Numerical Methods for 3-dimensional Magnetic Confinement Configurations using Two-Fluid Plasma Equations

Bhuvana Srinivasan

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

Doctor of Philosophy

University of Washington

2010

Program Authorized to Offer Degree:
Aeronautics & Astronautics

University of Washington
Graduate School

This is to certify that I have examined this copy of a doctoral dissertation by

Bhuvana Srinivasan

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Chair of the Supervisory Committee:

_____

Uri Shumlak

Reading Committee:

_____

Uri Shumlak

_____

Brian Nelson

_____

Richard Milroy

Date: _____

In presenting this dissertation in partial fulfillment of the requirements for the doctoral degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this dissertation is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for copying or reproduction of this dissertation may be referred to Proquest Information and Learning, 300 North Zeeb Road, Ann Arbor, MI 48106-1346, 1-800-521-0600, to whom the author has granted "the right to reproduce and sell (a) copies of the manuscript in microform and/or (b) printed copies of the manuscript made from microform."

Signature_____

Date_____

University of Washington

Abstract

Numerical Methods for 3-dimensional Magnetic Confinement Configurations using Two-Fluid Plasma Equations

Bhuvana Srinivasan

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

The 5-moment two-fluid plasma model uses Euler equations to describe the ion and electron fluids, and Maxwell's equations to describe the electric and magnetic fields. Two-fluid physics becomes significant when the characteristic spatial scales are on the order of the ion skin depth and characteristic time scales are on the order of the inverse ion cyclotron frequency. The two-fluid plasma model has disparate characteristic speeds ranging from the ion and electron speeds of sound to the speed of light. In addition, the characteristic frequencies in the system are the ion and electron plasma frequency, and the ion and electron cyclotron frequency. Explicit and implicit time-stepping schemes are explored for the two-fluid plasma model to study the accuracy and computational effectiveness with which they could capture two-fluid physics. The explicit schemes explored include the high resolution wave propagation method (a finite volume method) and the Runge-Kutta discontinuous Galerkin (RKDG) method (a finite element method). The ideal two-fluid model is a purely dispersive equation system with no physical or artificial dissipation. The dispersions are physical effects responsible for the wide variety of plasma waves; they are not numerical artifacts. This sets the two-fluid plasma model apart from other equation systems. The finite volume and finite element methods are compared for accuracy and computational expense for applications of the two-fluid plasma model. For realistic regimes, the explicit time-step for the two-fluid plasma model can be very restrictive making it computationally

expensive. This motivates the implicit time-stepping scheme. A semi-implicit two-fluid plasma model is developed using the discontinuous Galerkin method where the electron fluid equations and Maxwell's equations are evolved implicitly eliminating the restrictions set by the speed of light, and the electron plasma and cyclotron frequencies. Resolving all ion time-scales is a minimum to capture two-fluid physics, so the ion fluid equations are solved explicitly. This allows for accuracy and physics considerations alone to determine the time-step. Non-ideal terms are added to the two-fluid plasma model in the form of resistivity, viscosity, and heat flux to provide a self-consistent and physically relevant two-fluid plasma model and these are compared to solutions of the ideal two-fluid plasma model. The two-fluid plasma model is compared to the more commonly used Hall-MHD model for accuracy and computational effort using an explicit time-stepping scheme. Simulations of two-fluid instabilities in the Z-pinch and the field-reversed configuration are presented in 3-dimensions.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

The author wishes to express sincere appreciation to the Department of Aeronautics & Astronautics and especially to her committee chair, Professor Uri Shumlak, for the support, both financial and moral, and the guidance that he has provided during the course of the author's graduate career. The author also wishes to express gratitude to Dr. Ammar Hakim for his immense help with the code, WARPX, and all the advice that he has provided to the author over their years of collaboration. The support of Prof. Shumlak and Dr. Hakim made this dissertation possible. The author is thankful for the valuable suggestions and time of her committee members, Prof. Randall LeVeque, Prof. Arthur Mattick, Prof. Richard Milroy, Prof. Brian Nelson, and Prof. John Slough. The author would like to acknowledge the help of Dr. Loren Steinhauer in properly interpreting the two-fluid instability. The author would like to thank Dr. John Loverich for suggestions that proved valuable to the completion of this dissertation, and Robert Lilly for engaging in interesting and enlightening discussions ranging from plasma physics to history. The computing resources provided by the ICE cluster at the University of Washington and the Jaws and Mana clusters at the Maui High Performance Computing Center are gratefully acknowledged. The research resulting in this dissertation was funded by the AFOSR Grant FA9550-05-1-0159. The author is grateful for the constant support and encouragement from her parents, Geetha and Subbaraman Srinivasan, from her sister, Jaishri Srinivasan, and from all the amazing friends that made graduate school a pleasant experience, with a special mention for Colin Adams.

DEDICATION

To my parents, Geetha and Subbaraman Srinivasan.

Chapter 1

INTRODUCTION

1.1 Introduction

The majority of matter in the universe is in plasma state, dissociated into ions and electrons, motivating the study of plasma physics. Plasma physics is relevant to study stellar structures, interstellar media, and effects of the solar wind on the earth's magnetosphere, to state a few space physics and astrophysics applications. Plasma physics also plays a key role in the development of controlled thermonuclear fusion specifically magnetic confinement fusion and magneto-inertial fusion. In addition, a study of space propulsion systems employing plasma dynamics such as ion thrusters, magnetoplasmadynamic thrusters, Hall thrusters, Helicon thrusters, etc. requires an understanding of plasma physics.

