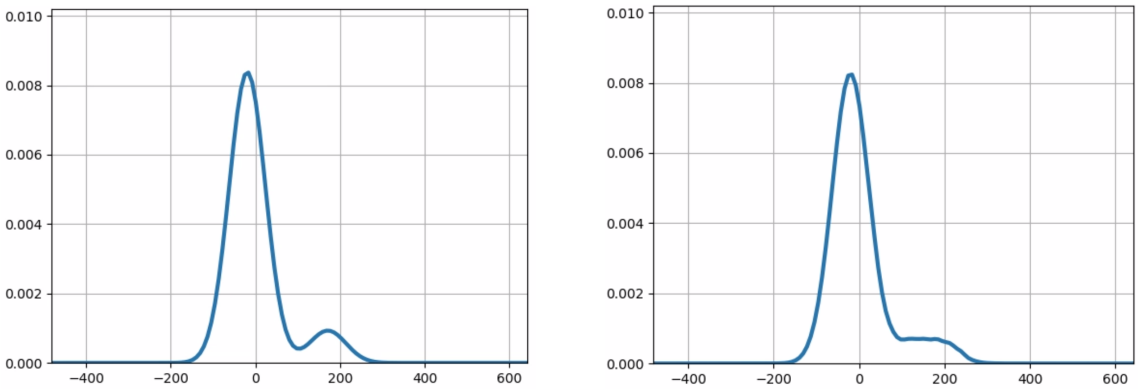


Figure 3.8: Initial Penrose-unstable distribution and distribution at saturation.

initial distribution in a bump-unstable configuration the following function was chosen,

$$n(x) = 1.0 + 0.01\cos(kx) \tag{3.8}$$

$$f(x,v) = n(x)\sqrt{\frac{1}{2\pi v_{th}^2}}\Big((1-\alpha)\exp\Big(-\frac{1}{2}\frac{(v-v_{main})^2}{v_{th}^2}\Big) + \alpha\exp\Big(-\frac{1}{2}\frac{(v-v_{bump})^2}{v_{th}^2}\Big)\Big) \tag{3.9}$$

where a density perturbation is necessary for the instability to manifest because all spatially-uniform distributions are equilibrium solutions to the Vlasov-Poisson equation. The main distribution and small drifting thermal population were given drift velocities so that the mean velocity at the beginning of the problem would be zero. To keep the initial mean velocity at zero, the main drift velocity is chosen as $v_{main} = \frac{-\alpha}{1-\alpha}v_{bump}$ where α is a parameter to set the relative importance of the bump. For the present simulations, $\alpha = 0.1$. The density of the plasma is kept normalized to one, including the bump particles. The initial average distribution function is shown in Fig. 3.8.

The results for the $32\lambda_D$ domain are shown in Figs. 3.9 and 3.10. After an initial period of Langmuir oscillations, the instability begins with a period of particle-trapping in the phase space. In the $32\lambda_D$ case, it was found that a single wave fit in the domain and continued to propagate. In contrast, Figs. 3.11 and 3.12 depict the results on a domain of length $128\lambda_D$ with the same plasma frequency. Given that the distributions and plasma frequency are the same as the $32\lambda_D$ case, the same spectrum of waves is excited but with a greater number of

