A Kinetic Vlasov Model for Plasma Simulation Using Discontinuous Galerkin Method on Many-Core Architectures

Noah Reddell

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

Doctor of Philosophy

University of Washington

2016

Reading Committee:

Uri Shumlak, Chair

Richard Milroy

Sett You

Program Authorized to Offer Degree:
Aeronautics and Astronautics

University of Washington

Abstract

A Kinetic Vlasov Model for Plasma Simulation Using Discontinuous Galerkin Method on Many-Core Architectures

Noah Reddell

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

Advances are reported in the three pillars of computational science achieving a new capability for understanding dynamic plasma phenomena outside of local thermodynamic equilibrium.

A continuum kinetic model for plasma based on the Vlasov-Maxwell system for multiple particle species is developed. Consideration is added for boundary conditions in a truncated velocity domain and supporting wall interactions. A scheme to scale the velocity domain for multiple particle species with different temperatures and particle mass while sharing one computational mesh is described. A method for assessing the degree to which the kinetic solution differs from a Maxwell-Boltzmann distribution is introduced and tested on a thoroughly studied test case.

The discontinuous Galerkin numerical method is extended for efficient solution of hyperbolic conservation laws in five or more particle phase-space dimensions using tensor-product hypercube elements with arbitrary polynomial order. A scheme for velocity moment integration is integrated as required for coupling between the plasma species and electromagnetic waves.

A new high performance simulation code WARPM is developed to efficiently implement the model and numerical method on emerging many-core supercomputing architectures. WARPM uses the OpenCL programming model for computational kernels and task paral-

lelism to overlap computation with communication. WARPM single-node performance and parallel scaling efficiency are analyzed with bottlenecks identified guiding future directions for the implementation.

The plasma modeling capability is validated against physical problems with analytic solutions and well established benchmark problems.

TABLE OF CONTENTS

Page

List of Figures . . . iv

List of Tables . . . xi

Chapter 1: Introduction . . . 1
- 1.1 A Kinetic Description of Plasma . . . 1
- 1.2 Simulation Architecture and Numerical Method for Next Generation of High Performance Computing . . . 2
- 1.3 Study of Plasma in the Presence of a Magnetic Field: Planar Wave Propagation, Current Sheet Instabilities, and Magnetic Reconnection . . . 3

Chapter 2: Plasma Models . . . 5
- 2.1 Vlasov - Maxwell Kinetic Model . . . 7
- 2.2 Euler - Maxwell Fluid Model . . . 7

Chapter 3: Numerical Method . . . 9
- 3.1 Particle-In-Cell dominance . . . 9
- 3.2 Motivation for discontinuous Galerkin versus alternatives . . . 10
- 3.3 The discontinuous Galerkin method . . . 12
- 3.4 Nodal Discontinuous Galerkin Semi-Discrete Implementation . . . 15
- 3.5 Element Type and Basis functions . . . 17
- 3.6 Optimizations for tensor product hypercubes . . . 18
- 3.7 High-Accuracy Discontinuous Galerkin Working Matrix Initialization . . . 23
- 3.8 Types of Errors With Discontinuous Galerkin Method . . . 28
- 3.9 Advanced discontinuous Galerkin Schemes . . . 36

Chapter 4: WARPM Simulation Code for Many-Core Architectures . . . 41
- 4.1 Next-Generation Simulation Code . . . 42

4.2 Multi-Level Domain Decomposition 43
4.3 Dynamic OpenCL Code Assembly 43
4.4 Minimized Data Movement 48
4.5 Support for different numerical methods, physical models, and domain geometries 51

Chapter 5: Kinetic Implementation Details 55
5.1 Considerations for discretized velocity dimension 55
5.2 Shared rectilinear velocity space mesh among species with tailored stretching and offset velocity 57
5.3 Domain decomposition that does not distribute velocity space 59
5.4 Solid Wall Boundary Condition for the Vlasov Equation 60
5.5 Symmetry Plane Boundary Condition 65
5.6 Sequencing of physical-space and phase-space evaluation 67

Chapter 6: Planar Plasma Waves 69
6.1 Planar wave propagation in a uniform unmagnetized plasma 69
6.2 Spatially uniform plasma gyration in the Vlasov-Maxwell model 81
6.3 Planar Wave Propagation Perpendicular to Magnetic Field 85
6.4 Streaming Weibel instability 87

Chapter 7: Current Sheets 90
7.1 Harris Current Sheet Equilibrium 90
7.2 Lower-Hybrid Drift Instability 92

Chapter 8: Magnetic Reconnection 107
8.1 GEM Magnetic Reconnection Challenge Problem 107

Chapter 9: WARPM Computational Performance 127
9.1 Weak Scaling 129
9.2 Strong Scaling 131
9.3 Single Node Performance 135

Chapter 10: Conclusion 142
10.1 Contributions 142

10.2 Suggestions for Future Study 145

LIST OF FIGURES

Figure Number Page

3.1 Illustration of the 2D reference square element including node ordering for element order ($2^{\text{nd}} \times 3^{\text{rd}}$). The element modal basis set is a tensor product of two 1-D Legendre polynomial basis sets. When the reference element is projected into physical space as a quadrilateral the transformation is bilinear such that lines through nodes in reference space remain lines through nodes in physical space. 20

3.2 This series of plots presents the projections of the function $1 + \cos(\pi r)$ onto the Legendre basis function set of indicated order versus r. The Legendre basis functions are ortho-normal on the domain $r \in [-1, 1]$. 30

3.3 The L^2 norm of the projection error resulting from projecting the function $1 + \cos(\pi r)$ onto the Legendre basis function set. The domain of integration is $r \in [-1, 1]$. The horizontal axis is the polynomial order of the basis set. An even-odd trend is made clear by the stair-step results beyond order one. . . . 30

3.4 The L^2 norm of the projection error resulting from projecting the function $1+\cos(\pi(x-vt))$ onto the Legendre basis function set. The domain of integration is one square reference element, $x \in [-1, 1]$ and $v \in [-1, 1]$. Basis polynomial order in x is constant and sufficient, while multiple order in v are tested: $[4^{\text{th}}, 7^{\text{th}}, 10^{\text{th}}, 13^{\text{th}}]$. 31

3.5 Gauss-Lobatto quadrature error for integration of $1 + \cos(2\pi x)$ over the interval $x \in [0, L]$ (Absolute value of Eq. (3.25)). Multiple cases of quadrature order are tested: $[4^{\text{th}}, 7^{\text{th}}, 10^{\text{th}}, 13^{\text{th}}]$. Also apparent is a numeric noise floor is observed around 1×10^{-16} consistent with 64-bit floating point arithmetic. . 32

3.6 This figure repeats the analysis performed for Figure 3.5 except that the floating point precision is improved to about 20 decimal digits using an arbitrary precision math tool. The floating point noise floor is reduced while the results otherwise remain unchanged. 33

3.7 The error as L^2 norm per Eq. (3.26) for the constant linear advection system with initial condition $u_0(x) = 1 + \frac{1}{2}\cos\left(\frac{1}{2}x\right)$ on the domain $x \in [0, 4\pi]$. . . . 35

4.1 The WARPM code decomposes the domain among available nodes and further subdivides the domain on a node into patches suitable for GPU computation. Current hardware supports simultaneous calculation and to/from memory transfer of other patches. 44

4.2 Organization of a dynamically assembled OpenCL program created by WARPM. 49

4.3 Compute assignments (patches) for one MPI process/node participating in a block-structured simulation. The node's compute hardware consists of a CPU and GPU. Each compute device is responsible for advancing the simulation of a fixed subdomain of the node's overall assignment (green). The patches assigned to the GPU are internal and the buffer arrangement on the GPU facilitates large-block contiguous transfers over the PCI bus. 52

4.4 NACA 0012 airfoil simulation domain set up as a structured mesh of quadrilateral elements. This is an 'O' mesh, that wraps around the airfoil with periodic boundary seam at the trailing edge. The simulation domain is much larger than the airfoil (center) in order to minimize boundary effects and is decomposed into quadrilateral elements $(n_r, n_\theta) = (60, 128)$. 53

4.5 Steady-state flow conditions around a NACA 0012 airfoil with zero angle-of-attack. Gas density is plotted. 54

5.1 Relationship between truncated velocity space and the three moments of a modeled species with Maxwellian distribution. 58

5.2 Depiction of the solid wall boundary condition for the phase space pdf with one physical dimension. The normal component of incident momentum is reflected. 61

5.3 Solid walls oblique to the physical coordinate cannot be implemented with the same simple reflection procedure. Some region of incident normal momentum would be reflected outside the modeled velocity space (hashed green area). This results in loss of conservation of mass, momentum and energy. Additionally, some emitted momentum region also has no corresponding incident momentum (blue), but this can be simply handled by assuming the pdf is zero there. 62

5.4 Incident and emitted pdf integration regions for an oblique wall. 64

6.1 Weak Landau damping resulting time series of electric field energy compared to predicted decay rate. Simulation domain is decomposed into rectangular elements $(n_x, n_v) = (20, 80)$. Polynomial basis function order are $(P_x, P_v) = (7, 10)$. 71

6.2 Weak Landau damping resulting time series of electric field energy compared to predicted decay rate. Simulation domain is decomposed into rectangular elements $(n_x, n_v) = (20, 80)$. Polynomial basis function order are $(P_x, P_v) = (7, 7)$. Time $70\omega_p^{-1}$ corresponds with feature size in the velocity direction becoming smaller than that resolvable with the selected grid resolution and polynomial order. 72

6.3 Strong Landau damping affect on distribution function in phase space presented as velocity versus position for multiple times as annotated. Simulation domain is decomposed into rectangular elements $(n_x, n_v) = (20, 80)$. Polynomial basis function order are $(P_x, P_v) = (7, 7)$. 74

6.4 Strong Landau damping affect on electric field energy time series. Simulation domain is decomposed into rectangular elements $(n_x, n_v) = (20, 80)$. Polynomial basis function order are $(P_x, P_v) = (7, 7)$. Additionally, two zones of linear growth are identified and plotted along with fit lines $\gamma_1 = -0.5904$ and $\gamma_2 = 0.1688$ (dashed lines). 75

6.5 Two stream initial conditions with unstable conditions, $k = 0.5$. Simulation domain is decomposed into rectangular elements $(n_x, n_v) = (20, 80)$. Polynomial basis function order are $(P_x, P_v) = (7, 7)$. 76

6.6 Electric field energy time series subject to the same conditions as in Figure 6.5. Simulation domain is decomposed into rectangular elements $(n_x, n_v) = (20, 80)$. Polynomial basis function order are $(P_x, P_v) = (7, 7)$. Also plotted is the theoretical linear growth rate $\gamma = 0.4817$ 77

6.7 Two stream initial conditions with stable conditions, $k = 2$. Simulation domain is decomposed into rectangular elements $(n_x, n_v) = (20, 80)$. Polynomial basis function order are $(P_x, P_v) = (7, 7)$. 79

6.8 Two-stream instability fully developed vortex at time $t = 45$ presented as phase space representation of distribution function (velocity versus position). Initial condition and domain are setup as per Ref. [21]. Simulation domain is decomposed into rectangular elements $(n_x, n_v) = (20, 80)$. Polynomial basis function order are $(P_x, P_v) = (7, 7)$. 80

6.9 $\omega_p^2/\omega_c^2 = 3$ 85

6.10 $\omega_p^2/\omega_c^2 = 3$ 86

6.11 Streaming Weibel instability result for parameter 'Case 1' reported in Figure 5.3 of Ref. [13] copied here for comparison purposes. 87

6.12 Mean field energy time series for streaming Weibel instability result simulated by WARPM Vlasov-Maxwell model with the same conditions as the published result in Figure 6.11. 88

6.13 Mean kinetic energy time series for streaming Weibel instability result simulated by WARPM Vlasov-Maxwell model with the same conditions as the published result in Figure 6.11. 89

7.1 Harris Current Sheet problem setup with number density, transverse current density and magnetic field all functions of x between the two conductors. . 90

7.2 Zero-contour intersections indicate potential eigenvalues for the odd-mode solutions of Eq. (7.34) with parameters $M = 1836$, $R = 100$, $U = 1$, $\tau = 0.1$ consistent with those for the solutions presented in Figure 3 of Ref. [49]. The lower-right branch intersections agree with the eigenvalues in the figure cited. 99

7.3 Zero-contour intersections indicate potential eigenvalues for the odd-mode solutions of Eq. (7.34) with parameters $M = 25$, $R = 100$, $U = 1$, $\tau = 0.1$. Only the intersection at $\omega = 0.16 + 0.24\hat{i}$ seems to correspond with an eigenmode from Ref. [49] with $M = 1836$. 100

7.4 Full set of even and odd mode eigenvalues identified and labeled for parameter set $M = 25$, $R = 100$, $U = 1$, $\tau = 0.1$. 101

7.5 Odd mode eigenfunction solutions for the eigenvalues presented in Figure 7.4 with parameter set $M = 25$, $R = 100$, $U = 1$, $\tau = 0.1$. Note that only Mode A has perturbation concentrated in the current sheet region $Z < 1$ associated with the LHDI 102

7.6 Even mode eigenfunction solutions for the eigenvalues presented in Figure 7.4 with parameter set $M = 25$, $R = 100$, $U = 1$, $\tau = 0.1$. Note that only Mode A has perturbation concentrated in the current sheet region $Z < 1$ associated with the LHDI. 103

7.7 Eigenfunction solutions for δE_y in Eq. (7.34) associated with the corresponding eigenvaluse in Figure 7.4. 104

7.8 Typical WARPM Vlasov-Maxwell simulation result with initialized with equilibrium perturbation adhearing to the solution for even mode ‘A’. The image shows the E_y electric field component amplitude. 105

8.1 Representative structured velocity mesh in v_x and v_y dimensions utilized in the GEM challenge simulations. There are twelve finite elements in each velocity dimension. The interior nodes are also depicted for the tensor-product 3$^{\text{rd}}$-order polynomial elements so there are four points per element per dimension. Element faces are the darker lines. Element vertex spacing is stretched per Eq. (8.9) to improve resolution near the centroid. 112

8.2 Ion number density at four representative time points during the simulation. The initial sheet peak density is $n_0 = 1.0$ plus an additional uniform background fraction of $n_b = 0.2$. Over the course of the reconnection event, the bulk plasma retreats along the sheet axis away from the X-point resulting in a higher number density. 113

8.3 Net charge density at four representative time points during the simulation. The ion number density in Figure 8.1.4 at the same time points is useful for establishing relative scaling. Some structures with net charge density are consistent over the duration of the magnetic reconnection event with regions of positive charge density collocated with the magnetic field separatrix and a region of negative charge surrounding the O-point containing the bulk plasma density. Also visible are small scale structures inside of the separatrix most pronounced at the time of fastest reconnection. 114

8.4 Total ion momentum as r.m.s. magnitude at four representative time points during the simulation. As the current sheet separates and retreats from the X-point, significantly more momentum is located at the retreating edge with sharp fall-off. An emerging asymmetry across the current sheet is visible at the final frame of simulated time. 115

8.5 Composite presentation of current density out of plane (j_z) along with magnetic field streamlines for in-plane B_x and B_y. At the initial condition, the applied magnetic field perturbation is noticeable as necking in the field lines mid-plane while the plasma is uniform along the sheet. At time $t = 32\ \omega_{ci}^{-1}$, the separatrix and X-point are both clear features in the magnetic topology. In the bulk current density region there are actually two magnetic islands giving clearest indication of broken symmetry along the current sheet and across the X-point. The out-of-plane current density is much less uniform than the ion number density in Figure 8.1.4 might lead one to guess. At final time $t = 40\ \omega_{ci}^{-1}$, a slight asymmetry across the current sheet is also visible as a kink near the right domain edge. 116

8.6 Normalized reconnected magnetic flux versus time per Eq. (8.10). Three WARPM Vlasov-Maxwell simulations are presented as well as three other published kinetic simulation model results [56, 57, 58]. Two WARPM simulations have comparable resolution; the one labeled HW or half-width implements a symmetry boundary condition along the current sheet mid-plane. A quite course simulation was also made with about half the resolution in all dimensions and also using the symmetry boundary condition. The fast reconnection rate and onset time are still well recovered. 118

8.7 Deviation of a any distribution (red) from the Maxwell-Boltzmann distribution (blue) can be assessed by the 1-norm, or shaded area in this cartoon with one velocity dimension. 120

8.8 Normalized Maxwell-Boltzmann error for ions per Eq. (8.17) at the times indicated. Over the course of the reconnection event, the hotter ion species' distribution deviates most from Maxwellian in the diffusion region and in the low density region along and outside the magnetic separatrix. 122

8.9 Normalized Maxwell-Boltzmann error for electrons per Eq. (8.17) at the times indicated. Over the course of the reconnection event, the colder electron species' distribution deviates most from Maxwellian in the diffusion region and along the magnetic separatrix. 123

8.10 Physical points at which the probability distribution functions have been sliced and analyzed in three-dimensional velocity space. Point labels overlay the Normalized Maxwell-Boltzmann error per Eq. (8.17) for ion species (top) and electron species (bottom) at the time $t = 25.6\ \omega_{ci}^{-1}$. 124

8.11 Probability distribution function for ions (top) and electrons (bottom) at slice point 'A' in Figure 8.1.6 at time $t = 25.6\ \omega_{ci}^{-1}$. Isovolumes indicate 0.75 and 0.25 fractions of the specie's number density at the slice point. The crosshairs are centered at the fluid velocity moment. 125

8.12 Probability distribution function for ions (top) and electrons (bottom) at slice point 'E' in Figure 8.1.6 at time $t = 25.6\ \omega_{ci}^{-1}$. Isovolumes indicate 0.75 and 0.25 fractions of the specie's number density at the slice point. The crosshairs are centered at the fluid velocity moment. 126

9.1 Representative problem geometry for parallel scaling performance studies. In all cases the simulated domain is square and surrounded by perfectly conducting solid walls with uniform element size, Δx. The domain size L_x is constant for the strong scaling problems. For weak scaling problems, the domain area increases linearly with number of processors, $L_x \propto \sqrt{N_{\text{nodes}}}$. 129

9.2 Weak scaling experiment results with tile of 3×3 elements per compute node with 8 elements in each velocity dimension. 132

9.3 Weak scaling experiment results with tile of 16×16 per compute node with 8 elements in each velocity dimension. 133

9.4 Strong scaling performance for problems based on 48 × 48 rectilinear mesh elements in physical space and 8 elements in each velocity dimension. Element polynomial order and number of velocity dimensions in the Maxwell-Vlasov model are indicated by the legend. Speedup is defined as the reduction in wall-time to compute one unit of time advance relative to single-node time. The purple annotations indicate the domain decomposition tile size on each node. 134

LIST OF TABLES

Table Number Page

3.1 Linear index to multi-index conversion function $M(n,d)$ for the 2^{nd}-order × 3^{rd}-order square element depicted in Figure 3.1. 19

3.2 Summary of floating point operations for DG element evaluation of time derivative on a hypercube element. The number of nodes per N_D-dimensional element with polynomial order P is $N_p = (P+1)^{N_D}$ and the number of face points is $N_{\mathrm{fp}} = 2N_D(P+1)^{N_D-1}$. 21

3.3 Example reduction in FLOP and storage requirements for DG tensor product hypercube element with element order, $P = 3$, and four-dimensional hypercube, $N_D = 4$. 21

3.4 Example reduction in FLOP and storage requirements for DG tensor product hypercube element with element order, $P = 3$, and five-dimensional hypercube, $N_D = 5$. 22

3.5 Example reduction in FLOP and storage requirements for DG tensor product hypercube element with element order, $P = 3$, and six-dimensional hypercube, $N_D = 6$. 22

3.6 Number of sub-steps required versus accuracy in explicit Runge-Kutta methods 36

5.1 Conditions associated with sausage mode or even symmetry along the $x = 0$ plane in the Vlasov-Maxwell model. 66

6.1 Mean velocity phase diagrams and electric field spectrum for spatially uniform plasma oscillation perpendicular to a magnetic field. The left column shows velocity normalized by the thermal velocity, with $\bar{v}_x$ on the horizontal and $\bar{v}_y$ vertically. The right column frequency spectrum is normalized to cyclotron frequency, ω_c. Each row represents a different initial condition that select both or single mode dynamics based on the initial electric field strength. For these examples, $\omega_p = 1$, $\omega_c = \frac{1}{\sqrt{20}}$. 84

9.1 NERSC *Edison* Configuration 128

9.2 Performance metrics collected from one two-socket 24-core Ivy Bridge node of *Edison* at NERSC. Domain geometry is 16 × 16 element tile in all cases, and 8 elements per velocity dimension. Hardware performance metrics were collected by the Intel VTune Amplifier 2016 performance sampling tool. . . 137

ACKNOWLEDGMENTS

The author has deep appreciation for the guidance and feedback from his dissertation committee and especially for his committee chair, Professor Uri Shumlak. The support that he has provided during the course of the author's graduate career, in all forms, was most influential in ultimate success. The support and organization of the William E. Boeing Department of Aeronautics & Astronautics was quite helpful making it through the program milestones.

The author would like to acknowledge the immeasurable contributions to physical understanding and mental health provided by fellow research lab students including Robert Lilly, Bhuvana Srinivasan, Eder Sousa, Sean Miller, and Genia Vogman.

It is unlikely much of this research would have been achievable without the terrific support from the Computational Science Graduate Fellowship program. The CSGF provided financial support, specialized high performance computing training, networking opportunities, and access to computing resources. The CSGF also organized a summer practicum at Princeton Plasma Physics Laboratory in 2011. The author is thankful to Dr. Guo Yong Fu for hosting the practicum and providing some of the author's early exposure to kinetic model simulations.

The author is grateful for opportunity to attend the *Computational Methods in High Energy Density Plasmas* long program hosted by the Institute of Pure and Applied Mathematics at UCLA in 2012. The technical program and deep interaction with other attendees were a terrific source of ideas at a formative time in this research endeavor.

This research used resources of the National Energy Research Scientific Computing

Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. The NERSC technical support staff were extremely helpful over the years.

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

The author is grateful for the unwavering support and encouragement from family, friends, and fellow graduate students that enabled graduate school experience to be enjoyable and reach the end-goal.

DEDICATION

This work is dedicated to my community of teachers who have nurtured, challenged, and inspired me to explore the unknown. The interactions have spanned decades, some will have eternal entanglement and others may never know where their support has lead me.

Thank you all the same.

Chapter 1
INTRODUCTION

The study of plasma, or the ionized state of matter, has been rich with discovery leading to both new understanding and new questions in contemporary theory. The real emergence of plasma science research was coincident with the beginnings of scientific computing and the two fields have been closely tied for more than half-a-century since.

Plasma phenomena of interest, especially in the application area of controlled fusion energy, have consistently proven to be more complex than each subsequent generation of models could capture. At the same time, computing capabilities and numerical methods have rapidly advanced to unlock the possibility of newer more complete models. This research intended to continue the trend from a point where both a change in modeling paradigm and computational approach were needed.

1.1 A Kinetic Description of Plasma

Most plasma modeling has considered the plasma to be a compressible fluid influenced additionally by electromagnetic forces [1, 2]. The behavior of the plasma is approximated by the bulk behavior of many individual particles' positions and velocities and the collective interaction with electric and magnetic fields. A fluid model imposes an additional restriction on the form of the distribution in velocity space. In local thermodynamic equilibrium, the velocity distribution of particles at a particular point in space is known to be a Maxwell-Boltzmann distribution which can be analytically expressed and is completely characterized by just three parameters: number density, average velocity, and temperature. The modeled fluid tracks the evolution of these parameters of the velocity distribution (commonly called the fluid variables) as a function of position and time.

The primary plasma model studied in this thesis is a kinetic model as opposed to a fluid representation. A statistical approximation is applied to the distribution of particles in both position and velocity space. The combination of dimensions is often called phase space and is most generally twice the number of spatial dimensions. A complete description, then, would be six-dimensional describing the probability density function (pdf) for particles over three space dimensions and three velocity dimensions. In the kinetic model, the pdf function is arbitrary and can take on any form. The distribution itself is evolved in time, greatly expanding the number of degrees of freedom and computational work, but capturing rich and important phenomena when the plasma is not in local thermodynamic equilibrium.

To further complicate the model, plasma is characterized by at least two particle species with differing charge and mass characteristics. For physical realism, the model must include at least separate electron and ion species. Sometimes it will be adequate to treat just one species with a kinetic model and the other with a fluid model. Either way, there is an open area for investigation into modeling the interaction between separate particle species.

The physical model is more fully described in Chapter 2.

1.2 Simulation Architecture and Numerical Method for Next Generation of High Performance Computing

The substantial increase in the physical model complexity paired with new trends in high performance computing architectures require careful numerical method selection and development of a new simulation code architecture. The code is WARPM: the *Washington Approximate Riemann Problem Solver – Many-core* edition.

Improvements in computing performance through increases in core clock frequency has been stalled for several years. Performance continued to improve, especially in scientific computing, through increased parallelism – first with many-node distributed memory clusters using Message Passing Interface, then with multiple shared memory cores on each node. Industry pressures and physical limitations on power density have started a new many-core architecture era with currently two main classes of devices. First, there are the modern graphics processors largely fueled by the gaming industry. A typical GPU might have over one-thousand compute cores[3], but the cores are lightweight. A lightweight core is not as functionally equipped or able to operate as independently as a traditional CPU core. One common reduction is that several cores are slaved lock-step to a single instruction scheduler; each core must execute the same instruction, but with independent data. Another reduction is that memory consistency for certain memory regions is not maintained between cores. Second, there are the new and upcoming CPU devices with sixty or more cores [4], supporting a greater number of threads per core and larger vector-width floating point operations. The clock speeds for these devices are considerably lower, requiring good use of the vector units and threads to realize improved performance.

Three important characteristics are common between both types of many-core devices. Memory movement is very expensive in terms of time and energy use compared to numeric operations, and multiple memory hierarchies exist including: memory accessible by a single core, memory shared among a few cores, memory accessible by all cores. Lastly, the highest floating point arithmetic performance is only achievable with vectorized operations. Software architecture and numerical methods that are well adapted to these three characteristics are best suited to perform well on today's and future high performance computing resources.

A numerical method well suited for the physical model and new high performance computing architectures is developed in Chapter 3 while a new simulation code designed to implement the numerical method with good performance is presented in Chapter 4.

1.3 Study of Plasma in the Presence of a Magnetic Field: Planar Wave Propagation, Current Sheet Instabilities, and Magnetic Reconnection

The advanced plasma model paired with high performance numerical method and simulation code opens up many new areas for investigation in plasma physics. Three general problem types are investigated in this thesis as both a proof of principle and to contribution to their physical understanding. The set is not accidental in that each type exercises a different model dimensionality in position or velocity space and thus also different computational problem scale.

First, planar plasma wave propagation and related limited cases are investigated in Chapter 6. These problems are adequately described with one physical dimension. For planar longitudinal plasma waves propagating parallel to a unidirectional magnetic field, only one velocity dimension is required to model the pdf dynamics; the modeled phase space is called $(1D + 1V)$. For planar plasma waves propagating perpendicularly to a unidirectional magnetic field, two velocity dimension are required to model the pdf dynamics including Larmor motion; the modeled phase space is called $(1D+2V)$. And for planar plasma waves propagating obliquely to a unidirectional magnetic field, three velocity dimensions are required and the modeled phase space is $(1D + 3V)$. Each of these sub-types has a spatially uniform limit where the wave number $\mathbf{k}$ is zero and only plasma gyration occurs with analytical solution.

Magnetic current sheets confine a plasma pressure gradient balanced by the Lorentz force and consistent current density along the sheet [5]. Current sheet instabilities that can develop due to perturbations along the current sheet direction are investigated in Chapter 7. With a magnetic field always perpendicular to the current-sheet plane, the problem can be modeled in $(2D + 2V)$.

Finally, the current sheet problem is rotated so that the magnetic field is in the simulation plane while the current flow is out of plane. This configuration allows the interesting kinetic study of magnetic reconnection[6] which is described in Chapter 8. The problem modeling is in $(2D + 3V)$.

The reconnection study proved to push the limit of currently available supercomputing resources. At the same time, experience gained studying WARPM performance at this scale resulted in ideas for new code optimizations and optimistic outlook that $(3D + 3V)$ simulations will soon be achievable. Performance analysis of relevant problem sizes is presented in Chapter 9.

Chapter 2
PLASMA MODELS

In considering development of a model for plasmas, there are several options for level of complexity and scale possible. The model should be intelligently selected to capture the significant physical processes in the problem at hand. Nevertheless, some models are more general than others and often the scope of physics is not fully known in advance. This work emphasizes kinetic model flexibility rooted in principled derivation and avoidance of overly restrictive approximations. At the same time, the ultimate goal is to analyze engineering systems of interest such as fusion plasma confinement, space propulsion systems, and industrial processes. At this scale, N-body[7] or molecular dynamic models[8] are still far out of reach of computational capability.

Instead of simulating the dynamics of every plasma particle directly, a kinetic simulation aims to capture the macroscopic behavior of the ensemble of particles. This is possible when the concentration of particles is very high compared to the physical scale of interest giving a statistically smooth distribution of particle states. A kinetic model then describes the evolution of the statistical average of many particles' states in both position and velocity. This statistical average is called the probability distribution function (pdf) with units of particles per space volume per velocity volume, $m^{-6}s^3$ in real-world $(3D + 3V)$ dimensions.

For kinetic models, the Vlasov equation system presents the simplest evolution of a single particle species affected by external forces and no collisions. The probability density function, $f_{\mathrm{s}}(\mathbf{x}, \mathbf{v})$, evolves in time according to,

$$\frac{\partial f_{\mathrm{s}}}{\partial t} + \mathbf{v} \cdot \frac{\partial f_{\mathrm{s}}}{\partial \mathbf{x}} + \frac{\mathbf{F}}{m_{\mathrm{s}}} \cdot \frac{\partial f_{\mathrm{s}}}{\partial \mathbf{v}} = 0, \tag{2.1}$$

where s identifies the species, $\mathbf{F}$ is the force vector, and m_{s} is the species' particle mass.

A more general implementation of the Boltzmann equation may be appropriate where collisions, radiation, and other physical effects necessitate the addition of a source term to the hyperbolic Vlasov equation [9]. When considered, the functional source term $S\left(\mathbf{x}, \mathbf{v}, \ldots\right)$ replaces the right-hand side of Eq. (2.1). The exact form of S could be quite complicated, perhaps causing the kinetic evolution to be stiff, operating non-locally in velocity, and may couple multiple plasma species.

In collisionless plasma, the most distinguishing forces are the macroscopic electromagnetic forces. The fields generally have spatial variation but are constant for any velocity coordinate, $\mathbf{v}$.

In some applications, the magnetic field does not change or changes so slowly compared to the time scale of interest; Poisson's simplified model for electrostatics can be used. The electric field $\mathbf{E}$, local charge density σ and potential ψ are simply related by

$$\nabla \cdot \mathbf{E} = \frac{\sigma}{\epsilon_0}, \tag{2.2}$$

$$\mathbf{E} = -\nabla \psi, \tag{2.3}$$

$$\nabla^2 \psi = -\frac{\sigma}{\epsilon_0}. \tag{2.4}$$

In applications with dynamic magnetic field, $\mathbf{B}$, the full electromagnetic dynamics of Maxwell's equations are required

$$\nabla \cdot \mathbf{E} = \frac{\sigma}{\epsilon_0}, \tag{2.5}$$

$$\nabla \cdot \mathbf{B} = 0, \tag{2.6}$$

$$\frac{\partial \mathbf{E}}{\partial t} = \frac{1}{\mu_0 \epsilon_0} \nabla \times \mathbf{B} - \frac{1}{\epsilon_0} \mathbf{J}, \tag{2.7}$$

$$\frac{\partial \mathbf{B}}{\partial t} = -\nabla \times \mathbf{E}. \tag{2.8}$$

Fluid velocity $\mathbf{u}_\mathrm{s}$ and the plasma number density n_s are reduced moments of the probability density function integrated over all velocity.

The local charge density σ is the summed contribution of all species with charge q_s and number density n_s,

$$\sigma = \sum_{s=1}^{N_\mathrm{s}} q_\mathrm{s} n_\mathrm{s}. \tag{2.9}$$

The current density $\mathbf{J}$ likewise results from the net effect of all charged species flowing at average velocity $\mathbf{u}_\mathrm{s}$,

$$\mathbf{J} = \sum_{s=1}^{N_\mathrm{s}} q_\mathrm{s} n_\mathrm{s} \mathbf{u}_\mathrm{s}. \tag{2.10}$$

Plasmas are typically made up of multiple particle species. Often models consider two: ions and electrons. Most real plasmas involve multiple elements at many different charge states. In some plasmas, a neutral gas population is significant. Such complexities are not considered in this work.

2.1 Vlasov - Maxwell Kinetic Model

A complete kinetic model for collisionless plasma is attained by combining the Vlasov model Eq. (2.1) with the Maxwell's Equations (2.5)-(2.8), and the local charge relations Eqs. (2.9) and (2.10). The evolution of the distribution function is then more completely specified as

$$\frac{\partial f_\mathrm{s}}{\partial t} + \mathbf{v} \cdot \nabla f_\mathrm{s} + \frac{q_\mathrm{s}}{m_\mathrm{s}} (\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \frac{\partial f_\mathrm{s}}{\partial \mathbf{v}} = 0. \tag{2.11}$$

2.2 Euler - Maxwell Fluid Model

In some systems, a plasma species is adequately approximated in local thermodynamic equilibrium where the probability density is a Maxwellian completely characterized by local number density n, average velocity $\mathbf{u}$, and temperature T. The probability density function is the Maxwell-Boltzmann distribution, including the famous Boltzmann constant k,

$$f_\mathrm{M}(n, \mathbf{u}, \mathbf{v}, T) = n \left(\frac{m}{2\pi kT} \right)^{\frac{3}{2}} \exp\left(\frac{-m|\mathbf{u} - \mathbf{v}|^2}{2kT} \right). \tag{2.12}$$

With fluctuations in physical space propagating only as perturbation of these macroscopic parameters, a fluid model is appropriate [2, 10, 11, 12]. The Euler-fluid plasma model equations adequately describe compressible inviscid fluid,

$$\frac{\partial \rho_{\mathrm{s}}}{\partial t} + \nabla \cdot (\rho_{\mathrm{s}} \mathbf{u}_{\mathrm{s}}) = 0, \tag{2.13}$$

$$\frac{\partial \rho_{\mathrm{s}} \mathbf{u}_{\mathrm{s}}}{\partial t} + \nabla \cdot (\rho_{\mathrm{s}} \mathbf{u}_{\mathrm{s}} \mathbf{u}_{\mathrm{s}} + p_{\mathrm{s}} \mathbf{I}) = \frac{\rho_{\mathrm{s}} q_{\mathrm{s}}}{m_{\mathrm{s}}} (\mathbf{E} + \mathbf{u}_{\mathrm{s}} \times \mathbf{B}), \tag{2.14}$$

$$\frac{\partial \epsilon_{\mathrm{s}}}{\partial t} + \nabla \cdot ((\epsilon_{\mathrm{s}} + p_{\mathrm{s}}) \mathbf{u}_{\mathrm{s}}) = \frac{\rho_{\mathrm{s}} q_{\mathrm{s}}}{m_{\mathrm{s}}} \mathbf{u}_{\mathrm{s}} \cdot \mathbf{E}. \tag{2.15}$$

An assumed closure relation such as the ideal gas law is required to complete the system with a relationship between kinetic energy density and momentum density.

Due to the extreme disparity between ion and electron mass, in many cases a fluid model is appropriate for one species while a kinetic model is necessary for another. A modeling system that can support a coupled combination of both fluid and kinetic models for different species could be useful.

Chapter 3
NUMERICAL METHOD

Having established the plasma model characterized by hyperbolic partial differential equations in high dimensional phase space, an appropriate numerical method must be selected to implement the model computationally.

The discontinuous Galerkin method is selected for its robustness to work well for kinetic equations in phase space, Maxwell's electromagnetic equations, and fluid equations in physical space with multiple contemporary examples for these application areas [12, 13, 14, 15, 16, 17, 18]. Additionally, the explicit method is straightforward to implement and well suited for emerging high performance computing architectures as described in Chapters 4 and 9.

3.1 Particle-In-Cell dominance

Before discussing discontinuous Galerkin, it must be acknowledged that there is a much more prevalent method in use for kinetic simulations. Particle-In-Cell (PIC) methods [19, 20] have been in use for kinetic simulations of plasma for a longer time. PIC is not just a pure numerical scheme – it is intertwined with the kinetic model.

For a system of N point particles, the probability distribution function, f, within phase space is given by the Klimontovich-Dupree equation,

$$f(\mathbf{x}, \mathbf{v}, t) = \sum_{j=1}^{N} \delta(\mathbf{x} - \mathbf{x}_j)\delta(\mathbf{v} - \mathbf{v}_j), \tag{3.1}$$

where each particle j has position $\mathbf{x}_j$ and velocity $\mathbf{v}_j$ and δ is the Dirac delta function. Each particle advances according to Lagrangian equations of motion,

$$\frac{d\mathbf{x}_j}{dt} = \mathbf{v}_j, \frac{d\mathbf{v}_j}{dt} = \frac{\mathbf{F}(x_j)}{m_j}, \tag{3.2}$$

where m_j is the particle mass and $\mathbf{F}(x_j)$ is the force experienced by particle j.

Solving the above equations with a very large number of particles should converge to the equivalent problem of solving the Vlasov Eq. for a smooth $f(\mathbf{x}, \mathbf{v}, t)$. PIC does not implement the Klimontovich-Dupree equation, however. The implementation of the force term, $\mathbf{F}(x_j)$, distinguishes PIC form N-body simulations. It is developed through the statistical contribution to charge density or current density by many nearby particles and then deposited on a physical grid of cells [20]. The number of particles per grid cell and cell size directly relates to statistical noise limitations.

Three limitations of PIC motivate search for an improved numerical method. The statistical noise problem has already been mentioned and is most prevalent in regions of the simulated system where the number density of particles is small. Additionally, PIC codes are able to conserve mass or conserve energy, not both. Finally, PIC codes have a challenge for good scaling on modern computing architectures due to the unpredictable communication involved when particles stream across computational domain boundaries of the physical grid.

3.2 Motivation for discontinuous Galerkin versus alternatives

In solving a continuous approximation of the conservative Vlasov equation system or a fluid model, possible methods include finite volume, semi-Lagrangian, discontinuous Galerkin, and continuous finite element method. All can be used to solve nonlinear systems of hyperbolic conservation laws of the type,

$$\frac{\partial q}{\partial t} + \nabla \cdot \mathbf{f}(q) = g, \ \ \mathbf{x} \in \Omega, \tag{3.3}$$

$$q(\mathbf{x}, 0) = q_0(\mathbf{x}). \tag{3.4}$$

Non-linear systems of conservation laws starting with continuous initial data can evolve into discontinuous solutions. Finite element methods that enforce solutions with C_0 continuity at cell boundaries can be eliminated from consideration due to their unsuitability to capture shock formation. They do offer many favorable qualities when it comes to developing high-order-of-accuracy methods and boundary conditions.

Semi-Lagrangian schemes for kinetic methods initially represent both particle distribution and field values on an Eulerian grid. The particle distribution is then advanced in a Lagrangian frame without restriction for time-step before final projection back onto the Eulerian grid. See Refs. [21] and [17] for a description of a semi-Lagrangian scheme applied to the Vlasov-Poisson system. The interpolation between Lagrangian and Eulerian frame introduces the most significant solution error in this scheme. Extending the scheme to higher dimensions than $(1D{+}1V)$ is complicated by the method's reliance on operator splitting techniques. Additionally, little progress is available for high temporal order-of-accuracy. When used with the Vlasov-Maxwell system of interest, the Lagrangian time-step advantages may not be significant compared to the speed of light CFL limit in the Eulerian frame.

The comparison of the remaining finite volume and discontinuous Galerkin methods warrants more careful evaluation. Overall, the formulation of a finite volume method is more straightforward but discontinuous Galerkin shows advantages when high spatial order-of-accuracy is desired. Finite volume methods store a single cell average value for each system variable between time steps. DG methods store multiple reconstruction coefficients per cell or element for each system variable. To achieve the same spatial accuracy, a finite volume approach would be required to use more (smaller) grid cells for the same problem domain.

To achieve higher spatial order of accuracy in finite volume methods, larger stencils are used in the reconstruction. Larger stencils require moving more data from more cells across every level of the communication hierarchy. DG utilizes a single-width stencil (meaning central element and adjacent neighbor elements only) requiring the fewest number of elements to be copied. One DG cell, however, must store a number of reconstruction coefficients proportional to the selected order of accuracy. At zeroth-order then, the actual amount of data required to be transferred could be the same for finite-volume and discontinuous Galerkin. (i.e. moving 5 stencil cells with one floating point value each is the same net movement as one cell with five floating point weighting values.) However, computer hardware effects lead DG to excel in this regard. For instance, in the implementation for this thesis, multiple DG nodal values for an element are stored contiguously in memory at all levels.

This is not true for finite volume cell values over a multi-dimensional stencil. Memory and cache systems are designed to fetch and store contiguous bytes in memory with fixed size called the read-line size. Typical read-line size for x86 CPU is 64 Bytes while for GPU they can by 128 Bytes or 32 Bytes [3]. An additional restriction is that read lines are always aligned to 64-byte address boundaries. A DG method may be implemented to use more data elements from the read-line than finite volume schemes.

Discontinuous Galerkin method is also better suited to achieve high-order-of-accuracy in complex geometries due to the more compact stencil compared to finite volume methods [22]. High-order finite volume methods using the *arbitrary high-order schemes using derivatives* (ADER) [23, 24] or *weighted essentially non-oscillatory* (WENO) [25] reconstruction techniques rely on quality stencil selection surrounding the evaluated cell. This becomes more difficult in higher dimensions, in the presence of grid distortion, and along physical boundaries. In the vicinity of a boundary the stencil must be one-sided or implemented to affect the boundary condition with multiple ghost cells.

For these reasons, the discontinuous Galerkin method was selected as the primary numerical method for PDE utilized in this thesis.

3.3 The discontinuous Galerkin method

3.3.1 Discontinuous Galerkin method background

The beginnings of the discontinuous Galerkin scheme started with development for steady neutron transport models by Reed and Hill [26]. Cockburn and Shu extended DG to general hyperbolic conservation laws in papers such as Refs. [27] and [28]. The description below follows the notation and order of operations of the instructive text by Hestaven and Warburton [22].

In a DG method, the global problem domain is approximated by a collection of K non-overlapping elements,

$$\Omega_h = \bigcup_{k=1}^{K} \mathsf{D}^k. \tag{3.5}$$

The elements D^k could be tetrahedral, hexahedral, or other shapes that tile the plane or fill the multidimensional volume.