A number of classical plasma models have been developed over the years to solve problems relevant to laboratory and space plasmas. The particle description of plasmas is the most fundamental model involving position and velocity tracking for each particle or superparticle. The Lorentz forces on the particles are calculated using Maxwell's equations of electromagnetics and the motions of the charged particles produce electromagnetic fields that interact with other particles in the system. In order to model bulk plasma behavior, the Vlasov model is used where the individual particles are replaced by a continuous distribution function. The Vlasov model contains an infinite number of moment equations and is a function of position and velocity which makes it six-dimensional. The full two-fluid plasma model described in this dissertation results from taking the 2^{nd} moment of the Vlasov model while assuming isotropic pressure for closure and ignoring higher-order tensors. Fluid models of plasmas have 3 dimensions compared to the 6 dimensions of the Vlasov model which make them computationally less intensive particularly for simulations of large-scale experiments. Although this reduction results in a loss of physics, the fluid description is able to capture an enormous amount of physics relevant to fusion and propulsion devices.

There are a number of fluid models of plasmas that are applicable in different regimes. The purpose of this dissertation is to study the ideal two-fluid plasma model and apply asymptotic approximations to reduce it to the more commonly used Hall-MHD model. The two models are compared for their ability to capture two-fluid physics and the computational effort required for problems such as the electromagnetic plasma shock, GEM challenge magnetic reconnection, the axisymmetric Z-pinch and the Field Reversed Configuration (FRC). Two-fluid effects become significant when the characteristic spatial scales are small compared to the ion skin depth and the characteristic time scales are short compared to the inverse ion cyclotron frequency. The full two-fluid plasma model has been used to study several plasma problems and has been benchmarked to previously published results. Hakim, Loverich, and Shumlak[1] implement an explicit finite volume method, the high resolution wave propagation method, for the two-fluid plasma model. Loverich and Shumlak[2] implement an explicit finite element method, the Runge-Kutta discontinuous Galerkin method, for the two-fluid plasma model. The two-fluid plasma model has been applied to study instabilities resulting from two-fluid effects in the Z-pinch[3] and in the FRC[4] using explicit time-stepping schemes. The study of the two-fluid model in this dissertation is extended to include non-ideal terms in the form of frictional forces, thermal forces, viscosity, thermal equilibration of the ion and electron species, and heat flux. This allows the two-fluid plasma model to capture physics associated with collisionality and provides a more complete physically relevant model for comparisons to benchmark results. A description of the fluid models is presented in Chapter 2.

The numerical method primarily used in this dissertation is the discontinuous Galerkin method. The equation systems studied here are described by balance laws of the form

$$\frac{\partial \mathbf{Q}}{\partial t} + \nabla \cdot \mathbf{F} = \mathbf{S}, \tag{1.1}$$

where $\mathbf{Q} \in \mathbf{R}^m$ represents the m conserved variables, $\mathbf{F} \in \mathbf{R}^{m \times d}$ represents fluxes where d is the number of spatial dimensions, and $\mathbf{S} \in \mathbf{R}^m$ represents the source terms. For all unit vectors $\boldsymbol{\omega} \in \mathbf{R}^d$ the flux Jacobian, $\partial(\mathbf{F} \cdot \boldsymbol{\omega})/\partial \mathbf{Q}$, is assumed to have real eigenvalues and a complete set of right eigenvectors. In the absence of source terms Eq. (1.1) is a hyperbolic

system of conservation laws[5, 6]. The conserved quantities for the two-fluid plasma model presented in this dissertation are the ion and electron densities, ion and electron momenta, ion and electron total energies, electric fields, and magnetic fields.

An additional objective of this dissertation is to explore various time integration schemes for the discontinuous Galerkin method particularly for applications of the two-fluid plasma model. In this dissertation, the Runge-Kutta time integration scheme is used with the discontinuous Galerkin method. An implicit time integration scheme is explored for the discontinuous Galerkin model in order to overcome the time-step restrictions of the speed of light and the electron plasma and cyclotron frequencies in the two-fluid plasma model. The numerical methods used in this dissertation and the implementation details are presented in Chapter 3. The advantages of using an implicit time integration over the Runge-Kutta time integration are described. Most of the implementations presented here use a Cartesian geometry with a rectangular grid. Some applications such as a 3-dimensional FRC require the use of general geometry body-fitted grids in order to eliminate the effect of the rectangular grid on the evolution of the circular geometry. The implementation details of a general geometry discontinuous Galerkin method are presented in Chapter 3.

In Chapter 4 of this dissertation, the high-resolution wave propagation method, a finite volume method, is compared to the Runge-Kutta discontinuous Galerkin (RKDG) method, a finite element method, for plasma fluid equations. The two-fluid plasma model has purely dispersive source terms that bear physical relevance in the form of the various plasma waves in the system. The numerical methods are compared for their ability to capture this physical dispersion.

In Refs.[1] and [2], the full two-fluid plasma model has been solved using explicit time-stepping schemes with the finite volume and the finite element methods respectively. When using explicit time-stepping schemes, the two-fluid plasma model encounters a severe time-step restriction governed by either the speed of light that shows up in Maxwell's equations or the electron plasma frequency that results from the inclusion of the electron mass in the equations. This provides the motivation to study a semi-implicit, two-fluid plasma model where the time-step is determined by accuracy and physics considerations alone. A semi-implicit, two-fluid plasma model is described in Chapter 6 and it is compared to the explicit

time integration scheme for the two-fluid plasma model.

Chapter 8 details the applications of the two-fluid plasma model explored as a part of this dissertation. A comparison between the two-fluid plasma model and Hall-MHD is provided here. A non-ideal two-fluid plasma model using Braginskii's transport terms are explored for some benchmark problems. The non-ideal two-fluid model is used to study several applications of the two-fluid plasma model that were previously explored without the inclusion of transport terms. A fully 3-dimensional implementation of the two-fluid model is presented using the Runge-Kutta discontinuous Galerkin method with general geometry to simulate plasma instabilities in a Z-pinch and an FRC.