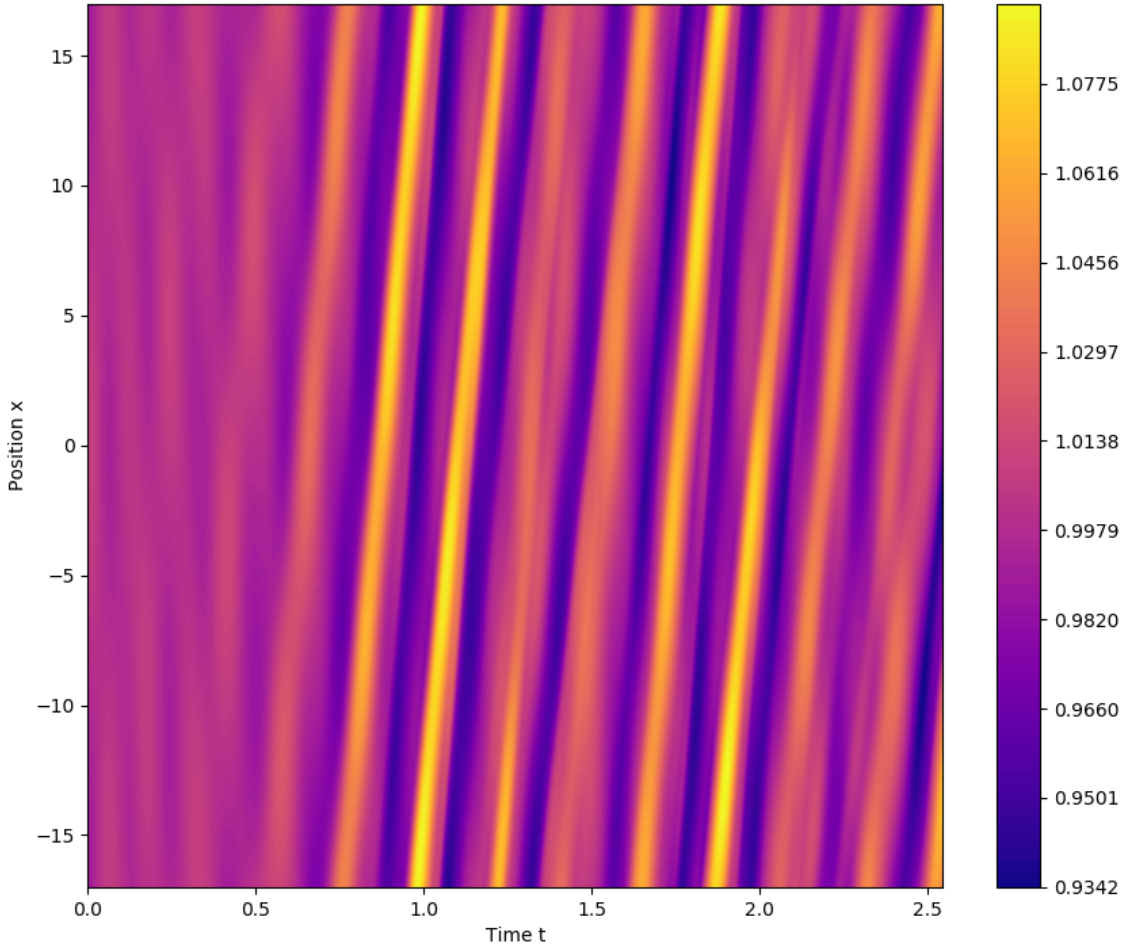
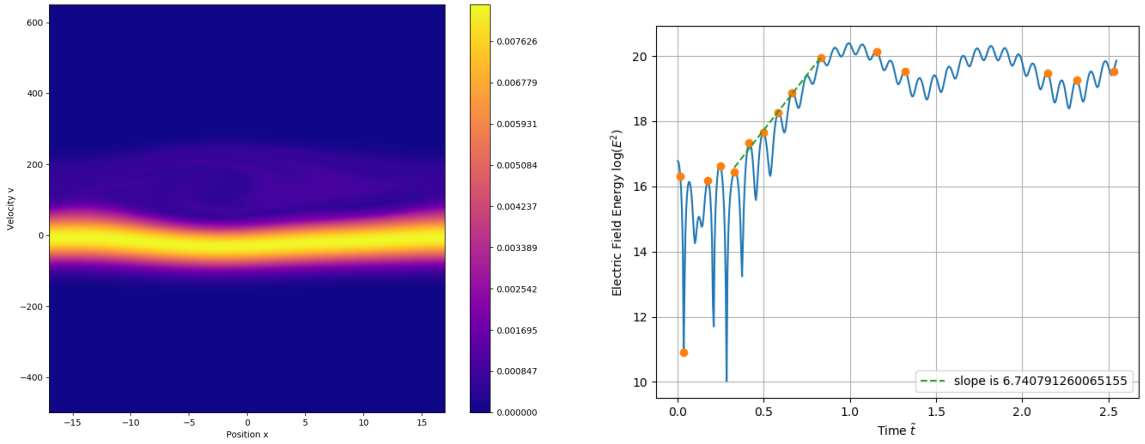


Figure 3.9: Evolution of the electron density for the bump-on-tail problem in a periodic domain of length $32\lambda_D$. The initial oscillations from the seeded perturbation are seen to evolve into an electron density cavity (bright region) within a few plasma periods from the start. The cavity transits quickly across the domain (bright vertical streaks). Variation of the cavity in time is seen to occur at the electron plasma frequency.



(a) Distribution function following saturation of the bump-on-tail instability. Particles are seen to be trapped in the forward-propagating wave.

(b) Evolution of field energy with time. Wave energy increases as free energy in the drifting population of electrons enters the Langmuir wave mode.

Figure 3.10: Structure of the distribution function and increase in field energy for the bump-on-tail instability on a domain of length $32\lambda_D$.

waves in the domain. Following saturation of the instability by particle-trapping, the waves collide and the wave-trapped particles begin to mix in the domain. This is an example of the evolution of one-dimensional Langmuir wave turbulence.

3.3.2 The electron two-stream instability, its saturation, and electron holes

Streaming instabilities are common occurances in plasma, occuring when one "crosses the beams". They are somewhat similar to the bump-on-tail instability, but represent rather a case when two plasma populations of similar density interpenetrate. Consequently there is greater available free energy in the two-stream instability than in the bump-on-tail instability and the excited wave has greater energy and more trapped particles.

The structure formed by the electron streaming instability is referred to as an electon