In the DG approach, the global solution to Eq. (3.3) is approximated by piecewise reconstructions within each element. Inside one element D^k, the solution is assumed to be a linear combination of basis functions. The approximation can specified as modal linear expansion using N_p weights $\hat{q}_n(t)$ on basis functions $\psi_n(\mathbf{x})$,

$$\mathbf{x} \in \mathsf{D}^k : q_h^k(\mathbf{x}, t) = \sum_{n=1}^{N_p} \hat{q}_n^k(t)\psi_n(\mathbf{x}), \tag{3.6}$$

or as a nodal reconstruction with known nodal values $q_h(\mathbf{x}_i, t)$ at each of N_p points weighting the Lagrange interpolation polynomials $\ell_i(\mathbf{x})$,

$$\mathbf{x} \in \mathsf{D}^k : q_h^k(\mathbf{x}, t) = \sum_{i=1}^{N_p} q_h^k(\mathbf{x}_i^k, t)\ell_i^k(\mathbf{x}). \tag{3.7}$$

The modal and nodal representation are connected through the Vandermonde matrix,

$$\mathbf{q} = \mathcal{V}\hat{\mathbf{q}}, \text{ where, } \mathcal{V}^T\ell(x) = \psi(x) \text{ and } \mathcal{V}_{ij} = \psi_j(x_i). \tag{3.8}$$

To develop a scheme that solves Eq. (3.3), a space of test functions V_h is defined and the residual is required to be orthogonal to all test functions in the space expressed as

$$\int_{\mathsf{D}^k} \overbrace{\left(\frac{\partial q_h^k}{\partial t} + \nabla \cdot \mathbf{f}_h^k - g_h^k\right)}^{\text{residual}} \phi_h(\mathbf{x})d\mathbf{x} = 0, \quad \forall \phi_h \in \mathsf{V}_h. \tag{3.9}$$

From this requirement, a system of equations local to each element is established and will be developed here for the nodal approach. This equation system is transformed into the DG numerical scheme through some manipulation. The Galerkin naming distinguishes that the space of test functions ϕ_h will be the same as the space of the set of basis functions ψ_n.

The flux term is separated into an integral and then integration by parts is applied to transfer the spatial derivative from the flux term to the basis function. One equation results for each basis function, ψ_n,

$$\int_{\mathsf{D}^k} \left(\frac{\partial q_h^k}{\partial t}\psi_n - \mathbf{f}_h^k \cdot \nabla\psi_n - g_h^k\psi_n\right) d\mathbf{x} = -\int_{\partial\mathsf{D}^k} \hat{n} \cdot \mathbf{f}^k\psi_n d\mathbf{x}, \quad 1 \le n \le N_p. \tag{3.10}$$

For a consistent system across all elements D^k, the surface flux on the right hand side must be in agreement for neighboring elements. The surface flux term is replaced, then, with an evaluated numerical flux, f^*, in the face-normal direction and based on the states from both sides of the element-element surface, generally a Riemann problem solution. This is the weak form expression for the discontinuous Galerkin method,

$$\int_{\mathsf{D}^k} \left(\frac{\partial q_h^k}{\partial t} \psi_n - \mathbf{f}_h^k \cdot \nabla \psi_n - g_h^k \psi_n \right) d\mathbf{x} = - \int_{\partial \mathsf{D}^k} f^* \psi_n d\mathbf{x}, \quad 1 \leq n \leq N_p. \tag{3.11}$$

An alternative but equivalent form can be developed by applying integration by parts once again. This is the strong form expression for the discontinuous Galerkin method,

$$\int_{\mathsf{D}^k} \left(\frac{\partial q_h^k}{\partial t} + \nabla \cdot \mathbf{f}_h^k - g_h^k \right) \psi_n d\mathbf{x} = \int_{\partial \mathsf{D}^k} \left(\hat{n} \cdot \mathbf{f}_h^k - f^* \right) \psi_n d\mathbf{x}, \quad 1 \leq n \leq N_p. \tag{3.12}$$

3.3.2 Basic steps for DG

To implement a scheme to solve the weak form in Eq. (3.11) or strong form in Eq. (3.12), a system of equations for all basis functions ψ_n and all elements D_k is constructed. Preliminary work can be done for a reference element leading to straightforward evaluation at run-time.

First a set of basis functions is identified that should be orthonormal. The normalized Legendre polynomials is a popular choice but not the only choice.

A coordinate transformation is performed for every element k, reshaping the general element into a reference element, e.g. square or equilateral triangle. By doing so, required basis function derivatives and products can be pre-computed.

The semi-discrete form of the DG scheme is an ordinary differential equation that involves explicit and local calculations of the hyperbolic equation flux functions within the element as well as a numerical flux consistent between two elements sharing a face. Many choices of numerical flux are possible and the use of approximate Riemann solvers as used in finite volume methods is a popular choice. The semi-discrete form is developed with more detail in the next section leading to Eq. 3.21.

Finally, the semi-discrete ODE is integrated forward in time using an ODE integrator selected to achieve order-of-accuracy balanced with the spatial order and stability compatible with the stiffness and oscillatory characteristics of the ODE system.

3.4 Nodal Discontinuous Galerkin Semi-Discrete Implementation

The discontinuous Galerkin strong form of equation Eq. (3.12) is developed here as a compact linear algebra system per-node and per-element to evaluate the time derivative $\frac{dq_i^k}{dt}$, where superscript k denotes the element index and subscript i is the node index within the element. The nodal implementation solves for the time derivative at specific node locations within the element, $\mathbf{x}_i^k$, using the flux and source terms evaluated at node locations satisfying a numerical integration (quadrature) rule and additionally a numerical flux at each node on each face. The numerical flux is normal to the face and must be consistently evaluated for two adjacent elements.

It is more practical to implement the DG equation system in terms of a standard size reference element. The volume and surface integrals of Eq. (3.12) can be implemented using the reference element in reference coordinates $\mathbf{r}$ and coordinate transformation, $\frac{\partial \mathbf{r}}{\partial \mathbf{x}}$.

$$\int_{\mathsf{D}^k} q d\mathbf{x} = \int_{\mathsf{I}} q J^k d\mathbf{r}, \tag{3.13}$$

where the geometric Jacobian $J^k = \left|\frac{\partial \mathbf{r}}{\partial \mathbf{x}}\right|$.

The volume and surface integration from Eq. (3.12) are computed using a numeric quadrature rule with weights w_i,

$$\begin{aligned} \sum_{i=1}^{N_p} w_i J_i^k \psi_n(\mathbf{r}_i) \left(\frac{dq^k(\mathbf{r}_i)}{dt} - g^k(\mathbf{r}_i) \right) = \\ - \sum_{i=1}^{N_p} w_i J_i^k \psi_n(\mathbf{r}_i) \nabla_{\mathbf{x}} \cdot \mathbf{f}^-(\mathbf{r}_i) + \sum_{m=1}^{N_{\mathrm{fp}}} w_m^s J_m^k \left(\hat{n} \cdot \mathbf{f}^-(\mathbf{r}_m) - f^*(\mathbf{r}_m) \right) \psi_n(\mathbf{r}_m), \\ 1 \le n \le N_p. \end{aligned} \tag{3.14}$$

Equation (3.14) can be expressed as a linear system of equations,

$$\begin{aligned}
&[\psi_n(\mathbf{r}_j)]\,[\mathrm{diag}(w_i)]\,\left[\mathrm{diag}(J_i^k)\right]\left(\left[\frac{dq^k(\mathbf{r}_i)}{dt}\right]-\left[g^k(\mathbf{r}_j)\right]\right)= \\
&-\sum_{d=1}^{N_{\mathrm{dims}}}[\psi_n(\mathbf{r}_j)]\,[\mathrm{diag}(w_i)]\,\left[\mathrm{diag}(J_i^k)\right]\left[\frac{\partial \mathbf{r}}{\partial \mathbf{x}}|_{\mathbf{r}_i}\right]\left[\frac{\partial (f^- \cdot \hat{n}_d)}{\partial r_d}|_{\mathbf{r}_i}\right] \\
&\quad + [\psi_n(\mathbf{r}_m)]\,[\mathrm{diag}(w_m^s)]\,\left[\mathrm{diag}(J_m^k)\right]\left[\hat{n}\cdot\mathbf{f}^-(\mathbf{r}_m)-f^*(\mathbf{r}_m)\right].
\end{aligned} \tag{3.15}$$

The matrix $[\psi_n(\mathbf{r}_j)]$ is the transpose of the Vandermonde matrix, $\mathcal{V}$, connecting the basis functions to interpolation functions. The Vandermonde matrix has elements,

$$\mathcal{V}_{ij} = \psi_j(\mathbf{r}_i). \tag{3.16}$$

Left-hand multiplying Eq. (3.15) by $\left(\mathcal{V}^T\right)^{-1}$, the inverse of the volume integral quadrature weights, and the inverse of the Jacobian determinate yields,

$$\begin{aligned}
\left[\frac{dq^k(\mathbf{r}_i)}{dt}\right]-\left[g^k(\mathbf{r}_j)\right] &= \\
&- \sum_{d=1}^{N_{\mathrm{dims}}}\left[\frac{\partial \mathbf{r}}{\partial \mathbf{x}}|_{\mathbf{r}_i}\right]\left[\frac{\partial (f^- \cdot \hat{n}_d)}{\partial r_d}|_{\mathbf{r}_i}\right] \\
&+ \left[\mathrm{diag}(J_i^k)\right]^{-1}[\mathrm{diag}(w_i)]^{-1}\left(\mathcal{V}^T\right)^{-1} \\
&\quad [\psi_n(\mathbf{r}_m)]\,[\mathrm{diag}(w_m^s)]\,\left[\mathrm{diag}(J_m^k)\right]\left[\hat{n}\cdot\mathbf{f}^-(\mathbf{r}_m)-f^*(\mathbf{r}_m)\right].
\end{aligned} \tag{3.17}$$

To evaluate the flux divergence term, properties of the reconstructed polynomial flux function are utilized exactly as for the conserved variable.

$$f_h(\mathbf{r}_i) = \sum_{n=1}^{N_p}\hat{f}_n\psi_n(\mathbf{r}_i). \tag{3.18}$$

$$\frac{\partial f_h}{\partial r_d}|_{\mathbf{r}_i} = \sum_{n=1}^{N_p}\hat{f}_n\frac{\psi_n}{\partial r_d}|_{\mathbf{r}_i}. \tag{3.19}$$

This introduces the Grad-Vandermonde matrix, $\mathcal{V}_{r_d} = \left[\frac{\psi_j}{\partial r_d}|_{\mathbf{r}_i}\right]$. With that and the connection of mode weights to nodal values described in Eq. (3.8), the flux divergence term is,

$$\left[\frac{\partial(f^{-}\cdot\hat{n}_d)}{\partial r_d}|_{\mathbf{r}_i}\right] = \mathcal{V}_{r_d}\mathcal{V}^{-1}\left[f^{-}\cdot\hat{n}_d\right] = \mathbf{D}_{r_d}\left[f^{-}\cdot\hat{n}_d\right], \tag{3.20}$$

where $\mathbf{D}_{r_d}$ is the differentiation matrix in direction d.

The full linear system for the semi-discrete ODE can now be compactly expressed,

$$\frac{d\mathbf{q}^k}{dt} = \mathbf{g}^k - \sum_{d=1}^{N_{\text{dims}}} \mathbf{D}_{r_d}\mathbf{J}^k\mathbf{f}^{-} + \mathbf{L}(\hat{n}\cdot\mathbf{f}^{-} - \mathbf{f}^{*})\frac{J_s}{J^k}. \tag{3.21}$$

3.5 Element Type and Basis functions

The element shape directly impacts the ease by which complex simulation domain geometries can be partitioned into an element mesh. The elements must tile or fill the domain without overlap or gaps. A mixture of element shapes is readily possible.

Triangles, tetrahedra, etc. are good for automated mesh generation of complex geometries and several free tools exist to assist the process [29, 30]. One draw back for this class of element shape is that the resulting mesh is unstructured, meaning that an ordered logical connection between elements does not exist. Instead, connectivity between one element face to another adjacent element must be established through a more sophisticated means. Square, cubes, and hypercubes are more challenging to use in mesh generation but can be utilized in structured meshes improving localized communication and data storage.

For the studied Vlasov-Maxwell application, hypercubes are used exclusively to discretize n-dimensional phase space. There are several benefits specific to this application. Quality sets of basis functions are easy to establish. A direct relationship between physical-space elements and phase-space elements of the same class is established making projections of electromagnetic fields and reduction of probability distribution function moments straightforward. Structured mesh discretization of velocity space is important for high performance implementation of the scheme where all of velocity space for a given physical point is contained in a shared memory domain and tens or hundreds of processor cores compute the DG update and moments in parallel.

Focus is on structured arrangements of hexahedra for this thesis due to straightforward workload decomposition and adaptability to GPU architectures targeted by the WARPM code. A capability for block structured meshes will be developed later to significantly improve flexibility to model complex physical domains.

The normalized Legendre polynomials as basis functions for the DG scheme are described in detail in Ref. [22]. In one variable, the n^{th}-order Legendre polynomials are the n^{th}-order Jacobi polynomials, $P_n^{(\alpha,\beta)}(r)$, when $\alpha = \beta = 0$.

Tensor-product hypercube elements represent the solution through a set of basis functions made by linear products of the one-dimensional Legendre polynomials,

$$\Psi_n(\mathbf{r}) = \prod_{d=1}^{N_D} P_M^{(0,0)}(r_d), \tag{3.22}$$

where $M(n, d)$ is a function that converts the multi-dimensional mode linear index, n, to a one-dimensional mode index in dimension d. Generally a row-major ordering scheme is utilized throughout. A two-dimensional example is depicted in Figure 3.1. For this example, the function $M(n, d)$ is listed in Table 3.1.

3.6 Optimizations for tensor product hypercubes

Generally, the flux at each node affects the solution's time derivative at all other nodes within the element. The working matrices $\mathbf{D}_{r_d}$, $\mathbf{L}$, and geometric Jacobian $\mathbf{J}^k$ are dense.

The selection of element class (e.g. triangles, hypercubes, etc.) and nodal location can have an extremely significant impact on practical storage requirements, cache locality, and operation count when working in higher dimensional space such as the five- and six-dimensional phase space required for the kinetic Vlasov model.

Tensor-product hypercubes yield potential for significant optimizations due to the linear combination of one-dimensional basis function polynomials and resulting orthogonal derivative operators. The floating point operations required per element and storage requirements are reduced as summarized in Table 3.2 due to predictable sparsity in the operator matrices.

Table 3.1: Linear index to multi-index conversion function $M(n, d)$ for the 2nd-order × 3rd-order square element depicted in Figure 3.1.

Dim. 1 mode	Dim. 2 mode
$M(1, 1) = 1$	$M(1, 2) = 1$
$M(2, 1) = 1$	$M(2, 2) = 2$
$M(3, 1) = 1$	$M(3, 2) = 3$
$M(4, 1) = 1$	$M(4, 2) = 4$
$M(5, 1) = 2$	$M(5, 2) = 1$
$M(6, 1) = 2$	$M(6, 2) = 2$
$M(7, 1) = 2$	$M(7, 2) = 3$
$M(8, 1) = 2$	$M(8, 2) = 4$
$M(9, 1) = 3$	$M(9, 2) = 1$
$M(10, 1) = 3$	$M(10, 2) = 2$
$M(11, 1) = 3$	$M(11, 2) = 3$
$M(12, 1) = 3$	$M(12, 2) = 4$

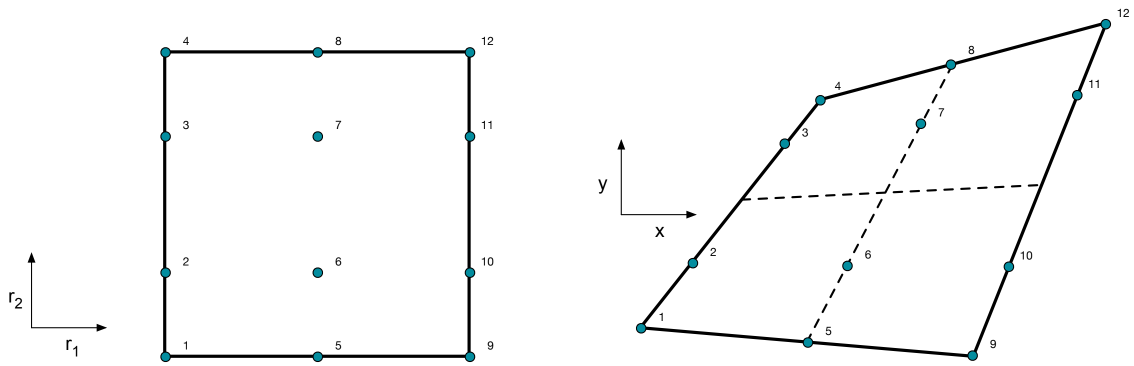

Figure 3.1: Illustration of the 2D reference square element including node ordering for element order ($2^{\text{nd}} \times 3^{\text{rd}}$). The element modal basis set is a tensor product of two 1-D Legendre polynomial basis sets. When the reference element is projected into physical space as a quadrilateral the transformation is bilinear such that lines through nodes in reference space remain lines through nodes in physical space.

Specific examples are given in Tables 3.3 through 3.5 highlighting the significance for high dimensional domains.

The practical performance improvement is ultimately limited by memory bandwidth. This optimization does not change the number of inputs (nodal flux evaluations) or outputs (nodal time derivatives) stored in global memory. It does reduce the storage size of the operator matrices $\mathbf{D}_r$ and $\mathbf{L}$ which reduce the pressure on memory read bandwidth and makes more cache available to the nodal data.

	General Dense Hypercube	Tensor Product Hypercube
$\mathbf{D}_r$ Storage	$N_D(P+1)^{2N_D}$	P odd: $\frac{1}{4}(P + 1)^2$ P even: $(\frac{P}{2}+1)^2$
$\mathbf{L}$ Storage	$2N_D(P+1)^{2N_D-1}$	(P+1)
Volume Integral FLOP	$\mathcal{O}(2N_D(P+1)^{2N_D})$	$\mathcal{O}(2N_D(P+1)^{(N_D+1)})$
Surface Integral FLOP	$\mathcal{O}(4N_D^2(P+1)^{2N_D-1})$	$\mathcal{O}(4N_D(P+1)^{N_D})$

Table 3.2: Summary of floating point operations for DG element evaluation of time derivative on a hypercube element. The number of nodes per N_D-dimensional element with polynomial order P is $N_p = (P+1)^{N_D}$ and the number of face points is $N_{\text{fp}} = 2N_D(P+1)^{N_D-1}$.

	General Dense Hypercube	Tensor Product Hypercube	Reduction
Number of nodes N_p	256	256	
Number of face points N_{fp}	512	512	
$\mathbf{D}_r$ Storage	2048 kB	128 B	6×10^{-5}
$\mathbf{L}$ Storage	1024 kB	32 B	3×10^{-5}
Volume Integral FLOP	5×10^5	8×10^3	1.6×10^{-2}
Surface Integral FLOP	1×10^6	4×10^3	4×10^{-3}

Table 3.3: Example reduction in FLOP and storage requirements for DG tensor product hypercube element with element order, $P = 3$, and four-dimensional hypercube, $N_D = 4$.

	General Dense Hypercube	Tensor Product Hypercube	Reduction
Number of nodes N_p	1024	1024	
Number of face points N_{fp}	2560	2560	
$\mathbf{D}_r$ Storage	42 MB	128 B	3×10^{-6}
$\mathbf{L}$ Storage	21 MB	32 B	2×10^{-6}
Volume Integral FLOP	1.0×10^{7}	4.1×10^{4}	3.9×10^{-3}
Surface Integral FLOP	2.6×10^{7}	2.0×10^{4}	8×10^{-4}

Table 3.4: Example reduction in FLOP and storage requirements for DG tensor product hypercube element with element order, $P = 3$, and five-dimensional hypercube, $N_D = 5$.

	General Dense Hypercube	Tensor Product Hypercube	Reduction
Number of nodes N_p	4096	4096	
Number of face points N_{fp}	12288	12288	
$\mathbf{D}_r$ Storage	805 MB	128 B	1.6×10^{-7}
$\mathbf{L}$ Storage	403 MB	32 B	3.2×10^{-7}
Volume Integral FLOP	2×10^{8}	2.0×10^{5}	9.8×10^{-4}
Surface Integral FLOP	6×10^{8}	9.8×10^{4}	1.6×10^{-4}

Table 3.5: Example reduction in FLOP and storage requirements for DG tensor product hypercube element with element order, $P = 3$, and six-dimensional hypercube, $N_D = 6$.

3.7 High-Accuracy Discontinuous Galerkin Working Matrix Initialization

As one part of reducing global solution error, it is worthwhile to give extra effort to preparing working matrices of the discontinuous Galerkin scheme with high accuracy. Even though the matrices are utilized with 64-bit double-precision floating point values repeatedly during time integration for computational efficiency, improved error performance can be gained by calculating the working matrix values initially using a precision higher than native double-precision floating point arithmetic provides. Then, after evaluation at high precision, the working matrices are down-converted to native double-precision numbers with net improvement in accuracy.

Use of an arbitrary precision tool or symbolic math manipulations can be used to conduct the high precision evaluations. The below list summarizes the required steps for initializing the discontinuous Galerkin working matrices based on the algorithms described in Ref. [22].

- Compute reference element node locations and quadrature weights associated with P^{th}-order Gauss-Lobatto quadrature ($P+1$ quadrature points). Points and weights are associated with roots of the Jacobi polynomial of type $\alpha = 0$ and $\beta = 0$.

  $$\mathbf{r}_{\text{1D}} = [-1, \text{JacobiGQ}(1, 1, P-2), 1]\,.$$

  Above, P is the quadrature order with respect to the reference dimension, r. The function JacobiGQ$(1, 1, N)$ returns a vector of length $N+1$. It can be evaluated as follows. Construct the diagonal matrix $\mathbf{Q}$ which is size $(N+1) \times (N+1)$ and is all zeros except for the following $2N$ entries adjacent to the main diagonal:

  $$Q_{n,(n+1)} = Q_{(n+1),n} = \frac{2}{2n+2}\sqrt{\frac{n^4+4n^3+5n^2+2n}{4n^2+8n+3}},$$

  where $n \in [1, N]$. The sorted eigenvalues of $\mathbf{Q}$ are the quadrature points, or the returned vector from JacobiGQ$(1, 1, N)$.

  Quadrature weights are expressed,

$$\mathbf{w}_{1\mathrm{D}} = \left[\frac{2}{P^2+P}, \frac{2P+1}{(P^2+P)(\mathrm{JacobiP}(\mathrm{JacobiGQ}(1,1,P-2),0,0,P))^2}, \frac{2}{P^2+P}\right].$$

The function $\mathrm{JacobiP}(x, \alpha, \beta, P)$ is the normalized Jacobi polynomial of order P and type α, β.

- Compute the multi-dimensional reference element node locations. The aim is to support simulation domains decomposed into elements that are quadrilaterals, hexahedra, or a higher dimensional generalization thereof. Many properties for the DG scheme can be pre-computed for a single reference element that is a line, square, cube, or hypercube with side length of 2 and centered on the origin.

  The reference element node coordinates are established by a rectilinear mesh from the one-dimensional quadrature points. For example, a two-dimensional (square) reference element is represented in Cartesian reference coordinates r_1 and r_2. Element polynomial order in r_1 is P_1 and order in r_2 is P_2. The coordinates are stored in two vectors $\mathbf{r}_1$ and $\mathbf{r}_2$ of length $N_{\mathrm{nodes}} = (P_1 + 1) \times (P_2 + 1)$. Node ordering is arbitrary but must be consistent. Here we use a row-major ordering scheme. The vector of quadrature weights $\mathbf{w}$ has the same length and ordering. At each node, the quadrature weight is the product of the two corresponding 1D weights. The below equivalent Matlab code describes the row-major node ordering and Figure 3.1 provides an illustration for a particular case of 2D quadrilateral.

```
for i=1:(P1+1)
    for j=1:(P2+1)
        n = (i-1)*(P1+1)+j;%row-major ordering
        r1(n) = r_1D_1(i); %node position in coordinate r0
        r2(n) = r_1D_2(j); %node position in coordinate r1
        w(n)  = w_1D_1(i) * w_1D_2(j);  %quadrature weight
    end
end
```

- Evaluate Vandermonde matrix $\mathbf{V}$ for the polynomial basis set that is the tensor product of 1D Legendre polynomial basis sets at the desired reference element node locations. Matrix $\mathbf{V}$ is size $N_{\text{nodes}} \times ((P_1+1)(P_2+1))$.

$$\mathbf{V}|_{\text{col}=(i-1)*(P_1+1)+j} = \text{JacobiP}(\mathbf{r}_1, 0, 0, i-1)\,\text{JacobiP}(\mathbf{r}_2, 0, 0, j-1),$$

where $i \in [1, P_1+1]$ and $j \in [1, P_2+1]$ and the right hand side operator is an element-by-element multiplication. The below equivalent MATLAB code may provide some clarity to the above notation.

```
for i=1:(P1+1)
    for j=1:(P2+1)
    n = (i-1)*(P2+1)+j;
    V(:,n) = JacobiP(r1(:),0,0,i-1).*JacobiP(r2(:),0,0,j-1);
    end
end
```

- The inverse of the Vandermonde matrix $\mathbf{V}^{-1}$ is required when continuing further setup for DG evaluation. In this case $\mathbf{V}$ is square – the interpolation points *are* the reference element nodes. The Vandermonde matrix is also used when interpolating the DG nodal solution to different points in the element. In this usage, the matrix inverse of the non-square matrix is not required.

- Evaluate the gradient-Vandermonde matrices $\mathbf{V}_r$ and $\mathbf{V}_s$ for the polynomial basis set that is the tensor product of 1D Legendre polynomial basis sets at the desired node locations. Element polynomial order in r_1 is P_1 and order in r_2 is P_2. The size of $\mathbf{V}_{r1}$ and $\mathbf{V}_{r2}$ is $N_{\text{nodes}} \times ((P_1+1)(P_2+1))$.

$$\begin{aligned} \mathbf{V}_{r1}|_{\text{col}=(i-1)*(P_2+1)+j} &= \frac{\partial}{\partial r_1}\big(\,\text{JacobiP}(\mathbf{r}_1,0,0,i-1)\big)\,\text{JacobiP}(\mathbf{r}_2,0,0,j-1) \\ &= \sqrt{i^2-i}\,\text{JacobiP}(\mathbf{r}_1,1,1,i-2)\,\text{JacobiP}(\mathbf{r}_2,0,0,j-1), \end{aligned}$$

  where $i \in [1, P_1+1]$ and $j \in [1, P_2+1]$.

- Evaluate the differentiation matrices $\mathbf{D}_{r1}$ and $\mathbf{D}_{r2}$.

$$\mathbf{D}_{r1} = \mathbf{V}_{r1}\mathbf{V}^{-1}.$$

$$\mathbf{D}_{r2} = \mathbf{V}_{r2}\mathbf{V}^{-1}.$$

- Evaluate a face Vandermonde matrix for each reference element face, $\mathbf{V}_f$, for points $\mathbf{t}$ along the face. The number of points on each face depends on the element order in every other dimension. For each of the two faces in dimension-α,

$$N_{\text{fp}}^{\alpha} = \prod_{\substack{d=1 \\ d\neq\alpha}}^{N_{\text{dims}}} P_d + 1.$$

  For square elements, each face is one-dimensional

$$\mathbf{V}_f|_{\text{col}=j} = \text{JacobiP}(\mathbf{t},0,0,j-1),$$

  where $j \in [1, P_t+1]$.

- Evaluate the 1D Mass matrix for each reference element face, $\mathbf{M}_f$.

$$\mathbf{M}_f = (\mathbf{V}_f\mathbf{V}_f^T)^{-1}.$$

- The edge mass matrix, $\mathbf{E}$, is size $N_{\text{p}} \times N_{\text{fp}}$ where

  $$N_{\text{fp}} = \sum_{\alpha=1}^{N_{\text{dims}}} 2N_{\text{fp}}^{\alpha}.$$

  Each column of $\mathbf{E}$ is populated by one column from a face mass matrix, $\mathbf{M}_f$, according to a whole-element face-node indexing scheme.

- The LIFT matrix, size $N_{\text{p}} \times N_{\text{fp}}$, is used to calculate surface flux contribution to the element.

$$\mathbf{L} = \mathbf{V}\mathbf{V}^T\mathbf{E}.$$

3.7.1 Example for 2D reference element of 2nd-order by 3rd-order.

To demonstrate the reference element construction, the following results are for a square element that is 2nd-order in r_0 and 3rd-order in r_1.

$$\mathbf{r}_{\text{1D}}^{\text{2nd}} = \begin{pmatrix} -1, & 0, & 1 \end{pmatrix}.$$

$$\mathbf{w}_{\text{1D}}^{\text{2nd}} = \begin{pmatrix} \frac{4}{3}, & \frac{1}{3}, & \frac{4}{3} \end{pmatrix}.$$

$$\mathbf{r}_{\text{1D}}^{\text{3rd}} = \begin{pmatrix} -1, & -\sqrt{\frac{1}{5}}, & \sqrt{\frac{1}{5}}, & 1 \end{pmatrix}.$$

$$\mathbf{w}_{\text{1D}}^{\text{3rd}} = \begin{pmatrix} \frac{1}{6}, & \frac{5}{6}, & \frac{5}{6}, & \frac{1}{6} \end{pmatrix}.$$

$$\mathbf{r}_1 = \begin{pmatrix} -1 \\ 0 \\ 1 \\ -1 \\ 0 \\ 1 \\ -1 \\ 0 \\ 1 \\ -1 \\ 0 \\ 1 \end{pmatrix}, \mathbf{r}_2 = \begin{pmatrix} -1 \\ -1 \\ -1 \\ -\sqrt{\frac{1}{5}} \\ -\sqrt{\frac{1}{5}} \\ -\sqrt{\frac{1}{5}} \\ \sqrt{\frac{1}{5}} \\ \sqrt{\frac{1}{5}} \\ \sqrt{\frac{1}{5}} \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

3.8 Types of Errors With Discontinuous Galerkin Method

Several different types of error are introduced using the discontinuous Galerkin scheme to discretize and solve hyperbolic partial differential equations. They are distinguished and described in the following sections.

3.8.1 Projection Error

Any exact solution not contained within the space of basis functions for the DG scheme will have an associated projection error – the difference between the exact solution and the reconstructed polynomial solution, u_h, that shares the same nodal values. This error can be quantified by the L^2 norm over an element D^k,

$$\|\mathcal{E}_{\text{proj.}}\|^2 = \int_{\mathsf{D}^k} (u_h(\mathbf{r}) - u(\mathbf{r}))^2 d\mathbf{r}. \tag{3.23}$$

This error is independent of the time integration by the discontinuous Galerkin scheme. Projection error can be introduced even in the initial condition of a simulation if the initial condition function is not in the space spanned by the basis functions. It is helpful to characterize this error, then, before considering error caused by the DG time integration scheme.

To illustrate projection error, a useful example is to look at the Legendre polynomial basis function projection onto the trigonometric function $u(r) = 1 + \cos(\pi r)$. Figure 3.2 depicts the basis function projection, $u_h(r)$ and exact function, $u(r)$, for several different order basis sets. Improved agreement with increasing order is visually obvious. To quantify the quality of the projection, Figure 3.3 presents the L^2 norm of the projection error as expressed by Eq. (3.23).

An even-odd trend is observed in Figure 3.3, where error improvement is only achieved with each new even-order of polynomial basis for the even example function $u(r)$. Similar behavior occurs for the odd function $1 + \sin(\pi r)$, except that improvement occurs with each new odd-order polynomial basis. This behavior is a special case not expected for general functions that are not odd nor even.

Projection error may also be introduced as a simulation is advanced in time if the exact solution leaves the space spanned by the basis set. The Vlasov-Advection equation is a useful example of this effect and relevant to the greater work of this thesis. The hyperbolic PDE describes the advection of a quantity u in a two-dimensional space (x, v) where the advection speed is the velocity coordinate v. Advection parallels the x-axis.

$$\frac{\partial u}{\partial t} + v\frac{\partial u}{\partial x} = 0. \tag{3.24}$$

This is a particularly useful system to study because it has an analytical solution. On an infinite domain, $u(x, v, t) = u_0(x - vt, v)$. A function periodic in x remains periodic with the same wave number, k_x, and the wave number k_v increases as the initial condition becomes increasing striated or shears repeatedly. Based on the previous observations, projection error should increase with time for a given element order in velocity.

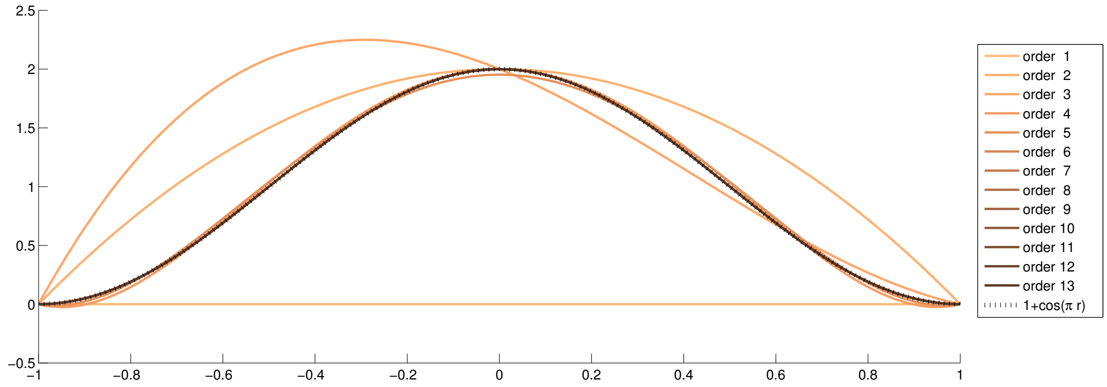

Figure 3.2: This series of plots presents the projections of the function $1 + \cos(\pi r)$ onto the Legendre basis function set of indicated order versus r. The Legendre basis functions are ortho-normal on the domain $r \in [-1, 1]$.

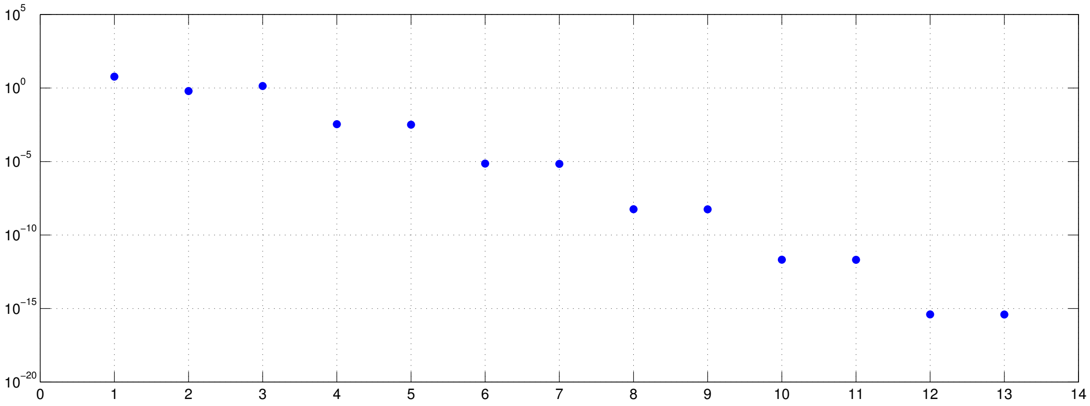

Figure 3.3: The L^2 norm of the projection error resulting from projecting the function $1 + \cos(\pi r)$ onto the Legendre basis function set. The domain of integration is $r \in [-1, 1]$. The horizontal axis is the polynomial order of the basis set. An even-odd trend is made clear by the stair-step results beyond order one.

Figure 3.4 demonstrates this effect showing increasing projection error as time increases, wave number in the velocity dimension increases, and the analytical function becomes undersampled. The initial condition under test is $u_0(x, v) = 1 + \cos(\pi x)$. Increasing the order of basis functions reduces the projection error, but for all cases, L^2 norm of projection error eventually grows to be on par with the L^2 norm of the analytical solution itself – the error is as big as the solution. L^2 norm of projection error is evaluated per Eq. (3.23), where $\mathbf{r} = [x, v]$.

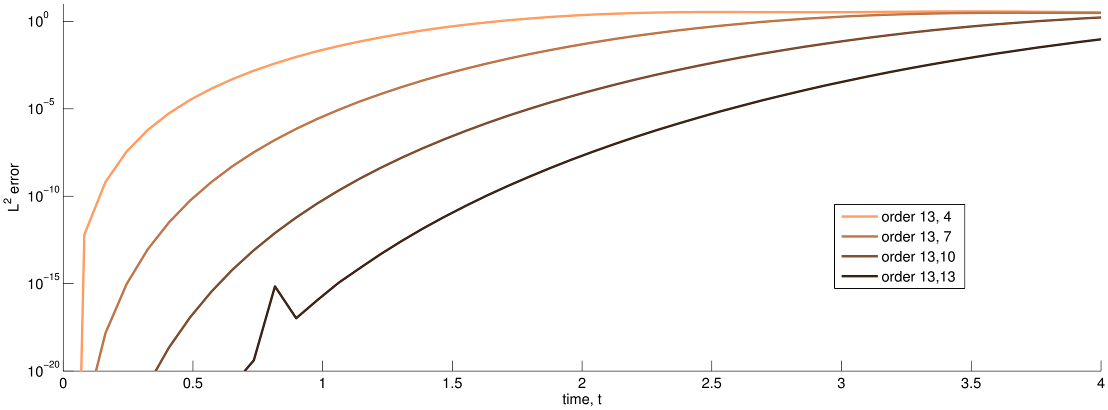

Figure 3.4: The L^2 norm of the projection error resulting from projecting the function $1 + \cos(\pi(x - vt))$ onto the Legendre basis function set. The domain of integration is one square reference element, $x \in [-1, 1]$ and $v \in [-1, 1]$. Basis polynomial order in x is constant and sufficient, while multiple order in v are tested: $[4^{\text{th}}, 7^{\text{th}}, 10^{\text{th}}, 13^{\text{th}}]$.

3.8.2 Quadrature Error

The discontinuous Galerkin scheme involves the use of numeric quadrature rules for integration of such quantities as surface flux and terms specific to the Vlasov equation system. In this work, the Gauss-Lobatto quadrature rule [31] is utilized giving accuracy for polynomials of order up to $2N - 3$, where N is the number of quadrature points. The Gauss-Lobatto rule is well suited for the nodal DG scheme because the quadrature points include the extremum of the range of integration. In some cases, the N-point rule is also referred to as $(N-1)$-order.

Integration by quadrature rule introduces error for polynomials exceeding the accuracy limit for the rule or non-polynomial integrands. The error is readily characterized,

$$\mathcal{E}_{quad} = \int_{x_1}^{x_2} u(x)dx - \frac{x_2 - x_1}{2} \sum_{n=1}^{N_p} w_n u(x_n). \tag{3.25}$$

Integration error for the non-polynomial function $u(x) = 1 + \cos(2\pi x)$ serves as a useful example. The error is presented in Figure 3.5 illustrating improved accuracy with increasing quadrature order as well as increasing error with increase of the integration domain length.

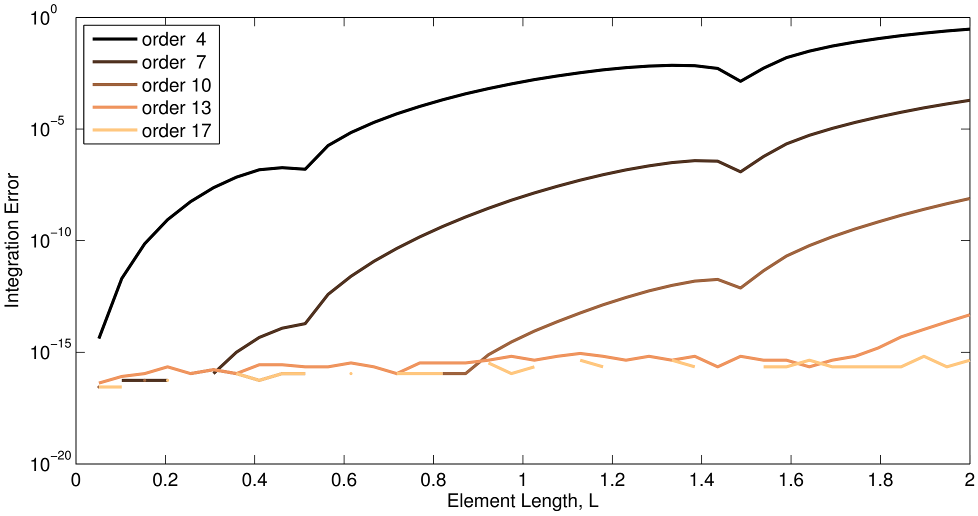

Figure 3.5: Gauss-Lobatto quadrature error for integration of $1+\cos(2\pi x)$ over the interval $x \in [0, L]$ (Absolute value of Eq. (3.25)). Multiple cases of quadrature order are tested: $[4^{\text{th}}, 7^{\text{th}}, 10^{\text{th}}, 13^{\text{th}}]$. Also apparent is a numeric noise floor is observed around 1×10^{-16} consistent with 64-bit floating point arithmetic.

3.8.3 Floating Point Arithmetic Precision

Also observed in Figure 3.5 is the limit on accuracy caused by the limited precision of floating point arithmetic. A numeric noise floor is observed around 1×10^{-16} consistent with 64-bit floating point arithmetic.

Figure 3.6 repeats the same quadrature analysis as Figure 3.5 except that the arithmetic is done using an arbitrary precision tool allowing the user to specify a level of numeric precision to maintain. The numeric noise floor is reduced to around 1×10^{-20} by directing the tool to maintain twenty decimal digits of precision.

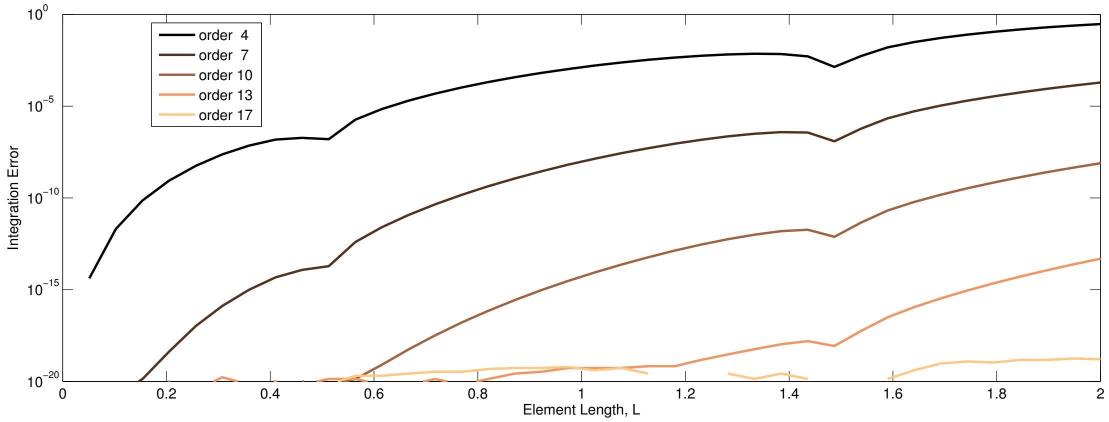

Figure 3.6: This figure repeats the analysis performed for Figure 3.5 except that the floating point precision is improved to about 20 decimal digits using an arbitrary precision math tool. The floating point noise floor is reduced while the results otherwise remain unchanged.