WARPX, Washington Approximate Riemann Plasma code that was developed at the University of Washington, is used for all the simulations in this dissertation. WARPX is a generalized geometry parallel C++ code and it is embarrassingly parallel for explicit simulations of the two-fluid plasma model. It uses MPI[7, 8], HDF5[9], LAPACK[10], PETSc[11] and SCons[12] as its main dependencies. Python and VisIt[13] are used for pre- and post-processing, and for visualization. The code is written with consideration for implementing any equation system with any numerical method with a focus on hyperbolic balance laws. This code contains a software framework that is general for both the wave propagation method and any desired order of the explicit or implicit DG method using the same flux, source and Riemann problem solvers for each hyperbolic equation system that is being simulated.

1.2 Objective

The objective of this dissertation is to study the two-fluid plasma model for applications towards experimental geometries. In order to do this, a numerical method capable of capturing the two-fluid physics needs to be developed and implemented. This provides the motivation for the comparisons between the finite volume and finite element methods specifically for the two-fluid plasma model that has been performed in this dissertation. A suitable time-integration scheme needs to be developed to account for the disparate speeds and frequencies in the two-fluid plasma model. This provides the motivation to study and implement implicit time-integration schemes for applications of the two-fluid plasma model and to determine

if the desired physics and accuracy can be achieved by taking large time-steps. Results of an implicit implementation are presented in this dissertation.

Once a suitable numerical method is developed, the two-fluid plasma model needs to be benchmarked. Hall-MHD has been increasingly used in the plasma physics community to capture two-fluid physics. A Hall-MHD model is developed within the WARPX framework as a part of this dissertation and is compared to the two-fluid plasma model for applications of the electromagnetic plasma shock and GEM Challenge magnetic reconnection. These benchmark applications contain previously published results and performing a comparison of the two-fluid plasma model to Hall-MHD sheds light on the difference in physics between the two models as well as the computational difficulties associated with implementing each of the models.

In order to make the model as complete as necessary, non-ideal terms are included in the two-fluid plasma model to account for the effects of transport and collisions. While a collisionless plasma assumption is sufficient for a number of problems, the effect of collisions need to be taken into account to accurately model plasma geometries in certain regimes. This dissertation explores the inclusion of transport in the two-fluid plasma model.

The objective of this dissertation is to develop a suitable numerical method and a physically relevant full two-fluid plasma model with generalized geometry to study two-fluid instabilities that develop in a 3-dimensional Z-pinch and an FRC.

Chapter 2

FULL TWO-FLUID PLASMA MODEL AND ASYMPTOTIC APPROXIMATIONS

2.1 Background

In this chapter, the ideal two-fluid plasma model is described and asymptotic approximations are applied to the full two-fluid plasma model to derive the Hall-MHD model. Analytical dispersion relations are presented and several intermediate models are explored. The non-ideal two-fluid plasma model is presented to include effects of transport and collisions.

The equations described here are the five-moment equations that result from taking the zeroth, first and second moments of the collisionless Boltzmann equation or the Vlasov equation

$$\frac{\partial f_s}{\partial t} + \mathbf{v} \cdot \nabla f + \frac{q}{m}(\mathbf{E} + \mathbf{v} \times \mathbf{B})\frac{\partial f}{\partial \mathbf{v}} = 0 \tag{2.1}$$

where $f = f(\mathbf{r}, \mathbf{v}, t)$ is the particle density distribution function as a function of space, velocity and time respectively such that $\int f d\mathbf{v} = n$. Here n is the number density of the species, with the species being ions and electrons. $\mathbf{E}$ and $\mathbf{B}$ represent the electric and magnetic fields and q and m represent the species charge and mass. This is a 6 dimensional equation and is coupled with Maxwell's equations for a continuum description of a collisionless plasma. Infinite moments of the Vlasov equation can be taken, however, it is possible to obtain a simplified model suitable for numerical modeling by making some approximations and truncating the series of equations.

The Vlasov equation is multiplied by a chosen set of moments and is integrated over all velocity space to obtain the fluid equations. The first moment is the mass of the species, m_s which gives rise to the continuity equation with a mass density ρ and a mean velocity $\mathbf{u}$ of the species. The next three equations are obtained by multiplying the Vlasov equation by the next moment, the mass momentum ($m\mathbf{u}$), and integrating in velocity space. This results

in the momentum equation with $\rho\mathbf{u}$ representing the momentum term. The last moment is obtained by multiplying the tensor product $m\mathbf{vv}$ to the Vlasov equation and integrating over velocity space. This would ordinarily result in 6 independent equations containing an anisotropic pressure tensor. In order to simplify the equation system and provide a closure, an isotropic pressure is assumed and an equation of state is used to describe the evolution of the pressure, P. This results in an equation for the energy, E. The resulting ideal two-fluid plasma model contains 5 equations to evolve the 5 conserved variables, continuity, momentum in 3 directions, and energy for each of the ion and electron species. Additional equations evolve the electric and magnetic fields.

2.2 Ideal Full Two-Fluid Plasma Model

The full two-fluid plasma model used in this dissertation is described by a complete set of Euler equations for the ions, a complete set of Euler equations for the electrons and a complete set of Maxwell's equations to evolve the electric and magnetic fields. The source terms couple the fluid and field variables. Writing these in balance law form described by Eq. (1.1),

$$\frac{\partial \rho_s}{\partial t} + \nabla \cdot (\rho_s \mathbf{u}_s) = 0 \tag{2.2}$$

$$\frac{\partial \rho_s \mathbf{u}_s}{\partial t} + \nabla \cdot (\rho_s \mathbf{u}_s \mathbf{u}_s + p_s \mathbf{I}) = \frac{\rho_s q_s}{m_s}(\mathbf{E} + \mathbf{u}_s \times \mathbf{B}) \tag{2.3}$$

$$\frac{\partial \epsilon_s}{\partial t} + \nabla \cdot ((\epsilon_s + p_s)\mathbf{u}_s) = \frac{\rho_s q_s}{m_s}\mathbf{u}_s \cdot \mathbf{E} \tag{2.4}$$

where subscript, s, denotes electron or ion species, q_s and m_s are the species charge and mass. The energy is defined as

$$\epsilon_s \equiv \frac{p_s}{\gamma - 1} + \frac{1}{2}\rho_s u_s^2. \tag{2.5}$$

It can be seen that the source terms of Eq. (2.3) contain the Lorentz forces on the electrons and ions. These source terms couple the fluid equations to the electromagnetic terms of Maxwell's equations. The Lorentz forces act as body forces on the electrons and ions. The evolving electromagnetic source terms can make the equation set and the solutions rather

complicated.