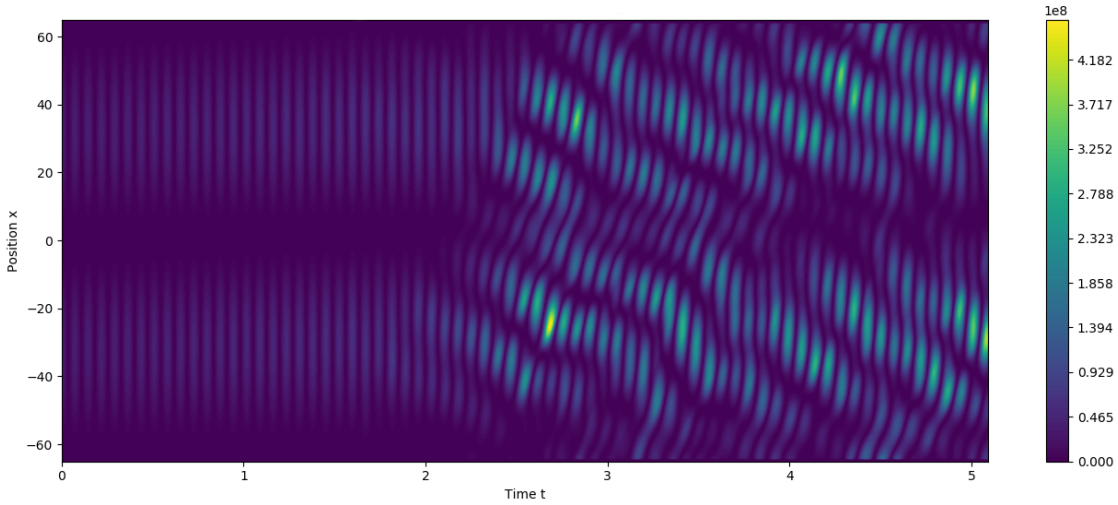
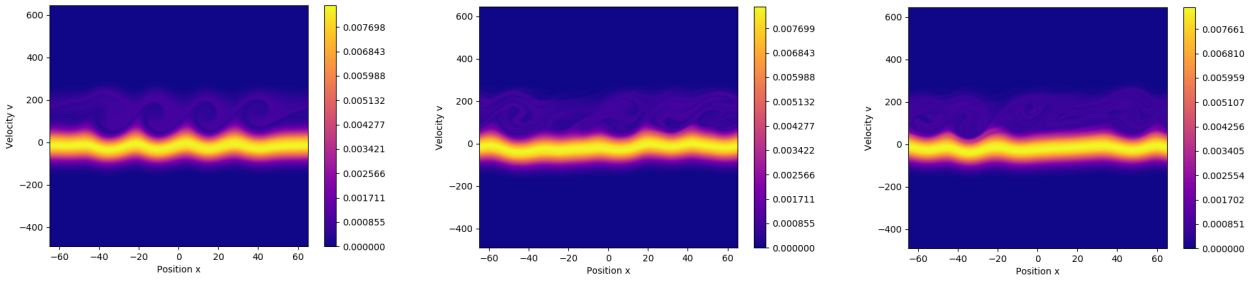


Figure 3.11: Propagation of the spectrum of Langmuir waves seen from the electric field intensity on an (x, t) diagram, for a $128\lambda_D$ periodic domain. The instability develops after many periods of the initial perturbation. Plasma waves emitted from bump-on-tail instability propagate with positive velocity, here seen as a tilt to the right. Modulation of the wave profile in time is indicative of wave dispersion and trapped particle mixing.



(a) Particles trapping within Langmuir waves of wavelength $\lambda \approx 20\lambda_D$ at time $t = \frac{1}{2}t_{final}$, seen as swirls.

(b) Wave dispersion leads to collision and mixing of trapped particles at time $t = \frac{3}{4}t_{final}$, as the swirls become chaotic.

(c) The waves are seen to coalesce after some time of evolution at time $t = t_{final}$, with larger coherent swirled regions.

Figure 3.12: Evolution of the distribution function's phase space structure in Langmuir turbulence following saturation of the bump-on-tail instability, for a domain length of $128\lambda_D$.

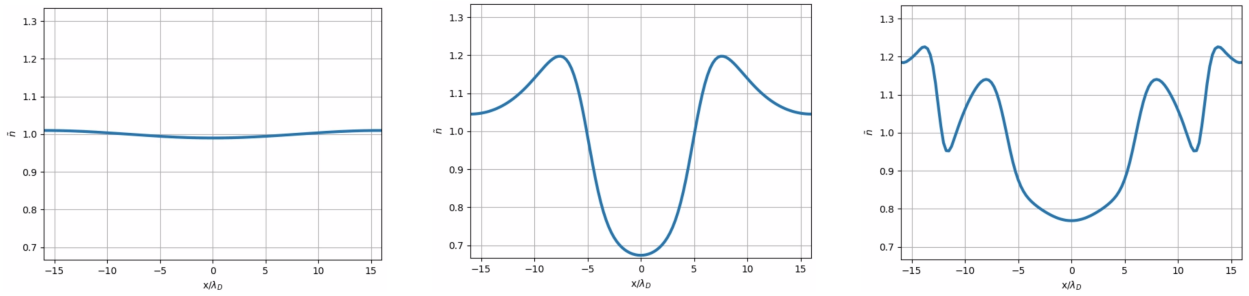
hole, characterized by a great number of trapped electrons accompanied by a stable hole or cavity in the electron density [11]. Electron holes have been observed since early plasma experiments, and are commonly observed in planetary magnetospheres. Trapping is considered essential for the stability of the wave, as the resonant particles are topologically bound to the plasma wave by the separatrix of the trapping region [22]. There are interesting soliton electron hole solutions, but this thesis considers periodic solutions.