The example here is quadrature evaluation, but the limits of floating point precision affect discontinuous Galerkin solution in many ways. In general, the error is unavoidable and just something to be aware of. Certain types of calculations like summing large and small numbers are more error prone.

It is generally not practical to perform time integration of a large scale simulation using arbitrary precision arithmetic. The computational cost compared to native double-precision calculations is very high.

Certain key working matrices of the discontinuous Galerkin scheme are pre-computed and utilized many times over in evaluating each time integration step. For these matrices, it is practical and beneficial to pre-compute their double-precision values using enhanced precision arithmetic. Doing so helped to eliminate very small but constantly growing errors in the conserved moments of the Vlasov-Poisson system. More details for this technique is described in Section 3.7.

3.8.4 Discretization Error

Like all numerical solvers for partial differential equations, the discontinuous Galerkin method is subject to discretization error for both discretized time and space.

For systems with analytical solution, $\tilde{u}$, the global solution error by the numerical scheme can be characterized by the L^2 norm,

$$\|\mathcal{E}(t)\|^2 = \int_\Omega \left(u_h(x,t) - \tilde{u}(x,t)\right)^2 dx. \tag{3.26}$$

The distinction between Eq. (3.23) and Eq. (3.26) considered here is that the later is a global evaluation over the full domain Ω which is discretized by many elements and error introduced by discrete time integration is also included.

Figure 3.7 presents this error characterization for the simple constant advection system,

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0, \tag{3.27}$$

with $a = 10$ and periodic boundaries. Six different simulation test cases are presented by varying the number of elements making up the domain and for two different element spatial order. Increasing the element order or increasing the number of elements decreases the global solution error. Since the solution to this system maintains the same shape as the initial condition, simply translated, any projection error should be effectively capped at an initial level allowing focus on the discretization error specifically.

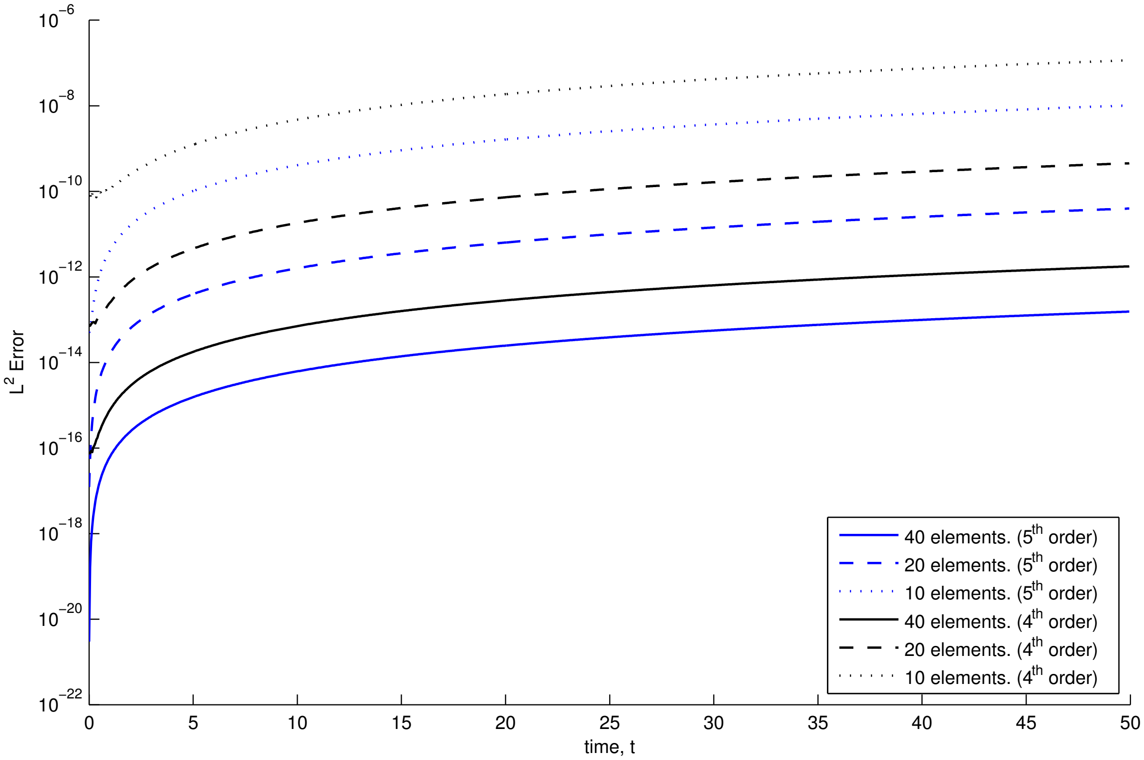

Figure 3.7: The error as L^2 norm per Eq. (3.26) for the constant linear advection system with initial condition $u_0(x) = 1 + \frac{1}{2}\cos\left(\frac{1}{2}x\right)$ on the domain $x \in [0, 4\pi]$.

3.9 Advanced discontinuous Galerkin Schemes

Generally, the discontinuous Galerkin spatial discretization is paired with a multi-step Runge-Kutta numerical ODE solver for temporal discretization [27, 28, 32, 33, 22, 34]. RKDG is conceptually simple and easy to implement, but it has drawbacks. First, being a multi-step method, it faces scaling challenges relative to single-step methods for large computations on distributed machines. Between each sub-step, data communication is required between adjacent elements. Second, when high order-of-accuracy is required, RK schemes become less efficient. Above fourth-order-of-accuracy the Butcher-barrier[35] is encountered; the number of sub-steps required exceeds the order of accuracy as seen in Table 3.6.

Order	$O(\Delta t^4)$	$O(\Delta t^5)$	$O(\Delta t^6)$	$O(\Delta t^7)$	$O(\Delta t^8)$	$O(\Delta t^9)$	$O(\Delta t^{10})$
sub-steps	4	6	7	9	11	12-17	13-17

Table 3.6: Number of sub-steps required versus accuracy in explicit Runge-Kutta methods

One-step numerical methods tailored to discontinuous Galerkin and finite volume methods have been developed over the last ten years that may prove more computationally scalable than RKDG while high order in space and time [36, 37, 38, 39].

3.9.1 One-Step High Order Methods

Based on manipulations of the equation system of interest, several one-step methods have been developed that can achieve, in principle, any order of temporal accuracy [40, 24, 41]. The common theme is the following. First, perform a Taylor series expansion of the base PDE system in time. Then, replace higher order time derivatives and mixed time and space derivatives with spatial derivatives only using the differentiated PDE and Lax-Wendroff (also Cauchy-Kowalewski) procedure [42, 43]. The resulting truncated Taylor series PDE is then solved using the presented DG method. Schemes as described were originally called Lax-

Wendroff discontinuous Galerkin (LWDG) and more recently are referred to as arbitrary high order schemes using derivatives (ADER) discontinuous Galerkin (ADER-DG).

The resulting method is single-step and thus the Butcher barrier is avoided. Even for fourth-order accuracy, higher performance can be achieved compared to RKDG. The ADER-DG method developed by Michael Dumbser[41], for example, reports 35% performance improvement over RKDG on a single-node simulation. The performance benefits are most pronounced when each step involves expensive computations like non-linear limiter applications.

The main disadvantage of the Lax-Wendroff based one-step methods is that the resulting scheme is more complex. Implementing a new temporal order of accuracy requires both analytical development which can be complicated for multidimensional systems and translation of the resulting new PDE system into computer software for the numerical method.

3.9.2 Suitability to many-core and GPU computation

As will be discussed more in the following chapter, GPU systems are characterized by high numerical performance limited by multiple memory hierarchies and bandwidth restrictions.

ADER-DG is more memory space efficient than RKDG as only one time history is required instead of a number fractional time steps based on the order of accuracy. In the aggregate, this not only means fitting a larger problem, it also means less memory communication and better scaling.

RKDG methods with high order accuracy in time are less suitable for GPU and many-core architectures due the multi-step approach in time. Implementing an explicit multistep method for PDE is disadvantaged in that global ghost cell syncing is required for each sub-step. Global ghost cell synching exercises every communication hierarchy including the inter-node network with the slowest bandwidth and greatest latency. Excess global communication versus the quantity of local computation degrades the scaling performance of the scheme on an HPC system.

3.9.3 Identified challenges

Qiu et al. pointed out in Ref. [40] that ADER-DG is not as developed for stiff PDE systems or those with diffusive parts. RKDG has much more developed literature in support of such PDE systems. They do claim ADER-DG could be made suitable through, "careful Taylor expansions."

The ADER-DG method applied to the Vlasov equation requires development of a generalized Riemann problem solution. Review of current literature has not identified existing published work. This is expected to be tenable given the simple nature of the flux terms in the Vlasov equation.

3.9.4 Prototype ADER-DG development for 2D Vlasov system

To demonstrate the Lax-Wendroff procedure and ADER-DG PDE preparation for a system of interest, the basic steps applied to the two-dimensional Vlasov PDE system are outlined below to third-order accuracy in time. The procedure follows that detailed by Qiu et al. [40]. The notation $f_x(q)$ means $\frac{\partial f}{\partial x}$ and notation $f'(q)$ means $\frac{df}{dq}$.

The two-dimensional (position x and velocity v) Vlasov system is a conservation law of the form,

$$q_t + f_x(q) + g_v(q) = 0, \tag{3.28}$$

where

$$\begin{aligned} f(q) = &\quad qv, \\ g(q) = &\quad q\frac{q_c}{m}E(x,t). \end{aligned} \tag{3.29}$$

The evolution of Eq. (3.28) can be evaluated through Taylor series expansion,

$$q(x,v,t+\Delta t) = q(x,v,t) + \Delta t q_t + \frac{1}{2}\Delta t^2 q_{tt} + \frac{1}{6}\Delta t^3 q_{ttt} + \cdots . \tag{3.30}$$

Differentiation rules and algebraic manipulations are used to reformulate the higher order time derivatives needed in the Taylor series expansion. The particular final forms expressed below are useful in the Lax-Wendroff procedure,

$$\begin{aligned}
q_t &= -f_x(q) - g_y(q) = -f'(q)q_x - g'(q)q_y, \\
q_{tt} &= -f_{tx}(q) - g_{ty}(q) = -(f'(q)q_t)_x - (g'(q)q_t)_y \\
&= -(f''(q)q_xq_t + f'(q)q_{xt} + g''(q)q_yq_t + g'(q)q_{yt}), \\
q_{xt} &= -\left(f''(q)(q_x)^2 + f'(q)q_{xx} + g''(q)q_xq_y + g'(q)q_{xy}\right), \\
q_{yt} &= -\left(g''(q)(q_y)^2 + g'(q)q_{yy} + f''(q)q_xq_y + f'(q)q_{xy}\right), \\
q_{ttt} &= -(f'(q)q_t)_{xt} - (g'(q)q_t)_{yt} \\
&= -\left(f''(q)(q_t)^2 + f'(q)q_{tt}\right)_x - \left(g''(q)(q_t)^2 + g'(q)u_{tt}\right)_y .
\end{aligned}$$

Truncating the expansion Eq. (3.30) yields an approximation expressed as

$$q(x,v,t+\Delta t) = q(x,v,t) - \Delta t(F_x + G_y), \tag{3.31}$$

where for third-order approximation,

$$\begin{aligned}
F &= f + \frac{\Delta t}{2} f'(q)q_t + \frac{\Delta t^2}{6}(f''(q)(q_t)^2 + f'(q)q_{tt}), \\
G &= g + \frac{\Delta t}{2} g'(q)q_t + \frac{\Delta t^2}{6}(g''(q)(q_t)^2 + g'(q)q_{tt}).
\end{aligned}$$

$$\begin{aligned}
q(x,v,t+\Delta t) &\approx q(x,v,t) - \Delta t(F_x + G_y) \\
&= q(x,v,t) + \Delta t(f_x + g_y) \\
&\quad + \frac{\Delta t^2}{2}\left(f''(q)q_xq_t + f'(q)q_{xt} + g''(q)q_yq_t + g'(q)q_{yt}\right) \\
&\quad + \frac{\Delta t^3}{6}\left(\left(f''(q)(q_t)^2 + f'(q)q_{tt}\right)_x - \left(g''(q)(q_t)^2 + g'(q)u_{tt}\right)_y\right) \\
&= q(x,v,t) + \Delta t q_t + \frac{1}{2}\Delta t^2 q_{tt} + \frac{1}{6}\Delta t^3 q_{ttt}.
\end{aligned}$$

The Vlasov-Poisson equation system has only linear dependence on q, so the required terms are particularly straight-forward to calculate,

$$\begin{aligned} f'(q) &= v, \\ f''(q) &= 0, \\ g'(q) &= \frac{q_c}{m}E(x), \\ g''(q) &= 0. \end{aligned}$$

To complete the ADER-DG scheme, equation Eq. (3.31) is developed into the discontinuous form using the same process as described in Section 3.3.2, but with the new flux functions F and G.

Chapter 4

WARPM SIMULATION CODE FOR MANY-CORE ARCHITECTURES

WARPM is a new simulation code developed as a principal component of this research and to enable improved simulation of plasmas and other systems of hyperbolic conservation laws. It has been designed from the ground-up targeting emerging many-core architectures. These modern architectures include GPU-accelerated systems and are characterized by multiple levels of memory hierarchy and high cost of data movement. Between memory levels, there is an increasing penalty for moving data as it moves farther from the functional unit relative to the numerical operation performance. This is easy to imagine given energy and bandwidth considerations and is the most significant trend in HPC affecting computational science. While raw floating-point arithmetic performance continues to increase, memory bandwidth to functional units and between distributed nodes lags FLOPS and will continue to do so. At the same time, the energy cost for a floating-point calculation is far outweighed by the energy needed to move the operand data.

In contemporary processors, the energy consumed in one floating point operation is around 50 pJ while memory movement and other overhead associated with the operation can consume 1000-10,000 pJ if the operands come from outside the processor's register space. In future computing roadmaps, this disparity only grows.

Current GPU systems have the GPU interconnected to the host CPU through a Peripheral Component Interconnect (PCI) express bus with up to 8 GB/s host-to-GPU and 8 GB/s GPU-to-host simultaneous data transfer rate. Using the specifications for the NVIDIA Fermi M2050 with peak double-precision floating point performance of 515 GFLOP/s:

- 515 floating point operations needed per operand transferred between GPU and host to achieve peak FLOPS

(515 GFLOP/s ÷ 8 GB/s × 8 bytes/double)

- 28 floating point operations are needed per operand transferred between GPU RAM and core to achieve peak flops

  (Global Memory bandwidth: 148 GB/s to cores)

It is clear that for a code to effectively utilize the GPU numerical capabilities, it must involve a high degree of calculation compared to the amount of data transferred.

4.1 Next-Generation Simulation Code

WARPM is based on the operational principles of its predecessor WARPX [44] developed in the same research group and shares a significant amount of infrastructure source code. The functional characteristics and features maintained include:

- Key properties of the simulation are chosen at run-time by the user

  Physical model

  Numerical model

  Boundary Conditions

- Designed for simulation on HPC distributed memory systems

- Provides common infrastructure for hyperbolic problems

  Loading initial conditions

  Saving snapshots of simulation conditions at specified times

  Variable time stepping based on CFL condition

  Logging capabilities

WARPM builds on these capabilities. It was developed as a pairing between C++ host code to orchestrate the simulation and OpenCL source for calculations implementing the numerical method and physical equations on GPUs or many-core CPUs. Significant new technologies implemented in the code include multi-level domain decomposition, dynamic OpenCL code assembly, and task parallelism.

4.2 Multi-Level Domain Decomposition

The problem domain is divided among nodes in the cluster and then again further subdivided on each node into smaller patches. This provides several benefits. Exterior patches on a given node are sequenced first and generally processed by the CPU so that ghost cell communication over MPI between nodes occurs while interior patch computation continues. This hides the MPI communication bottleneck. The GPU, if available, is assigned a fixed region of the interior of the node's region of responsibility. This region is divided into two patches – one is smaller and surrounds the exterior interface with the CPU patches the other is much larger and purely internal. This scheme is illustrated in Figure 4.1.

Another scheme was first tested in which the work domain on a given node was subdivided into many patches. The patch was served to available compute devices (e.g. installed GPU and CPU) as they are available for more work giving a form of dynamic load balancing on the node and heterogeneous computation. Actually, on GPU devices, three patches were processed in a pipeline to utilize simultaneous compute and bi-directional PCI bus transfer capabilities. This scheme required 100% of patch data used by the GPU to be transferred round trip on the PCI bus each step. The impact was much more data movement than the current scheme and performance was only about 10% of the current patch-resident scheme with static compute assignments on the same hardware.

4.3 Dynamic OpenCL Code Assembly

WARPM supports a variety of physical models, numerical methods and simulation domain geometries in one compiled application through use of C++ object-oriented mechanisms,

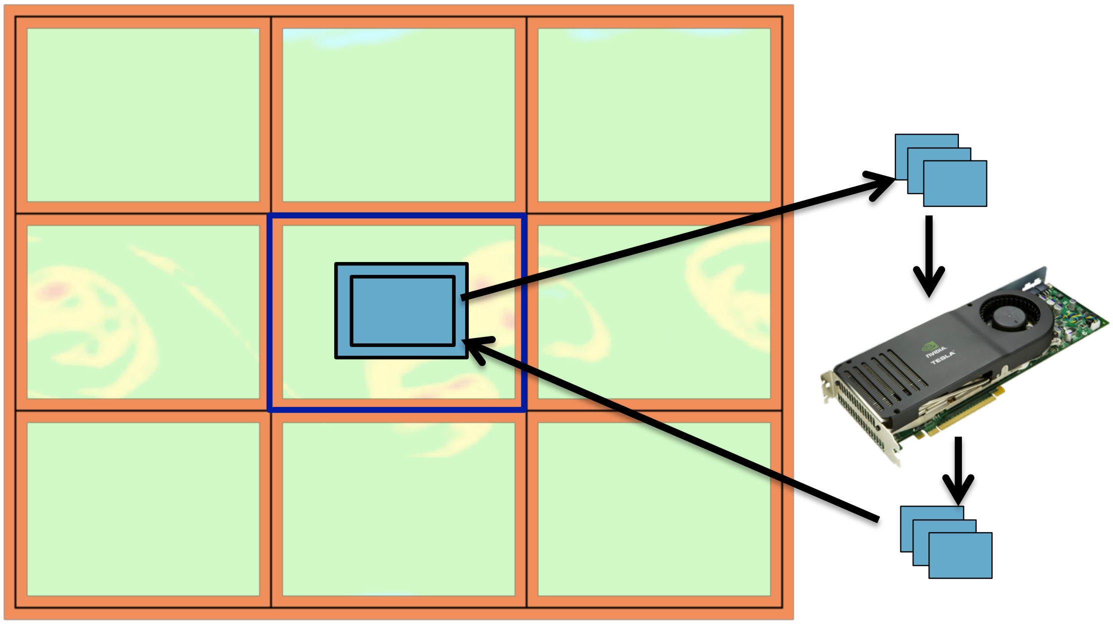

Figure 4.1: The WARPM code decomposes the domain among available nodes and further subdivides the domain on a node into patches suitable for GPU computation. Current hardware supports simultaneous calculation and to/from memory transfer of other patches.

namely polymorphism. However, the vast majority of numerical evaluations in WARPM are implemented with the OpenCL language which is based in the C99 language and does not include any object-oriented features. Even if OpenCL supported the same polymorphic model, which it does not, the approach is not suited for high performance. The WARPM implementation takes advantage of natural support for runtime compilation of OpenCL source. The OpenCL kernel source code is dynamically assembled from segments of source code contributed by a paired C++ class during the initialization phase of simulation execution. This all takes place in the first few seconds of execution (including the run-time compilation of the fully assembled OpenCL kernel). The result is a compiled OpenCL kernel targeted for the actual hardware that implements the precise physics and numerical method selected by the user in one compilation unit. The OpenCL compiler ideally sees a completely predictable kernel execution sequence.

There is another important benefit of this consolidated kernel approach over another technique used in the alternative CUDA language. One could sequence multiple smaller kernels as steps selected at run-time to match the numerical method and physical model. Between kernel execution, register space, cache, and shared memory on the device is cleared. This means data passed from kernel to kernel must be via GPU global memory.

The single consolidated kernel minimizes memory movement between the computational cores and device global memory as well as reduces the storage requirements for intermediary results. Rather than computing and storing the intermediary results of $x \rightarrow y = f(x) \rightarrow w = g(y)$, WARPM computes $w = g(f(x))$ with better performance and smaller global memory footprint. The tradeoff for this is that the compiled kernels can have a large local and private memory footprint on the compute hardware and more complex control flow. If the kernel is too big, a core may not have enough register space or local memory available to launch the kernel. This may cause ”register spilling” of fast private memory to much slower global memory, or perhaps worse, the kernel will not run at all.

The combination of C++ host code and dynamically assembled OpenCL code increases the complexity of WARPM and adds to the learning curve for developers who would implement new models and numerical methods for WARPM. For users of existing capability, the added complexity is largely hidden. The simulation configuration input file make-up is almost the same as for the preceding code WARPX. A new and separate compute configuration input file has been created with WARPM to allow the user to parameterize how the simulation kernels will be mapped to the available compute hardware. Currently, performance can be significantly improved by using this compute configuration file versus the default behavior.

Every WARPM run will create one or more dynamically assembled OpenCL programs reflecting the model and numerical method. The OpenCL program has a common structure as depicted in Figure 4.2. Each section of the program is described as follows:

Device Specific and User Specified Macro Definitions The compiled program is tailored to the compute hardware by three macro parameters adjusting the number of work items per workgroup, the number of work items that collectively process an element in parallel, and the number of serial element sets per workgroup. Additionally, the developer can define custom macros through the compute configuration file useful in troubleshooting.

Code Module Macro Definitions Every code module can declare and define macros. This is more useful in code modules that should only have one instance in an OpenCL program such as the domain geometry.

Common Utility Function Declarations Every program has access to several useful functions for interacting with the WARPM patching system and variable buffers. These functions are declared to allow use by all code modules and the kernel itself.

`#include` Code Module Source Files Each code module is developed with both a C++ `.cc` and `.h` file for the host code and a single OpenCL `.cl` source file to encapsulate the module's source code in one easy to read and modify text file. A series of `#include` statements here include module's source code as part of the declaration pass. Each code module source file is divided into a declaration section and a definition section only active if the macro `DEFINITION_PASS` is defined.

`constant` Instance Variables Initialized For code modules that support multiple instances in the same OpenCL program, an instance is characterized by an instance structure defined by the code module. All instances access the corresponding instance structure in an array of such structures in `constant` memory by the instance index. An example use of this scheme is how multiple instances of the Vlasov equation is supported in one hyperbolic equation set for multiple species. Each instance of Vlasov equation is characterized by different species mass and charge. The Vlasov equation instance structure includes a particle mass and particle charge value.

`#define DEFINITION_PASS` At this point, a universal macro is defined that disables subsequent declaration sections of code modules and enables the definition sections.

Kernel: Function Name The kernel function name is specified to match the name of the compute sequenced group in the user's input file for clarity.

Kernel: Arguments List Code modules declare variables expected as input and output arguments to the kernel. These, as well as a few common arguments such as time and step size, are combined to form the kernel function arguments list. For better clarity, the variable buffer argument name matches the simulation variable's name.

Kernel: Common Work Item Setup The first kernel code translates the scalar global work item id and global offset to a base element id that the workgroup will process.

Kernel: `local` Memory Declarations Local memory used in any OpenCL functions must be declared inside the kernel function body or supplied as a kernel argument. Code modules can declare local memory needs which are then declared in the kernel body here. A pointer to each local memory variable has to be passed to the code module function that is going to use it.

Kernel: Subsolver Step Calls A kernel is the serial sequence to process one compute sequenced group declared in the simulation input file. Each subsolver in the compute sequenced group will relate to a Code Module instance and will declare one or more function names and argument lists known as the sub solver step call. This series of function calls leads to the vast majority of processing done by the kernel.

Kernel: Reduced Output Processing Some kernels produce special output that involves a reduce operation over all of the elements processed by the workgroup. The reduce function is specified in this section. Existing examples are the suggested time step minimum reduce and the moment integration of current density from phase space probability density function in the kinetic systems.

Common Utility Function Definitions Definitions of the already declared utility functions.

`#include` Code Module Source Files Again, the code module files are included but with `DEFINITION_PASS` defined, the function definitions are active.

4.4 Minimized Data Movement

The plasma models developed in this thesis use high-order explicit numerical methods involving only predictable nearest-neighbor communication and maximally utilize floating point operations per operand in order to minimize data movement at every scale.

Already discussed are the benefits of a single consolidated compute kernel reducing the amount of memory transfer from device global memory to the compute core.

Several different memory hierarchies are exposed in the WARPM model:

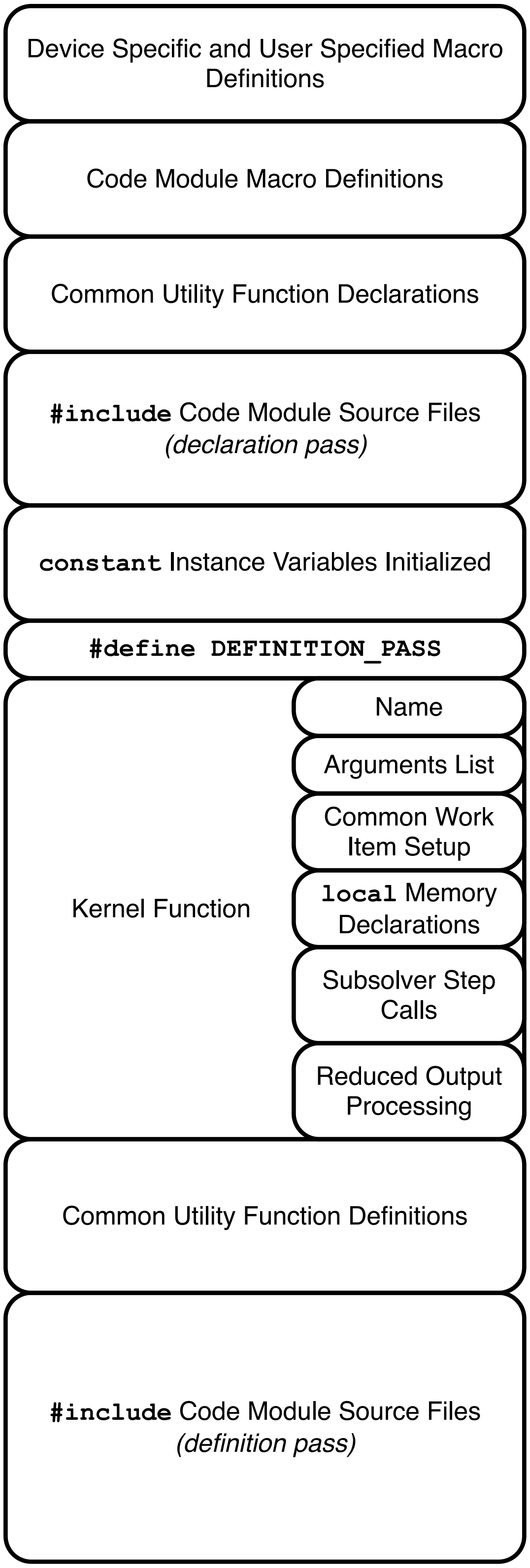
Device Specific and User Specified Macro Definitions
Code Module Macro Definitions
Common Utility Function Declarations
#include Code Module Source Files (declaration pass)
constant Instance Variables Initialized
#define DEFINITION_PASS
Kernel Function
Name
Arguments List
Common Work Item Setup
local Memory Declarations
Subsolver Step Calls
Reduced Output Processing
Common Utility Function Definitions
#include Code Module Source Files (definition pass)

Figure 4.2: Organization of a dynamically assembled OpenCL program created by WARPM.

- Distributed memory is shared and reduced by MPI.

- Host DRAM is managed by C++ new / malloc allocations.

- Device OpenCL global memory corresponds to DRAM on the GPU / Accelerator. This is allocated and copied through the OpenCL API calls made by the WARPM C++ code. Device constant memory acts in the same way, but can only be read by the kernel.

- Device OpenCL local memory is accessible by all work items in a workgroup. This corresponds on a GPU to around 32kB of CUDA shared memory on a Streaming Multiprocessor. local memory is allocated when the kernel is executed, based on either compile-time fixed array sizes in the OpenCL program and/or kernel arguments which are local memory arrays.

- Device OpenCL private memory is only accessible by one work item. This corresponds to registers on a GPU core and may be just a few kB on a CUDA Streaming Multiprocessor.

- In our programming model, distributed memory always has to be transferred to/from host DRAM.

Pairing an explicit numerical scheme with multi-level domain decomposition means that only ghost-cell values need to be transferred between neighboring regions at each level of domain decomposition. At the highest level, ghost-cell values are transferred between nodes using MPI communication over the interconnect. On a given node, ghost-cell values are loaded to device memory with every patch.

The arrangement of variable data in GPU device memory is per patch assignment which is more conducive to fast transfer of contiguous regions of memory. In current AMD and NVIDIA GPU hardware, a special direct memory access (DMA) controller is able to transfer data between the host and GPU at near maximum PCI bus throughput and without slowdown of GPU computation if certain conditions are met [3]. WARPM is designed to satisfy these conditions by using complementary host-side buffers that are ‘pinned’ by the operating system kernel paging system and by transferring data in large contiguous chunks.

The WARPM design aspects to minimize the impact of data movement between host and GPU are highlighted in Figure 4.3 which depicts the compute assignments for CPU and GPU on one node for a hypothetical multiblock structured mesh simulation. One process per compute node is ideal. The mesh region assigned to the MPI process is subdivided into separate patches. A patch represents one assignable work range for an OpenCL kernel. The GPU work is divided into two patches: one with smaller volume adjacent to the CPU patches and one with most of the volume which can be processed without any data transfer or synchronization with the host. The GPU device global memory buffer arrangement for a variable is optimized for fast contiguous transfer from host to device of dependent ghost elements and from device to host of periphery elements. The remaining interior region follows the same row-major array arrangement as host memory.

4.5 Support for different numerical methods, physical models, and domain geometries

While this thesis almost exclusively focuses on the simulation of the Vlasov-Maxwell kinetic model with the discontinuous Galerkin numerical method. It must be pointed out that WARPM was developed to support a flexible combination of numerical methods, physical models, and domain geometries. Each of these can be selected by the user at run-time based on a simulation configuration file largely similar to the predecessor code WARPX.[44]

As an illustrative example of capability not generally covered in this thesis, airflow over a wing cross section is simulated. The finite volume scheme and Euler fluid model in two-dimensions are combined along with a structured quadrilateral mesh discretization of the simulation domain. All of this is specified by the user in the simulation configuration file as well as which boundary conditions to apply.

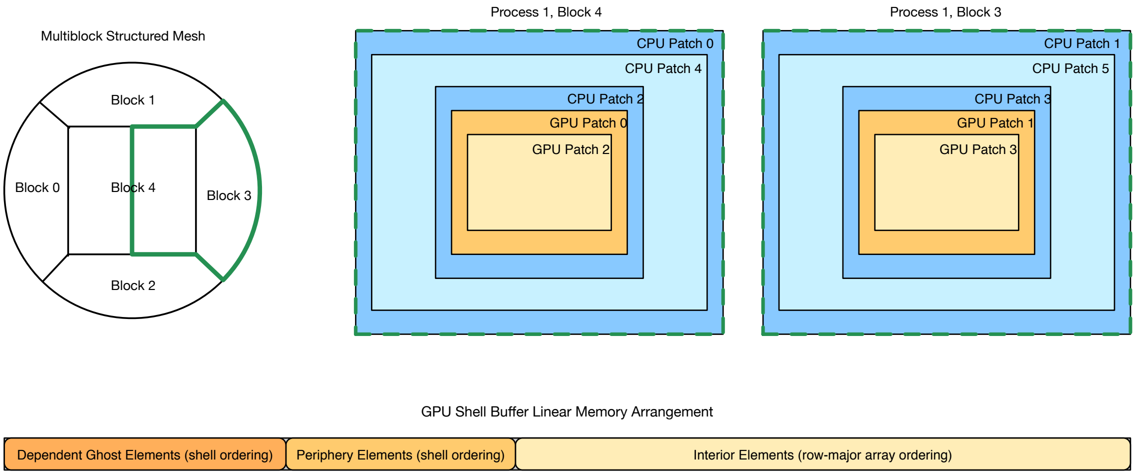
Multiblock Structured Mesh
Block 0
Block 1
Block 2
Block 3
Block 4
Process 1, Block 4
CPU Patch 0
CPU Patch 4
CPU Patch 2
GPU Patch 0
GPU Patch 2
Process 1, Block 3
CPU Patch 1
CPU Patch 5
CPU Patch 3
GPU Patch 1
GPU Patch 3
GPU Shell Buffer Linear Memory Arrangement
Dependent Ghost Elements (shell ordering)
Periphery Elements (shell ordering)
Interior Elements (row-major array ordering)

Figure 4.3: Compute assignments (patches) for one MPI process/node participating in a block-structured simulation. The node's compute hardware consists of a CPU and GPU. Each compute device is responsible for advancing the simulation of a fixed subdomain of the node's overall assignment (green). The patches assigned to the GPU are internal and the buffer arrangement on the GPU facilitates large-block contiguous transfers over the PCI bus.

At runtime, the Euler fluid model, finite volume scheme, and quadrilateral mesh code modules each provide OpenCL source code segments that are assembled into a single compiled kernel at runtime. The compiled program for simulation has run successfully on CPU, GPU, and both.

Figures 4.4 and 4.5 illustrate the structured quadrilateral domain capabilities and results for an exemplary simulation in WARPM. Starting from an initially uniform flow field, this steady state solution develops for Mach 0.9.

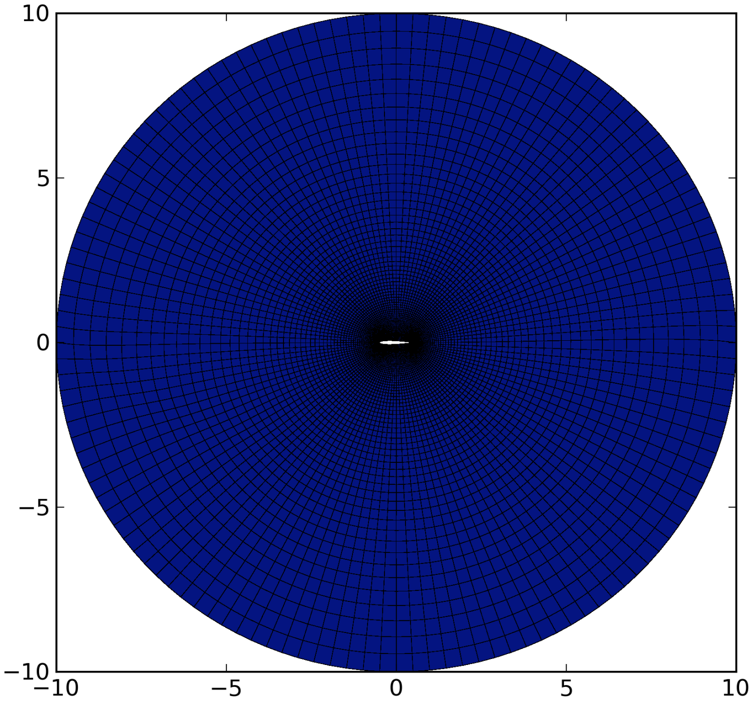

Figure 4.4: NACA 0012 airfoil simulation domain set up as a structured mesh of quadrilateral elements. This is an 'O' mesh, that wraps around the airfoil with periodic boundary seam at the trailing edge. The simulation domain is much larger than the airfoil (center) in order to minimize boundary effects and is decomposed into quadrilateral elements $(n_r, n_\theta) = (60, 128)$.

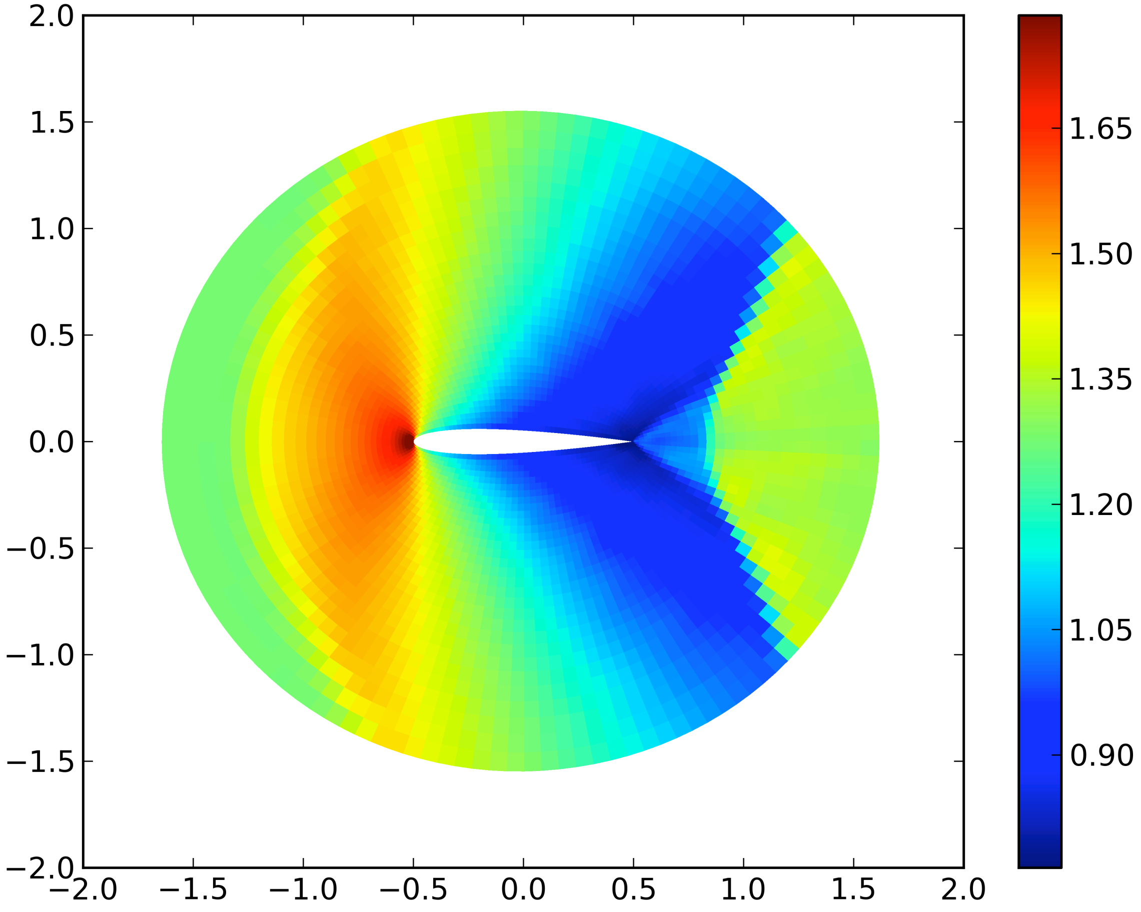

Figure 4.5: Steady-state flow conditions around a NACA 0012 airfoil with zero angle-of-attack. Gas density is plotted.

Chapter 5
KINETIC IMPLEMENTATION DETAILS

5.1 *Considerations for discretized velocity dimension*

5.1.1 Truncated velocity extent

Solving the kinetic Vlasov equation directly on a realizable discretized phase space grid requires truncating velocity space to some restricted extent, $\pm V_{\max}$. In the most common and straightforward representation, the modeled probability distribution function is assumed to have compact support within the restricted extent, such that $f(v) = 0|_{x \notin [-V_{\max}, V_{\max}]}$. Particle-in-cell methods do not involve such a truncation requirement as each macro-particle can represent an arbitrary velocity vector without any bound.

Maxwellian functions are desirable probability distribution functions to resolve and simulate despite their lack of compact support due to their representation of species in local thermodynamic equilibrium. In light of this, it is worth looking at the impact of velocity space truncation on the three moments of the Maxwellian distribution. With a better understanding of the impacts of restricted velocity extent, the velocity domain can be scaled appropriately to well capture the physics of thermalized systems.

The Maxwellian distribution in full three velocity dimensions can be expressed as a product of one-dimensional Maxwellian distributions.

$$f(\mathbf{x}, \mathbf{v}, t) = n(\mathbf{x}, t) f_{v_x}(v_x) f_{v_y}(v_y) f_{v_z}(v_z), \tag{5.1}$$

where,

$$f_{v_i}(v_i) = \sqrt{\frac{m}{2\pi k T_i}} \exp\left(\frac{-m(v_i - \bar{u}_i)^2}{2kT_i} \right). \tag{5.2}$$

The total distribution can have different fluid drift velocity $\bar{u}_i$ and temperature T_i in each dimension. Alternative expression can use the so-called thermal velocity, $v_{\mathrm{T}} = \sqrt{\frac{kT}{m}}$, which in a statistical context is also the standard deviation for the distribution.

Analysis of the impact of a truncated velocity space on the distribution moments can be simplified to just one velocity dimension and no spatial variation without loss of utility. Additionally, it is easiest to assume zero drift velocity but this assumption must be remembered when modeling drifting species especially if the drift velocity is comparable or faster than the thermal velocity. The drift velocity directly shifts the Maxwellian centroid away from zero and closer to the velocity domain extent.

As a consideration for the velocity mesh resolution of the Maxwellian, note that the full-width at half-maximum of a Maxwellian distribution scales linearly with the thermal velocity or square root of temperature, FWHM $= 2\sqrt{2\ln 2}v_{\mathrm{T}} \approx 2.355 v_{\mathrm{T}}$.

Number density

The number density, or number of particles per unit of spatial volume, is evaluated as the 0^{th}-moment of the distribution. For the Maxwellian,

$$n = \int_{-\infty}^{\infty} f(v)dv = \int_{-\infty}^{\infty} \frac{n_0}{v_{\mathrm{T}}\sqrt{2\pi}} \exp\left(\frac{-v^2}{2v_{\mathrm{T}}^2}\right) dv = n_0. \tag{5.3}$$

The truncated Maxwellian captures a smaller part of the number density,

$$\tilde{n} = \int_{-V_{\max}}^{V_{\max}} f(v)dv = n_0 \operatorname{erf}\left(\frac{V_{\max}}{v_{\mathrm{T}}\sqrt{2}}\right). \tag{5.4}$$

To capture a specified fraction of the Maxwellian number density, $\gamma_n = \frac{\tilde{n}}{n}$, then the velocity domain extent is set as follows,

$$\frac{V_{\max}}{v_{\mathrm{T}}} = \sqrt{2}\operatorname{erf}^{-1}(\gamma_n). \tag{5.5}$$

Notice that $V_{\max}$ scales linearly with v_{T}.