Maxwell's equations are used to evolve the electric and magnetic fields using Faraday's law and Ampere's law described by

$$\frac{\partial \mathbf{B}}{\partial t} + \nabla \times \mathbf{E} = 0 \tag{2.6}$$

$$\frac{1}{c^2}\frac{\partial \mathbf{E}}{\partial t} - \nabla \times \mathbf{B} = -\mu_0 \sum_{\mathbf{s}} \frac{\mathbf{q_s}}{\mathbf{m_s}} \rho_{\mathbf{s}} \mathbf{u_s} \tag{2.7}$$

Additional constraints are required to preserve divergence,

$$\nabla \cdot \mathbf{E} = \frac{\varrho_c}{\varepsilon_0} \tag{2.8}$$

$$\nabla \cdot \mathbf{B} = 0. \tag{2.9}$$

Although Eqs.(2.6) and (2.7) provide sufficient information to solve the equation system and advance the solution, divergence errors can develop in the solution. Often, these appear as small numerical errors initially and then grow with time. Solving Eqs.(2.8) and (2.9) additionally leads to an over-specified system of equations. Hence, the perfectly hyperbolic Maxwell's equations[14] are used to evolve the electric and magnetic fields while maintaining the divergence constraints as shown in Eqs.(2.12) and (2.13)

$$\frac{\partial \mathbf{B}}{\partial t} + \nabla \times \mathbf{E} + \gamma \nabla \Psi = 0 \tag{2.10}$$

$$\frac{1}{c^2}\frac{\partial \mathbf{E}}{\partial t} - \nabla \times \mathbf{B} + \chi \nabla \Phi = -\mu_0 \mathbf{J} \tag{2.11}$$

$$\frac{1}{\chi}\frac{\partial \Phi}{\partial t} + \nabla \cdot \mathbf{E} = \frac{\varrho_c}{\varepsilon_0} \tag{2.12}$$

$$\frac{1}{\gamma c^2}\frac{\partial \Psi}{\partial t} + \nabla \cdot \mathbf{B} = 0 \tag{2.13}$$

where ϱ_c and $\mathbf{J}$ are the charge density and the current density defined by

$$\varrho_c \equiv \sum_s \frac{q_s}{m_s} \rho_s \tag{2.14}$$

$$\mathbf{J} \equiv \sum_s \frac{q_s}{m_s} \rho_s \mathbf{u}_s. \tag{2.15}$$

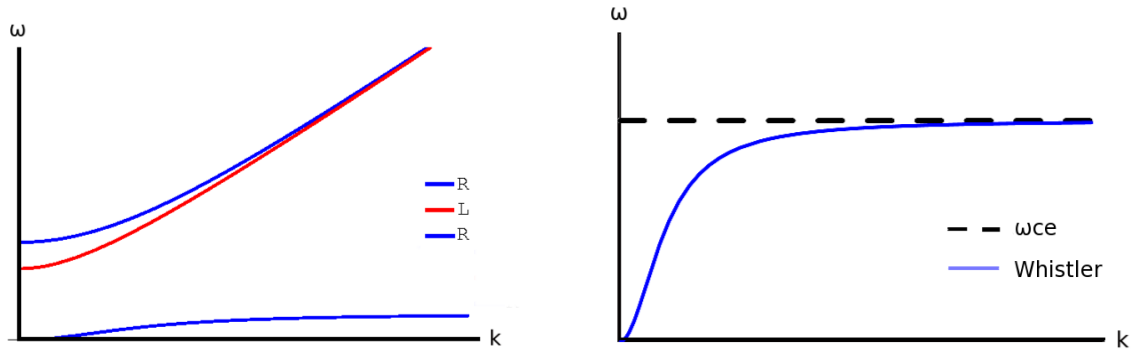

Figure 2.1: Parallel propagation dispersion relation of ω vs k for the two-fluid plasma model. Left plot shows R- and L-mode waves and right plot shows the low-frequency R-mode wave, the whistler wave, that has an asymptote at the electron cyclotron frequency. The right-hand-side plot is an expanded scale of the left-hand-side plot.

In Eqs.(2.12) and (2.13), Φ and Ψ are the scalar error correction potentials for the divergence of $\mathbf{E}$ and divergence of $\mathbf{B}$ respectively. It is noted that as the error correction coefficients, χ and γ are large, the divergence constraints are closer to being maintained. For many problems, the error correction coefficients can be set to 1 such that the errors are propagated out of the domain at the speed of light. This is often a sufficient condition, however, for some problems better divergence correction is required to capture the evolution of the electromagnetic waves. In such situations increasing the error correction coefficients leads to a more restrictive time-step than the speed of light. For problems with electric and magnetic fields, it is important to ensure that the divergence constraints are satisfied initially.

Figure 2.1 shows the dispersion diagrams for parallel propagation using the two-fluid model. The R-mode, L-mode and whistler waves, which are a part of the low-frequency spectrum of the R-mode wave, are shown and it is seen that the whistler wave has an asymptote at the electron cyclotron frequency.