Figure 3.15a shows the initial distribution consisting of a superposition of two Maxwellians normalized to a unit density. A sinusoidal perturbation is applied to the density, as in Fig. 3.13a. The instability quickly saturates as a standing Langmuir wave, an electron hole, seen at $t = \frac{3}{4}t_{final}$ in Figs. 3.13b and 3.15b. While such a hole is an equilibrium state, the electron plasma overshoots and continues oscillating in the nonlinear phase of the instability's evolution. As wave energy oscillates, some particles are detrapped and exit the hole to subsequently interact with the neighboring hole, where some particles are then retrapped. This process continues, producing finer and finer structures within the phase space as in Fig. 3.15c.

An important consequence of the two-stream instability is seen by considering the spatially average distribution function $\langle f \rangle$ as in Fig. 3.15. As the particles of the two interpentrating plasma streams mix together, the resulting distribution takes on a thermal spread much greater than either initial beam as the bulk kinetic energy of the streams is converted to plasma internal energy and Langmuir wave energy. In this way the two-stream instability is said to be a mechanism of collisionless heating in plasma.

3.3.3 Return of Landau damping: Response of protons to a standing electron hole

While the electron two-stream instability can evolve in a complex nonlinear fashion for a long time, it is interesting to observe two-species effects by considering the electron streams to have been crossed in the presence of a thermal population of protons stationary with respect to the streams. The result is that the large amplitude standing Langmuir wave decays into a pair of counter-propagating ion-acoustic waves as the initially uniform protons are

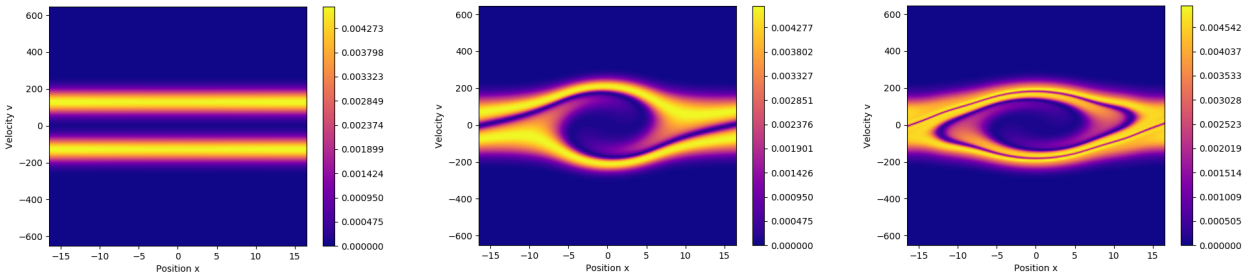


(a) Initial perturbation applied to the electron plasma.

(b) Characteristic hollow shape of an electron hole.

(c) Later time sees detrapping of some particles.

Figure 3.13: Formation of an electron hole, a standing Langmuir wave of large amplitude with many trapped particles, due to saturation of the electron two-stream instability.

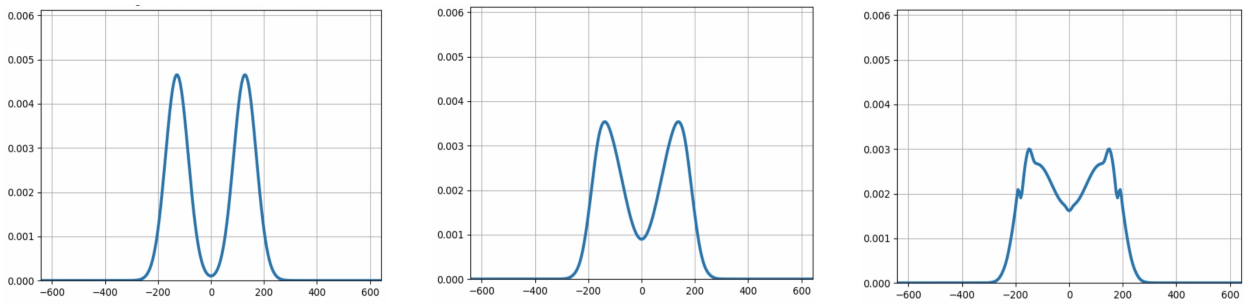


(a) Initial state of interpenetrating electron beams.

(b) A phase space vortex describes an electron hole.

(c) Finer phase space structures form per oscillation.

Figure 3.14: Phase space structure of the two-stream instability's evolution. An electron hole is formed at saturation of the instability. The system then oscillates around the hole equilibrium, so that some particles are detrapped and others retrapped per plasma period producing complex evolution of a large-amplitude standing Langmuir wave.



(a) Initial Penrose-unstable distribution.

(b) The instability evolves to fill in the unstable dip.

(c) The resulting $\langle f \rangle$ is hotter than either initial beam.

Figure 3.15: Inspection of the average distribution function for the two-stream instability demonstrates that the final spatially-averaged distribution is hotter than either distribution initially. The directed kinetic energy of the beams has been converted to a combination of plasma thermal energy and electrostatic Langmuir wave energy.

accelerated by the wave potential. That the electron hole is disrupted by ions and converted into ion-acoustic waves has been previously observed by other authors, who investigated the break-up of an electron hole soliton [20]. Figures 3.17 and 3.18 show the (x, t) diagrams of the protons and electrons, respectively. The proton space-time diagram is characteristic of counter-propagating simple waves. Figure 3.19 confirms this by showing that the proton distribution function takes on a shape much like that of the case of nonlinear Landau damping of an electron plasma wave, while Fig. 3.16 shows that low-frequency (proton) oscillations with superimposed high-frequency (electron) oscillations are born following saturation of the electron two-stream instability. As observed in [20], remnants of the broken-up electron hole persist in the electron phase space distribution, seen in Fig. 3.20.