Momentum density

For assessing the truncation impact on the first-moment, or momentum density, consider only the positive-directed momentum since the net momentum is zero.

$$p^{+} = \int_{0}^{\infty} mvf(v)dv = \int_{0}^{\infty} \frac{mvn_0}{v_{\mathrm{T}}\sqrt{2\pi}} \exp\left(\frac{-v^2}{2v_{\mathrm{T}}^2}\right) dv = \frac{n_0 m v_{\mathrm{T}}}{\sqrt{2\pi}}. \tag{5.6}$$

The truncated Maxwellian captures a smaller part of the momentum density,

$$\gamma_p = \frac{\tilde{p}^{+}}{p^{+}} = \frac{\int_0^{V_{\max}} mvf(v)dv}{p^{+}} = 1 - \exp\left(\frac{-V_{\max}^2}{2v_{\mathrm{T}}^2}\right). \tag{5.7}$$

To capture a specified fraction of the Maxwellian momentum density, γ_p, then the velocity domain extent is set as follows,

$$\frac{V_{\max}}{v_{\mathrm{T}}} = \sqrt{-2\ln(1-\gamma_p)} \tag{5.8}$$

Thermal energy density

For assessing the truncation impact on the second-moment, or thermal energy density,

$$e = \int_{-\infty}^{\infty} \frac{1}{2} mv^2 f(v)dv = \int_{-\infty}^{\infty} \frac{\frac{1}{2}mv^2 n_0}{v_{\mathrm{T}}\sqrt{2\pi}} \exp\left(\frac{-v^2}{2v_{\mathrm{T}}^2}\right) dv = \frac{1}{2} m v_{\mathrm{T}}^2 n_0. \tag{5.9}$$

The truncated Maxwellian captures a smaller part of the thermal energy density,

$$\gamma_e = \frac{\tilde{e}}{e} = \frac{\int_{-V_{\max}}^{V_{\max}} \frac{1}{2}mv^2 f(v)dv}{e} = \operatorname{erf}\left(\frac{1}{\sqrt{2}}\frac{V_{\max}}{v_{\mathrm{T}}}\right) - \frac{V_{\max}}{v_{\mathrm{T}}}\sqrt{\frac{2}{\pi}} \exp\left(-\frac{1}{2}\frac{V_{\max}^2}{v_{\mathrm{T}}^2}\right). \tag{5.10}$$

A close form solution for $\frac{V_{\max}}{v_{\mathrm{T}}}$ as a function of γ_e is not provided. Instead, notice the asymptotic approach to just the error function term; the term makes up 99.9% of γ_e by $\frac{V_{\max}}{v_{\mathrm{T}}} = 4$.

In summary, for all three moments, the error effect of the truncated velocity space can be expressed as functions of the ratio $\frac{V_{\max}}{v_{\mathrm{T}}}$. Additionally, the ratio equal to 5 gives a adequate ballpark range that encloses about 99.999% of mass, momentum, and thermal energy relative to the Maxwellian. Each fraction is plotted in Figure 5.1.

5.2 Shared rectilinear velocity space mesh among species with tailored stretching and offset velocity

Given the reality that particle different species have quite different different mass, temperature, thermal velocity, and mean fluid velocity, it is regularly appropriate to model the

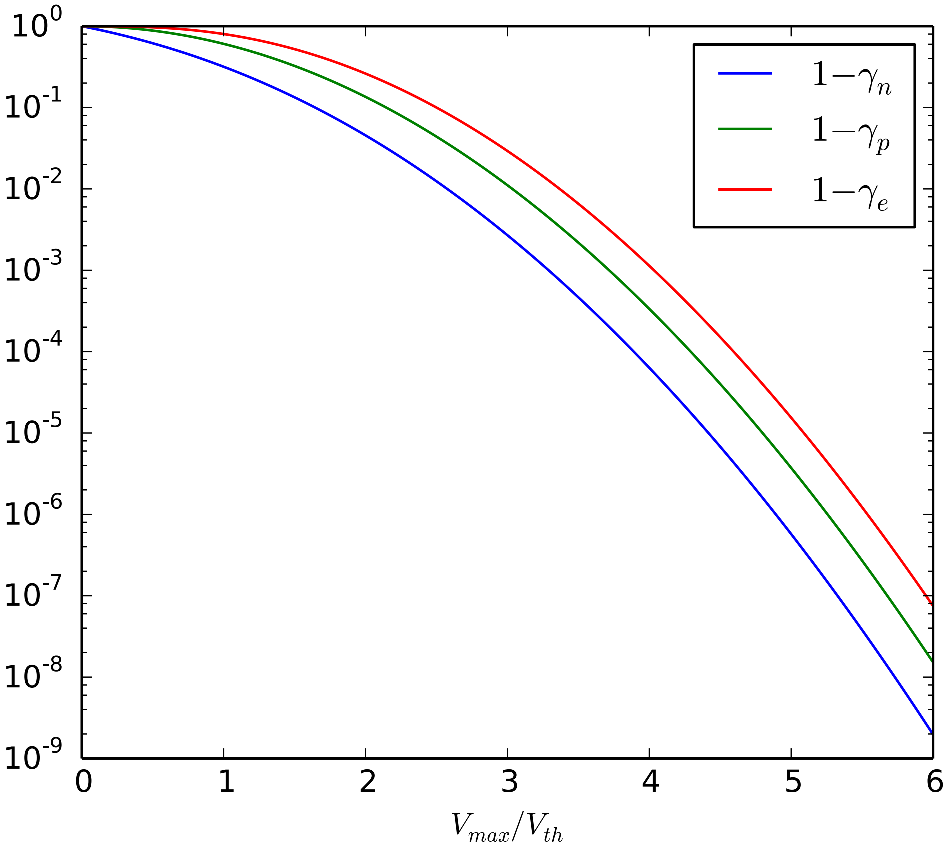

Figure 5.1: Relationship between truncated velocity space and the three moments of a modeled species with Maxwellian distribution.

pdf of separate species on a different truncated velocity domain. In terms of computational implementation, it is beneficial to compute the numerical solution of both species on the same mesh. This includes moment integral calculations.

To satisfy both, a customized scaling is introduced for each specie's velocity coordinate based upon a common shared velocity-like mesh. The common mesh is chosen to always have extents $w \in [-1, 1]$ for each velocity dimension.

Each species' Vlasov equation is rescaled linearly based on an additional selected scaling property that becomes associated with the species just as its mass or charge. Then the actual implemented equation becomes Eq. (5.14).

$$\frac{\partial f_{\mathrm{s}}}{\partial t} + \nabla_x \cdot [\mathbf{v} f_{\mathrm{s}}] + \nabla_v \cdot [\frac{q_{\mathrm{s}}}{m_{\mathrm{s}}}(\mathbf{E} + \mathbf{v} \times \mathbf{B}) f_{\mathrm{s}}] = 0. \tag{5.11}$$

$$\mathbf{v} = \gamma_v \mathbf{w} \tag{5.12}$$

$$\nabla_v = \frac{1}{\gamma_v} \nabla_w \tag{5.13}$$

$$\frac{\partial f_{\mathrm{s}}}{\partial t} + \nabla_x \cdot [\gamma_v \mathbf{w} f_{\mathrm{s}}] + \nabla_w \cdot [(\frac{q}{\gamma_v m}\mathbf{E} + \frac{q_{\mathrm{s}}}{m_{\mathrm{s}}}\mathbf{w} \times \mathbf{B}) f_{\mathrm{s}}] = 0. \tag{5.14}$$

Typically, we will want to model two species with different mass but similar temperature. In order to resolve both distributions similarly, i.e. same number of velocity elements over the Maxwellian half-width, then $\gamma_v = \frac{1}{\sqrt{m_{\mathrm{s}}}}$.

5.3 Domain decomposition that does not distribute velocity space

All of $3V$ velocity space is held by a compute unit, including being on a GPU. Large contiguous transfers of ghost cell data for position space ghost elements only. Moment integration operates on locally held data only.

5.4 Solid Wall Boundary Condition for the Vlasov Equation

There is interest in supporting an idealized solid wall boundary condition imposed on the Vlasov equation model for plasma. An idealized solid wall should perfectly reflect incident particles with opposite momentum to the wall and not affect momentum tangential to the wall. The main motivation to support the idealized solid wall is that several benchmark problems for plasma simulation specify it. To compare to the benchmark results, the boundary condition should be implemented. There is not so much justification to use this boundary condition for structurally confined plasmas with wall temperatures much colder than the plasma temperature.

When using an upwind numerical flux, an idealized solid wall can be simply implemented by imposing exterior face node values on the probability distribution function, f, such that $f^{+}(\mathbf{x}, \mathbf{v}) = f^{-}(\mathbf{x}, -(\hat{n} \cdot \mathbf{v})\hat{v})$, where the plus-sign superscript indicates the exterior element face value and the negative-sign superscript indicates the interior element face value. This will have the affect that probability density advecting into the wall does so freely, but then emerges from the wall with opposite normal momentum and the same tangential momentum. Implementation of the boundary condition for one physical dimension is depicted in Figure 5.2.

The one-dimensional implementation is straight-forward, but does not generalize to two or three physical dimensions. The approach will still work if the idealized wall boundary is perpendicular to a physical coordinate (i.e. $\hat{x}$, $\hat{y}$, or $\hat{z}$). With velocity space discretized as a rectilinear mesh perpendicular to the physical coordinates, the same reflection of normal-momentum can be done as in one-dimension. If the wall is oblique to any physical coordinate, then the reflection approach cannot be implemented correctly. Parts of the incident normal momentum space do not reflect back into the modeled momentum space. The difficulty is depicted in Figure 5.3. The region reflected out of the domain represents loss of mass, momentum, and energy. In addition to this problem, when using high-order nodal elements, the reflection procedure would require sophisticated interpolation of the reconstructed poly-

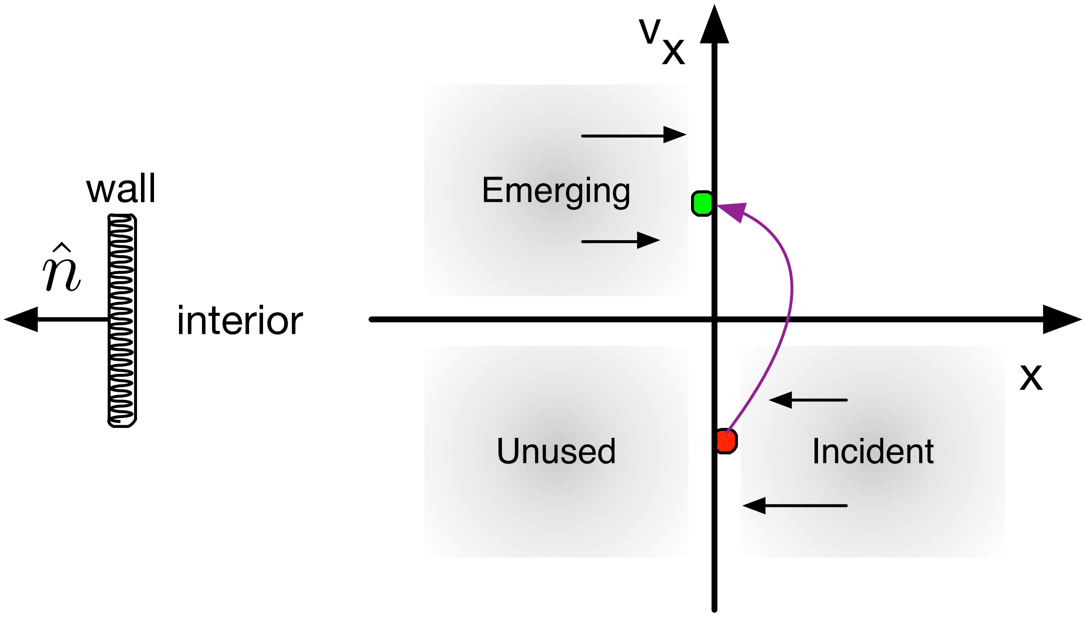

Figure 5.2: Depiction of the solid wall boundary condition for the phase space pdf with one physical dimension. The normal component of incident momentum is reflected.

nomial solution in the incident region onto the nodal locations in the reflected region – the node mapping is not one-to-one.

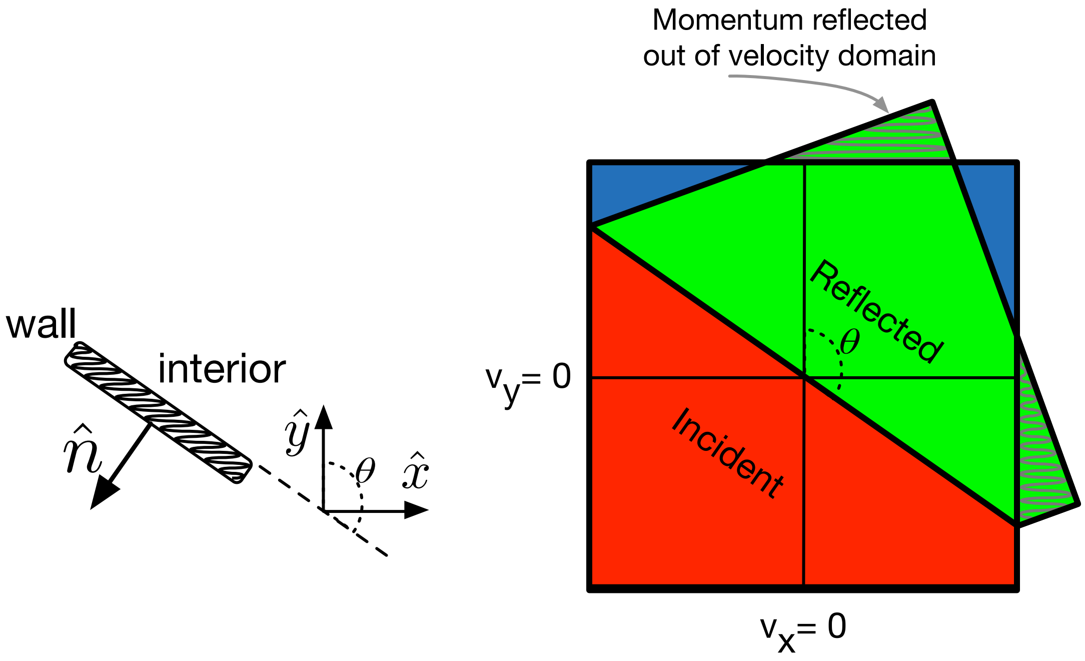

Figure 5.3: Solid walls oblique to the physical coordinate cannot be implemented with the same simple reflection procedure. Some region of incident normal momentum would be reflected outside the modeled velocity space (hashed green area). This results in loss of conservation of mass, momentum and energy. Additionally, some emitted momentum region also has no corresponding incident momentum (blue), but this can be simply handled by assuming the pdf is zero there.

One alternative boundary condition that maintains similar properties as the idealized solid wall would be to have the wall thermalize the incident pdf The emitted pdf would be Maxwellian in the face-normal direction with matching number density, opposite momentum, and matching kinetic energy as the incident distribution. This technique would be conservative, but destroys the velocity space structure of the incident pdf

The flux of number density normal to the face, $\hat{F}$, must balance between the incident and emitted integrals,

$$\hat{F}_{\text{incident}} = \int_{\Omega_{\text{incident}}} (\mathbf{v} \cdot \hat{n}) f^{-} d\mathbf{v} = \hat{F}_{\text{emitted}} = \int_{\Omega_{\text{emitted}}} (\mathbf{v} \cdot \hat{n}) f^{+} d\mathbf{v}. \tag{5.15}$$

The flux of momentum density in the face normal direction normal to the face, $\hat{P}$, must be opposite between the incident one emitted integrals for elastic collisions with the wall,

$$\hat{P}_{\text{incident}} = \int_{\Omega_{\text{incident}}} (\mathbf{v} \cdot \hat{n})(\mathbf{v} \cdot \hat{n}) f^{-} d\mathbf{v} = -\hat{P}_{\text{emitted}} = \int_{\Omega_{\text{emitted}}} (\mathbf{v} \cdot \hat{n})(\mathbf{v} \cdot \hat{n}) f^{+} d\mathbf{v}. \tag{5.16}$$

The flux of total kinetic energy density normal to the face, $\hat{E}$, must balance between the incident and emitted integrals,

$$\hat{E}_{\text{incident}} = \int_{\Omega_{\text{incident}}} \mathbf{v}^2 (\mathbf{v} \cdot \hat{n}) f^{-} d\mathbf{v} = \hat{E}_{\text{emitted}} = \int_{\Omega_{\text{emitted}}} \mathbf{v}^2 (\mathbf{v} \cdot \hat{n}) f^{+} d\mathbf{v}. \tag{5.17}$$

For the above equations, the integration domains are depicted in Figure 5.4. A Maxwellian pdf for f^{+} can be parameterized such that the above constraints are satisfied.

Integration of f^{-} over the incident domain would require development of a new quadrature rule which would be a function of wall angle θ.

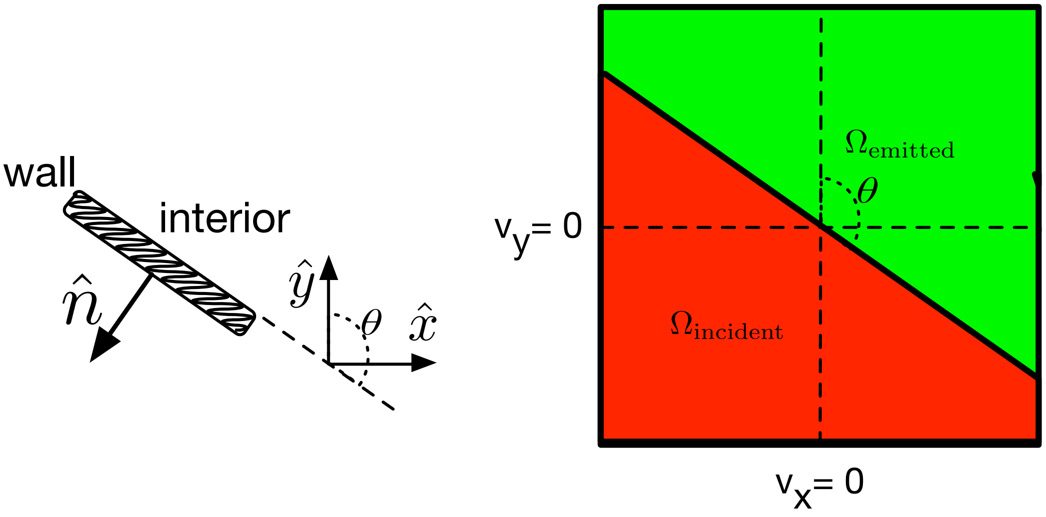

Figure 5.4: Incident and emitted pdf integration regions for an oblique wall.

5.5 Symmetry Plane Boundary Condition

For problems with a physical plane of symmetry analytically maintained, half of the computational effort can be eliminated by implementing a symmetry boundary condition at the plane of symmetry rather than simulating the full domain. The current sheet problems in Chapter 7 and 8 are examples of this case.

Considering a sausage mode or even symmetry along the plane $x = 0$ and the modeled domain is $x \geq 0$, then the associated conditions are as summarized in Table 5.1. The discontinuous Galerkin method makes imposing such a condition on the solution quite easy and precise with an adjustment. Rather than place an element edge at the symmetry plane, center the element on the symmetry plane. Thus these elements centered along the plane should always demonstrate the desired symmetry in there internal solution. The boundary condition can be imposed though appropriate copy and perhaps negative of Dirichlet values from the face nodes of the second interior element to the mirrored face nodes on the ghost element. For each property, $g\left(-\left(\frac{\Delta x}{2}\right), y, z\right) = \pm g\left(\left(\frac{\Delta x}{2}\right), y, z\right)$. The boundary condition for the pdf is slightly more complicated but simply repeats the idea already presented for the solid wall boundary,

$$f\left(-\left(\frac{\Delta x}{2}\right), y, z, v_x, v_y, v_z\right) = \pm f\left(\left(\frac{\Delta x}{2}\right), y, z, -v_x, v_y, v_z\right) \tag{5.18}$$

Table 5.1: Conditions associated with sausage mode or even symmetry along the $x = 0$ plane in the Vlasov-Maxwell model.

n	$\frac{\partial}{\partial x} = 0$
j_x	$j_x\|_{x=0} = 0$
j_y, j_z	$\frac{\partial}{\partial x} = 0$
E_x	$E_x\|_{x=0} = 0$
E_y, E_z	$\frac{\partial}{\partial x} = 0$
B_x	$\frac{\partial}{\partial x} = 0$
$B_y,$	$B_y\|_{x=0} = 0$
B_z	$B_z\|_{x=0} = 0$

5.6 Sequencing of physical-space and phase-space evaluation

Evaluation of physical space hyperbolic system (i.e. Maxwell's equations) is performed as a separate computation step than the evaluation of the phase-space hyperbolic system (i.e. Vlasov's equation). The implementation has the evaluation performed by separate instances of the discontinuous Galerkin scheme. The combination of the Vlasov-Maxwell model is still one coupled system, however. Coupling is through the first-moment (current density).

The current density term leaves a couple options for when to evaluate the first-moment integration over velocity-space. In ether case, it is desirable to implement the moment integration at the same time as Vlasov's equation evaluation. This allows the algorithm to traverse velocity space just once and gives reuse of the distribution function values while held locally in memory.

Two sequencing options are presented below with some tradeoffs identified. The compute stages are represented by the following symbols and three time periods, n, $*$, and $n+1$.

- M - evaluation of Maxwell's equations in physical space.
- V - evaluation of Vlasov's equation in phase space.
- A - integration of the pdf first-moment

In this first sequence option, the initial j^n must be boot-strapped by separate evaluation of the pdf 1st-moment.

$$\begin{aligned} \text{boot strap:} \quad A(f^n) &\rightarrow (j^n) \\ M(E^n, j^n) &\rightarrow (E^*) \\ V(f^n, E^n) &\rightarrow (f^*) \\ A(f^*) &\rightarrow (j^*) \\ M(E^*, j^*) &\rightarrow (E^{n+1}) \\ V(f^*, E^*) &\rightarrow (f^{n+1}) \\ A(f^{n+1}) &\rightarrow (j^{n+1}) \end{aligned}$$

The next sequence option does not require a preliminary boot-strap evaluation of $A()$ but on the other hand may need a special final evaluation if it is desired to save the current density state at end of simulation consistent with the final pdf. The sequencing should be more easily extended to add a collision model provided the model conserves momentum. The steps integrating the previous current moment in function A could also potentially calculate correlation or higher-moment terms supporting the collision operator to apply to f^* in parallel with other evaluations.

$$\begin{aligned} V(f^n, E^n) &\rightarrow (f^*) \\ A(f^n) &\rightarrow (j^n) \\ M(E^n, j^n) &\rightarrow (E^*) \\ V(f^*, E^*) &\rightarrow (f^{n+1}) \\ A(f^*) &\rightarrow (j^*) \\ M(E^*, j^*) &\rightarrow (E^{n+1}) \end{aligned}$$

Chapter 6

PLANAR PLASMA WAVES

This chapter considers preliminary problems and simplified one-dimensional geometries to demonstrate the core technologies and numerical method implementation. Work presented here serves as a stepping stone to later higher-dimension simulations.

6.1 Planar wave propagation in a uniform unmagnetized plasma

A prototype implementation of the discontinuous Galerkin method was developed to validate solution of Vlasov-Poisson model problems for longitudinal waves with known analytical solution.

The Vlasov-Poisson model described in Chapter 2 has been specified for the simplified case of one particle species, one physical space dimension and one velocity space dimension ($1D+1V$). A nodal discontinuous Galerkin method was implemented in Matlab to solve the time evolution of the model system. The implementation provided ability to adjust the finite element basis set to arbitrary polynomial order and independently in physical dimension and velocity dimension. The initial conditions were chosen to reproduce classic test cases of weak Landau damping, strong Landau damping, and the evolution of the two-stream instability with results in Ref. [45], which also provides a more thorough analytical treatment.

6.1.1 Weak Landau damping

Landau damping describes a stabilizing phenomena in plasma where waves in the plasma interact with charged particles with velocity close to the wave phase velocity [46]. For a particle velocity distribution that is approximately Maxwellian, there will be more particles with velocity slightly less than the wave phase velocity than particles with velocity slightly greater. Thus more particles are accelerated by the wave than give up energy to the wave; the wave is damped (exponential decay in time).

A rich set of analytical work has been done exploring this phenomena; the damping factor can be analytically determined. This makes for a nice benchmarking problem for the kinetic model and numerical method.

The problem is set up as a perturbed Maxwellian distribution on a periodic spatial domain $x \in [-2\pi, 2\pi]$. The initial distribution function is,

$$f = \frac{1}{\sqrt{2\pi}} e^{(-v^2/2)} (1 + \alpha \cos(kx)), \tag{6.1}$$

where $\alpha = 0.01$ is the amplitude of initial perturbation and $k = \frac{1}{2}$ is the wave number of the initial perturbation. The velocity domain is truncated at ten times the thermal velocity, $v \in [-v_{\max}, v_{\max}]$ with $v_{\max} = 10v_{\mathrm{T}}$ and zero-flux boundary conditions applied at $\pm v_{\max}$.

Landau damping can be readily confirmed by time evolution of the domain electric field energy,

$$U_E = \frac{\epsilon}{2} \int_{-2\pi}^{2\pi} E^2(x) dx. \tag{6.2}$$

The Landau damping rate for the prescribed conditions is predicted to be -0.3066 in Ref. [45]. The electric field energy time series are plotted for two different velocity dimension order of accuracy in Figures 6.1 and 6.2 along with the analytical damping rate, $y(t) = 7 \times 10^{-4} \times \exp(-0.3066t)$.

The results show good agreement with theory and previously published works which used different numerical methods such as Refs. [45], [21], and [47]. The pair of simulation conditions producing Figures 6.1 and 6.2 are chosen to highlight the long-time limitation of the discontinuous Galerkin numerical method applied to this model. The simulations only differ by the basis functions' polynomial order in the velocity dimension. With Figure 6.2 using lower order polynomials (7^{th} versus 10^{th}), continued adherence to Landau damping breaks down around $t = 70\omega_p^{-1}$. This is because the system solution becomes increasingly striated in the velocity dimension until the basis set and grid resolution combination can no longer resolve the smallest feature size.

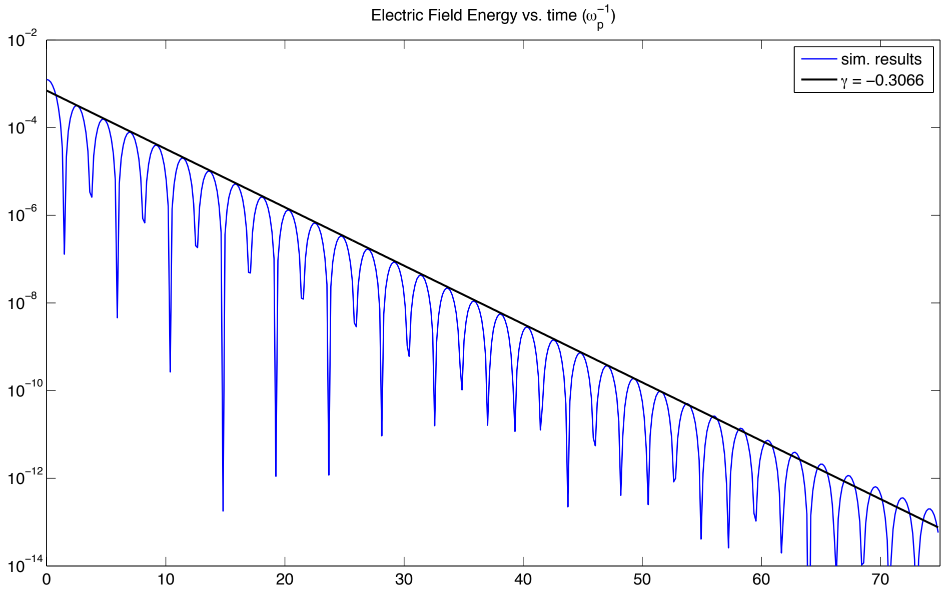

Figure 6.1: Weak Landau damping resulting time series of electric field energy compared to predicted decay rate. Simulation domain is decomposed into rectangular elements $(n_x, n_v) = (20, 80)$. Polynomial basis function order are $(P_x, P_v) = (7, 10)$.

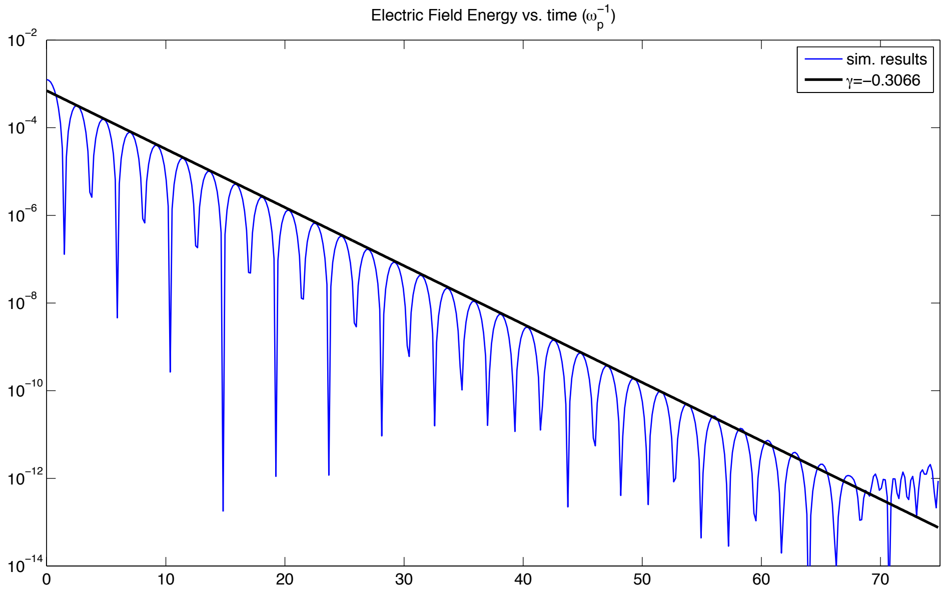

Figure 6.2: Weak Landau damping resulting time series of electric field energy compared to predicted decay rate. Simulation domain is decomposed into rectangular elements $(n_x, n_v) = (20, 80)$. Polynomial basis function order are $(P_x, P_v) = (7, 7)$. Time $70\omega_p^{-1}$ corresponds with feature size in the velocity direction becoming smaller than that resolvable with the selected grid resolution and polynomial order.

In reality, some amount of collisions occur between particles of the same species that work to smooth out the smallest features in the velocity dimension. Thus, such a high resolution of velocity space is not generally required.

6.1.2 Strong Landau damping

Strong Landau damping [46] follows the same setup as the previous section, except that the initial perturbation is larger, $\alpha = 0.5$, such that the transition to non-linear behavior occurs. Onset time of non-linear effects is approximately $1/\sqrt{\alpha}$ [47].

Figures 6.3 and 6.4 present simulation results for strong Landau damping in a manner mimicking that of Ref. [21] in order to facilitate qualitative comparison showing very good agreement. Figure 6.4 also plots the fit linear damping rates in two regions. Initially linear damping occurs ($\gamma_1 = -0.5904$) until particle trapping dominates and linear growth ($\gamma_2 = 0.1688$) begins. These slopes agree with the semi-Lagrangian results published by Rossmanith and Seal [21], and compare well to the results of Cheng and Knorr (-0.562 and 0.168) [45].

6.1.3 Two-stream instability

Cheng and Knorr also consider the dynamics of two opposing warm particle beams [45]. The problem is set up as two opposing perturbed Maxwellian beams on a periodic spatial domain $x \in [0, 2\pi/k]$. The initial distribution function is,

$$f = \frac{1}{\sqrt{2\pi}} e^{(-v^2/2)} (1 + \alpha \cos(kx)), \tag{6.3}$$

where $\alpha = 0.05$ is the amplitude of initial perturbation and k is the wave number of the initial perturbation. The velocity domain is truncated at ten times the thermal velocity, $v \in [-v_{\max}, v_{\max}]$ with $v_{\max} = 10 v_{\mathrm{T}}$ and zero-flux boundary conditions applied at $\pm v_{\max}$.

Different values of k give unstable ($k < 1$) and stable ($k > 1$) behavior for the linear mode. A commonly analyzed unstable case $k = 0.5$ was used in the simulations producing Figures 6.5 and 6.6.

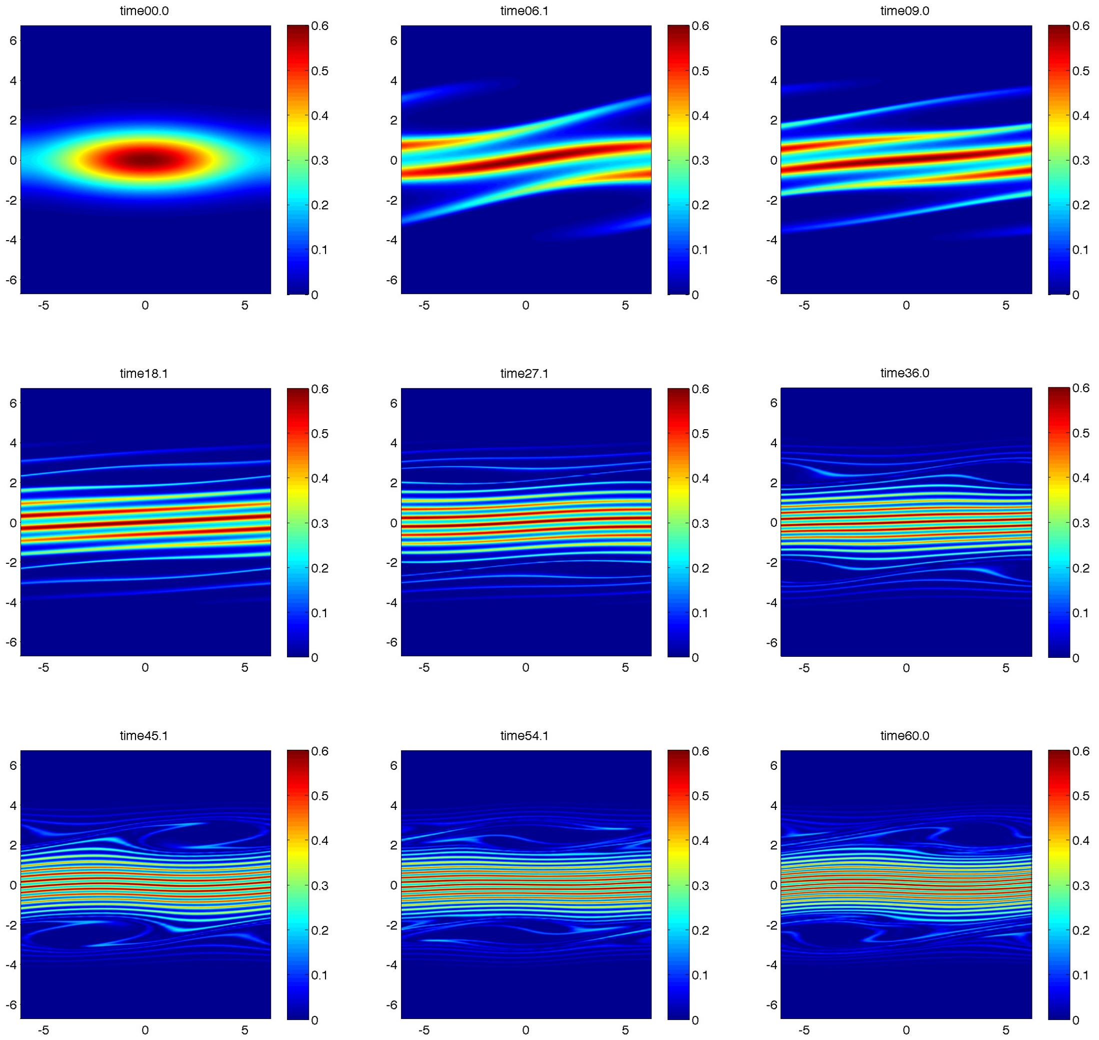

Figure 6.3: Strong Landau damping affect on distribution function in phase space presented as velocity versus position for multiple times as annotated. Simulation domain is decomposed into rectangular elements $(n_x, n_v) = (20, 80)$. Polynomial basis function order are $(P_x, P_v) = (7, 7)$.

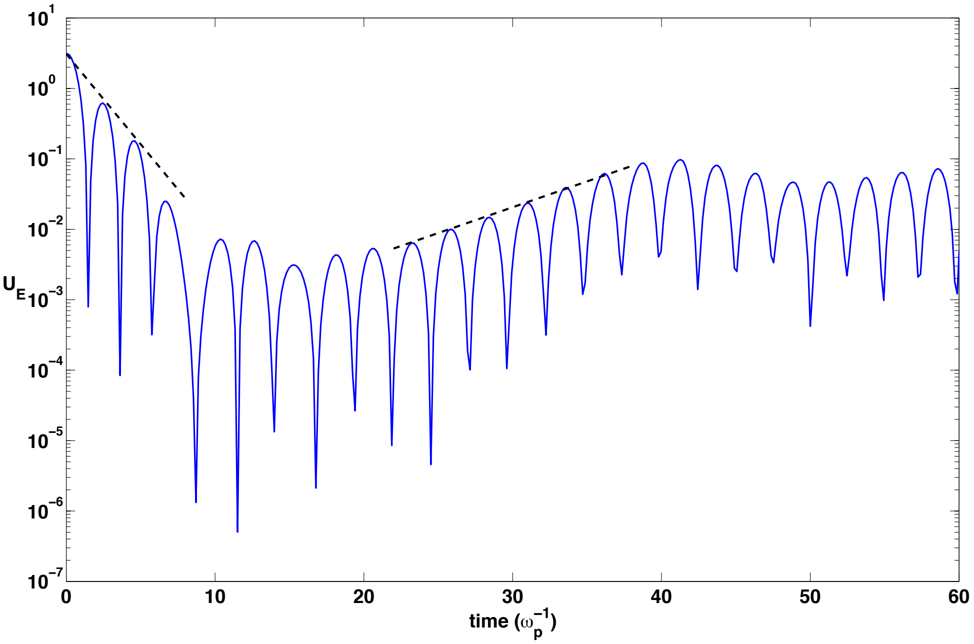

Figure 6.4: Strong Landau damping affect on electric field energy time series. Simulation domain is decomposed into rectangular elements $(n_x, n_v) = (20, 80)$. Polynomial basis function order are $(P_x, P_v) = (7, 7)$. Additionally, two zones of linear growth are identified and plotted along with fit lines $\gamma_1 = -0.5904$ and $\gamma_2 = 0.1688$ (dashed lines).

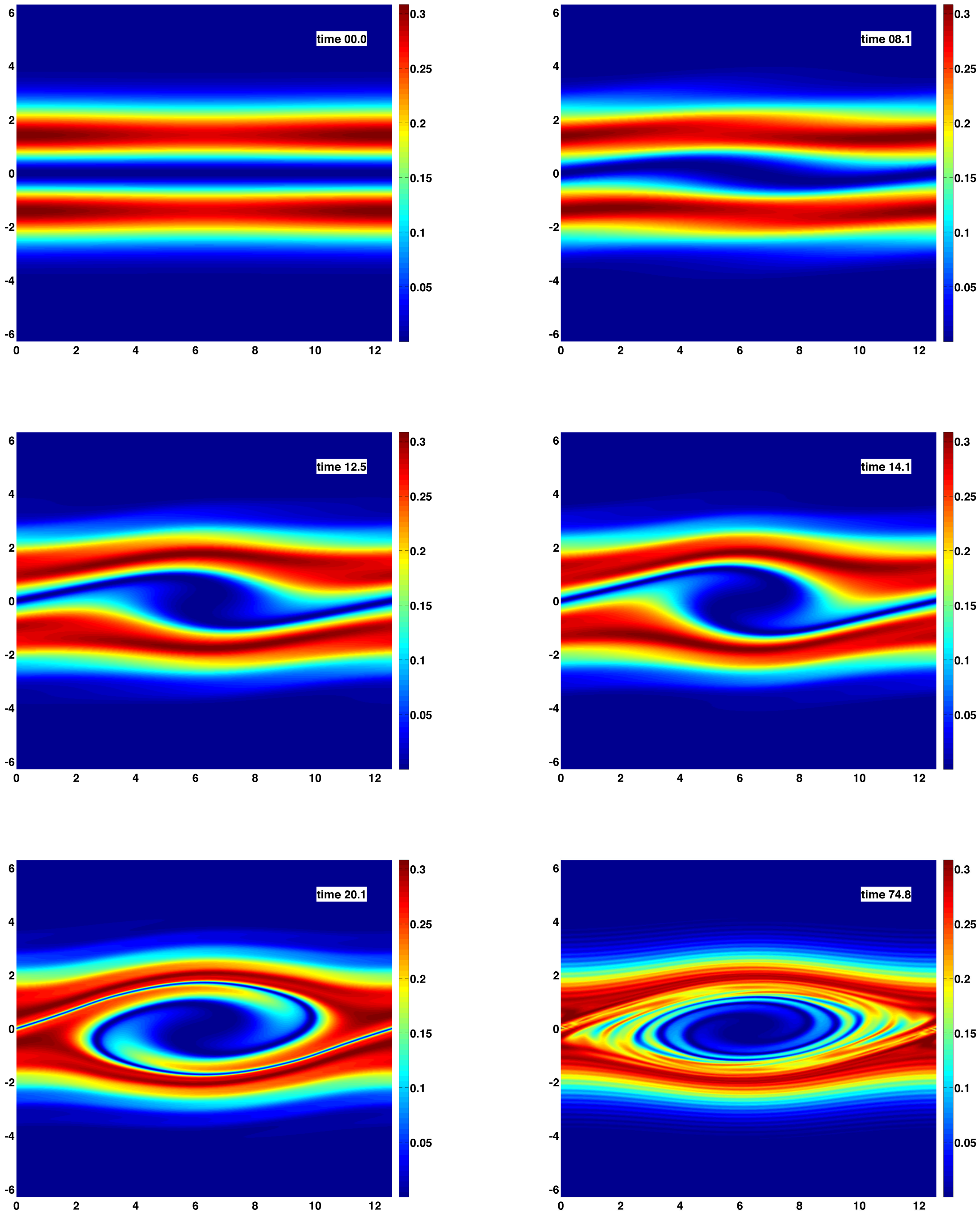

Figure 6.5: Two stream initial conditions with unstable conditions, $k = 0.5$. Simulation domain is decomposed into rectangular elements $(n_x, n_v) = (20, 80)$. Polynomial basis function order are $(P_x, P_v) = (7, 7)$.