For the ideal two-fluid plasma model, the characteristic speeds in the system are the fluid speeds of sound and the speed of light. The numerical method used to solve the two-fluid plasma model needs to be capable of resolving the physics in the presence of such disparate characteristic speeds. Furthermore, the characteristic frequencies in the system include the electron and ion plasma frequencies, and the electron and ion cyclotron frequencies. All

the speeds and frequencies must be resolved in the system to capture full two-fluid physics when using an explicit time-integration scheme. The explicit time step is described by

$$\Delta t = min\left(CFL\frac{\Delta x}{c}, \frac{0.1}{\omega_{pe}}, \frac{0.1}{\omega_{ce}}, \frac{0.1}{\omega_{pi}}, \frac{0.1}{\omega_{ci}}\right) \tag{2.16}$$

where c is the light speed,

$$\omega_{ps} = \sqrt{\frac{n_s q_s^2}{\varepsilon_0 m_s}}, \quad \omega_{cs} = \frac{q_s \mathbf{B}}{m_s}, \tag{2.17}$$

CFL is the Courant condition for stability, and the factor of 0.1 for each of the frequencies is to ensure that the Nyquist condition for proper sampling is maintained for resolving these time scales.

The two-fluid model is valid for simulations involving electron demagnetization due to the inclusion of electron inertia. Electron inertia allows the electrons to break the frozen-in flux condition. The two-fluid model is also valid in regimes where the ion Larmor radius is much smaller than the scale-length of interest as well as when the ion Larmor radius is much larger than the scale-length of interest. For large ion Larmor radii, single-fluid models are no longer relevant since the Larmor radius is assumed to be negligible in the single-fluid regime and two-fluid models are required to capture the physics. To include effects of anisotropic pressure, finite Larmor radius effects need to be included and higher moments of the fluid equations need to be taken to explore this regime.

The two-fluid plasma equation system is different from Euler equations or Navier-Stokes equations. The two-fluid plasma equations are dispersive and not dissipative. The dispersion is not a numerical artifact. The dispersive nature is a physical effect that leads to the wide variety of plasma waves. Mathematically, the dispersive effects are generated from the source terms of the two-fluid plasma equations. Explicit methods are often challenging to use in the presence of such dispersions because increasing the grid resolution can trigger waves of smaller wavelengths making it difficult to capture these physical dispersions accurately.

2.2.1 Source Terms of the Two-Fluid Plasma Model

The two-fluid plasma model has physical dispersion that comes about from the presence of dispersive source terms. The source Jacobian for equation Eq. (1.1) is $\frac{\partial \mathbf{S}}{\partial \mathbf{Q}}$. The first three eigenvalues of the source Jacobian are $0, \pm i\omega_p$ where $\omega_p^2 = \omega_{pe}^2 + \omega_{pi}^2$. The plasma frequency is defined as

$$\omega_{ps} = \sqrt{\frac{n_s q_s^2}{\varepsilon_0 m_s}}, \tag{2.18}$$

where subscript s represents each species (electrons and ions). The remaining six eigenvalues are all imaginary roots of a 6^{th} order polynomial described as follows with respect to λ,

$$\begin{aligned} -\frac{1}{M^2}\lambda\left(\lambda^2+\omega_{pe}^2+\omega_{pi}^2\right)\Big(B^4\lambda^2 r_i^4 + M^2\lambda^2\left(\lambda^2+\omega_{pe}^2+\omega_{pi}^2\right)^2 \\ +B^2 r_i^2\left(M^2\lambda^4+\lambda^4+2M^2\omega_{pe}^2\lambda^2+M^2\omega_{pe}^4+\omega_{pi}^4+2\left(\lambda^2-M\omega_{pe}^2\right)\omega_{pi}^2\right)\Big) = 0 \end{aligned} \tag{2.19}$$

where M is the electron-to-ion mass ratio, r_i is the ion charge-to-mass ratio, and B is the total magnitude of the magnetic field. All non-zero eigenvalues, λ, are imaginary.

Since the source Jacobian has only imaginary eigenvalues, the waves of the two-fluid model are not damped. This plasma model is not diffusive but instead is dispersive with undamped oscillations. These dispersions are not numerical but are physical and are responsible for the wide variety of plasma waves. In Chapter 4, the wave propagation and RKDG methods are studied for their abilities to capture these physical dispersions accurately. Explicit methods are often unstable when such oscillations are present because refining the grid can excite waves of smaller wavelengths making it difficult to capture the dispersions accurately.

2.3 Asymptotic Approximations

Two asymptotic approximations are applied to the full two-fluid plasma model described previously to obtain the reduced fluid models. These approximations are negligible electron inertia and infinite light speed. Quasi-neutrality follows from the infinite light speed approximation. In this section, each asymptotic approximation is applied independently and

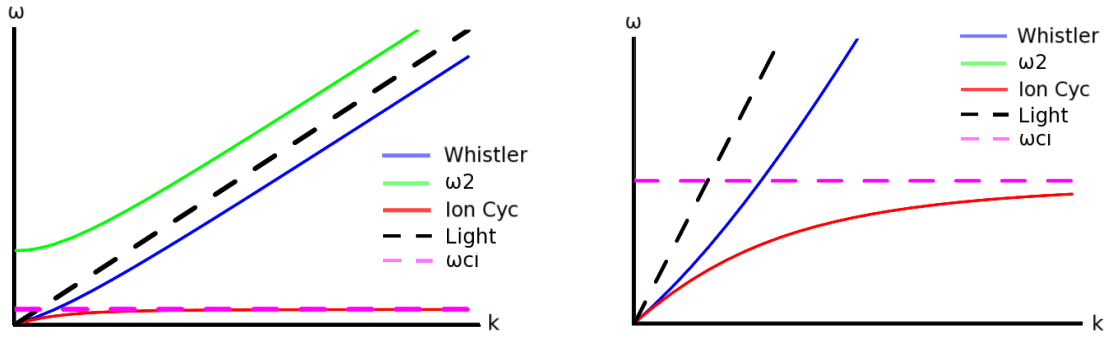

Figure 2.2: Parallel propagation dispersion relation of ω vs k when electron inertia is ignored in the two-fluid plasma model. The dashed black line represents the speed of light and is included for scale. Right plot has an expanded scale to show the solution of the left plot for smaller values of k. The blue line is the whistler wave and reaches an asymptote at the speed of light. The red line is the ion cyclotron wave. The green line denoted by $\omega 2$ is an additional wave of the dispersion relation that has an asymptote at the speed of light.

the resulting dispersion relations are studied. Applying both approximations together gives the Hall-MHD model.