This result can be considered as an extension of the Landau damping process to multiple species, as the bulk kinetic energy of the electron beams finds its way into more plasma modes. It also demonstrates a wave-wave process whereby a single wave can decay into two others. Note that the wavevector was conserved in this process, as $k_L = 0 = k_{IA,1} + k_{IA,2} \implies$

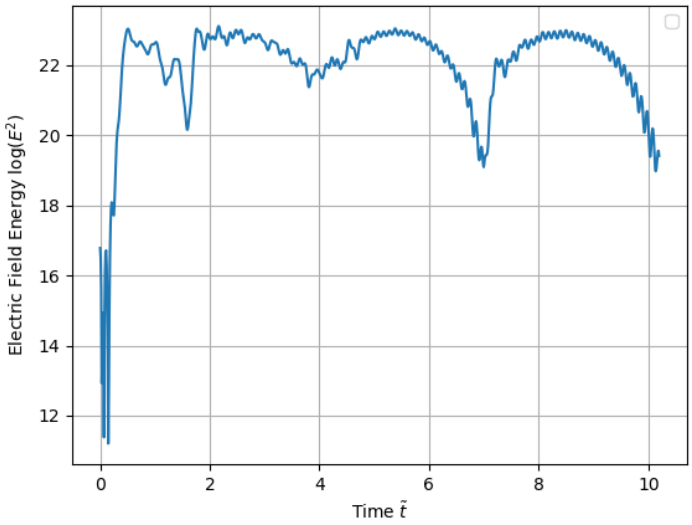


Figure 3.16: Long-time evolution of the electric field energy in the two-stream instability with excited proton plasma waves. Following instability saturation, ion-acoustic waves are excited by the standing Langmuir wave in their frame. Oscillations at the proton plasma frequency are seen in this plot.

$k_{IA,1} = -k_{IA,2}$, and indeed the ion-acoustic waves are counter-propagating [21].

3.4 Consequences of Collisionless Plasma Instability

The problems considered in this thesis have demonstrated a number of important consequences of the instability of collisionless plasma. A key result emphasized in the above problems has been the heating of plasma due to a redistribution between thermal and electric energies as a consequence of wave-particle interactions. This section reviews physics takeaways seen in the simulation results and mentions applications to the engineering of plasma devices.

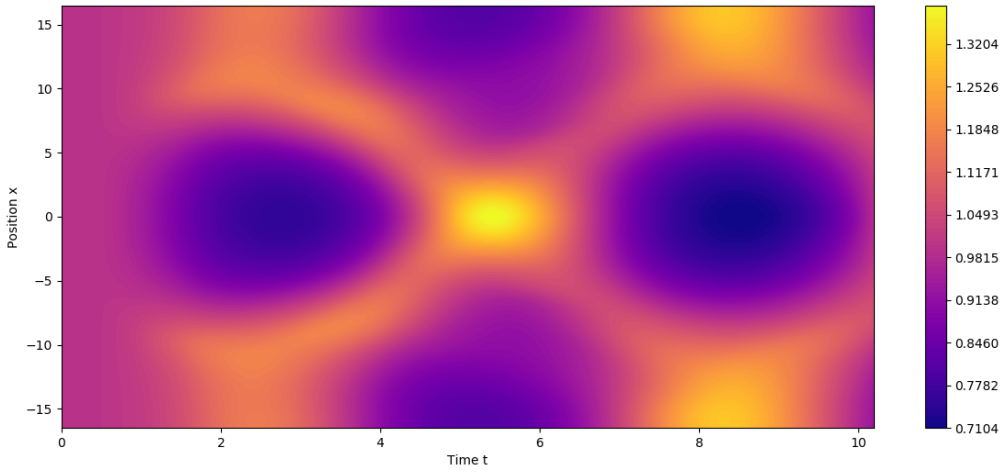


Figure 3.17: Excitation of ion-acoustic waves by a stationary Langmuir wave viewed on an (x, t) diagram. An initially uniform ion density profile evolves into oscillating peaks and troughs, indicative of counter-propagating waves of equal intensity.

3.4.1 Streaming instability as a mechanism of collisionless heating

The streaming instability may be thought of as a dynamic friction resulting from relative velocity between thermal populations in a collisionless plasma with mean field interaction. The friction promotes relaxation of non-equilibrium (i.e. non-Maxwellian) distributions, resulting in an increase in average thermal energy relative to the undisturbed thermal population along with emission of a turbulent wave spectrum which may decay into other kinds of plasma waves. In this way, plasma streaming processes can heat plasma collisionlessly, which is thought to be an important process in space plasma such as the solar wind.

3.4.2 Landau resonance as a heating and current driving mechanism

The resonant damping and excitation of plasma waves may be used in engineering applications to heat plasma and drive electric currents. For example, consider the Landau damping of a Langmuir wave propagating in one direction with phase velocity v_{ph}. As the average distribution function flattens, the distribution begins to look like the final state of Fig. 3.8.