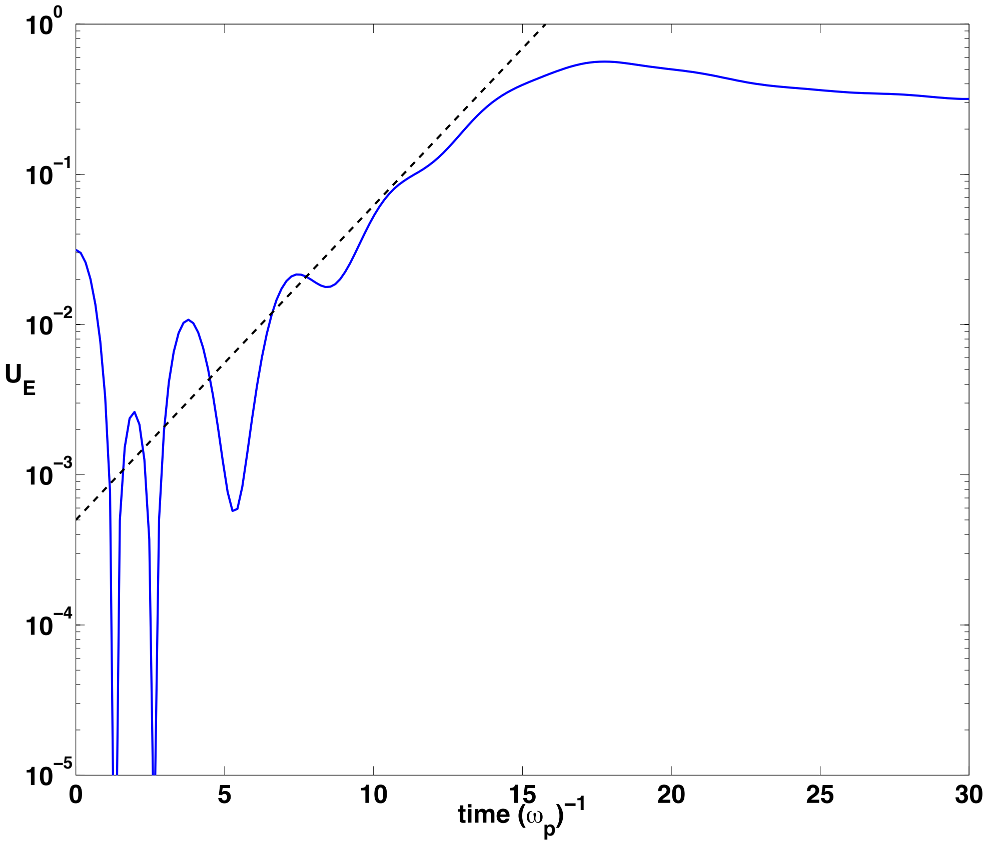

Figure 6.6: Electric field energy time series subject to the same conditions as in Figure 6.5. Simulation domain is decomposed into rectangular elements $(n_x, n_v) = (20, 80)$. Polynomial basis function order are $(P_x, P_v) = (7, 7)$. Also plotted is the theoretical linear growth rate $\gamma = 0.4817$

Figure 6.7 shows the results for a stable case, $k = 2$ in which small scale striation develops in the velocity profile, but the two steams remain stable.

Lastly, unstable conditions matching that of Ref. [21] are simulated resulting in Figure 6.8 which agrees qualitatively with the published result and highlights the fine scale structures in the solution that is resolved by the DG method.

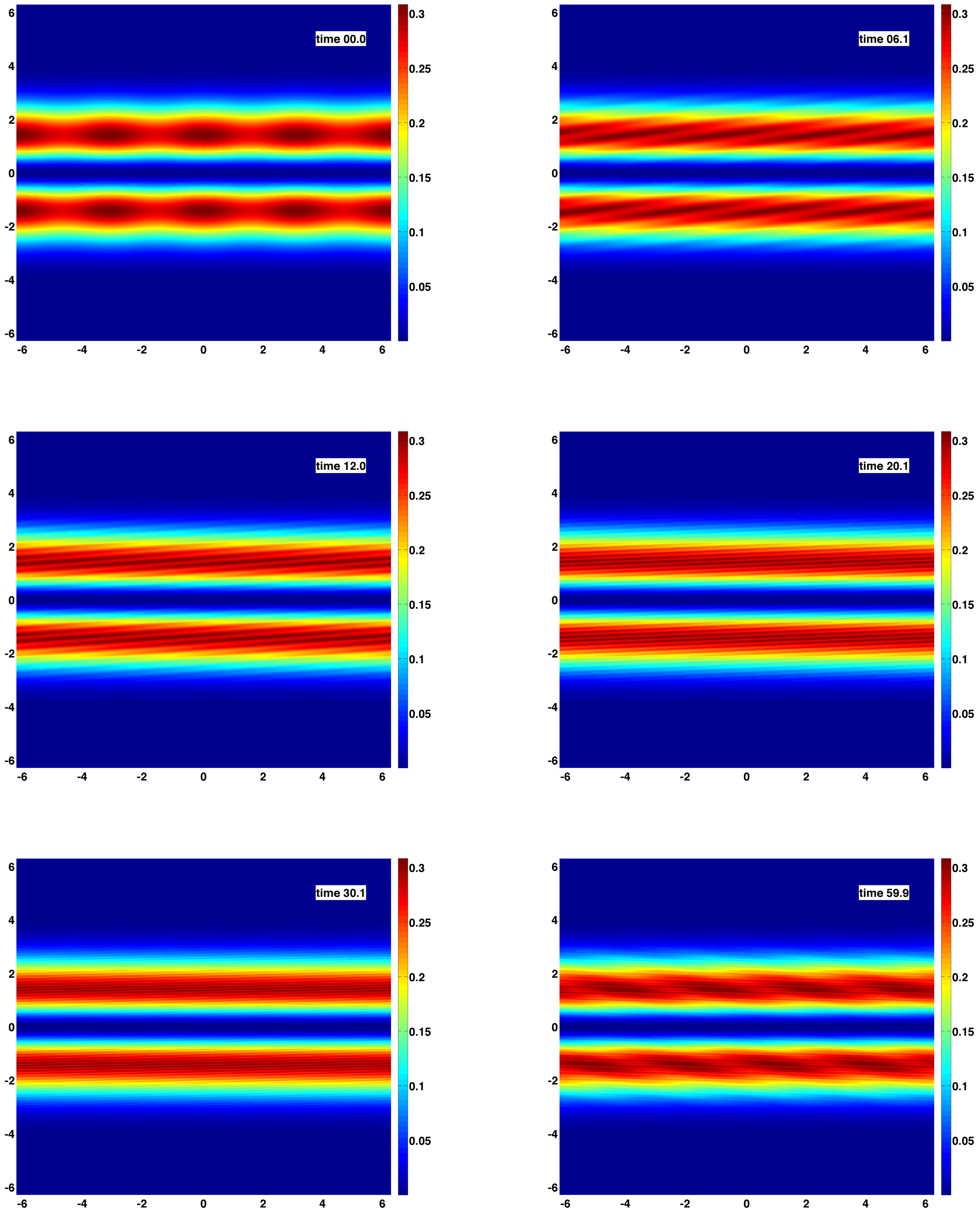

Figure 6.7: Two stream initial conditions with stable conditions, $k = 2$. Simulation domain is decomposed into rectangular elements $(n_x, n_v) = (20, 80)$. Polynomial basis function order are $(P_x, P_v) = (7, 7)$.

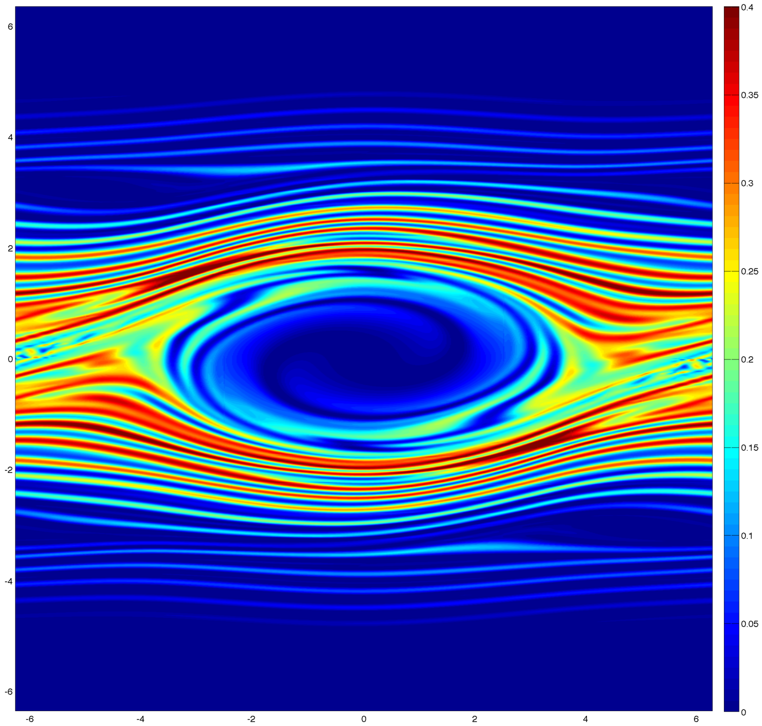

Figure 6.8: Two-stream instability fully developed vortex at time $t = 45$ presented as phase space representation of distribution function (velocity versus position). Initial condition and domain are setup as per Ref. [21]. Simulation domain is decomposed into rectangular elements $(n_x, n_v) = (20, 80)$. Polynomial basis function order are $(P_x, P_v) = (7, 7)$.

6.2 Spatially uniform plasma gyration in the Vlasov-Maxwell model

An interesting physical reduction for the Vlasov-Maxwell model is to study the dynamics of a spatially uniform plasma in the presence of electric and magnetic fields. Without spatial variation, the shape of the pdf cannot change; in velocity space it can only translate and rotate about the line $\mathbf{v} \times \mathbf{B} = 0$. This problem has utility because it has an analytic solution and can be configured to focus on the effects of velocity space discretization or the truncated velocity extent. Additionally, oscillation modes are admitted by the model that are not present in the electrostatic case.

Taking the first moment of the Vlasov equation in a coordinate system with the magnetic field oriented along the z-axis and limited to the spatially uniform case, $\frac{\partial \cdot}{\partial \mathbf{x}} = 0$, yields the following PDE system for the evolution of the centroid $\bar{v}_x$, $\bar{v}_y$, and $\bar{v}_z$.

$$\begin{aligned} \frac{\partial \bar{v}_x}{\partial t} &= \frac{q}{m}(E_x + \bar{v}_y B_z) \qquad (6.4) \\ \frac{\partial \bar{v}_y}{\partial t} &= \frac{q}{m}(E_y - \bar{v}_x B_z) \qquad (6.5) \\ \frac{\partial \bar{v}_z}{\partial t} &= \frac{q}{m}E_z. \qquad (6.6) \end{aligned}$$

The relevant remaining terms for Maxwell's equations are,

$$\begin{aligned} \frac{\partial E_x}{\partial t} &= -\frac{1}{\epsilon_0} j_x = -\frac{q n_0}{\epsilon_0}\bar{v}_x \qquad (6.7) \\ \frac{\partial E_y}{\partial t} &= -\frac{1}{\epsilon_0} j_y = -\frac{q n_0}{\epsilon_0}\bar{v}_y \qquad (6.8) \\ \frac{\partial E_z}{\partial t} &= -\frac{1}{\epsilon_0} j_z = -\frac{q n_0}{\epsilon_0}\bar{v}_z. \qquad (6.9) \end{aligned}$$

Equations (6.4) through (6.6) can be transformed into second order PDE.

$$\begin{aligned}
\frac{\partial^2 \bar{v}_x}{\partial t^2} &= \frac{q}{m}(-\frac{qn_0}{\epsilon_0}\bar{v}_x + \frac{\partial \bar{v}_y}{\partial t}B_z) && (6.10)\\
\frac{\partial^2 \bar{v}_y}{\partial t^2} &= \frac{q}{m}(-\frac{qn_0}{\epsilon_0}\bar{v}_y - \frac{\partial \bar{v}_x}{\partial t}B_z) && (6.11)\\
\frac{\partial^2 \bar{v}_y}{\partial t^2} &= -\frac{q^2 n_0}{m\epsilon_0}\bar{v}_z. && (6.12)
\end{aligned}$$

Using $w_x = \frac{\partial \bar{v}_x}{\partial t}$, $w_y = \frac{\partial \bar{v}_y}{\partial t}$, and $w_z = \frac{\partial \bar{v}_z}{\partial t}$ the following linear first order ODE system can be created of the form $\frac{\partial U}{\partial t} = AU$,

$$\frac{\partial}{\partial t}\begin{bmatrix} \bar{v}_x \\ \bar{v}_y \\ \bar{v}_z \\ w_x \\ w_y \\ w_z \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \frac{-q^2 n_0}{m\epsilon_0} & 0 & 0 & 0 & \frac{qB_z}{m} & 0 \\ 0 & \frac{-q^2 n_0}{m\epsilon_0} & 0 & \frac{-qB_z}{m} & 0 & 0 \\ 0 & 0 & \frac{-q^2 n_0}{m\epsilon_0} & 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} \bar{v}_x \\ \bar{v}_y \\ \bar{v}_z \\ w_x \\ w_y \\ w_z \end{bmatrix} \qquad (6.13)$$

The eigenvalues of the matrix A in Eq. (6.13) are complete with three purely imaginary pairs yielding three oscillation frequencies. Expressed in terms of ω_p^2 and ω_c^2 they are,

$$\begin{gathered}
\pm\hat{\imath}\omega_p \\
\pm\hat{\imath}\tfrac{1}{\sqrt{2}}\sqrt{2\omega_p^2 + \omega_c^2 + \operatorname{sgn}(qB_z)\sqrt{\omega_c^2(4\omega_p^2 + \omega_c^2)}} \\
\pm\hat{\imath}\tfrac{1}{\sqrt{2}}\sqrt{2\omega_p^2 + \omega_c^2 - \operatorname{sgn}(qB_z)\sqrt{\omega_c^2(4\omega_p^2 + \omega_c^2)}}.
\end{gathered} \qquad (6.14)$$

The corresponding eigenvectors are not orthogonal, but are linearly independent. These algebraic substitutions are useful for compactness: $A = \sqrt{2\omega_p^2 + \omega_c^2 + \sqrt{\omega_c^2(4\omega_p^2 + \omega_c^2)}}$, $B = (\omega_c + \sqrt{\omega_c^2 + 4\omega_p^2})$, $C = \sqrt{2\omega_p^2 + \omega_c^2 - \sqrt{\omega_c^2(4\omega_p^2 + \omega_c^2)}}$, and $D = (\omega_c - \sqrt{\omega_c^2 + 4\omega_p^2})$. The vector ordering follows that of the eigenvalues, with the negative imaginary eigenvalue coming before its pair.

$$\begin{bmatrix} 0 \\ 0 \\ \frac{-\hat{\imath}}{\omega_p} \\ 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ \frac{\hat{\imath}}{\omega_p} \\ 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} \frac{-2}{B} \\ \frac{-\hat{\imath}\sqrt{2}}{A} \\ 0 \\ \frac{-\hat{\imath}\sqrt{2}A}{B} \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} \frac{-2}{B} \\ \frac{\hat{\imath}\sqrt{2}}{A} \\ 0 \\ \frac{\hat{\imath}\sqrt{2}A}{B} \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} \frac{2}{D} \\ \frac{-\hat{\imath}\sqrt{2}}{C} \\ 0 \\ \frac{\hat{\imath}\sqrt{2}C}{D} \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} \frac{2}{D} \\ \frac{\hat{\imath}\sqrt{2}}{C} \\ 0 \\ \frac{-\hat{\imath}\sqrt{2}C}{D} \\ 1 \\ 0 \end{bmatrix}. \tag{6.15}$$

Solutions to the linear system have the form,

$$\begin{bmatrix} \bar{v}_x \\ \bar{v}_y \\ \bar{v}_z \\ w_x \\ w_y \\ w_z \end{bmatrix} = \sum_{i=1}^{6} c_i \mathbf{v}_i \exp(\lambda_i t), \tag{6.16}$$

where the purely imaginary eigenvalues λ_i lead to harmonic oscillation. Clearly, displacement along the magnetic field direction leads to simple plasma frequency oscillation. Displacement perpendicular to the magnetic field leads to one or two excited oscillation modes with frequencies greater than both the plasma frequency and cyclotron frequency. With just one perpendicular mode excited, electric field energy and kinetic energy remain constant, whereas when both modes are excited, an oscillating exchange of the two types of energy occurs. Table 6.1 shows demonstrative phase diagrams for the plasma mean velocity for different mode combinations.

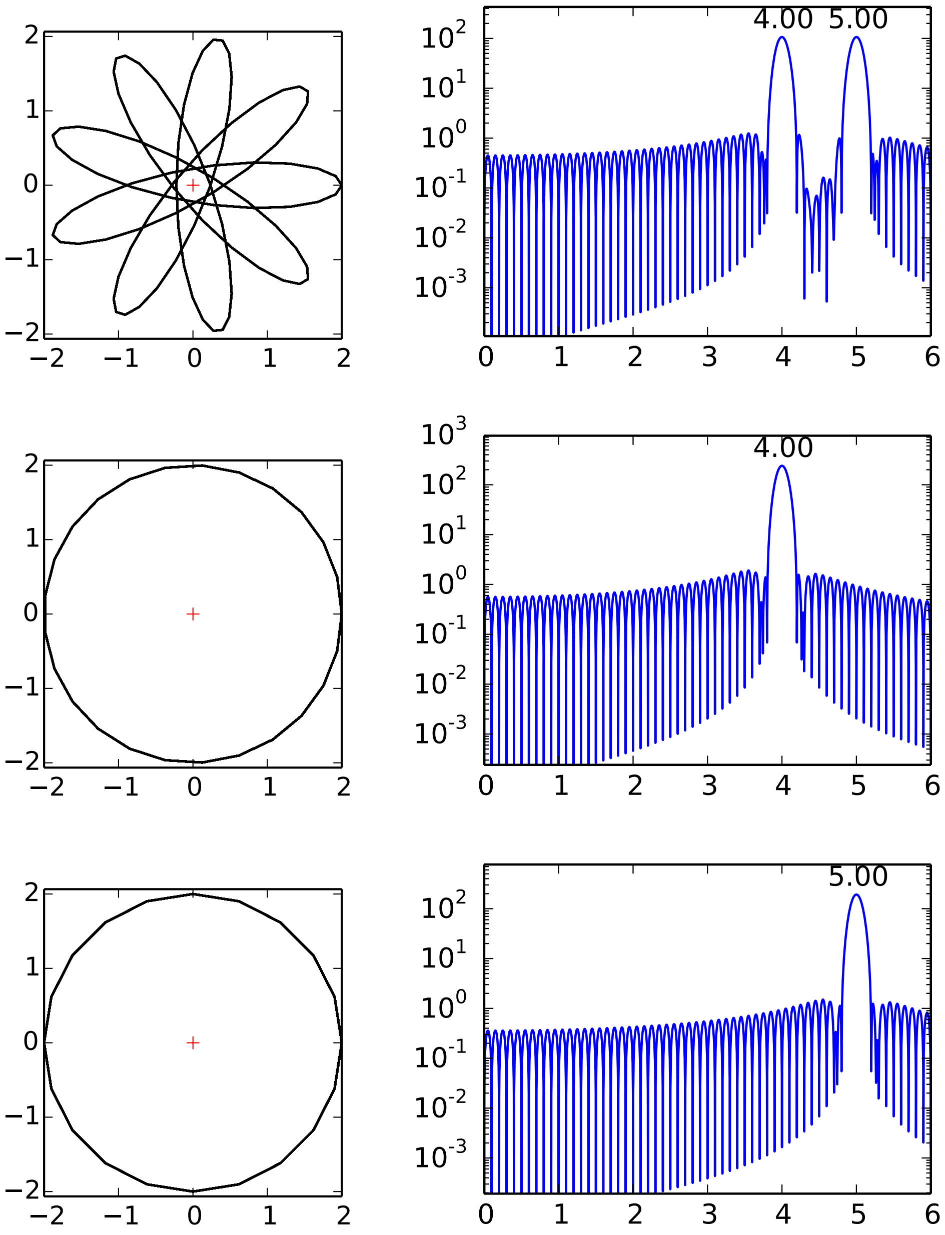

Table 6.1: Mean velocity phase diagrams and electric field spectrum for spatially uniform plasma oscillation perpendicular to a magnetic field. The left column shows velocity normalized by the thermal velocity, with $\bar{v}_x$ on the horizontal and $\bar{v}_y$ vertically. The right column frequency spectrum is normalized to cyclotron frequency, ω_c. Each row represents a different initial condition that select both or single mode dynamics based on the initial electric field strength. For these examples, $\omega_p = 1$, $\omega_c = \frac{1}{\sqrt{20}}$.

6.3 Planar Wave Propagation Perpendicular to Magnetic Field

[48]

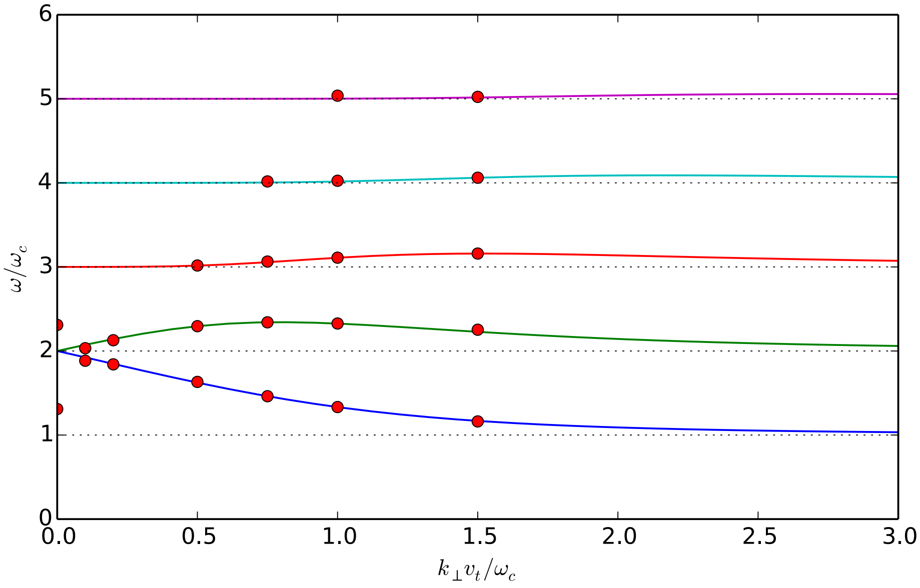

Figure 6.9: $\omega_p^2/\omega_c^2 = 3$

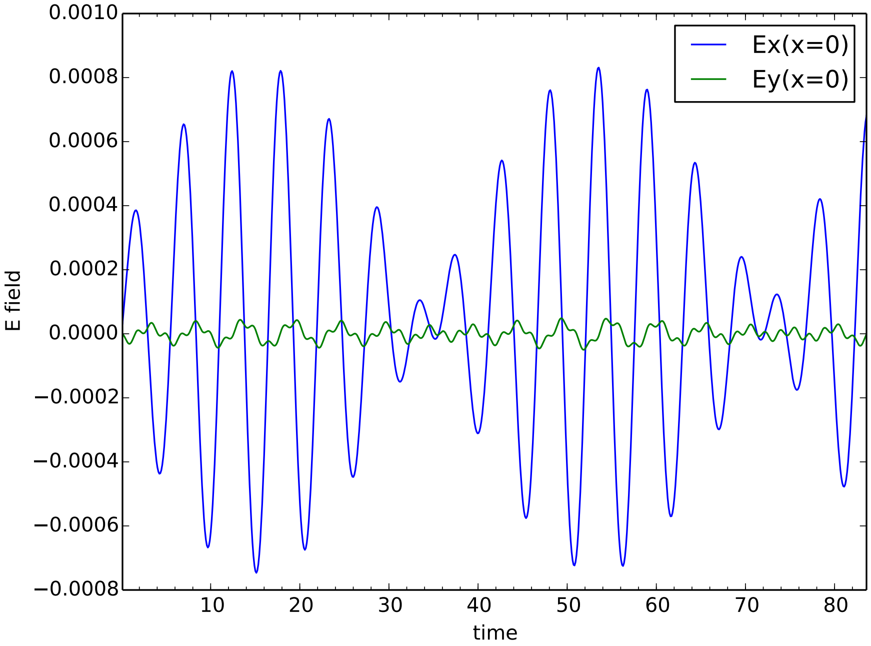

Figure 6.10: $\omega_p^2/\omega_c^2 = 3$

6.4 *Streaming Weibel instability*

The Streaming Weibel instability result for parameters labeled ‘Case 1’ in Ref. [13] was reproduced as one means to validate the WARPM Vlasov-Maxwell model implementation in $(1D + 2V)$.

The domain discretization utilized was 100 elements in all three dimension, and the DG finite elements were second-order polynomial tensor products with 27 nodes each.

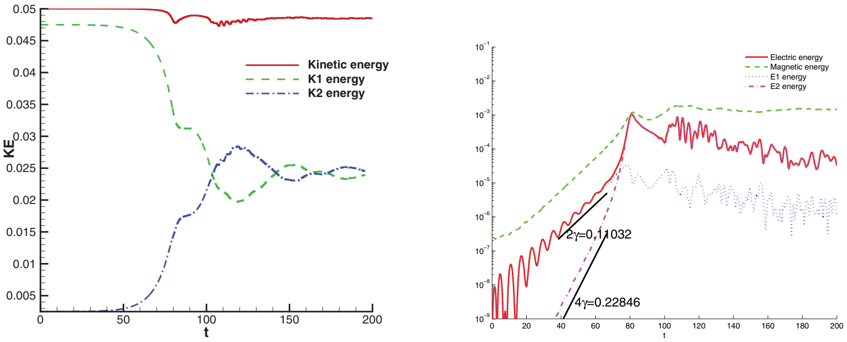

Figure 6.11: Streaming Weibel instability result for parameter ‘Case 1’ reported in Figure 5.3 of Ref. [13] copied here for comparison purposes.

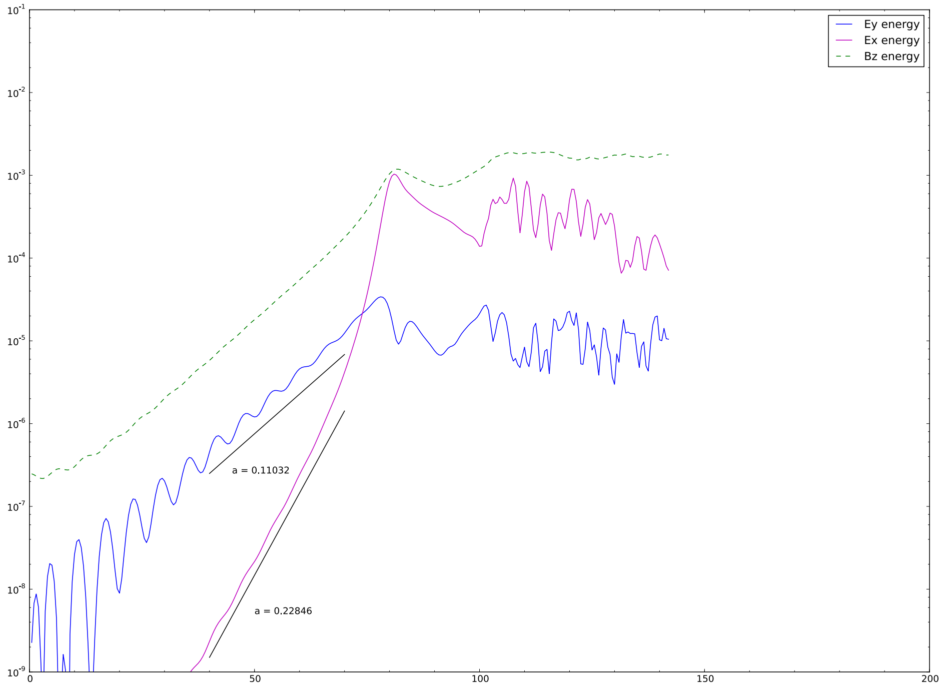

Figure 6.12: Mean field energy time series for streaming Weibel instability result simulated by WARPM Vlasov-Maxwell model with the same conditions as the published result in Figure 6.11.

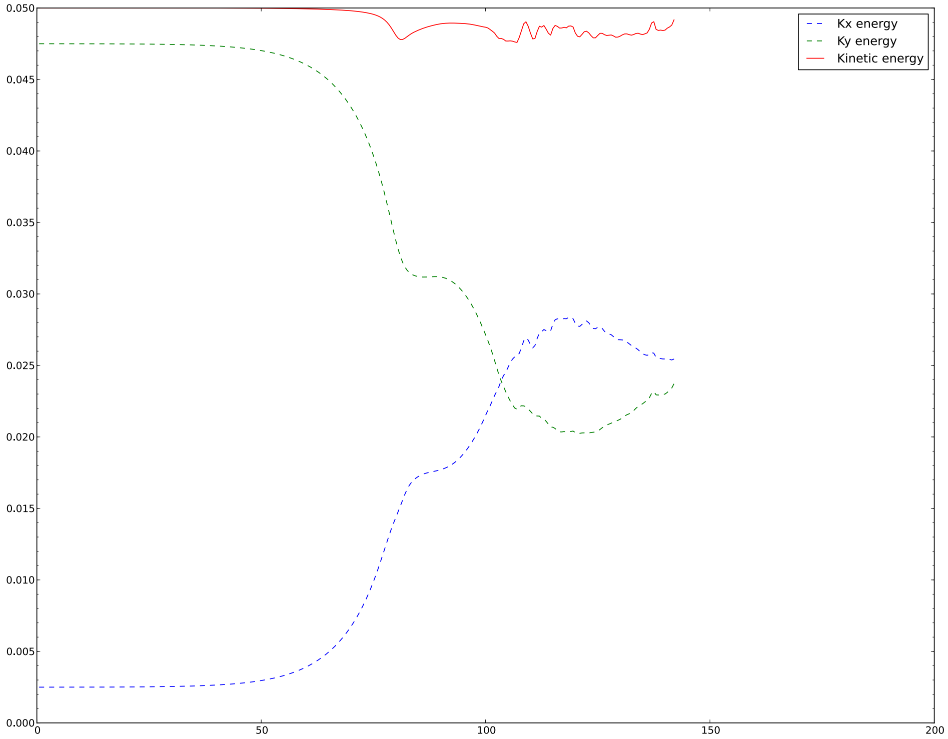

Figure 6.13: Mean kinetic energy time series for streaming Weibel instability result simulated by WARPM Vlasov-Maxwell model with the same conditions as the published result in Figure 6.11.

Chapter 7
CURRENT SHEETS

7.1 Harris Current Sheet Equilibrium

The Harris current sheet is an equilibrium condition that is useful for both verification and as a building block for more sophisticated simulations. The physical setup can be represented in one physical dimension and two velocity dimensions. The physical simulation space exists between two perfectly conducting walls with a transverse magnetic field and current-carrying fluid velocity transverse to both as depicted in Figure 7.1.

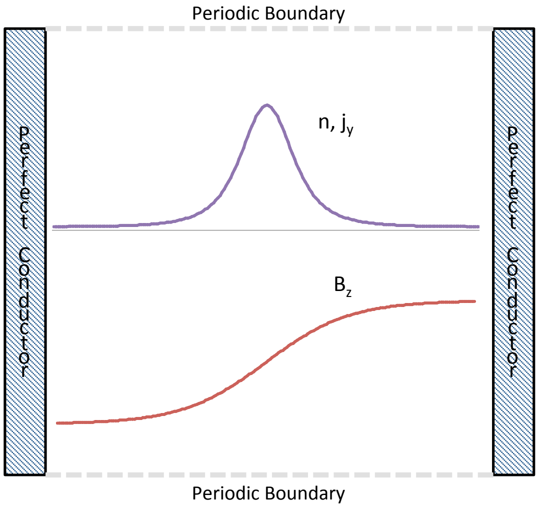

Figure 7.1: Harris Current Sheet problem setup with number density, transverse current density and magnetic field all functions of x between the two conductors.

The magnetic field gradient is balanced by a tangential current density so $\nabla \times \mathbf{B} = \mu_0 \mathbf{j}_{\text{net}}$ and no electric field is generated. The magnetic field strength varies across the sheet as

$\mathbf{B} = B_0 \tanh(x/\lambda)\hat{z}$, where λ is the sheet width scaling parameter.

Each plasma species is in thermodynamic equilibrium, i.e. a Maxwellian distribution. The equilibrium condition with no electric field requires that each species' pressure be in balance with the $q\mathbf{v} \times \mathbf{B}$ force. We'll consider a uniform fluid velocity, $\bar{u}_y$, and species temperature across the domain such that the number density varies with position,

$$f_{\mathrm{s}}(x, v_x, v_y, t = 0) = \frac{n_0}{2\pi v_{\mathrm{Ts}}^2} \exp\left(\frac{-(v_x^2 + (v_y - \bar{u}_{\mathrm{s}y})^2)}{2v_{\mathrm{Ts}}^2}\right) \frac{1}{\cosh^2(x/\lambda)}. \tag{7.1}$$

The $1D-2V$ Vlasov equation is then solved for an equilibrium condition for each species,

$$\cancelto{0}{\frac{\partial f_{\mathrm{s}}}{\partial t}} + \overset{\textcircled{1}}{\frac{\partial v_x f_{\mathrm{s}}}{\partial x}} + \frac{q_{\mathrm{s}}}{m_{\mathrm{s}}} \frac{\partial}{\partial v_x} \overset{\textcircled{2}}{\left(f_{\mathrm{s}} \cancelto{0}{E_x} + v_y B_z f_{\mathrm{s}}\right)} + \frac{q_{\mathrm{s}}}{m_{\mathrm{s}}} \frac{\partial}{\partial v_y} \overset{\textcircled{3}}{\left(f_{\mathrm{s}} \cancelto{0}{E_y} - v_x B_z f_{\mathrm{s}}\right)} = 0. \tag{7.2}$$

which requires,

$$B_0 = -\frac{2m_{\mathrm{s}} v_{\mathrm{Ts}}^2}{q_{\mathrm{s}} \bar{u}_{\mathrm{s}y} \lambda}. \tag{7.3}$$

The current density required for equilibrium balance of Ampere's law is,

$$j_y = \frac{B_0}{\mu_0 \lambda \cosh^2(x/\lambda)}. \tag{7.4}$$

Under the prescribed conditions with two species, ion and electron, at uniform fluid velocities, the current density is equivalently,

$$j_y = (q_i n \bar{u}_{\mathrm{i}y} + q_e n \bar{u}_{\mathrm{e}y}) = \frac{B_0}{\mu_0 \lambda \cosh^2(x/\lambda)}. \tag{7.5}$$

One notable solution for proton and electron plasma at the same temperature and zero net charge density requires that each species carry an equal fraction of the net current density. An initial condition in which the electrons carry all the current and thus protons have no fluid velocity cannot be an equilibrium without an electric field E_x to balance the proton pressure.

7.2 Lower-Hybrid Drift Instability

It is interesting to consider the stability of the previously described Harris current sheet equilibrium. This question has been studied in several contexts and research efforts giving opportunity to compare the developed model to existing predictions and simulations. Lower-hybrid drift instability (LHDI) refers to a perturbation growing in the intermediate frequency between electron and ion gyro-frequencies and driven by the different drift speeds of the two species.

Some analytical predictions can be made using the results of linear perturbation analysis. The analysis presented by Yoon, Lui, and Sitnov in Ref. [49] is repeated here with adaptation to a lower mass ratio and m.k.s. units.

7.2.1 Physical Scaling

Before proceeding with the linear perturbation analysis, the non-dimensional scaling conditions defined in Ref. [49] are repeated here and related to the equilibrium conditions already presented.

$$\begin{aligned} &X \equiv x/\lambda, \quad w \equiv \omega/\omega_{LH}, \quad q \equiv \frac{ck_y}{\omega_{pe0}}, \quad R \equiv \frac{\omega_{pe0}^2}{\Omega_{e0}^2}, \\ &U \equiv \bar{u}_{iy}/v_A, \quad M \equiv m_i/m_e, \quad \tau \equiv T_e/T_i. \end{aligned} \tag{7.6}$$

Appearing in the above are the lower-hybrid frequency, $\omega_{LH} = \left|\Omega_{i0}\Omega_{e0}\right|^{1/2}$, the electron cyclotron frequency outside of the sheet, $\Omega_{e0} = \frac{q_e B_0}{m_e}$, the electron plasma frequency at the sheet's peak density, $\omega_{pe0} = \sqrt{\frac{n_0 q_e^2}{\epsilon_0 m_e}}$, and the Alfvén speed also at the peak density, $v_A = \frac{B_0}{\sqrt{\mu_0 n_0 m_i}}$. Two terms to be introduced in more detail in the next section are the complex mode frequency, ω, and the perturbation wavelength along the sheet, k_y. They do not affect the equilibrium condition.

Each of the non-dimensional parameters above have a fixed value for the linear perturbation analysis in the following section. The Harris current equilibrium conditions expressed in terms of these parameters are as follows:

$$\begin{aligned} R = \frac{\omega_{pe0}^2}{\Omega_{e0}^2} &= \frac{n_0 m_e}{\epsilon_0 B_0^2}, \\ B_0^2 &= \frac{n_0 m_e}{\epsilon_0 R}, \\ B_0 &= \sqrt{\frac{n_0 m_e}{\epsilon_0 R}}. \qquad (7.7) \\ & \qquad (7.8) \end{aligned}$$

From Eq. (7.7),

$$\bar{u}_{\text{i}y} = U v_A. \qquad (7.9)$$

From the ratio of pressure balance Eq. (7.3) for both species,

$$\frac{q_e \bar{u}_{\text{e}y}}{q_i \bar{u}_{\text{i}y}} = \frac{T_e}{T_i} = \tau. \qquad (7.10)$$

From Eq. (7.5),

$$\lambda = \frac{B_0}{\mu_0 n_0 (q_i \bar{u}_{\text{i}y} + q_e \bar{u}_{\text{e}y})} = \frac{B_0}{\mu_0 n_0 q_i \bar{u}_{\text{i}y} (1+\tau)} \qquad (7.11)$$

In the above, this leaves unspecified the physical constants m_i, ϵ_0, μ_0, q_i and a seemingly arbitrary peak number density, n_0.

Examining the relationship between the equilibrium physical scaling and the dimensionless quantities reveals the following relationships are fixed only by the dimensionless parameters, i.e. no other physical scaling has an effect.

$$\left(\frac{v_{\text{Ti}}^2}{c^2}\right) = \frac{1}{2MR(1+\tau)}. \qquad (7.12)$$

$$\left(\frac{v_{\text{Te}}^2}{c^2}\right) = \frac{\tau}{2R(1+\tau)}. \qquad (7.13)$$

7.2.2 Linearized Euler-Maxwell Equations for Plasma

Assume small amplitude two-dimensional perturbations from the Harris current sheet equilibrium condition of the form,

$$\delta\mathbf{p}(\mathbf{x},t) = \delta\mathbf{p}(x)\exp\left(-\hat{\imath}\omega t + \hat{\imath}k_y y\right). \tag{7.14}$$

The linearized Euler-Maxwell system for plasma is presented below, where for compactness,

$$\Omega_j = \frac{q_j B}{m_j}, \quad W_j = \omega - \mathbf{k}\cdot\mathbf{u}_j$$

.

The system of linearized equations for electric field $\delta\mathbf{E}$, magnetic field $\delta\mathbf{B}$, and for each species number density δn_j, and momentum $m_j\delta\mathbf{u}_j$ are listed below.

Linearized Faraday's law:

$$\omega\delta\mathbf{B} = -\hat{\imath}\nabla\times\delta\mathbf{E}. \tag{7.15}$$

Linearized Ampere's law:

$$\omega\delta\mathbf{E} - \hat{\imath}c^2\nabla\times\delta\mathbf{B} = -\hat{\imath}\sum_j \frac{q_j}{\epsilon_0}\left(n\delta\mathbf{u}_j + \delta n_j\mathbf{u}_j\right). \tag{7.16}$$

Linearized Gauss's law:

$$\nabla\cdot\delta\mathbf{E} = \frac{1}{\epsilon_0}\sum_j q_j\delta n_j. \tag{7.17}$$

Absence of magnetic field divergence:

$$\nabla\cdot\delta\mathbf{B} = 0. \tag{7.18}$$

Linearized continuity equation:

$$W_j\delta n_j = -\hat{\imath}\nabla\cdot(n\delta\mathbf{u}_j). \tag{7.19}$$

Linearized momentum equation:

$$m_j\left[W_j\delta\mathbf{u}_j + \hat{\imath}\Omega_j(\hat{\mathbf{b}}\times\delta\mathbf{u}_j)\right] = \hat{\imath}q_j(\delta\mathbf{E} + \mathbf{u}_j\times\delta\mathbf{B}) - \hat{\imath}T_j\nabla\left(\frac{\delta n_j}{n}\right). \tag{7.20}$$

For compactness and alternate expression of the linearized momentum equation, it is useful to define

$$\delta \mathbf{a}_j = \delta \mathbf{E} + \mathbf{u}_j \times \delta \mathbf{B} - \frac{T_j}{q_j} \nabla \left(\frac{\delta n_j}{n} \right), \tag{7.21}$$

and

$$\mathbf{U}_j = \delta \mathbf{a}_j - \left(\frac{\hat{\imath} \Omega_j}{W_j} \right) \left(\hat{\mathbf{b}} \times \delta \mathbf{a}_j \right) - \left(\frac{\Omega_j}{W_j} \right)^2 \left[\left(\hat{\mathbf{b}} \cdot \delta \mathbf{a}_j \right) \hat{\mathbf{b}} \right]. \tag{7.22}$$

Now the linearized momentum equation can be equivalently expressed,

$$m_j \delta \mathbf{u}_j = \hat{\imath} q_j \frac{W_j}{W_j^2 - \Omega_j^2} \mathbf{U}_j. \tag{7.23}$$

7.2.3 Approximated solution for LHDI

Define the following three spatial parameters

$$\chi_j = \frac{\omega_{pj}}{W_j^2 - \Omega_j^2}, \quad \eta_j = \frac{\Omega_j}{W_j} \chi_j, \quad \sigma_j = \frac{\Omega_j^2}{W_j^2} \chi_j. \tag{7.24}$$

The term $\delta \mathbf{a}_j$ can be equivalently expressed as

$$\delta \mathbf{a}_j = \delta \mathbf{E} + \mathbf{u}_j \times \delta \mathbf{B} - n \lambda_{Dj}^2 \nabla \rho_j. \tag{7.25}$$

$$\begin{aligned} \mathbf{D}_j &= \chi_j \delta \mathbf{a}_j - \hat{\imath} \eta_j \hat{\mathbf{b}} \times \delta \mathbf{a}_j - \sigma_j \left(\hat{\mathbf{b}} \cdot \delta \mathbf{a}_j \right) \hat{\mathbf{b}} && (7.26) \\ &= \chi_j \mathbf{U}_j. && (7.27) \end{aligned}$$

A useful transformation,

$$\frac{T_j}{e_j} \nabla \left(\frac{\delta n_j}{n} \right) = n \lambda_D^2 \nabla \rho_j. \tag{7.28}$$

With the following physical quantities,

$$\omega_{pj}^2 = \frac{n q_j^2}{\epsilon_0 m_j}, \mathbf{N} = \frac{c \mathbf{k}}{\omega}, \mathbf{V}_j = \frac{\mathbf{u}_j}{c}, \rho_j = \frac{q_j \delta n_j}{\epsilon_0 n}, \lambda_{Dj}^2 = \frac{\epsilon_0 T_j}{n q_j^2}. \tag{7.29}$$

The linearized system with the above definitions transforms to the following.

$$n \rho_j = \nabla \cdot \mathbf{D}_j. \tag{7.30}$$

$$\omega\delta\mathbf{B} = -\hat{\imath}\nabla \times \delta\mathbf{E}. \tag{7.31}$$

$$\nabla \cdot \delta\mathbf{E} = \sum_j \nabla \cdot \mathbf{D}_j. \tag{7.32}$$

$$\omega\delta\mathbf{E} - \hat{\imath}c^2\nabla \times \delta\mathbf{B} = \sum_j \left[\omega\left(1 - \mathbf{N}\cdot\mathbf{V}_j\right)\mathbf{D}_j - \hat{\imath}c\mathbf{V}_j\nabla\cdot\mathbf{D}_j\right]. \tag{7.33}$$

The authors of Ref. [49] make a simplifying approximation neglecting the λ_{Dj}^2 term in the definition of δa_j, Eq. (7.25). The validity of this approximation is considered in Section 7.2.6.