2.3.1 Negligible Electron Inertia

Realistic ion-to-electron mass ratio is approximately 1836 when the ion is a single proton. Since the ions are more massive than the electrons, electron inertia is neglected in a majority of plasma fluid models. Neglecting electron inertia reduces the electron momentum equation described in Eq. (2.3) to the generalized Ohm's law,

$$n_e q_e \mathbf{E} = \nabla p_e - \mathbf{J}_e \times \mathbf{B}. \tag{2.20}$$

where $\mathbf{J}_e = n_e q_e \mathbf{u}_e$. This approximation also eliminates the kinetic energy term in the electron energy described by Eq. (2.5).

The dispersion diagram for parallel propagation when ignoring electron inertia in the two-fluid model is shown in Fig. 2.2. The red line is the ion cyclotron wave and it has an asymptote at the ion cyclotron frequency. The dispersion wave represented by the blue line is the whistler wave and it reaches an asymptote at the speed of light. The dispersion wave

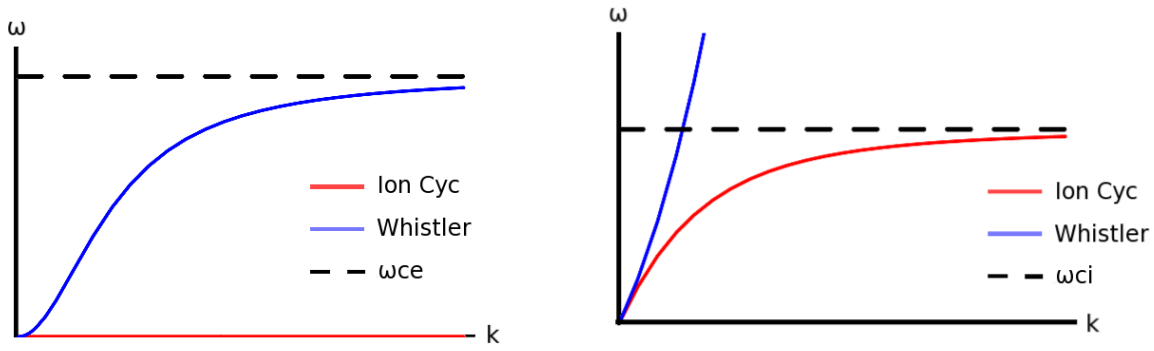

Figure 2.3: Parallel propagation dispersion relation of ω vs k when speed of light is assumed to be infinite in the two-fluid plasma model. The left plot shows the whistler wave that has an asymptote at the electron cyclotron frequency and the right plot shows the ion cyclotron wave with an asymptote at the ion cyclotron frequency. Right plot has an expanded scale to show the solution of the left plot for smaller values of k and shows the ion cyclotron wave.

represented by the green line also reaches an asymptote at the speed of light.

2.3.2 Infinite Speed of Light

The infinite speed of light approximation is used to ignore the high frequency electromagnetic waves so that the regime of interest lies in the lower frequency plasma waves. This approximation is achieved by assuming that the vacuum permittivity of free space is 0 thus implying infinite speed of light. Applying this approximation eliminates the displacement currents from Ampere's law in Eq. (2.11) and results in

$$\mathbf{J} = \frac{1}{\mu_0}\nabla \times \mathbf{B} \tag{2.21}$$

such that $\mathbf{J} = \mathbf{J}_i + \mathbf{J}_e$.

The dispersion diagrams for parallel propagation are shown in Fig. 2.3. Quasineutrality is automatically implied through the mathematics of this approximation from Eq. (2.12) because of the $\varepsilon_0 \to 0$ assumption. The parallel dispersion relations for the infinite light speed assumption yield a whistler wave with an asymptote at the electron cyclotron frequency and an ion cyclotron wave with an asymptote at the ion cyclotron frequency. The high frequency

R- and L-mode waves do not appear in this approximated model since $\varepsilon_0 \to 0$ eliminates the ion and electron plasma frequencies as well as the speed of light. No displacement currents are present in this model.

From Poisson's equation,

$$\varepsilon_0 \nabla \cdot \mathbf{E} = q_i(n_i - n_e). \tag{2.22}$$

For $\varepsilon_0 \to 0$, $n_i = n_e$, where n represents the number density of each species. Therefore, the infinite light speed approximation implies neutrality.

2.3.3 Reduction to Hall-MHD

Applying all 3 of the above approximations together gives Hall-MHD. In the Hall-MHD model, the electron momentum equation reduces to the generalized Ohm's law, Ampere's law reduces to the form described in Eq. (2.21), and the electron continuity equation is eliminated. Hall-MHD as implemented in this dissertation is described by a complete set of Euler equations for ions, an electron energy equation, the reduced Ampere's law (described by Eq. (2.21)) and Faraday's law (described by Eq. (2.10)).