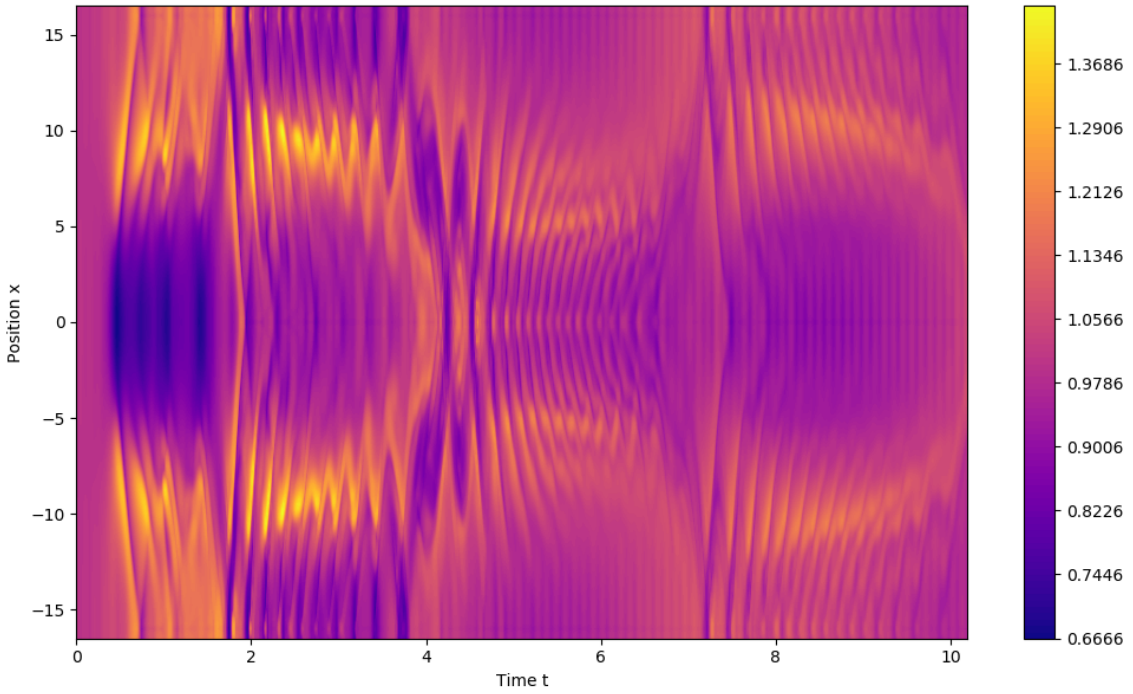
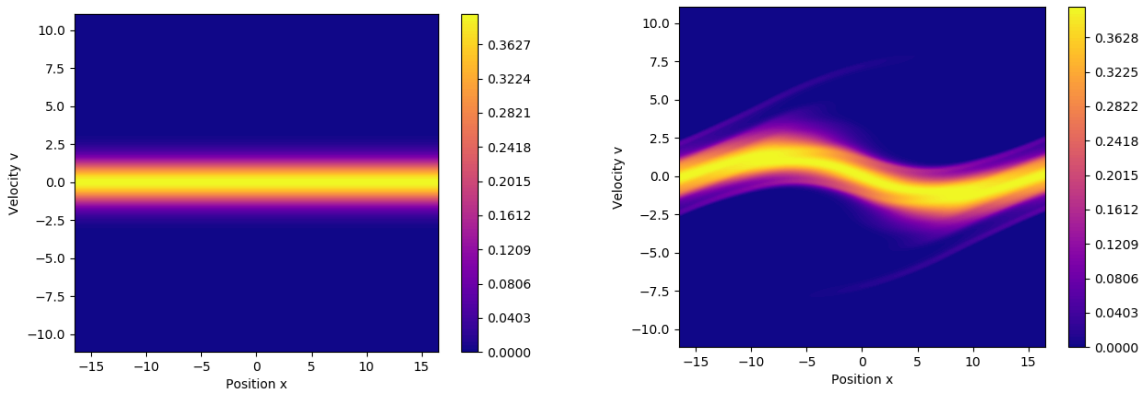


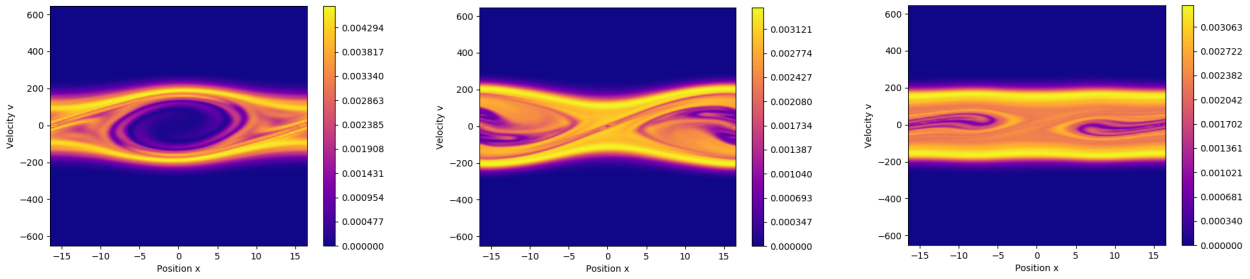
Figure 3.18: The electron hole resulting from saturation of the electron two-stream instability is seen to decay into a pair of counter-propagating ion-acoustic waves on this (x, t) diagram. The electron hole is abruptly disrupted at $t = 1$ by the ion response, demonstrating that the protons respond at the proton plasma timescale. The motion of initially uniform ions dominates the late time evolution as electrons follow bulk proton motion due to their small inertia, seen in comparison to Fig. 3.17. Oscillations around the ion motion at the electron plasma frequency are seen here as rapid striations.