Taking into account the assumed perturbation form of Eq. (7.14), a complete linear equation system is formed by taking the spatial derivative of Eqs. (7.31) and (7.33) z-component with respect to position across the current sheet, x. The system is presented in Eq. (7.34). The solution of this system along with the relations for momentum perturbation Eq. (7.23) fully specify the perturbation eigen-mode based on the solution of δE_y only.

$$\left[\begin{array}{c:c:c:c:c:c}
-k_y & \omega & & & & \\ \hdashline
 & & -k_y & \omega & & \\ \hdashline
 & & -1 & & \frac{q_i}{\epsilon_0} & \frac{q_e}{\epsilon_0} \\ \hdashline
\begin{array}{c}\hat{\imath}W_i\eta_i\\ +\hat{\imath}W_e\eta_e\end{array} & \begin{array}{c}-\hat{\imath}W_i\eta_i u_{iy}\\ -\hat{\imath}W_e\eta_e u_{ey}\end{array} & & \hat{\imath}c^2 & -\hat{\imath}\frac{1}{\epsilon_0}q_i u_{iy} & -\hat{\imath}\frac{1}{\epsilon_0}q_e u_{ey} \\ \hdashline
\begin{array}{c}-\omega\\ +W_i\chi_i\\ +W_e\chi_e\end{array} & \begin{array}{c}c^2k_y\\ -\chi_i u_{iy}W_i\\ -\chi_e u_{ey}W_e\end{array} & & & & \\ \hdashline
\begin{array}{c}W_i\chi_i'\\ +W_e\chi_e'\end{array} & \begin{array}{c}-W_i\chi_i' u_{iy}\\ -W_e\chi_e' u_{ey}\end{array} & \begin{array}{c}-\omega\\ +W_i\chi_i\\ +W_e\chi_e\end{array} & \begin{array}{c}c^2k_y\\ -W_i\chi_i u_{iy}\\ -W_e\chi_e u_{ey}\end{array} & &
\end{array}\right]
\left[\begin{array}{c}
\delta E_z\\ \delta B_x\\ \delta E_z'\\ \delta B_x'\\ \delta n_i\\ \delta n_e
\end{array}\right] =$$

$$\begin{bmatrix} \hat{\imath}\delta E_y' \\ \hat{\imath}\delta E_y'' \\ \hat{\imath}k_y\delta E_y \\ \\ \delta E_y\left(\omega - W_i\chi_i - W_e\chi_e\right) \\ \\ \\ \delta E_y\left(\hat{\imath}W_i\eta_i + \hat{\imath}W_e\eta_e\right) \\ \\ \\ \\ \left(\hat{\imath}W_i\eta_i' + \hat{\imath}W_e\eta_e'\right)\delta E_y + \left(\hat{\imath}W_i\eta_i + \hat{\imath}W_e\eta_e\right)\delta E_y'' \\ \\ \end{bmatrix} \tag{7.34}$$

7.2.4 Numerical solutions for LHDI Eigenmodes

Solution of the linearized system Eq. (7.34) proved quite challenging due to the stiffness of the equation system. Focus was directed toward odd modes solutions only. Ultimately a shooting method was used to solve for the complex eigenvalues of the system. In all cases, shooting from the current sheet centerline resulted in the eigenfunction growing unboundedly to $\pm\infty$ a few sheet widths away from centerline. It turns out that a manifold could be detected near the eigenvalue depending on which direction the solution diverged. One example set of solutions is presented in Figure 7.2 of which a subset (the lower right arm) agree well with the solutions presented in Ref. [49]. Additional potential eigenvalues are identified by this method, all with potentially higher growth rate based on the imaginary part. The implications of these should be explored further.

The realistic mass ratio of $M = 1836$ is quite high for a numerical simulation at this time and so the solution approach was repeated with $M = 25$. Actually, the approach was repeated for numerous mass ratios in the range 1836 to 25. The eigenvalues were tracked during the parameter progression. The final eigenvalues are presented in Figure 7.3. Only one eigenvalue value was identified that evolved from one at $M = 1836$.

The full set of eigenvalues identified with mass ration $M = 25$ are presented in Figure 7.4.

To solve for the eigenfunction, a different technique was utilized based on Chebyshev polynomial solution to the boundary value problem. First a mode of interest was identified and the eigenvalue refined to five or more significant digits by successive graphical reduction of the search space centered around the previous solution.

7.2.5 Numerical solution with WARPM Vlasov-Maxwell model

With one motivation being to validate the WARPM Vlasov-Maxwell model implementation in $(2D + 2V)$, it was attempted to simulate the single even eigenmode ‘A’ identified by the linearized analysis. This ultimately proved unsuccessful for reasons warranting further study. Rather than observing the predicted linear growth rate or eigenfunction form as predicted

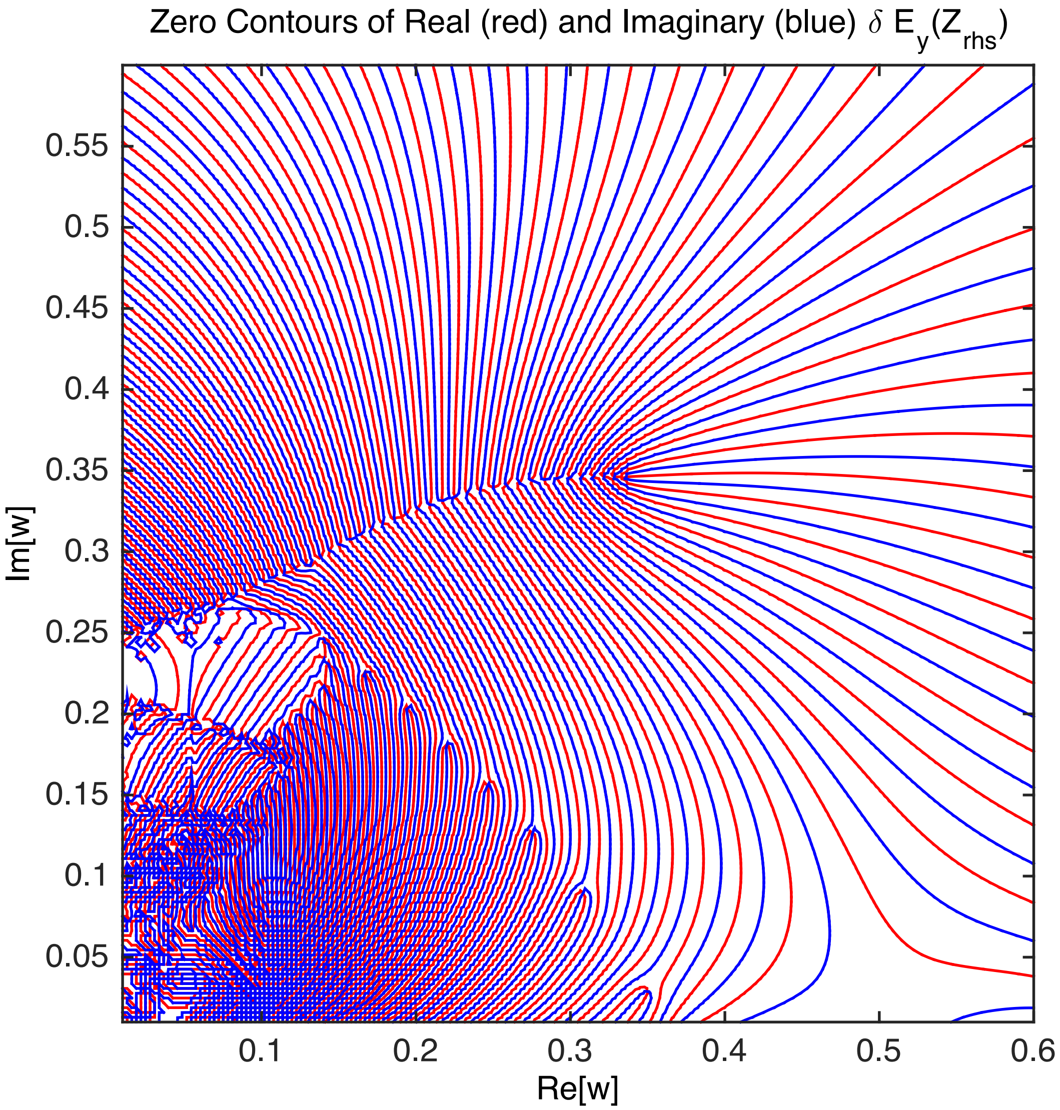

Figure 7.2: Zero-contour intersections indicate potential eigenvalues for the odd-mode solutions of Eq. (7.34) with parameters $M = 1836$, $R = 100$, $U = 1$, $\tau = 0.1$ consistent with those for the solutions presented in Figure 3 of Ref. [49]. The lower-right branch intersections agree with the eigenvalues in the figure cited.

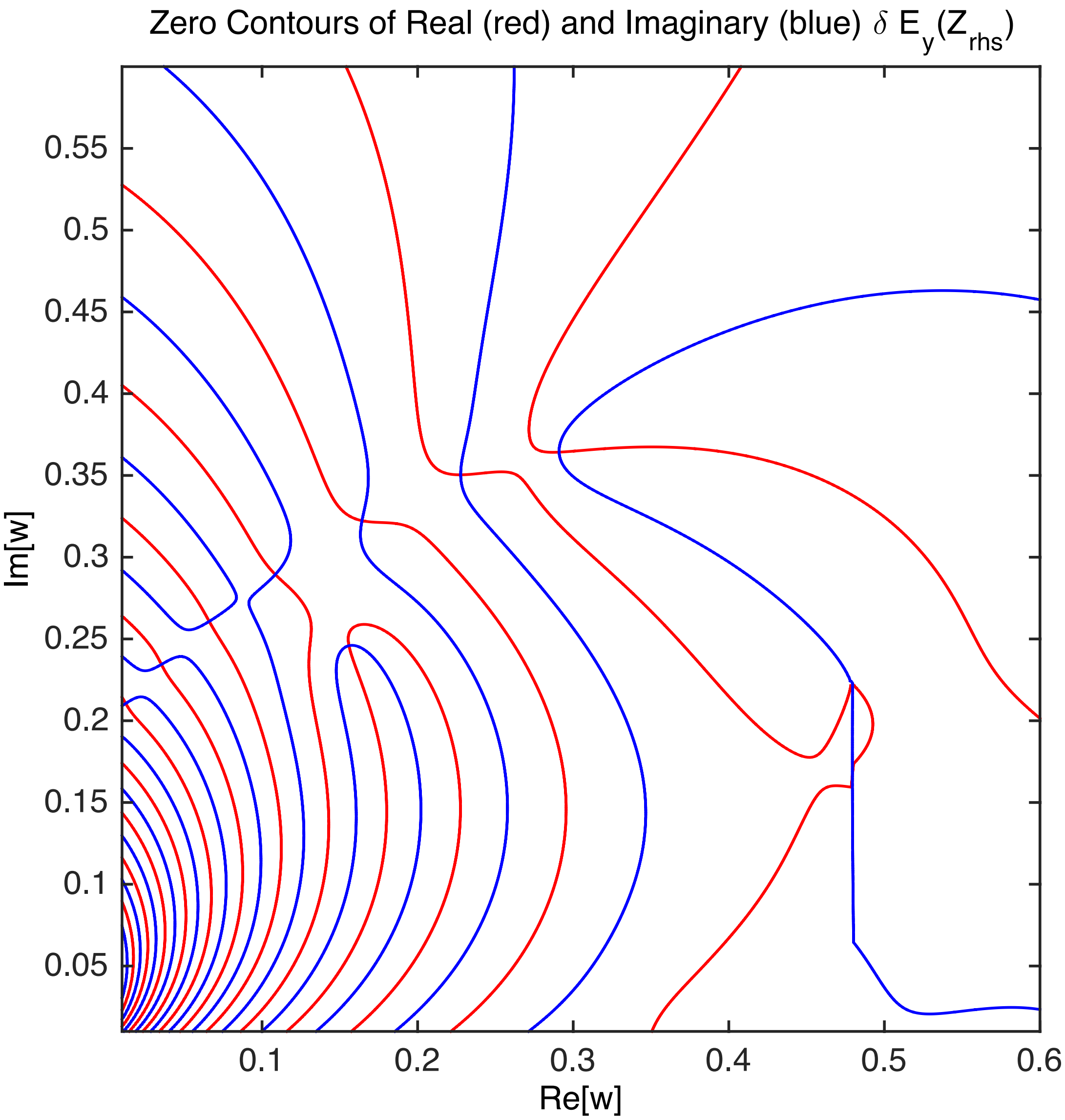

Figure 7.3: Zero-contour intersections indicate potential eigenvalues for the odd-mode solutions of Eq. (7.34) with parameters $M = 25$, $R = 100$, $U = 1$, $\tau = 0.1$. Only the intersection at $\omega = 0.16 + 0.24\hat{i}$ seems to correspond with an eigenmode from Ref. [49] with $M = 1836$.

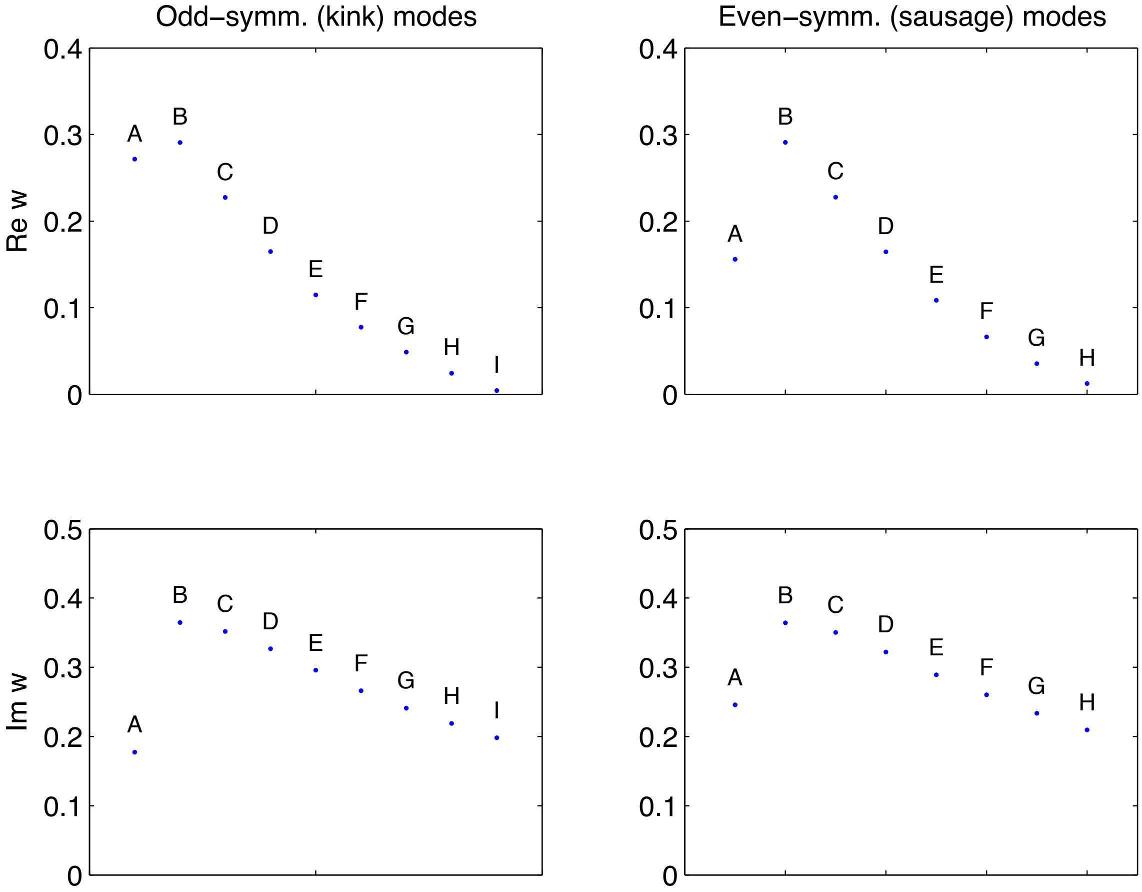

Figure 7.4: Full set of even and odd mode eigenvalues identified and labeled for parameter set $M = 25$, $R = 100$, $U = 1$, $\tau = 0.1$.

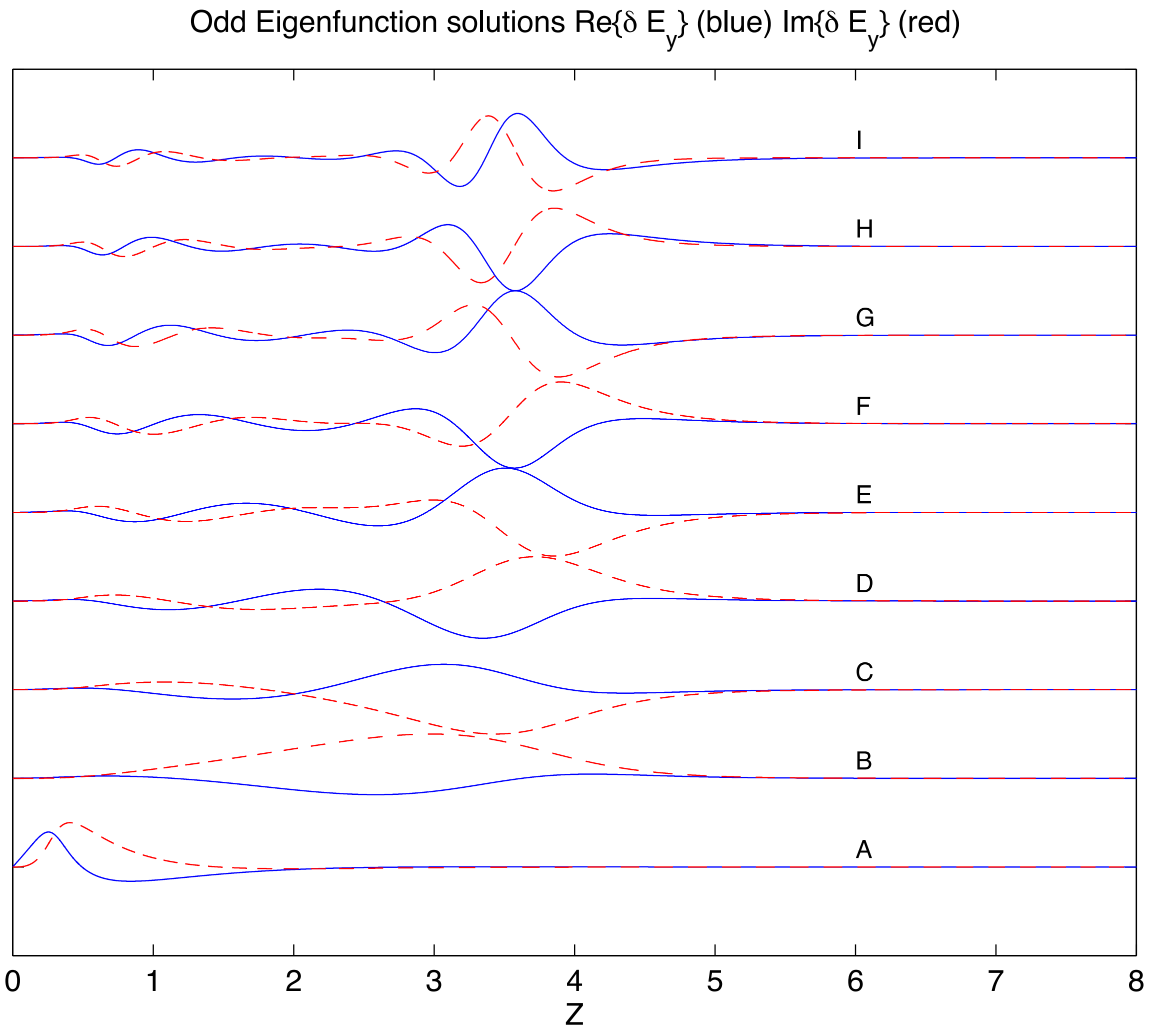

Figure 7.5: Odd mode eigenfunction solutions for the eigenvalues presented in Figure 7.4 with parameter set $M = 25$, $R = 100$, $U = 1$, $\tau = 0.1$. Note that only Mode A has perturbation concentrated in the current sheet region $Z < 1$ associated with the LHDI

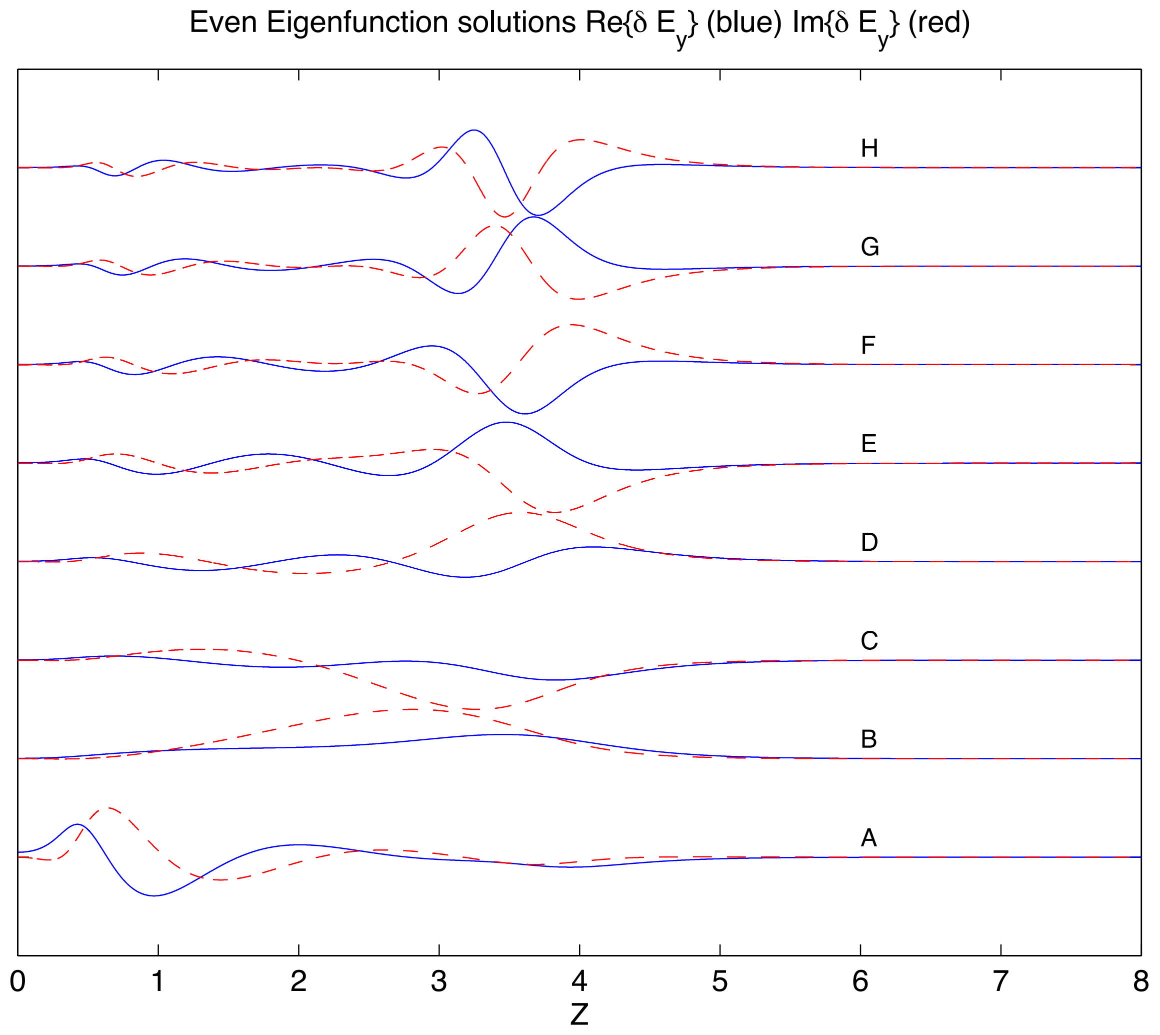

Figure 7.6: Even mode eigenfunction solutions for the eigenvalues presented in Figure 7.4 with parameter set $M = 25$, $R = 100$, $U = 1$, $\tau = 0.1$. Note that only Mode A has perturbation concentrated in the current sheet region $Z < 1$ associated with the LHDI.

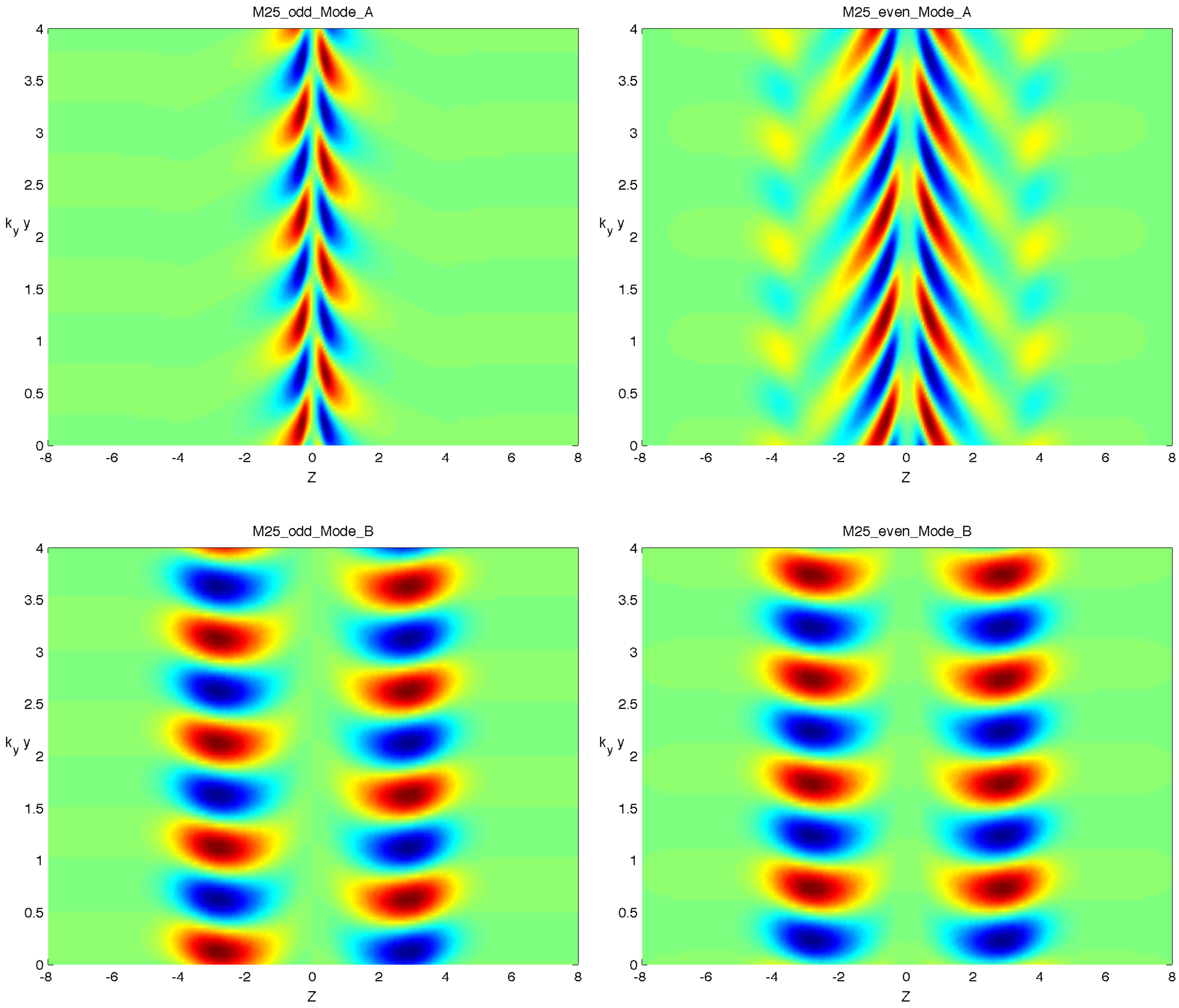

Figure 7.7: Eigenfunction solutions for δE_y in Eq. (7.34) associated with the corresponding eigenvaluse in Figure 7.4.

by linear analysis, a growth rate 2-3 times faster was observed with the electric field energy exclusively outside of the current sheet region. An exemplary early developing simulation result is presented in Figure 7.8. This type of result seems to develop for a variety of initial conditions.

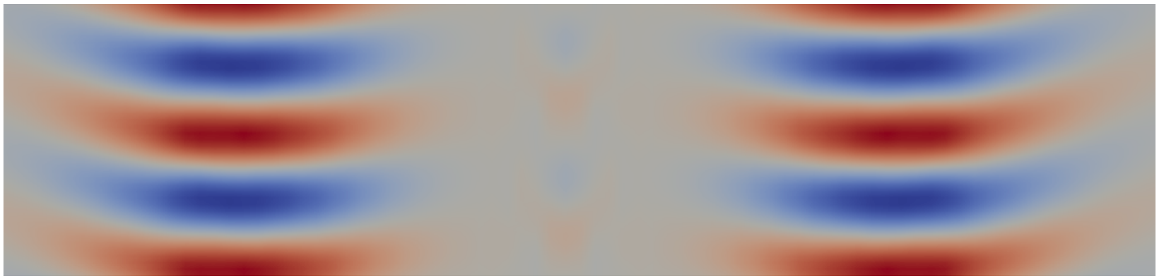

Figure 7.8: Typical WARPM Vlasov-Maxwell simulation result with initialized with equilibrium perturbation adhearing to the solution for even mode ‘A’. The image shows the E_y electric field component amplitude.

Two possibilities explored were that the initial perturbation amplitude was too strong or imprecise and that the simplifying assumptions made in Yoon’s linearized analysis are not satisfied.

To consider the first case, it is possible to evaluate the time derivative of the Vlasov-Maxwell system analytically given the eigenfunction initial perturbation and compare that result to the time derivative predicted by the eigenvalue linear growth rate. It was discovered that the two only balanced when the perturbation amplitude was extremely small – on the order of one part per million for the out-of-plane magnetic field perturbation. A potential cause for this sensitivity is discussed in the next section.

Such a small perturbation is not achievable with any accuracy in the presence of discretization and projection errors for the initial field components and pdf. One strategy that could be pursued would be to implement a so-called delta-f method, which only tracks deviation from an initial condition. There would be no initial discretization or projection error associated with the equilibrium.

7.2.6 Considerations for the simplifying approximation

This term in Eq. (7.25) must be small to satisfy the Yoon linear perturbation approximation.

For the electron $\hat{y}$-component,

$$\left|\left(\frac{u_{the}^2}{c^2}\right) cq \frac{\sqrt{m_e}}{\sqrt{\epsilon_0}} \frac{\sqrt{n_0}}{n} \left(\frac{\delta n_e}{\delta E_y}\right)\right| << 1. \tag{7.35}$$

For ion $\hat{y}$-component,

$$\left|\left(\frac{u_{thi}^2}{c^2}\right) cq \frac{|q_e|}{q_i} \frac{m_i}{\sqrt{m_e \epsilon_0}} \frac{\sqrt{n_0}}{n} \left(\frac{\delta n_e}{\delta E_y}\right)\right| << 1. \tag{7.36}$$

The $\hat{x}$-component approximation has less utility due to the gradient component across the sheet. Each species requires,

$$\left| - \frac{\frac{T_j}{n_0 q_j} \left(\frac{\partial \delta n_j}{\partial x} \cosh^2(x/\lambda) + \delta n_j \frac{2}{\lambda} \cosh(x/\lambda) \sinh(x/\lambda) \right)}{\delta E_x + u_{yj} \delta B_z} \right| << 1. \tag{7.37}$$

For all parameters explored, the $\hat{y}$-component of the dropped second-order term in Eq. (7.25) was in fact three to five times larger than the retained first-order terms. No physical scaling parameter was identified that affected this condition. An attempt was made to continue the linearized analysis with the term retained, but was ultimately unsuccessful.

Chapter 8

MAGNETIC RECONNECTION

A natural extension to the current sheet physics described in Chapter 7 is the study of magnetic field reconnection [50]. A now thoroughly studied configuration for reconnection experiments is also based on perturbation of the Harris current sheet equilibrium [6] .

Magnetic reconnection is characterized by a change in topology or connection of magnetic field lines in different closed domains. In a perfectly conducting medium, this is not possible. In very low resistivity plasma, observed reconnection rates far exceed predictions of resistive MHD. The developed Vlasov-Maxwell model can be applied to the fast magnetic reconnection phenomena to improve understanding of the dynamics. The problem geometry is rotated compared to the earlier Harris current sheet setup with an out-of-plane current and thus requires a $2D + 3V$ phase space simulation.

8.1 GEM Magnetic Reconnection Challenge Problem

The Geospace Environmental Modeling (GEM) magnetic reconnection challenge initially specified in Ref. [6] describes a two-species plasma initial configuration targeted for the study of reconnection. The problem has been studied many times with models ranging from ideal MHD, multi-fluid plasma [12], extended moment models [51, 52, 53, 54], and kinetic models. Kinetic models include Particle-in-Cell [55, 56], Vlasov-Darwin [57], and Vlasov-Maxwell [58].

8.1.1 Initial Condition

The initial condition and physical domain setup for the GEM Challenge is very similar to the Harris current sheet equilibrium described in Chapter 7. The primary difference is that the current sheet is rotated such that current is out-of-plane in the $\hat{z}$-direction while magnetic field is in-plane in the $\hat{y}$-direction. In addition to the two-species plasma current sheet, a background two-species uniform plasma with zero fluid velocity is introduced with 20% of the number density of the current-carrying plasma and same uniform temperature.

$$\begin{aligned} f_{\mathrm{s}}(x, v_x, v_y, v_z, t=0) &= \frac{n_0}{(2\pi)^{\frac{3}{2}}\, v_{\mathrm{Ts}}^3} \exp\left(\frac{-\left(v_x^2 + v_x^2 + (v_z - \bar{u}_{z_{\mathrm{s}}})^2\right)}{2 v_{\mathrm{Ts}}^2}\right) \frac{1}{\cosh^2(x/\lambda)} \\ &+ \frac{n_b}{(2\pi)^{\frac{3}{2}}\, v_{\mathrm{Ts}}^3} \exp\left(\frac{-(v_x^2 + v_x^2 + v_z^2)}{2 v_{\mathrm{Ts}}^2}\right). \end{aligned} \tag{8.1}$$

where the current sheet width is again given the symbol, λ, the each species' thermal velocity is v_{Ts}, and each species mean velocity is $[0, 0, \bar{u}_{z_{\mathrm{s}}}]$.

The bulk magnetic field satisfies the equilibrium requirement described in Chapter 7 and includes a perturbation in the planar field components,

$$\mathbf{B} = -B_0 \tanh(x/\lambda)\hat{y} + \delta\mathbf{B}. \tag{8.2}$$

The perturbation has spatial dependence on the width between conductors, L_x, and the length in the periodic direction, L_y.

$$\delta\mathbf{B} = -\hat{z} \times \nabla\psi, \tag{8.3}$$

where

$$\psi(x, y) = \psi_0 \cos\left(\frac{\pi x}{L_x}\right) \cos\left(\frac{2\pi y}{L_y}\right). \tag{8.4}$$

The expanded expression for the perturbation makes clear the periodic nature along the sheet and half-period across the conductors. The perturbed field is tangential to the conducting walls and normal to the centerline with zero net flux across the centerline.

$$\begin{aligned} \delta\mathbf{B} = -\frac{2\pi}{L_y}\psi_0 \cos(\frac{\pi}{L_x}x)\sin(\frac{2\pi}{L_y}y)\hat{x} \qquad (8.5) \\ +\frac{\pi}{L_x}\psi_0 \sin(\frac{\pi}{L_x}x)\cos(\frac{2\pi}{L_y}y)\hat{y}. \qquad (8.6) \end{aligned}$$

8.1.2 Normalization

It is common for the GEM magnetic reconnection challenge to be presented in normalized units. Time is normalized to the ion cyclotron radial frequency, ω_{ci}^{-1}, and distance by the ion inertial length per radian, d_i, with definitions based on bulk conditions.

$$\omega_{ci} = \frac{q_i B_0}{m_i}. \tag{8.7}$$

$$d_i = \frac{c}{\omega_{pi}} = \frac{c}{\sqrt{\frac{n_0 q_i^2}{m_i \epsilon_0}}}. \tag{8.8}$$

8.1.3 Physical Parameters

$$B_0 = 1$$

$$n_0 = 1$$

$$n_b = 0.2n_0$$

$$q_i = 1$$

$$q_e = -1$$

$$m_i = 1$$

$$M = \frac{m_i}{m_e} = 25$$

$$\tau = \frac{T_e}{T_i} = 0.2$$

$$\psi_0 = 0.1B_0$$

$$\lambda = 0.5d_i$$

$$L_x = 12.8d_i$$

$$L_y = 25.6d_i$$

$$\epsilon_0 = 0.015$$

$$\mu_0 = 0.015$$

$$c = 66.6666$$

$$d_i = 8.1649658$$

8.1.4 Simulated Solution

The $2D+3V$ domain is discretized in a rectilinear arrangement of hyper-rectangles as follows:

Across the sheet and between the perfectly conducting walls, $N_x = 48$ elements. The element spacing is not uniform. Element spacing is smallest at the centerline, $x = 0$, and grows linearly until the elements along the conducting wall are twice as wide. Along the sheet, $N_y = 30$ elements are arranged with uniform spacing and periodic boundaries at the O-point. The number of elements in each velocity dimension are equal, $N_{v_x} = N_{v_y} = N_{v_z} = 12$. The total extent of each truncated velocity dimension is limited to $2V_{\mathtt{max},i} = 12v_{\mathrm{th},i}$ and $2V_{\mathtt{max},e} = 12v_{\mathrm{th},e}$. For each species, the velocity mesh is centered and symmetric around a non-zero offset $\mathbf{v}_0$. To efficiently utilize the limited discretized velocity space, $\mathbf{v}_0$ is chosen to be the initial fluid velocity for each species. Thus, for out-of-plane equilibrium current density the velocity $v_z \in [-V_{\mathtt{max}} + v_{z0}, V_{\mathtt{max}} + v_{z0}]$. The velocity mesh is rectilinear but non-uniform. The element spacing is smallest at the velocity center $\mathbf{v}_0$ and stretches up to a factor of 3 at the extrema according to the cubic polynomial,

$$an^3 + bn^2 + cn + d = 0, \tag{8.9}$$

where n is the element index. The factors a,b, and c are solved to give desired stretch, smallest elements at the velocity center and symmetry about the velocity center. This is done to improve the velocity domain resolution about the centroid of each pdf with trade-off of less resolution near the $V_{\mathtt{max}}$ extrema where smaller features are less expected. The resulting rectilinear mesh is shown in Figure 8.1.

Each high-order element is a tensor product of 3$^{\mathrm{rd}}$-order Legendre polynomials. The total degrees of freedom for each species' probability distribution function is $48 \times 30 \times 12^3 \times 4^5 = 2.55 \times 10^9$ nodes.

Physical solution results from the full domain simulation are presented in Figures 8.1.4 through 8.5 with more discussion in each figure caption. Considering the magnetic field streamlines in Figure 8.5, no magnetic island formation is observed nor expected. Numerical models with greater numerical dissipation can introduce anomalous stable magnetic islands at or in vicinity of the X-point [12].

8.1.5 Reconnected Flux

In expanded form, it is clear that the initial magnetic field has net zero flux through the centerline, in the $\hat{x}$-direction. This will remain true and thus the absolute value is taken to

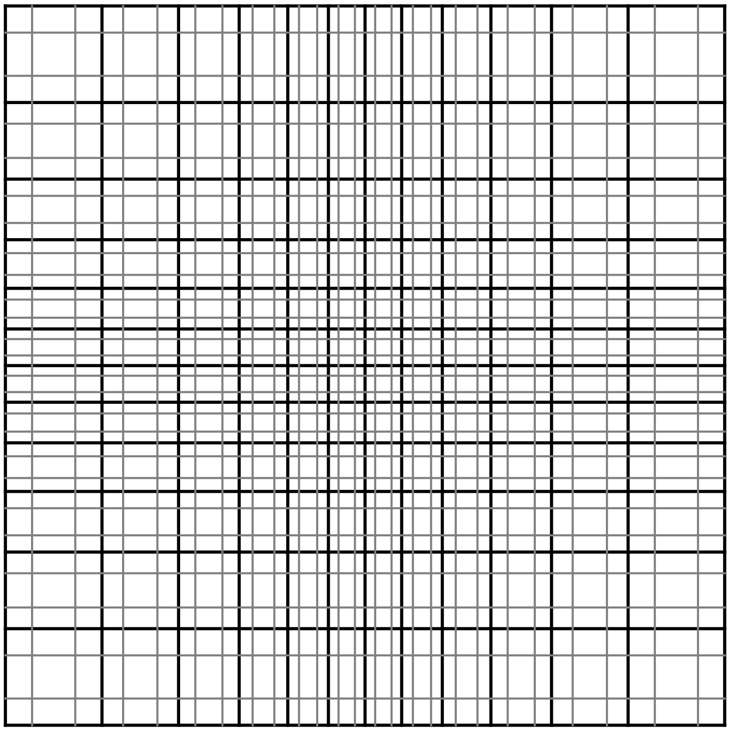

Figure 8.1: Representative structured velocity mesh in v_x and v_y dimensions utilized in the GEM challenge simulations. There are twelve finite elements in each velocity dimension. The interior nodes are also depicted for the tensor-product 3$^{\text{rd}}$-order polynomial elements so there are four points per element per dimension. Element faces are the darker lines. Element vertex spacing is stretched per Eq. (8.9) to improve resolution near the centroid.