Ref.[15] details a hyperbolic divergence cleaning method for the MHD equations based on the perfectly hyperbolic Maxwell's equations[14]. This technique is applied to the Hall-MHD equation system where the fluid equations are

$$\frac{\partial \rho_i}{\partial t} + \nabla \cdot (\rho_i \mathbf{u}_i) = 0 \tag{2.23}$$

$$\frac{\partial \rho_i \mathbf{u}_i}{\partial t} + \nabla \cdot (\rho_i \mathbf{u}_i \mathbf{u}_i + p_i \mathbf{I}) = n_i q_i \mathbf{E} + \mathbf{J}_i \times \mathbf{B} \tag{2.24}$$

$$\frac{\partial \epsilon_i}{\partial t} + \nabla \cdot ((\epsilon_i + p_i)\mathbf{u}_i) = \mathbf{J}_i \cdot \mathbf{E} \tag{2.25}$$

$$\frac{\partial \epsilon_e}{\partial t} + \nabla \cdot ((\epsilon_e + p_e)\mathbf{u}_e) = \mathbf{J}_e \cdot \mathbf{E}. \tag{2.26}$$

Here $\mathbf{J}_i$ and $\mathbf{J}_e$ are the ion and electron currents. Assuming infinite speed of light leads to $n_i = n_e$, eliminating the need for a separate electron continuity equation. The assumption of negligible electron inertia eliminates the electron kinetic energy from the electron energy

equation and the electron momentum equation reduces to the generalized Ohm's law

$$n_e q_e \mathbf{E} = -\mathbf{J}_e \times \mathbf{B} + \nabla p_e. \tag{2.27}$$

Faraday's law contains the divergence correction potential Ψ

$$\frac{\partial \mathbf{B}}{\partial t} + \nabla \times \mathbf{E} + \nabla \Psi = 0 \tag{2.28}$$

$$\frac{\partial \Psi}{\partial t} + \Gamma^2 \nabla \cdot \mathbf{B} = -\zeta \Psi. \tag{2.29}$$

Here Γ is the speed at which the divergence of $\mathbf{B}$ errors are propagated out of the domain and ζ provides dissipation for the divergence errors. To obtain a perfectly hyperbolic divergence correction formulation, ζ is often assumed to be 0 such that the error is propagated out of the domain without dissipation similar to the implementation for the full two-fluid plasma model.

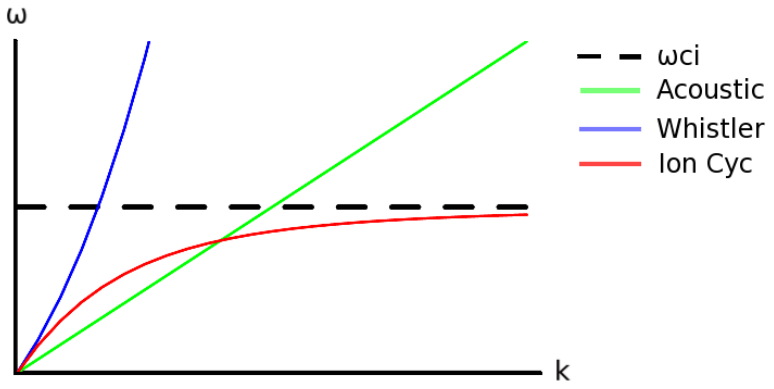

Figure 2.4: Parallel propagation dispersion relation of ω vs k when all 3 assumptions are applied to the two-fluid model to reduce it to Hall-MHD. The green line is the ion acoustic wave and the red line is the ion cyclotron wave. The whistler wave, denoted by the blue line, grows quadratically without bound.

The dispersion diagram for parallel propagation is shown in Fig. 2.4. Figure 2.4 shows that the whistler wave for the Hall-MHD model grows quadratically without bound in regimes where the ion skin depth, δ_i, and the ion cyclotron frequency, ω_{ci}, become significant.

If the scale length of interest $L \gg \delta_i$ the dispersion diagram resembles that of ideal-MHD and is only described near the origin of Fig. 2.4. This may be better seen by studying the dispersion relation for parallel propagation for Hall-MHD,

$$\omega = \frac{1}{2}\omega_{ci}\delta_i^2 k^2 + \sqrt{v_A^2 k^2 + \frac{1}{2}\omega_{ci}^2 \delta_i^4 k^4} \tag{2.30}$$

where v_A is the Alfvén velocity. For $L \gg \delta_i$ and $\omega \ll \omega_{ci}$, the Alfvén wave dispersion relation varies linearly with k and resembles ideal-MHD. However, for sufficiently large k, the whistler wave becomes,

$$\omega = \omega_{ci}\delta_i^2 k^2. \tag{2.31}$$

Therefore, for a regime where $L \lesssim \delta_i$ and $\omega \gtrsim \omega_{ci}$, the whistler wave grows quadratically without bound.

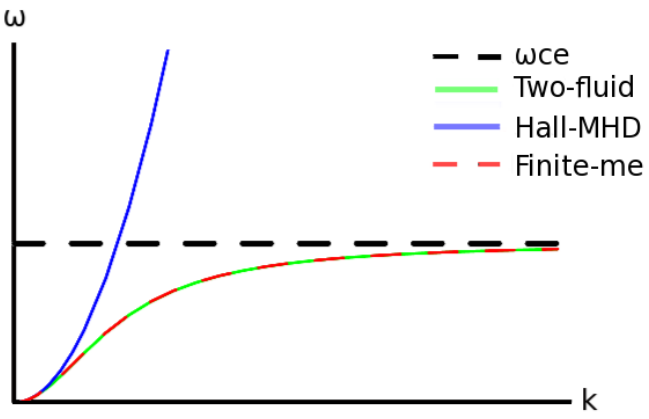

Figure 2.5: The whistler waves of the two-fluid model (green line), Hall-MHD (blue line) and an infinite light speed with finite electron mass model (red dashed line) are compared. Finite electron mass allows the whistler wave to have an asymptote at the electron cyclotron frequency whereas ignoring electron mass causes it to grow without bound as seen in Hall-MHD.