(a) Initial proton distribution with $\langle v \rangle = 0$. The electron hole has $v = 0$ in their frame.

(b) Final proton distribution after several proton plasma periods.

Figure 3.19: Structure of the proton distribution function initially and finally. Note the similarity of the final phase space structure to the Langmuir wave nonlinear Landau damping problem, suggesting that protons are being trapped in the ion-acoustic wave potential.



(a) Electron hole at $t = \frac{1}{4} t_{final}$ before the proton response has altered its topology.

(b) The electron hole is squeezed by the proton response to its potential.

(c) After several proton periods, the electron distribution has nearly completely relaxed.

Figure 3.20: Evolution of the electron distribution following saturation of the two-stream instability when ion-acoustic waves are excited by the electron hole's standing potential.

The mean velocity of such a distribution is greater than zero, so that plasma is not only heated by collisionless wave absorption but also dragged along in the direction of wave propagation. This is the notion of *current drive*, whereby the absorption of waves by plasma electrons can drive electric currents in the direction of wave propagation. Although the electrostatic case is only an analogy to the EM waves excited in magnetized plasmas, the basic idea continues to hold, e.g. for RF current drive in tokamaks.

3.5 Mediation of Landau Damping through BGK Collisionality

An important question is how the collisionless damping of plasma waves is effected by a finite collisionality. To understand this problem, the Landau damping case with a wavelength of $\lambda = 10\lambda_D$ was repeated using a BGK collision frequency of $\nu_e\tau = 0$, $\nu_e\tau = \omega_e\tau$, and $\nu_e\tau = 100\omega_e\tau$. The results of this study are shown in Fig. 3.21 which depicts the field energy of these cases as well as the electron density for the collisionless case and for $\nu_e\tau = 100\omega_e\tau$. It is seen that the influence of a finite collisionality is that Landau damping is slowed down. In fact, for $\nu_e \to \infty$ the wave damping rate decreases entirely.

Figure 3.22 sheds light on the reason that Landau damping is arrested by a finite particle collisionality. The high collisionality causes the characteristic phase space structure of a collisionless system, phase mixed filiments, to be erased. This causes the wave dissipation to be irreversible rather than reversible, which is certainly a difference between the two systems. However, collisionality is seen to slow down the wave damping process by interfering with phase mixing. Particle collisions are somewhat paradoxically seen to inhibit collisionless wave dissipation.

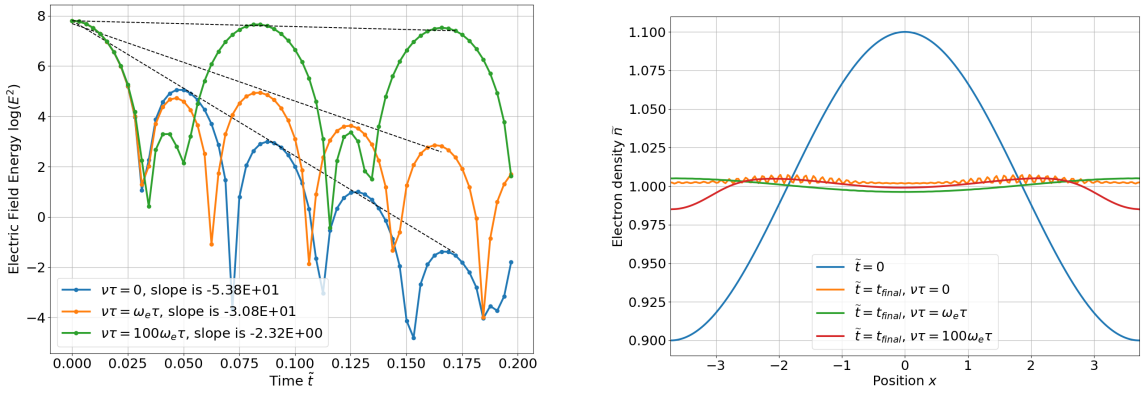


Figure 3.21: Field energy and electron density of damping Langmuir waves of wavelength $\lambda = 10\lambda_D$. The influence of collisions is seen to decrease the effective wave dissipation.

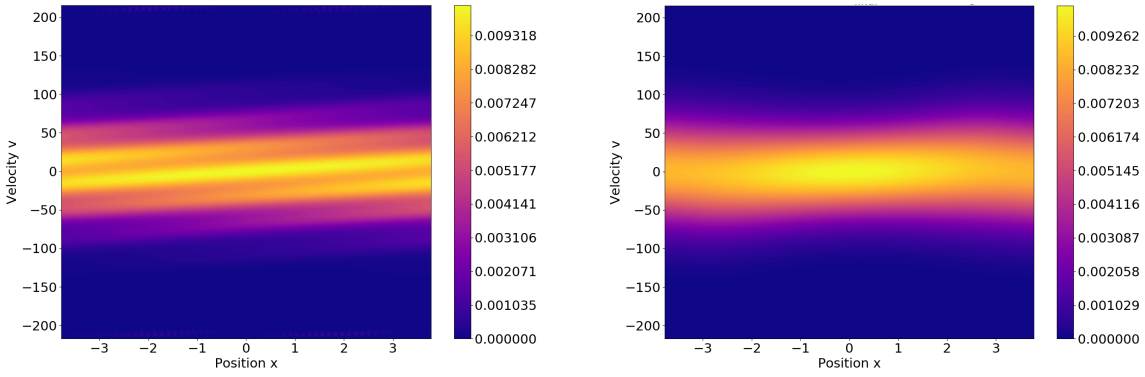


Figure 3.22: Comparison of the phase structures of damped Langmuir waves in the collisionless ($\nu_e = 0$) and collisional ($\nu_e\tau = 100\omega_e\tau$) cases.