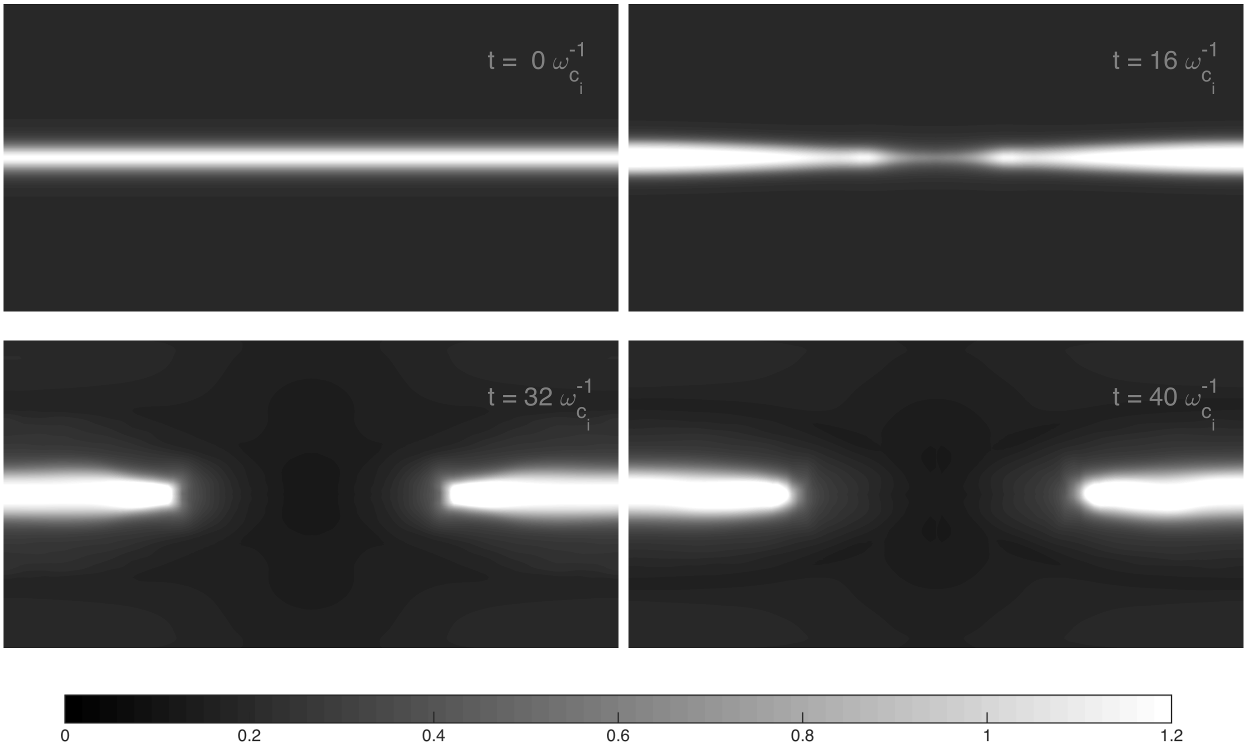

Figure 8.2: Ion number density at four representative time points during the simulation. The initial sheet peak density is $n_0 = 1.0$ plus an additional uniform background fraction of $n_b = 0.2$. Over the course of the reconnection event, the bulk plasma retreats along the sheet axis away from the X-point resulting in a higher number density.

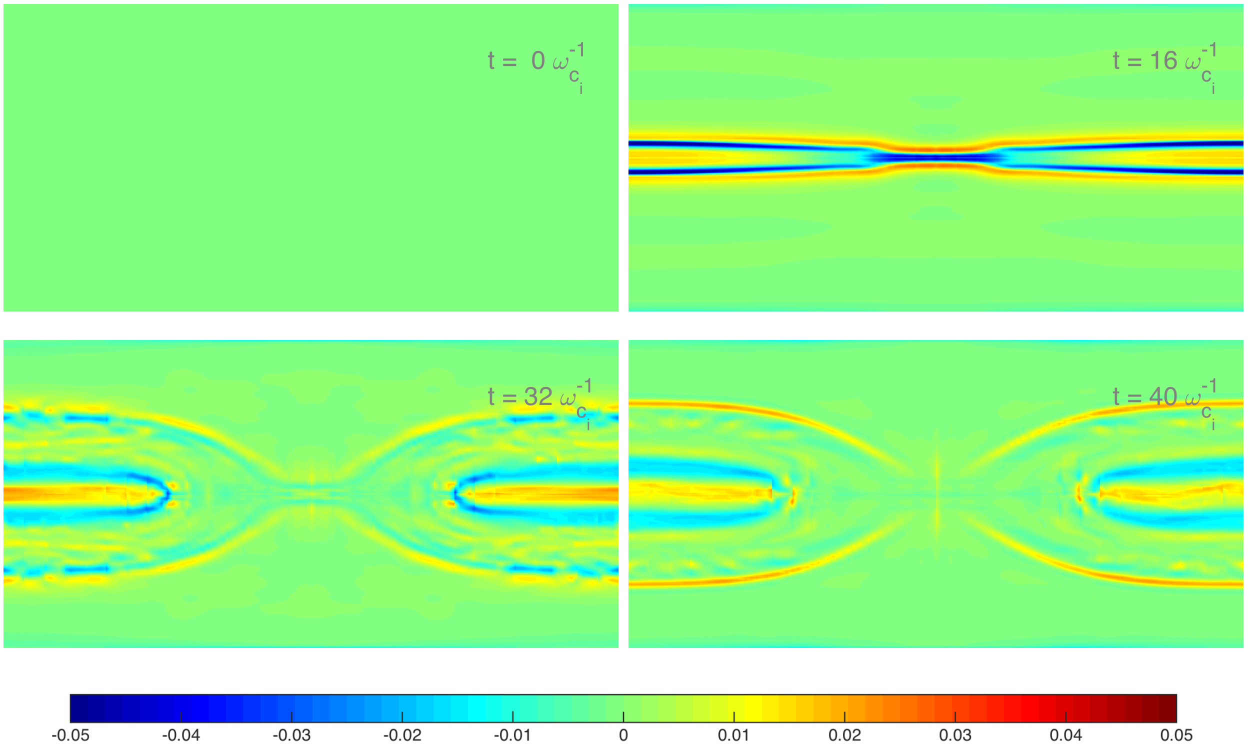

Figure 8.3: Net charge density at four representative time points during the simulation. The ion number density in Figure 8.1.4 at the same time points is useful for establishing relative scaling. Some structures with net charge density are consistent over the duration of the magnetic reconnection event with regions of positive charge density collocated with the magnetic field separatrix and a region of negative charge surrounding the O-point containing the bulk plasma density. Also visible are small scale structures inside of the separatrix most pronounced at the time of fastest reconnection.

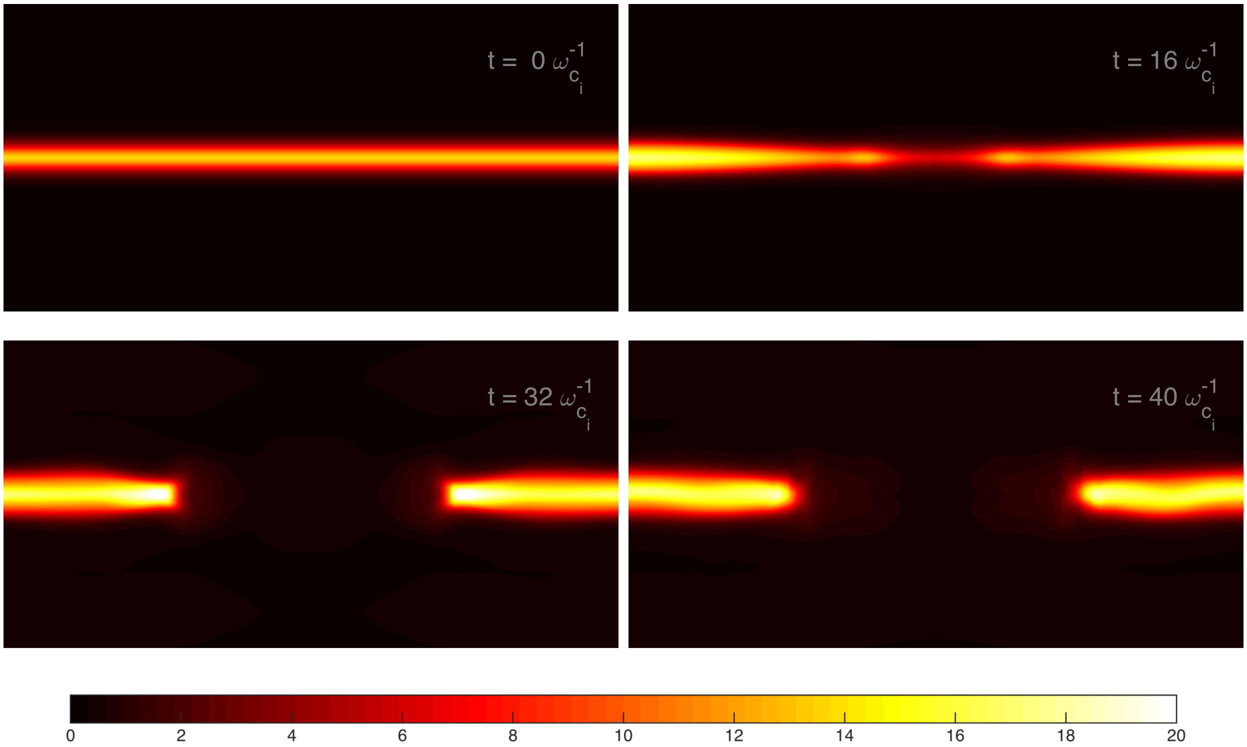

Figure 8.4: Total ion momentum as r.m.s. magnitude at four representative time points during the simulation. As the current sheet separates and retreats from the X-point, significantly more momentum is located at the retreating edge with sharp fall-off. An emerging asymmetry across the current sheet is visible at the final frame of simulated time.

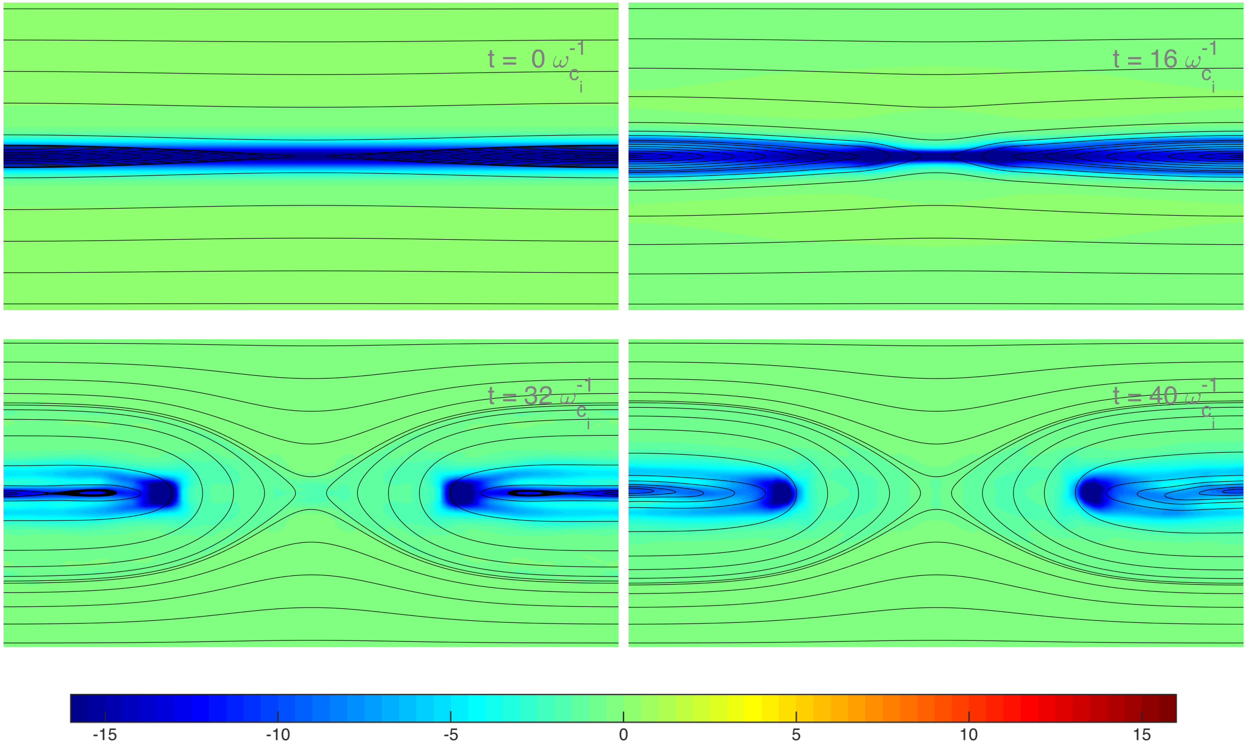

Figure 8.5: Composite presentation of current density out of plane (j_z) along with magnetic field streamlines for in-plane B_x and B_y. At the initial condition, the applied magnetic field perturbation is noticeable as necking in the field lines mid-plane while the plasma is uniform along the sheet. At time $t = 32\ \omega_{ci}^{-1}$, the separatrix and X-point are both clear features in the magnetic topology. In the bulk current density region there are actually two magnetic islands giving clearest indication of broken symmetry along the current sheet and across the X-point. The out-of-plane current density is much less uniform than the ion number density in Figure 8.1.4 might lead one to guess. At final time $t = 40\ \omega_{ci}^{-1}$, a slight asymmetry across the current sheet is also visible as a kink near the right domain edge.

assess the gross flux,

$$\theta(t) = \frac{\frac{1}{2}\int_{x=0} |B_x| \, dy}{L_y}. \tag{8.10}$$

Simulation results for this reconnected flux metric are plotted in Figure 8.6 along with other kinetic model simulation results for comparison.

8.1.6 Assessing Kinetic Qualities of the Solution PDF

For this GEM magnetic reconnection challenge problem and many other plasma physics investigations, it is a useful and relevant question to consider what level of model complexity is sufficient to capture the relevant physical processes. The kinetic simulation with three velocity dimensions is certainly more computationally expensive than a fluid model simulation of the same problem.

One of several possible ways to consider this question can be made after the kinetic solution has been computed as a post-processing step. With the full kinetic solution probability density function available at many simulated time points, an analysis is made to consider how much the pdf deviates from a Maxwell-Boltzmann distribution with the same fluid moments.

Other researchers have considered considered diagonal and off-diagonal pressure tensor components such as the Vlasov-Darwin simulation results presented in Ref. [57]. A ten-moment fluid model simulation presented in Ref. [53] demonstrates good qualitative agreement with the kinetic simulation reduced result. One question that cannot be directly considered in comparing these two results is how much specific qualities of the kinetic solution agree with the the reduced 10-moment distribution. A new metric is developed and described here to consider how well the kinetic solution agrees with a reduced moment model. For most straightforward explanation, the kinetic solution is compared to a corresponding five-moment fluid distribution though the method could readily be extended to higher-moment comparison.

The five-moment Maxwell-Boltzmann probability distribution function for species s with

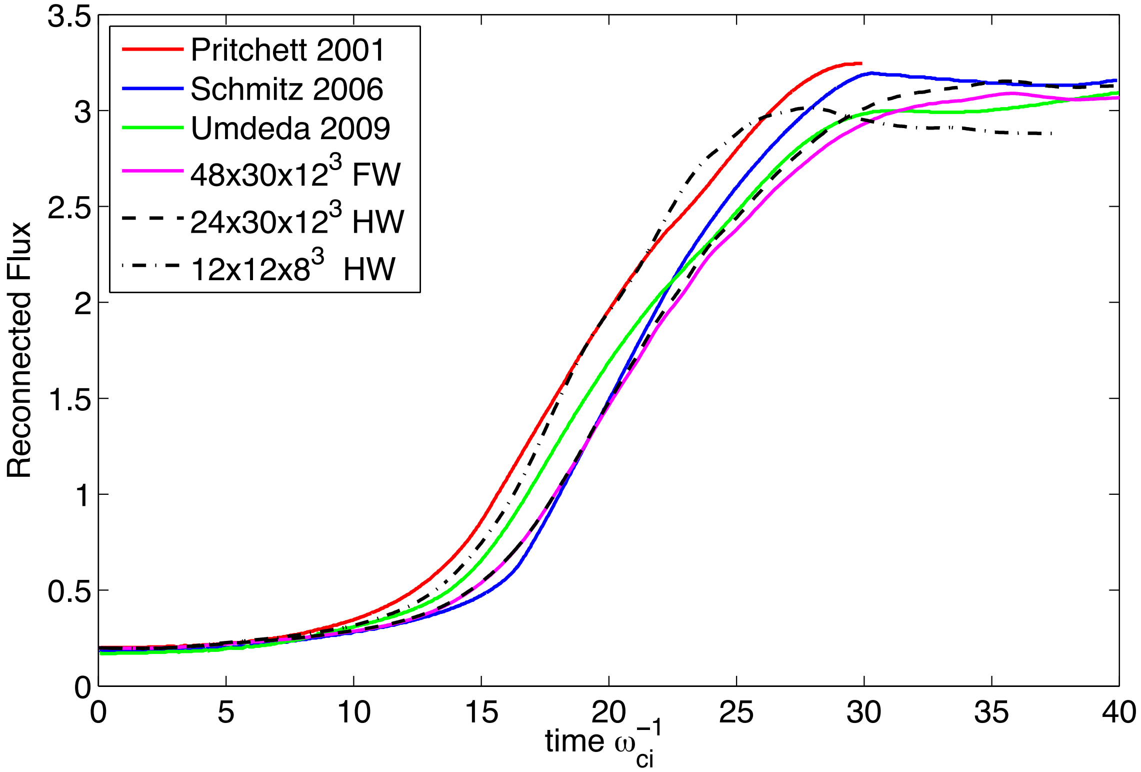

Figure 8.6: Normalized reconnected magnetic flux versus time per Eq. (8.10). Three WARPM Vlasov-Maxwell simulations are presented as well as three other published kinetic simulation model results [56, 57, 58]. Two WARPM simulations have comparable resolution; the one labeled HW or half-width implements a symmetry boundary condition along the current sheet mid-plane. A quite course simulation was also made with about half the resolution in all dimensions and also using the symmetry boundary condition. The fast reconnection rate and onset time are still well recovered.

particle mass m_s, number density n_s, mean velocity $\mathbf{u}_\mathrm{s}$, and scalar temperature T_s is expressed as,

$$f_M = \frac{n_\mathrm{s}}{\left(\sqrt{2\pi}\sqrt{\frac{k_B T_\mathrm{s}}{m_\mathrm{s}}}\right)^{\mathtt{NV}}} \exp\left(\frac{-(\mathbf{v}-\mathbf{u}_\mathrm{s})\cdot(\mathbf{v}-\mathbf{u}_\mathrm{s})}{2\frac{k_B T_\mathrm{s}}{m_\mathrm{s}}}\right), \tag{8.11}$$

where the number of velocity dimensions $\mathtt{NV} = 3$ for the case at hand. The temperature for a monatomic gas relates to average random particle velocity, or thermal velocity, as

$$k_B T_\mathrm{s} = \frac{\mathrm{NV}}{2} m_\mathrm{s} v_{\mathrm{Ts}}^2. \tag{8.12}$$

The necessary parameters can be evaluated by three moments integrals for any distribution as

$$n_\mathrm{s} = \int f_\mathrm{s} d\mathbf{v}, \tag{8.13}$$

$$\mathbf{u}_\mathrm{s} = \frac{\int \mathbf{v} f_\mathrm{s} d\mathbf{v}}{n_\mathrm{s}}, \tag{8.14}$$

and

$$U = U_x + U_y + U_z = \int \frac{m_\mathrm{s}}{2}(\mathbf{v}\cdot\mathbf{v}) f_\mathrm{s} d\mathbf{v}. \tag{8.15}$$

Total kinetic energy is the sum of thermal energy associated with random motion about the mean velocity vector $\mathbf{u}_\mathrm{s}$ and the fluid mean flow energy,

$$U = n_\mathrm{s} k_B T_\mathrm{s} + n_\mathrm{s} \frac{1}{2} m_\mathrm{s} (\mathbf{u}_\mathrm{s}\cdot\mathbf{u}_\mathrm{s}). \tag{8.16}$$

For any distribution function f_s a corresponding Maxwell-Boltzmann distribution is determined through evaluation of the above integrals and algebraic solution of Eq. (8.16) for temperature.

To assess deviation from the Maxwell-Boltzmann distribution, 1-norm error integral is evaluated and normalized by number density,

$$\chi_{\rm s} = \frac{\int_{\Omega_v} |f_{\rm s} - f_M| \, d\mathbf{v}}{n_{\rm s}}, \tag{8.17}$$

where the domain of integration, Ω_v, is the truncated velocity space. The error integral is depicted in Figure 8.7. For the GEM magnetic reconnection challenge problem, the discretized velocity space is centered about the initial fluid velocity vector and so,

$$\Omega_v = [-V_{\tt max}, V_{\tt max}] \times [-V_{\tt max}, V_{\tt max}] \times [-V_{\tt max} + u_{z0}, V_{\tt max} + u_{z0}] \, . \tag{8.18}$$

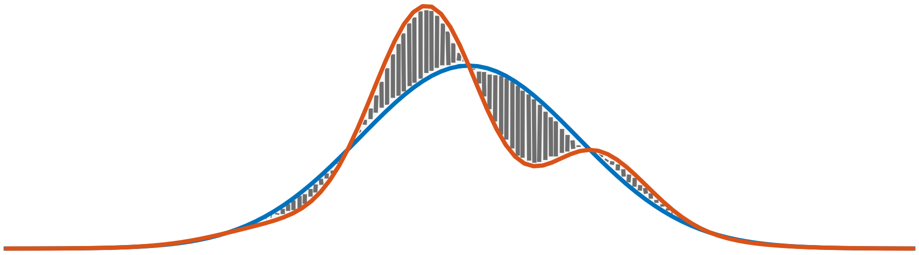

Figure 8.7: Deviation of a any distribution (red) from the Maxwell-Boltzmann distribution (blue) can be assessed by the 1-norm, or shaded area in this cartoon with one velocity dimension.

Figures 8.8 through 8.9 show the error χ_i and χ_e at select time points over the duration of the magnetic reconnection event. Even the initial condition for protons is significantly non-Maxwellian around the edges of the current sheet. This is due to the initial background population with 20% of the current sheet peak number density, but no mean flow velocity. The initial condition is thus a sum of two Maxwell-Boltzmann distributions with different mean velocity centers.

Once particular physical locations are identified for further investigation, another analysis can be made by taking slices of the distribution function at that location reducing the five-dimensional pdf to a three-dimensional slice in (v_x, v_y, v_z). A Maxwell-Boltzmann distribution iso-volume would present as a sphere in this analysis. For examples, slices of both species' distributions are made at two points as labeled in Figure 8.10. The point labeled 'A' is the X-point and is dead-center in the domain; pdf slices are presented in Figure 8.11. The point labeled 'E' is initially outside of the current sheet and the separatrix but ends up inside the separatrix over the course of the reconnection event; pdf slices are presented in Figure 8.12. To aid the reader in visually understanding the pdf structure, two iso-volumes are rendered for each slice. The innermost is adjusted to encompass 25% of the number density at that location while the outer iso-volume encompasses 75% of the number density. Both species' pdf in Figure 8.11 have complicated structures that are neither spherical nor oblate spheroids which can be characterized by a 5-moment or 13-moment fluid model. Each has some aspect of counter-streaming populations across the current sheet. In Figure 8.12, the electron species is most kinetic with a multi-humped distribution and significantly different temperature along the magnetic field sheet than transverse to it.

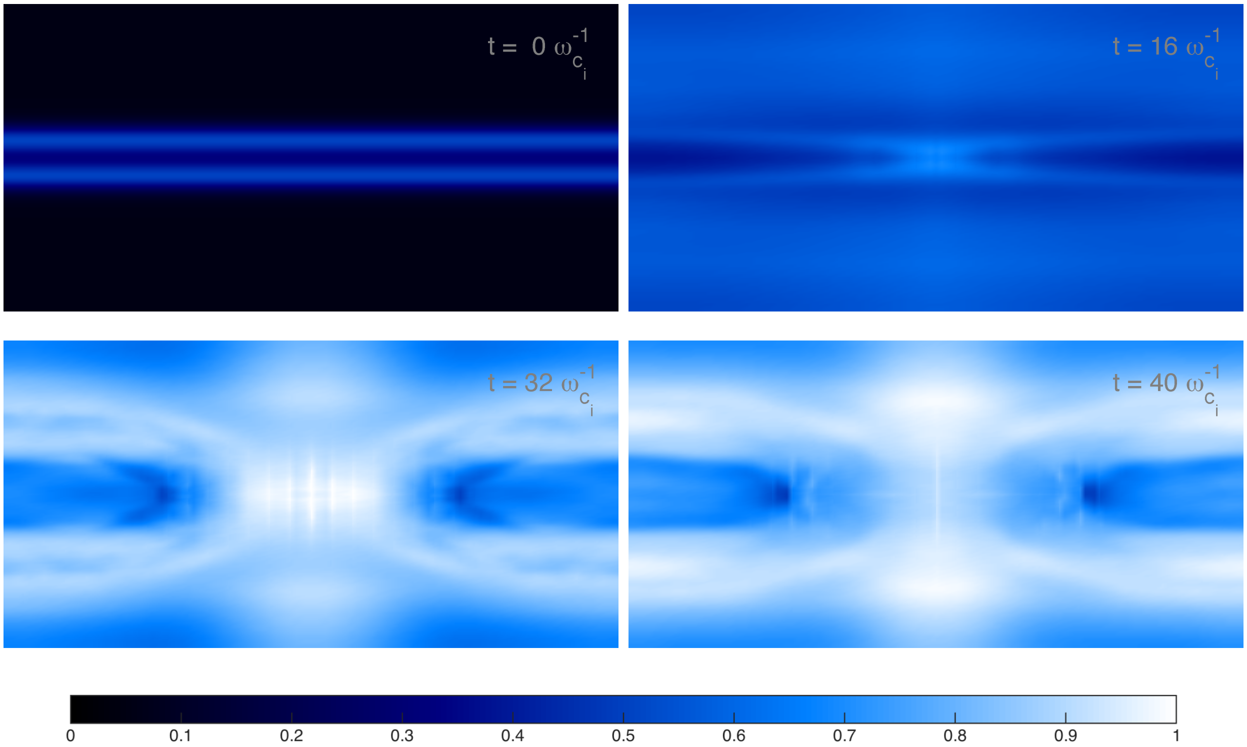

Figure 8.8: Normalized Maxwell-Boltzmann error for ions per Eq. (8.17) at the times indicated. Over the course of the reconnection event, the hotter ion species' distribution deviates most from Maxwellian in the diffusion region and in the low density region along and outside the magnetic separatrix.

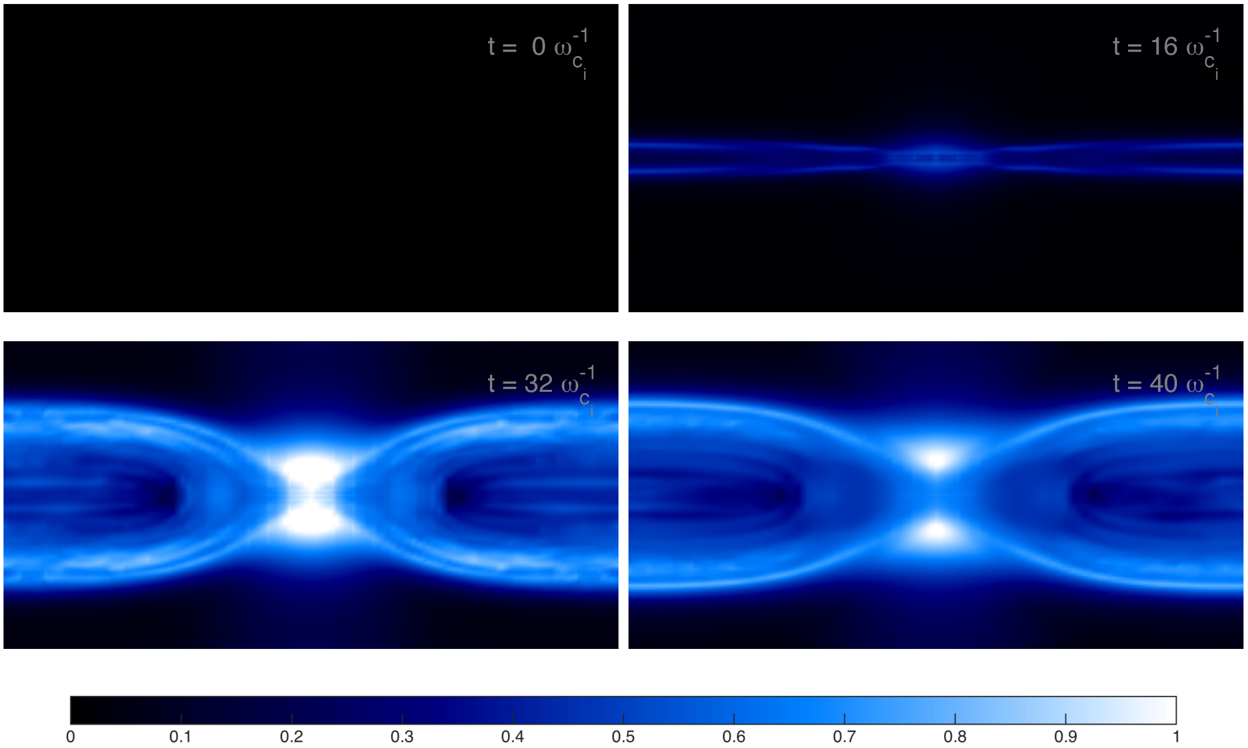

Figure 8.9: Normalized Maxwell-Boltzmann error for electrons per Eq. (8.17) at the times indicated. Over the course of the reconnection event, the colder electron species' distribution deviates most from Maxwellian in the diffusion region and along the magnetic separatrix.

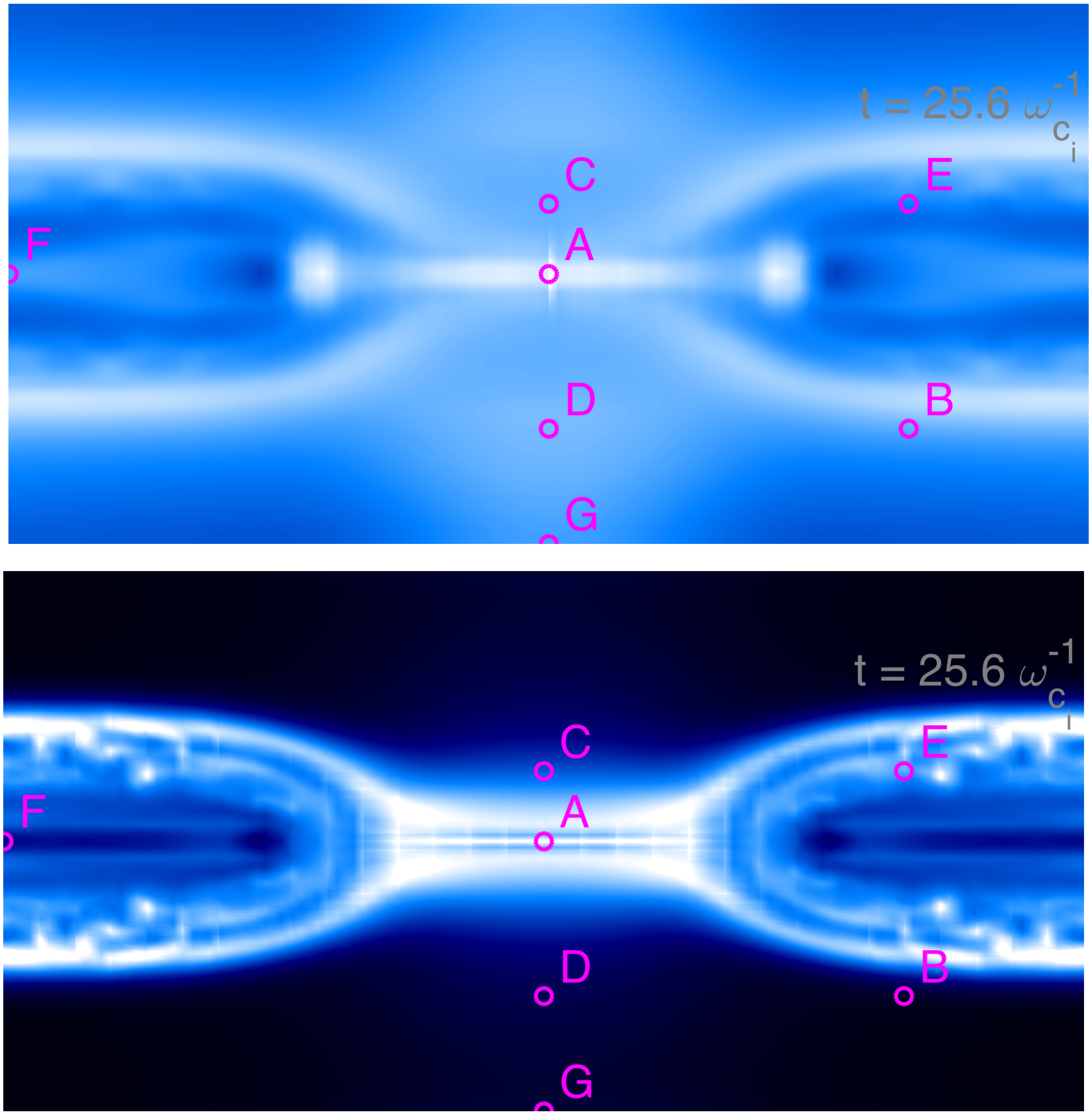

Figure 8.10: Physical points at which the probability distribution functions have been sliced and analyzed in three-dimensional velocity space. Point labels overlay the Normalized Maxwell-Boltzmann error per Eq. (8.17) for ion species (top) and electron species (bottom) at the time $t = 25.6\ \omega_{ci}^{-1}$.

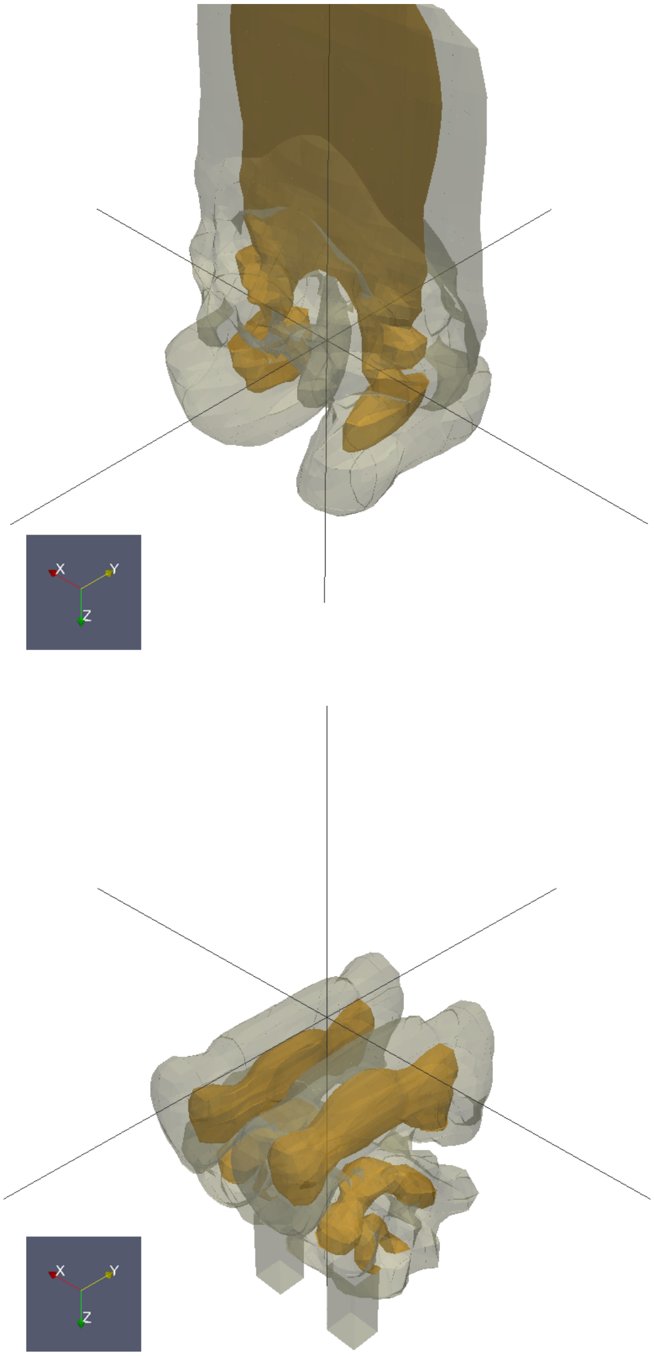

Figure 8.11: Probability distribution function for ions (top) and electrons (bottom) at slice point ‘A’ in Figure 8.1.6 at time $t = 25.6\ \omega_{ci}^{-1}$. Isovolumes indicate 0.75 and 0.25 fractions of the specie’s number density at the slice point. The crosshairs are centered at the fluid velocity moment.

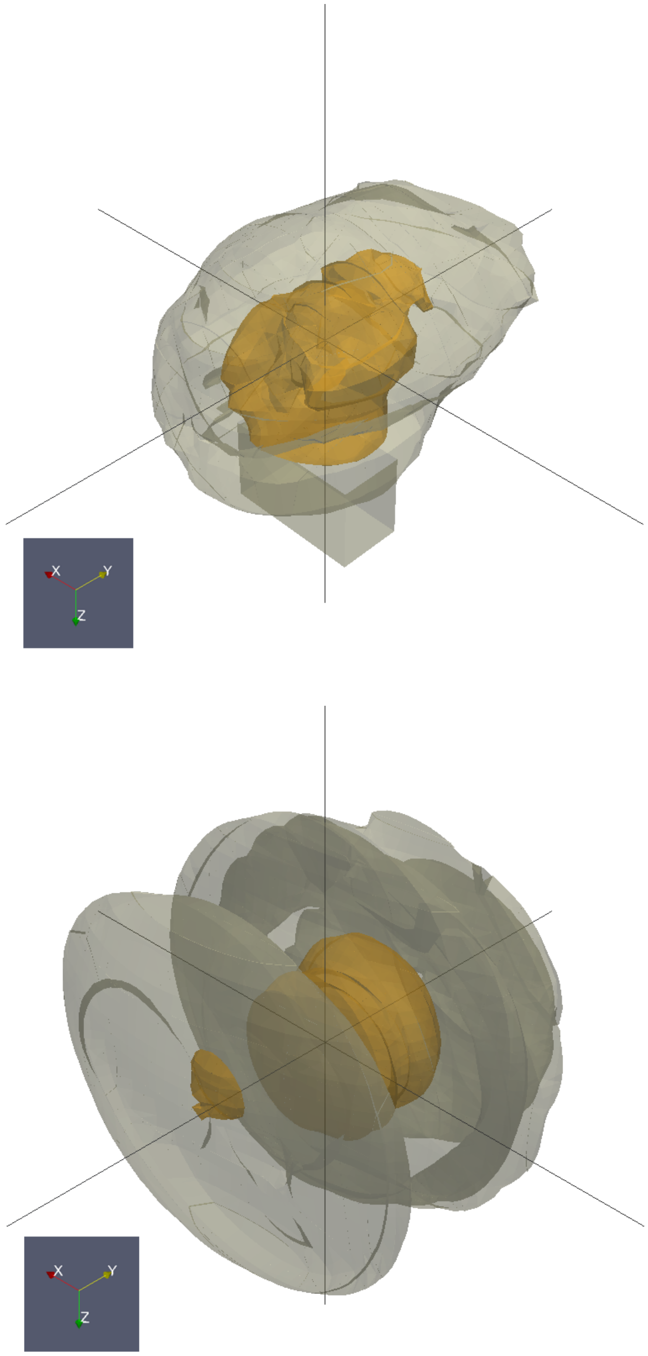

Figure 8.12: Probability distribution function for ions (top) and electrons (bottom) at slice point ‘E’ in Figure 8.1.6 at time $t = 25.6\ \omega_{ci}^{-1}$. Isovolumes indicate 0.75 and 0.25 fractions of the specie’s number density at the slice point. The crosshairs are centered at the fluid velocity moment.

Chapter 9
WARPM COMPUTATIONAL PERFORMANCE

WARPM was designed to compute solutions of large kinetic simulations on modern supercomputers and emerging many-core architectures. A suite of scaling studies and single-node performance sampling demonstrate observed performance on one supercomputer, the Cray XC30 *Edison* operated by the National Energy Research Scientific Computing Center (NERSC). Technical characteristics of *Edison* are summarized in Table 9.1.

A simple two-dimensional square physical domain with perfectly conducting solid walls is simulated as depicted in Figure 9.1. Multiple physical models are simulated on the same domain with two different DG element basis function polynomial orders. Two kinetic models both include two species, protons and electrons, with an artificial mass ratio of 25. The model labeled as "2D+2V" includes two velocity dimensions in phase space, so x, y, v_x, v_y, and 8^2 velocity space elements per physical element. The model labeled as "2D+3V" includes all three velocity dimensions in phase space, so x, y, v_x, v_y, v_z, and 8^3 velocity space elements per physical element. This discretization of phase space is coarser than other simulated problems of physical interest in this dissertation, but sufficient to demonstrate the relevant scaling effects. The weak scaling studies also consider relatively small simulations of Maxwell's equations in a planar vacuum labeled as "Maxwell-only".

Two different element polynomial reconstructions are considered. For results labeled "P=2", the computed solution is a piecewise quadratic polynomial projection in each dimension. For those labeled "P=3", it is piecewise cubic in each dimension. The number of nodal values per element is $(P+1)^{N_{\text{dims}}}$ and thus the two sets of computations represent very different levels of work in terms of both floating point operations and data movement.

The square domain allows WARPM to subdivide each process's assigned elements into

Total Compute Nodes	5576
CPU	Two-socket 12-core Intel *Ivy Bridge* processors at 2.4 GHz.
Threads	SMT Hyperthreads with 48 active threads per node.
Floating Point Units	256 bits wide vector unit. (Four double-precision) 19.2 GFLOPS/core, 460.8 GFLOPS/node peak.
Memory	64 GB DDR3 1866 MHz memory per node.
Cache	Each core has its own L1 and L2 caches, with 64 KB (32 KB instruction cache, 32 KB data) and 256 KB, respectively; A 30 MB L3 cache shared between 12 cores on the *Ivy Bridge* processor Cache bandwidth per core: L1/L2/L3 = 100/40/23 GB/s
Node Operating System	Cray Linux Environment 5.2
Interconnect	Cray Aries Interconnect with Dragonfly topology with approximately 8 GB/s bandwidth per node.
Working Storage	Cray Sonexion 1600 Lustre file system.
Installed	Summer 2013

Table 9.1: NERSC *Edison* Configuration

interior and exterior patches. The benefit of overlapping MPI communication of ghost element data with interior patch computation is clear in both the strong scaling study and weak scaling studies.

In addition to parallel scaling performance, the reported simulations also demonstrate WARPM's capability to compute very large simulations. The largest simulation reported contained 620 billion degrees of freedom advanced in time over 302 million DG elements.

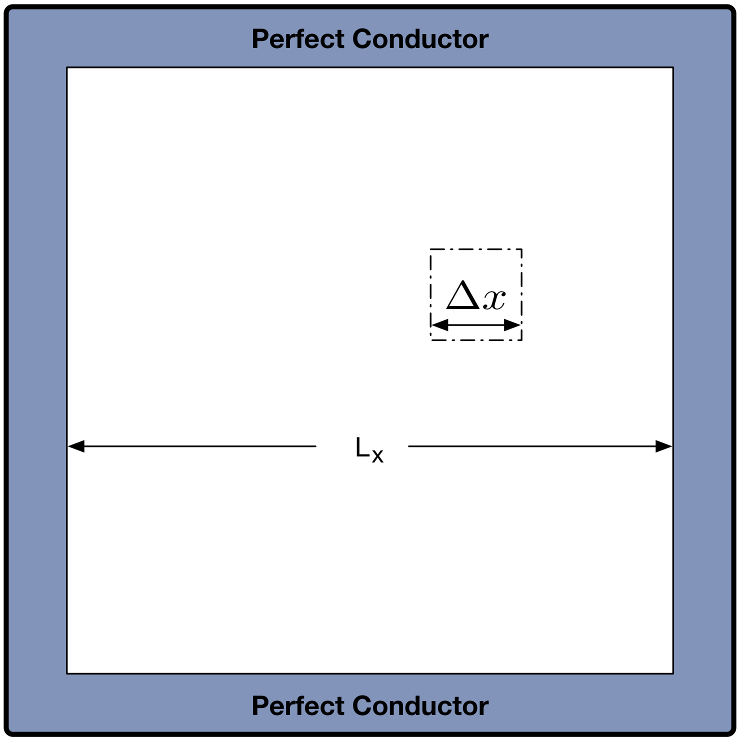

Figure 9.1: Representative problem geometry for parallel scaling performance studies. In all cases the simulated domain is square and surrounded by perfectly conducting solid walls with uniform element size, Δx. The domain size L_x is constant for the strong scaling problems. For weak scaling problems, the domain area increases linearly with number of processors, $L_x \propto \sqrt{N_{\text{nodes}}}$.

9.1 *Weak Scaling*

To assess the weak scaling performance of WARPM, a constant tile size is maintained per compute node (one MPI process element) while the number of nodes is increased. The degrees of freedom, domain area, and computational work all increase linearly with number of nodes.

Two different tile sizes are chosen purposefully to examine the impact of the tile surface area to volume ratio or more precisely the ratio between the number of elements adjacent to the tile boundary and the total number of tile elements. Only elements on the tile exterior require ghost value communication using MPI message passing and the machine high-speed interconnect while all tile elements have equal computational cost in floating point operations. As described in Section 4.4, WARPM is designed for good weak scaling performance by overlapping ghost communication with continued computation of tile interior elements.

The two tile sizes studied are 3×3 and 16×16 with exterior element to interior element ratios of $8 : 1$ and $60 : 196$, respectively. Comparing Figures 9.2 and 9.3, it is clear that better weak-scaling efficiency is maintained at large scale in the 16×16 tile computations. The data also demonstrates that the more complex models with five phase-space dimensions maintain better scaling at the largest problem size than the four-dimensional models. This is expected as more computational work is required relative to the ghost communication data size.

In Figures 9.2 and 9.3 parallel scaling efficiency is calculated relative to the wall time required to advance one time step on a single tile on one node,

$$\eta = \frac{t_0}{t_{\text{step}}}. \tag{9.1}$$

Between six and ten time steps are averaged depending on the physical model. The ideal efficiency is unity representing no incurred overhead due to ghost element communication or other factors.

The largest calculations utilized nearly one-half of the machine *Edison*. Machine allocations of this size are difficult to schedule and costly and so the data presented represent single-run observations rather than averaged performance and no attempt is made to assess performance variance across one machine allocation to the next. It is understood that specifics of the machine allocation can have an impact on the repeatability of the results presented. The machine's Aries high-speed network interconnect has a three-tier topology [59] and so both the set of nodes allocated and MPI rank ordering among the nodes allocated can have a significant impact on the effective nearest-neighbor ghost element communication bandwidth and latency. In all studies presented, a custom MPI rank reordering was performed in attempt to maximize communication performance considering the specific simulated domain decomposition and Cray Aries network topology. WARPM runs best in the configuration of one MPI rank per node, with thread parallelism utilizing all cores on the node with simultaneous multi-threading (SMT) (two hyperthread per core) enabled.

The result of this study confirms the capability of WARPM to simulate very large kinetic problems efficiently and gives guidance to choosing the right balance of node count versus net simulation wall time.

9.2 Strong Scaling

Often a problem of interest has a particular domain decomposition and computational size based on physical scales and accuracy requirements. A question becomes how fast can we efficiently compute the simulation to arrive at desired answer. This is a strong scaling study where the problem size is fixed and more computational resource is applied.

Strong scaling experiments for multiple physical models were performed on a fixed domain decomposed into 48×48 elements and the results are summarized in Figure 9.4. We see very good speedup for only the largest tile sizes and significant roll-off in efficiency starting at 6×6 tile size. Similar to the indications in the weak scaling studies, the larger models with five phase-space dimensions maintain better scaling efficiency than those with four at smaller tile sizes.

The strong scaling experiments highlight the importance of overlapped ghost data communication with element time advance computation. Several potential code changes should be considered in more detail to further improve this design feature of WARPM:

- The current implementation transfers every nodal value for each ghost element during

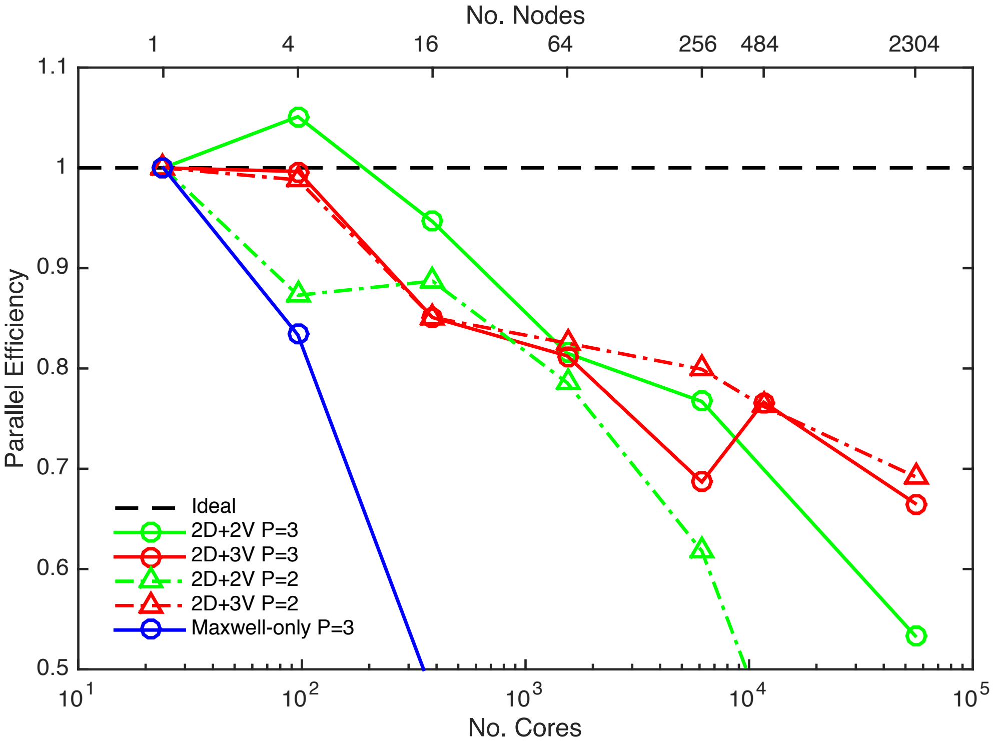

Figure 9.2: Weak scaling experiment results with tile of 3 × 3 elements per compute node with 8 elements in each velocity dimension.

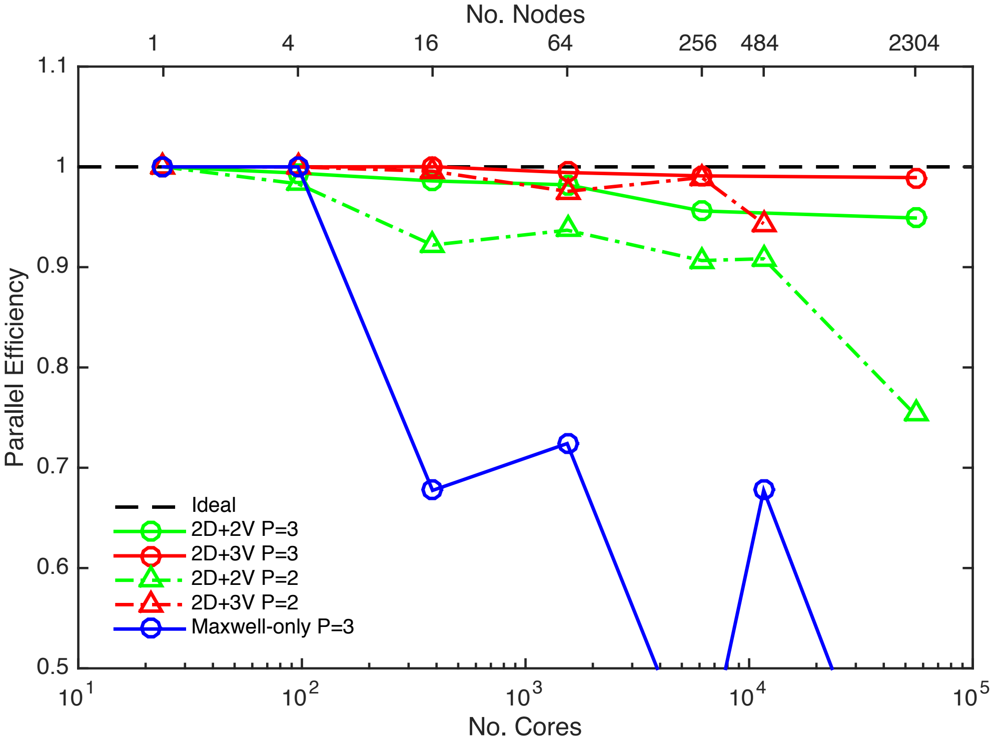

Figure 9.3: Weak scaling experiment results with tile of 16 × 16 per compute node with 8 elements in each velocity dimension.

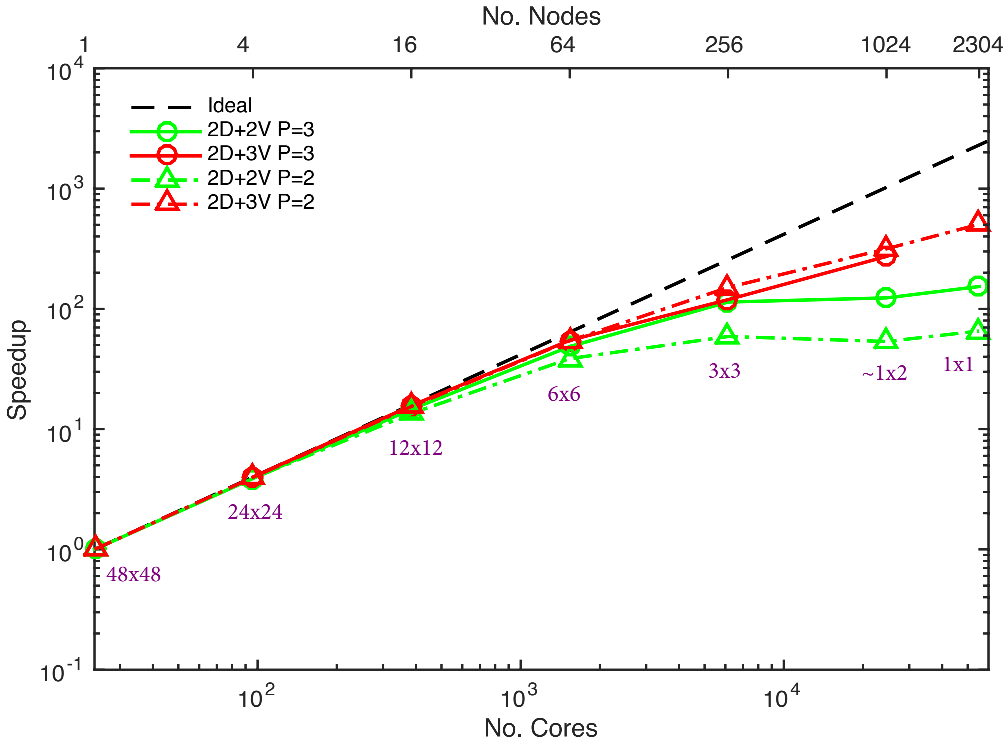

Figure 9.4: Strong scaling performance for problems based on 48 × 48 rectilinear mesh elements in physical space and 8 elements in each velocity dimension. Element polynomial order and number of velocity dimensions in the Maxwell-Vlasov model are indicated by the legend. Speedup is defined as the reduction in wall-time to compute one unit of time advance relative to single-node time. The purple annotations indicate the domain decomposition tile size on each node.

the ghost synchronization while the existing numerical scheme only utilizes the surface node values in computing the surface numerical flux. Transferring only surface nodal values would reduce the data volume by a factor of $(P + 1)$. Ghost element synchronization for the kinetic models are limited by interconnect bandwidth so this optimization would likely be significant.

- The current implementation does not commence ghost element synchronization in phase space until all velocity space elements are completed. This reduces the number of messages and increases the contiguous message size but delays the commencement of communication. An alternative implementation to consider would commence communication as soon as a given element face is ready rather than wait for all velocity space elements.

- A different time discretization scheme might be used making better use of the full ghost element solution such as a one-step ADER method with high-order temporal reconstruction of numerical flux [41].

- Taking advantage of the one-directional nature of the Vlasov equation's advection flux across physical faces, a purely upwind numerical flux can be fully specified with one element nodal value alone. Through specialized treatment considering the normal velocity coordinate, this numerical flux scheme could reduce the volume of data synchronized by a factor of two.

9.3 Single Node Performance

Computational performance of a fluid dynamics domain decomposition code like WARPM relies fundamentally on achieving best single-node performance that maximally utilizes the floating-point capability of the hardware to implement the numerical method. Good weak and strong scaling efficiency can still be achieved with poor single-node performance but the cheapest and sometimes most surprising performance gains can be achieved by understanding the hardware bottlenecks and making implementation adjustments to ease them.

Core performance sampling tools enable powerful insight into the inner-workings of the compute core and memory system during application execution. They utilize hardware performance counters incorporated in the core circuitry intended to count a variety of core architecture events during the otherwise normal execution of a user application. The tool briefly interrupts the application execution periodically to observe the region of program instructions being execution and to save this along with the performance counter snapshot for later analysis.

WARPM was studied with the Intel VTune Amplifier performance sampling tool on the same NERSC *Edison* compute nodes used in the scaling studies. Tests only considered the Vlasov-Maxwell model problems and only 16×16 physical elements on the node. The simulated problem setup was the same as in the scaling studies. Sampling was enabled over the period of the second of two time advance frames in order to allow a warm-up before collection. For the $P = 2$ models, five time steps with four Runge-Kutta sub-steps are sampled; for the $P = 3$ models, the advance is ten time steps.

Important problem size attributes, time step performance, and hardware counter observations are summarized in Table 9.2 with discussion and analysis for each item as follows. For each of the sampling experiments, over 90% of instructions are associated with the single combined OpenCL kernel evaluating internal flux, numerical (face-normal) flux, and kinetic moment integration leaving the remaining instructions mostly to the boundary condition application and 2$^{\text{nd}}$-stage moment integration kernels. The metrics can thus be largely considered specific to the main OpenCL compute kernel.

	$(2D+2V)$		$(2D+3V)$	
Polynomial Order per dim.:	$P=2$	$P=3$	$P=2$	$P=3$
Nodes per element	81	256	243	1024
Face nodes per element	216	512	810	2560
PDF Solution Storage (MB)	21.2	67.1	509.6	2147.4
Wall time per node-step (s)	2.96×10^{-8}	2.64×10^{-8}	3.58×10^{-8}	3.19×10^{-8}
Wall time per face node-step (s)	1.11×10^{-8}	1.32×10^{-8}	1.07×10^{-8}	1.28×10^{-8}
Instructions per node-step	5.06×10^{3}	4.81×10^{3}	6.84×10^{3}	6.21×10^{3}
CPU Utilization	93.0%	94.8%	98.5%	98.6%
Hardware GFLOPS	10.56	12.95	13.06	14.99
FPU Utilization	2.1%	2.5%	2.5%	2.8%
Memory Bound	12.5%	14.0%	13.1%	13.3%
L1 Bound	18.2%	20.0%	18.9%	19.4%
Core Bound	43.2%	41.9%	43.2%	42.8%
Front-End Bound	8.4%	8.1%	7.7%	7.0%
Back-End Bound	55.7%	55.9%	56.3%	56.1%

Table 9.2: Performance metrics collected from one two-socket 24-core Ivy Bridge node of *Edison* at NERSC. Domain geometry is 16 × 16 element tile in all cases, and 8 elements per velocity dimension. Hardware performance metrics were collected by the Intel VTune Amplifier 2016 performance sampling tool.

Element node count: It is informative to note the ratio of face nodes to element nodes. Face nodes are simply element nodes which additionally require a numerical face-normal flux evaluation in conjunction with the adjacent element. An element node on an edge or corner can be multiple face nodes. The ratio increases with increasing number of phase dimensions. Higher-order polynomial elements reduce the ratio of communication to floating point operations.

Probability distribution function (PDF) storage: The memory footprint of the two-species pdf for each problem size is indicated. It is important to observe that the footprint in all but the smallest case exceeds the L3 cache size. The DG consolidated kernel design was intended to reuse data as much as possible while computing the time advance for one element in order to maximize cache reuse.

Wall time per node-step: This is computed as the average time to advance one Runge-Kutta sub-step divided by the total number of nodes. The time is surprisingly flat for each test case given that higher polynomial order and higher dimensions both require more floating point operations in the numerical method. This is the first of several metrics listed that suggests a performance bottleneck unrelated to floating point performance.

Wall time per face node-step: This is computed as the average time to advance one Runge-Kutta sub-step divided by the total number of face nodes. Remarks are same as for the above.

Instructions per node-step: This is computed dividing the hardware counters for instructions retired by all cores for one Runge-Kutta sub-step divided by the total number of nodes. This metric actually decreases for the higher polynomial order cases even though more floating point operations required by the numerical method.

CPU Utilization: This metric reports the fraction of time in which all CPU cores are occupied with work over the observation period and is an assessment of how well threaded the application is versus serial code. It does not assess how effectively the CPU resources are utilized, only that it is occupied by a thread.

GFLOPS: The number of floating point operations tallied by the hardware counters divided by the observation period. The test hardware provides 460.8 GFLOPS ideal peak, and thus the observed performance for all cases falls far below the processor peak capabilities.

FPU Utilization: The fraction of observed GFLOPS versus ideal peak.

Memory Bound: This composite metric measures a fraction of clocks where the core functional pipelines are stalled due to load or store instructions. It shows how memory subsystem issues affect the performance and accounts mainly for incomplete memory loads that coincide with execution starvation. This fraction is surprisingly low for all tests cases in that it was anticipated that the numerical method would be memory bandwidth bound.

L1 Bound: The core L1 cache has the shortest latency in general but in certain cases like loads blocked on older stores to the cache, a load might suffer a high latency even though it is being satisfied by the L1. The observations are nominal and consistent for each test case.

Core Bound: This composite metric represents how much core non-memory issues were a bottleneck. It can indicate the core ran out of out-of-order resources, certain execution units are overloaded, or dependencies for long-latency arithmetic operations are limiting performance. Combined with the low FPU utilization result, this high indication of on-core bottleneck unrelated to memory bandwidth suggests the core functional units are overwhelmed by integer, logical, and pointer operations such array index calculations, masks, and branch conditions.

Front-End Bound: The Intel Ivy-Bridge (Xeon) processor used in this study can be conceptually divided into the ‘front-end’, where instructions are fetched and decoded into the micro operations that constitute them, and the ‘back-end’, where the operations are carried out by a limited number of pipelines. Each clock cycle, the front-end generates up to four of these micro operations placed into pipeline slots and moved to the back-end. With this metric quite low for all test cases, instruction decoding and branch mis-prediction are not leading the delays that starve the back-end pipelines of work.

Back-End Bound: This metric identifies the fraction of time in which new micro-operations cannot be issued to a processing pipeline due to a lack of required resources –

pipelines with the needed functional units are all full. A pipeline can stall due to data-cache misses which by a separate metric is observed to be the case only about half of the time. Long-latency operations or other contention for execution resources is understood to make up the remaining delay with particular focus on non-floating point operations.

9.3.1 Reflection on observations

The combined observations of very low floating point unit utilization and low fraction of memory bound execution reveals performance problems in the kinetic compute kernel as implemented and opportunity for significant improvement.

Inspection of the assembly code produced by the Intel OpenCL compiler for the kernels studied revealed that only 128-bit vector instructions are being generated even though the studied processor is capable of 256-bit vector operations (e.g. four double-precision floating point multiplies or adds simultaneously). Further experimentation with very simple kernels revealed that the compiler does not vectorize even trivial loop operations beyond 128-bit unless the OpenCL vector intrinsic types are explicitly specified. (i.e. double4, double8, etc.) This is not a natural way to express the DG algorithm, but could be done with some effort in order to better utilize the hardware capabilities. Newer processors with 512-bit vector widths make this an important aspect to consider in any re-design work for the compute kernels.

The high back-end utilization without delivering floating-point performance suggests the kernel implementation is too complex to utilize the hardware effectively. The experience suggests a new implementation splitting the combined compute kernel into multiple kernels with more specialized function and simpler code expression could significantly improve the on-core performance. As mentioned previously, there are additional scaling and physical model reasons motivating adoption of a distinct kernel for numerical flux evaluation.

In considering options to sequence multiple kernels, two possibilities have been considered. Compact groups of elements could be processed through all compute stages in entirety by the same compute unit. This is closest to the current implementation. Or, a compute stage could be applied over the entire patch before moving on to the next compute stage. There are two hardware trends to be considered in making such a design choice. First, latest GPU hardware and many-core CPU roadmaps do not have an L3 cache. Second, the number of simultaneous threads in the hundreds or thousands means a lot of parallel work must be exposed. Threading in units of physical elements may not be sufficient to assign work to all cores. Different parts of the DG algorithm and moment integration in the Vlasov Maxwell model would be best decomposed for threads in different ways. This all suggests that the simpler sub-stage kernels and sequencing applying to whole patches would ultimately perform better.

Chapter 10
CONCLUSION

10.1 Contributions

10.1.1 WARPM code developed

The open-source research code WARPM was developed to achieve a new level of computational scale supporting advanced kinetic plasma simulations. WARPM was based on the principle flexibility and user interface of its predecessor WARPX [44]. Both codes are meant to support a variety of physical models, domain geometries, numerical methods.

WARPM is based on a new computational kernel and memory buffer management model targeting new and emerging supercomputing architectures based on GPU accelerator and many-core CPU with multi-level memory hierarchy.

During scaling studies, WARPM computed the highest number of degrees of freedom for any Vlasov simulation to date [60, 58, 57]. Through the course of this thesis WARPM was used to evaluate both fluid and kinetic models, DG and finite volume methods, and geometries from one to six dimensions.

Run-time flexibility with compile-time optimization benefits to DG numerical method

Flexibility for the user to specify physical model, domain geometry, and numerical method is maintained with improved compiler performance, vectorization, and thread parallelism. The user parameterizes existing kernel templates and chains together input and output variables before the final OpenCL source code is compiled at run-time just before the start of simulation.

WARPM additionally adds support for task-level parallelism that the user can express in terms of startup, per-step, finishing, and restart graphs. This is one way the data communication and file I/O can now be overlapped with computation.

New CFD code architecture designed to maximize performance on both GPU accelerated supercomputers and future 'many-core' machines

WARPM implements computational kernels and memory buffer models using the OpenCL computing language and API. Data communication between nodes is through the Message-Passing-Interface (MPI) standard. These two programming models are supported by every major supercomputing architecture currently and in the near future. The same computational kernels compile for both GPU and CPU.

10.1.2 Discontinuous Galerkin implementation in four, five, and six-dimensional phase space

The nodal Discontinuous Galerkin method based on hyper-cubes and tensor-product orthonormal Legendre polynomial basis functions was developed and implemented in a way such that the number of dimensions could be varied from one to six. Thus, full $3D + 3V$ phase space can be discretized with high order finite elements.

The tensor-product basis set is not optimal in terms of minimizing degrees of freedom for a given error convergence rate. However, a surprising property was discovered that reduces volume integral and surface integral operations to an effective line integral in the direction normal to each face. This enables a dramatic reduction in the number of floating point operations required and the storage space needed for the constant DG operator matrices. The reduction scales geometrically with the number of dimensions making it especially beneficial for Vlasov model simulations.

The DG kernel implements an algorithmic way to compute internal and external flux integrals supporting a variety of domain and mesh types at different number of dimensions. Implementation of the geometric Jacobian is also custom to each mesh type such that optimizations can be made when the matrix is not fully general. For example, the rectilinear velocity space Jacobian is understood to be diagonal and computed by evaluating expressions of the element indices rather than reading mesh vertex coordinates from memory.

10.1.3 Boundary condition developments for V_{max}

Vlasov model simulations in a discrete phase space introduce complex boundary condition requirements along both physical boundaries and at the extrema of the truncated velocity domain, $V_{\mathtt{max}}$.

The implemented $V_{\mathtt{max}}$ boundary condition is conservative for mass and gives qualitatively better numerical stability than solid wall (i.e. zero flux).

The solid wall boundary condition is conservative and implemented in such a way that a more sophisticated wall interaction could be computed such as scattering, charge exchange, or knock-off interactions.

10.1.4 Component contributions to numerical error in DG solutions

To support analyzing the overall solution accuracy and moment conservation properties of discontinuous Galerkin method, first the different contributions to numerical error in the DG method were studied.

Highly accurate evaluations of the DG operator matrices were developed. This helped to lessen asymmetries in the surface and volume flux numerical integration, improving equilibrium solution quality, and reducing one source of conservation error.

10.1.5 Highest resolution kinetic simulation of the GEM Challenge magnetic reconnection

The Geospace Environmental Modeling (GEM) reconnection challenge problem was simulated for the first time combining the Vlasov-Maxwell model and discontinuous Galerkin numerical method. The combination of element count, concentrated resolution in vicinity of the current sheet, and high-order polynomial elements results in the highest resolution magnetic reconnection simulation in both physical space and velocity space [60, 58, 57]. The magnetic reconnection rate predicted agrees well with other published kinetic models and numerical methods simulating the same problem. This simulation result served to validate the model and numerical method implementation in $2D + 3V$ dimensions.

The improved physical and velocity space resolution enabled by the WARPM design and effective utilization of more powerful computational resources revealed new qualities of this well studied problem.

Both the ion and electron species were shown to qualitatively and quantitatively deviate from thermal Maxwell-Boltzmann distribution and even probability distributions captured by extended moment fluid models.

A new post-processing analysis for assessing the probability distribution function deviation from an associated Maxwell-Boltzmann distribution was introduced. Utilizing the kinetic solution moments of number density, momentum density, and kinetic energy density, a thermal Maxwell-Boltzmann distribution is parameterized to have matching moments. Then, a 1-norm of the difference between the two distributions over velocity space is computed at each point in the physical plane and normalized to number density. The result proved to be an effective metric to assess and discover solution regions that are more 'kinetic' than others.

Guided by the new metric, slices of the pdf solution were collected at choice physical locations resulting in samples of the pdf in three velocity space dimensions. Computed and rendered iso-volumes revealed pdf structures that were indeed far from Maxwellian and even higher-moment fluid distributions.

10.2 Suggestions for Future Study

With promising results, there are clear and exciting opportunities to continue this line of research. A few of them are suggested below out of recent memory and collective experience working with the numerical method, computational implementation, and physical model. A perspective motivating all of the following suggestions is that Vlasov-Maxwell simulations of full $3D + 3V$ problems is nearing reasonable possibility with trends in supercomputing. An approach designed for $1D + 1V$ or $2D + 2V$ problem can be quite different than $3D + 3V$.

In design trade-off decisions, the later should be favored.

10.2.1 Split DG kernel into multiple parts

The current WARPM implementation assembles multiple component parts into a single numerical OpenCL kernel implementing a Runge-Kutta time step. The original idea was that input components, i.e. nodal solution values, could be reused repeatedly in lowest level cache reducing demands on global memory bandwidth. The values are used for evaluating the source terms, internal flux, numerical (surface) flux, and velocity moment integrands. Experience has revealed that the resulting *monolithic* compute kernel is complicated, difficult to maintain, and problematic for the GPU compilers. More importantly, with DG element exceeding one thousand nodes, the techniques to maintain data locality and cache presence must be different.

It is planned to implement separate kernel stages for each of the following:

- Surface flux evaluation kernel
- Runge-Kutta time integration
- Discontinuous Galerkin right-hand-side evaluation
- Velocity moment integration

Separating the surface flux evaluation from the DG kernel also opens up more flexibility to impose Von Neumann boundary conditions which impose a specified flux value directly. Currently, boundary condition flux has to be indirectly achieved through applied Dirichlet values and knowledge of the numerical flux method. This will open more options for the $V_{\mathtt{max}}$ boundary physics and wall boundaries.

10.2.2 Initiate inter-zone communication as a wave

The WARPM domain decomposition and patching algorithm was tailored for Vlasov model calculations in phase space by only patching in the physical dimensions. In doing so, it is assured that velocity space storage is contiguous and local at any physical dimension. This greatly simplifies the efforts required to integrate the pdf velocity moments needed evaluate current density as a source term to Maxwell's equations.

The current implementation does not initiate ghost or halo communication until all patch periphery elements have been updated. No halo exchange is required across velocity faces, but very large amounts of data must be transferred across physical faces at patch boundaries. In the limiting case of $3D + 3V$, data volume on the order of 100MB must be exchanged every step per element physical face.

Whether considering the inter-node network or the PCI bus between GPU and host, this data volume results in significant delay limiting strong scaling performance of WARPM as more and more interior element computation is required to mask the halo exchange time.

This halo exchange algorithm could be modified in conjunction with the patch processor element sequence to start halo communication sooner as smaller chunks complete. This would result in more smaller messages exchanged over the full duration of the time step. Strong scaling performance should improve.

10.2.3 Adopt communication pattern tailored for Vlasov-Maxwell upwind flux

Memory bandwidth at all levels limits the computational performance of WARPM and the Vlasov-Maxwell model. A somewhat specialized halo exchange algorithm and numerical flux method could be adopted considering that Vlasov-Maxwell system flux in all dimensions is purely upwind. Across physical faces, the flux direction is always determined by the velocity coordinate and across velocity faces the direction is determined by the time dependent Lorentz force.

A new halo exchange algorithm leveraging this property could reduce exchanged data volume by one half.

10.2.4 Implement collisions as a source term

Incorporating intra-species and inter-species collisions into the physical model would expand the range of model validity and open the possibility to study new and interesting problems. A Vlasov code should be more readily adaptable to incorporate a collision operator than a Particle-In-Cell method because the velocity space slice of a species' probability distribution functions at a particular position is compactly known and resolved with the same resolution everywhere. Different collision operators could be implemented balancing computational effort with the inner product integration and physical accuracy.

BIBLIOGRAPHY

[1] J P Freidberg. Ideal magnetohydrodynamics. January 1987.

[2] Uri Shumlak and J Loverich. Approximate Riemann solver for the two-fluid plasma model. *Journal of Computational Physics*, 187(2):620–638, May 2003.

[3] NVIDIA's Next Generation CUDA Compute Architecture: Kepler GK110. Technical report, NVIDIA Corporation, 2012.

[4] Intel. *Knights Corner: Your Path to Knights Landing*, September 2014.

[5] E G Harris. On a Plasma Sheath Separating Regions of Oppositely Directed Magnetic Field. *Nuovo Cimento*, 23(1):115–+, 1962.

[6] J Birn, J F Drake, M A Shay, B N Rogers, R E Denton, M Hesse, M Kuznetsova, Z W Ma, A Bhattacharjee, and A Otto. Geospace Environmental Modeling (GEM) magnetic reconnection challenge. *Journal of Geophysical Research: Space Physics (1978–2012)*, 106(A3):3715–3719, 2001.

[7] L Greengard and V Rokhlin. A Fast Algorithm for Particle Simulations. *Journal of Computational Physics*, 135(2):280–292, August 1997.

[8] B J Alder and T E Wainwright. Studies in Molecular Dynamics. I. General Method. *The Journal of Chemical Physics*, 31(2):459–466, August 1959.

[9] AGR Thomas, M Tzoufras, A P L Robinson, R J Kingham, C P Ridgers, M Sherlock, and A R Bell. A review of Vlasov-Fokker-Planck numerical modeling of inertial confinement fusion plasma. *Journal of Computational Physics*, 231(3):1051–1079, 2011.

[10] Ammar Hakim, J Loverich, and Uri Shumlak. A high resolution wave propagation scheme for ideal Two-Fluid plasma equations. *Journal of Computational Physics*, 219(1):418–442, November 2006.

[11] Ammar Hakim and Uri Shumlak. Two-fluid physics and field-reversed configurations. *Physics of Plasmas*, 14(5):055911, 2007.

[12] John Loverich, Ammar Hakim, and Uri Shumlak. A Discontinuous Galerkin Method for Ideal Two-Fluid Plasma Equations. *Communications in Computational Physics*, 9(2):240–268, February 2011.

[13] Yingda Cheng, Irene M Gamba, Fengyan Li, and Philip J Morrison. Discontinuous Galerkin Methods for the Vlasov-Maxwell Equations. *SIAM Journal on Numerical Analysis*, 52(2):1017–1049, April 2014.

[14] Yingda Cheng, Andrew J Christlieb, and Xinghui Zhong. Energy-conserving discontinuous Galerkin methods for the Vlasov–Ampère system. *Journal of Computational Physics*, 256(C):630–655, January 2014.

[15] Yingda Cheng, Irene M Gamba, Fengyan Li, and Philip J Morrison. Discontinuous Galerkin Methods for the Vlasov-Maxwell Equations. *arXiv.org*, February 2013.

[16] Yingda Cheng, Irene M Gamba, and Philip J Morrison. Study of conservation and recurrence of Runge–Kutta discontinuous Galerkin schemes for Vlasov–Poisson systems. *Journal of Scientific Computing*, 56(2):319–349, January 2013.

[17] David C Seal. *Discontinuous Galerkin methods for Vlasov models of plasma*. PhD thesis, University of Wisconsin - Madison, University of Wisconsin - Madison, May 2012.

[18] R E Heath, I M Gamba, Philip J Morrison, and C Michler. A discontinuous Galerkin method for the Vlasov–Poisson system. *Journal of Computational Physics*, 231(4):1140–1174, February 2012.

[19] John M Dawson. Particle simulation of plasmas. *Rev. Mod. Phys.*, 55(2):403–447, 1983.

[20] Charles K Birdsall and A Bruce Langdon. *Plasma physics via computer simulation.* McGraw-Hill, New York, NY, 1985.

[21] J A Rossmanith and David C Seal. A positivity-preserving high-order semi-Lagrangian discontinuous Galerkin scheme for the Vlasov-Poisson equations. *Journal of Computational Physics*, 230(16):6203–6232, July 2011.

[22] Jan S Hesthaven and Tim Warburton. *Nodal Discontinuous Galerkin Methods*, volume 54 of *Texts in Applied Mathematics*. Springer New York, New York, NY, 2008.

[23] Michael Dumbser and Martin Käser. Arbitrary high order non-oscillatory finite volume schemes on unstructured meshes for linear hyperbolic systems. *Journal of Computational Physics*, 221(2):693–723, February 2007.

[24] Vladimir A Titarev and Eleuterio F Toro. ADER schemes for three-dimensional non-linear hyperbolic systems. *Journal of Computational Physics*, 204(2):715–736, April 2005.

[25] Vladimir A Titarev and Eleuterio F Toro. Finite-volume WENO schemes for three-dimensional conservation laws. *Journal of Computational Physics*, 201(1):238–260, November 2004.

[26] W H Reed and T R Hill. Triangular mesh methods for the neutron transport equation. *Proceedings of the American Nuclear Society*, January 1973.

[27] B Cockburn and Chi-Wang Shu. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws II: General Framework. *Math Comput*, 52:411–435, 1989.

[28] B Cockburn, Suchung Hou, and Chi-Wang Shu. The Runge-Kutta local projection

discontinuous Galerkin finite element method for conservation laws IV: the multidimensional case. *Math. Comp*, 54(190):545–581, 1990.

[29] Jonathan Shewchuk. Triangle.

[30] Christophe Geuzaine and Jean-François Remacle. Gmsh.

[31] Irene A Stegun and Milton Abramowitz. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. Dover books on intermediate advanced mathematics. US. Nat. Bureau Stand., New York, NY, 1964.

[32] B Cockburn. Discontinuous Galerkin methods. *ZAMM*, 83(11):731–754, November 2003.

[33] Jianxian Qiu and Chi-Wang Shu. Runge–Kutta Discontinuous Galerkin Method Using WENO Limiters. *SIAM Journal on Scientific Computing*, 26(3):907–929, January 2005.

[34] Jun Zhu, Xinghui Zhong, Chi-Wang Shu, and Jianxian Qiu. Runge-Kutta discontinuous Galerkin method using a new type of WENO limiters on unstructured mesh. Technical Report 2012-8, Brown University, Providence, RI, March 2012.

[35] J Butcher. *The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods*. Wiley, January 1987.

[36] Michael Dumbser and Claus-Dieter Munz. Building Blocks for Arbitrary High Order Discontinuous Galerkin Schemes. *Journal of Scientific Computing*, 27(1-3):215–230, December 2005.

[37] Michael Dumbser, Cedric Enaux, and Eleuterio F Toro. Finite volume schemes of very high order of accuracy for stiff hyperbolic balance laws. *Journal of Computational Physics*, 227(8):3971–4001, April 2008.

[38] Gregor Gassner, F Lörcher, and Claus-Dieter Munz. A Discontinuous Galerkin Scheme based on a Space-Time Expansion II. Viscous Flow Equations in Multi Dimensions. *Journal of Scientific Computing*, 34(3):260–286, December 2007.

[39] Gregor Gassner, Michael Dumbser, Florian Hindenlang, and Claus-Dieter Munz. Explicit one-step time discretizations for discontinuous Galerkin and finite volume schemes based on local predictors. *Journal of Computational Physics*, 230(11):4232–4247, May 2011.

[40] Jianxian Qiu, Michael Dumbser, and Chi-Wang Shu. The discontinuous Galerkin method with Lax–Wendroff type time discretizations. *Computer Methods in Applied Mechanics and Engineering*, 194(42-44):4528–4543, October 2005.

[41] Michael Dumbser. *Arbitrary high order schemes for the solution of hyperbolic conservation laws in complex domains.* Stuttgart University Ph.D. Thesis. Shaker Verlag GmbH, Germany, August 2005.

[42] Michael E Taylor. *Partial Differential Equations I: Basic Theory*, volume 115 of *Applied Mathematical Sciences.* Springer, New York, 2011.

[43] Jianxian Qiu. A Numerical Comparison of the Lax–Wendroff Discontinuous Galerkin Method Based on Different Numerical Fluxes. *Journal of Scientific Computing*, 30(3):345–367, September 2006.

[44] Noah Reddell, Robert Lilly, Uri Shumlak, Eder Sousa, and Bhuvana Srinivasan. Computational Algorithm for the Multi-Fluid Plasma Model - WARPX. *APS Meeting Abstracts*, November 2010.

[45] C Z Cheng and Georg Knorr. The integration of the vlasov equation in configuration space. *Journal of Computational Physics*, 22(3):330–351, November 1976.

[46] L D Landau. On the vibrations of the electronic plasma. *J. Phys.(USSR)*, 10:25–34, 1946.

[47] F Filbet and Eric Sonnendrücker. Comparison of eulerian Vlasov solvers. *Computer Physics Communications*, 150(3):247–266, 2003.

[48] J A Tataronis and F W Crawford. Cyclotron harmonic wave propagation and instabilities: I. Perpendicular propagation. *Journal of Plasma Physics*, 4(02):231–248, May 1970.

[49] Peter H Yoon, Anthony T Y Lui, and Mikhail I Sitnov. Generalized lower-hybrid drift instabilities in current-sheet equilibrium. *Physics of Plasmas (1994-present)*, 9(5):1526–1538, 2002.

[50] Eric Priest and Terry Forbes. Magnetic Reconnection. *Magnetic Reconnection, by Eric Priest , Terry Forbes, Cambridge, UK: Cambridge University Press, 2007*, February 2007.

[51] Liang Wang, Ammar Hakim, A Bhattacharjee, and K Germaschewski. Comparison of multi-fluid moment models with particle-in-cell simulations of collisionless magnetic reconnection. *Physics of Plasmas*, 22(1):012108, January 2015.

[52] E A Johnson and J A Rossmanith. Magnetic reconnection for 10-moment two-fluid versus kinetic simulations. *Physics of Plasmas*, 2006.

[53] E Alec Johnson and J A Rossmanith. Ten-moment two-fluid plasma model agrees well with PIC/Vlasov in GEM problem. *arXiv.org*, October 2010.

[54] Evan Alexander Johnson. *Gaussian-Moment Relaxation Closures for Verifiable Numerical Simulation of Fast Magnetic Reconnection in Plasma.* PhD thesis, University of Wisconsin - Madison, September 2014.

[55] Michael Hesse, Karl Schindler, Joachim Birn, and Masha Kuznetsova. The diffusion region in collisionless magnetic reconnection. *Physics of Plasmas*, 6(5):1781–16, 1999.

[56] P L Pritchett. Geospace Environment Modeling magnetic reconnection challenge: Simulations with a full particle electromagnetic code. *Journal of Geophysical Research: Space Physics (1978–2012)*, 106(A3):1–16, 2001.

[57] H Schmitz and R Grauer. Kinetic Vlasov simulations of collisionless magnetic reconnection. *Physics of Plasmas*, 13(9):092309, 2006.

[58] Takayuki Umeda, Kentaro Togano, and Tatsuki Ogino. Two-dimensional full-electromagnetic Vlasov code with conservative scheme and its application to magnetic reconnection. *Computer Physics Communications*, 180(3):365–374, March 2009.

[59] Greg Faanes, Abdulla Bataineh, Duncan Roweth, Tom Court, Edwin Froese, Bob Alverson, Tim Johnson, Joe Kopnick, Mike Higgins, and James Reinhard. Cray Cascade: a Scalable HPC System based on a Dragonfly Network. In *Supercomputing 2012*, pages 1–9, August 2012.

[60] T Minoshima, Y Matsumoto, and T Amano. A finite volume formulation of the multi-moment advection scheme for Vlasov simulations of magnetized plasma. *Computer Physics Communications*, 2015.

VITA

Noah Reddell grew up outside the town of Darrington in rural western Washington. He was born to parents Jerry and Mary Ann and provided a brother Eli to torment until his superior growth rate prevailed. Noah moved far away for college – earning a B.S. in Electrical Engineering at the United States Naval Academy and appreciation for all things nautical. He developed an interest in research at Stanford University studying plasma-wave interactions in the Earth's magnetosphere and earned an M.S. in Electrical Engineering. He left academics for applied science and sailor-management as a U.S. Navy submarine officer for six years. On this last pass as an bona fide student, he traveled, he married, he settled in, and he took the long road.