Chapter 4
CONCLUSION

It is hoped that study of the material presented in this thesis can aid in the understanding of topics in collisionless plasma physics. With a continuum solver of the collisionless kinetic equation, problems exploring the foundations of plasma kinetic theory can be solved to build intuition for real world problems at scales unreachable by computer simulation for the time being. Ideas such as collisionless wave damping and excitation by wave-particle resonance, proton-electron plasma wave coupling, and the non-equilibrium processes of non-Maxwellian distributions can all be understood through use of a program such as that described in this thesis.

To review the concepts considered, the first chapter consisted of an introduction to the Vlasov-Poisson system, a coupled hyperbolic-elliptic system of PDEs, beginning with the equation of motion for a single particle. Following this introduction, the second chapter considered numerical methods for the solution of such a system. Coding techniques for fast and simple implementation of parallel code in Python were then presented. The third chapter consisted of applying the developed code to the simulation of several problems of interest in plasma physics.

It should be noted that the nonlinear evolution of many of these plasma waves, such as electron holes, remains a topic of ongoing research through numerical simulation. Plenty of future work exists in the simulation of plasma wave solitons and their application in various areas. Another fruitful area for future work is in the study of plasma-wall interfaces and plasma double layers. Higher dimensional problems are also in need of study. Any of these problems are good directions for further research.

BIBLIOGRAPHY

[1] V.I. Arnol'd. *Mathematical Methods of Classical Mechanics*. Springer, 1997.

[2] R. Balescu. *Statistical Dynamics: Matter Out of Equilibrium*. Imperial College Press, 1997.

[3] C.K. Birdsall and A.B. Langdon. *Plasma Physics via Computer Simulation*. McGraw-Hill, 1985.

[4] D. Bohm and E.P. Gross. Theory of Plasma Oscillations. A. Origin of Medium-Like Behavior. *Phys. Rev.*, 75(1851), 1949.

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

[6] B. Cockburn and C. Shu. Runge-Kutta Discontinuous Galerkin Methods for Convection-Dominated Problems. *Journal of Scientific Computing*, 16(3), 2001.

[7] P.Henri et al. Low-energy Langmuir cavitons: Asymptotic limit of weak turbulence. *EPL*, 96(99004), 2011.

[8] R.J. Goldston and P.H. Rutherford. *Introduction to Plasma Physics*. CRC Press, 1995.

[9] M. Hénon. Vlasov Equation? *Astron. Astrophys.*, 114:211–212, 1982.

[10] J.S. Hesthaven and T. Warburton. *Nodal Discontinuous Galerkin Methods*. Springer, 2010.

[11] I.H. Hutchinson. Electron holes in phase space: What they are and why they matter. *Physics of Plasmas*, 24(055601), 2017.

[12] M.D. Kruskal I.B. Bernstein, J.M. Greene. Exact Nonlinear Plasma Oscillations. *Phys. Rev.*, 108(546), 1957.

[13] S. Ichimaru. *Statistical Plasma Physics*. Frontiers in Physics. Addison-Wesley Publishing, 1992.

[14] L.D. Landau. On the vibrations of the electronic plasma. *J. Phys. USSR*, 10(26), 1946.

[15] R.J. Leveque. *Finite Volume Methods for Hyperbolic Problems*. Cambridge University Press, 2002.

[16] Sean Miller. *Modeling collisional processes in plasmas using discontinuous numerical methods*. PhD thesis, University of Washington, 2016.

[17] D.R. Nicholson. *Introduction to Plasma Theory*. Krieger Publishing, 1992.

[18] O. Penrose. Electrostatic Instabilities of a Uniform Non-Maxwellian Plasma. *Phys. Fluids*, 3(258), 1960.

[19] D.D. Ryutov. Landau damping: half a century with the great discovery. *Plasma Phys. Control. Fusion*, 41, 1991.

[20] K. Saeki and H. Genma. Electron-Hole Disruption due to Ion Motion and Formation of Coupled Electron Hole and Ion-Acoustic Soliton in a Plasma. *Phys. Rev. Lett.*, 80(6), 1998.

[21] R.Z. Sagdeev and A.A. Galeev. *Nonlinear Plasma Theory*. W.A. Benjamin, Inc., 1969.

[22] H. Schamel. Cnoidal electron hole propagation: Trapping, the forgotten nonlinearity in plasma and fluid dynamics. *Physics of Plasmas*, 19(020501), 2012.

[23] U. Shumlak. Extensible Normalization for Multi-Species Plasma Models. *Unpublished*, 2016.