Figure 2.5 compares the whistler wave of the two-fluid plasma model, Hall-MHD, and the asymptotic model that contains electron inertia but assumes infinite speed of light. It

is seen that neglecting the electron inertia is the cause of the unbounded whistler wave and assuming finite electron mass makes this whistler wave have an asymptote at the electron cyclotron frequency, ω_{ce}. This unbounded whistler wave is problematic for simulations that use the Hall-MHD model because waves of higher and higher frequency need to be resolved as the wave number increases. This requires smaller and smaller time steps in order to resolve the two-fluid physics and hence, longer simulation times. The explicit time-step for Hall-MHD is described by

$$\Delta t = min\left(CFL\frac{\Delta x}{v_W}, CFL\frac{\Delta x}{v_M}, CFL\frac{\Delta x}{v_H}, \frac{0.1}{\omega_{ci}}\right) \tag{2.32}$$

where

$$v_W = \frac{kv_A^2}{\omega_{ci}}, \quad v_M = \sqrt{v_A^2 + c_{si}^2}, \quad v_H = -\frac{J}{nq} \tag{2.33}$$

Here, v_A is the Alfvén velocity, v_W is the whistler velocity, v_M is the magnetosonic velocity, and v_H is the Hall velocity. J is the magnitude of the total current. To compute the whistler wave, the wave number, k, is chosen to be at the grid scale such that $k = \pi/\Delta x$.

In order to overcome the grid scale dependence of the whistler wave, a cut-off frequency can be chosen such that any phenomena occurring on higher frequencies will not be resolved. This does not lead to an asymptote for the whistler wave like with the two-fluid plasma model. Instead, it just cuts-off the dispersion relation above the specified wave number. This is done by employing a hyper-resistivity similar to the one described in Ref.[16] which is purely included for numerical reasons. This modifies the generalized Ohm's law such that

$$n_e q_e \mathbf{E} = -\mathbf{J}_e \times \mathbf{B} + \nabla p_e + \nu \nabla \cdot \nabla \mathbf{J} \tag{2.34}$$

where ν is the hyper-resistivity parameter, which is most often held constant through all space and time. Hyper-resistivity includes another time-scale,

$$\Delta t < \frac{\Delta x^2}{2.0\nu} \tag{2.35}$$

that needs to be checked for stability of explicit methods.

The Hall-MHD model does not include electron inertia, so it does not describe electron demagnetization. However, it does apply to regimes where the Larmor radius is much smaller or much larger than the scale-length of interest.

2.3.4 Reduction to Ideal-MHD

Hall-MHD is further reduced to Ideal-MHD by neglecting the Hall and diamagnetic drift terms. Ideal-MHD is described by,

$$\begin{aligned}
&\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \\
&\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u} + P\bar{\mathbf{I}} - \frac{\mathbf{B}\mathbf{B}}{\mu_0} + \frac{B^2}{2\mu_0}\bar{\mathbf{I}}) = 0 \\
&\frac{\partial \varepsilon}{\partial t} + \nabla \cdot \left[\left(\varepsilon + P + \frac{B^2}{2\mu_0} \right) \mathbf{u} - \frac{(\mathbf{B} \cdot \mathbf{u})}{\mu_0} \mathbf{B} \right] = 0 \\
&\frac{\partial \mathbf{B}}{\partial t} + \nabla \cdot (\mathbf{u}\mathbf{B} - \mathbf{B}\mathbf{u}) = 0
\end{aligned}$$

which results in Ohm's law reducing to,

$$\mathbf{E} = -\mathbf{u} \times \mathbf{B}. \tag{2.36}$$

where $\mathbf{u}$ represents the bulk fluid velocity. This reduction of the generalized Ohm's law results because of the small Larmor radius assumption that estimates that the ratio of the Hall term to the $\mathbf{u} \times \mathbf{B}$ term goes as

$$\frac{\frac{\mathbf{J}}{ne} \times \mathbf{B}}{\mathbf{u} \times \mathbf{B}} \sim \frac{r_{Li}}{L} \tag{2.37}$$

where r_{Li} is the ion Larmor radius and L is the scale-length of fluid motion. A similar approximation is applied to eliminate the diamagnetic drift term from Eq. (2.20) assuming the Hall term and diamagnetic drift term are approximately the same order of magnitude. Also, ideal-MHD assumes infinite conductivity.

The MHD model is used for a number of problems involving large-scale plasma mo-

tions. However, it is not applicable in regimes where two-fluid effects become significant such as turbulence through microinstabilities, or in regions where Larmor radius effects or high frequency modes become important. Ideal-MHD is not applicable in regimes where ion demagnetization becomes important such as in Hall thrusters. The tearing and tilt instabilities in an FRC (field reversed configuration) are other examples of applications where ideal-MHD is not applicable. The tearing mode requires breaking up and reconnection of field lines which cannot happen with infinite conductivity. Ideal-MHD is highly unstable to the tilt mode instability while the instability is believed to be stabilized by kinetic effects.

The dispersion diagram for ideal-MHD sits near the origin of the Hall-MHD dispersion diagram in Fig. 2.4 at the $L \gg \delta_i$ limit.

2.4 Advantages of Full Two-Fluid Model

In summary, the two-fluid plasma model retains the two-fluid physics by including the Hall and diamagnetic-drift terms that are neglected in single fluid plasma models. The two-fluid plasma model is different from the commonly used Hall-MHD model in three specific ways. The two fluid model includes

- finite electron mass,
- finite speed of light, and
- non-neutral effects.

Retaining the electron inertia allows the whistler wave to reach an asymptote at the electron cyclotron frequency and this allows for ease of computation. Since the asymptote for the whistler wave now lies at the electron plasma frequency, there is no need to set an artificial resistivity or viscosity to choose a cut-off frequency for the whistler wave. This allows for two-fluid solutions to be obtained without the effect of artificial parameters introduced for purposes of numerical difficulties.

Secondly, retaining the speed of light in Maxwell's equations allows displacement currents to be captured in the system. The relevance of this term is often argued and it has

been pointed out that not including relativistic effects makes this model incomplete. However, the interest lies in capturing the physics associated with the ion and electron fluids. Electromagnetic waves can be captured without including relativistic effects and having waves in the system that propagate at speeds close to the speed of light does not require relativistic effects. Relativistic effects are important if the fluid motions occur at speed of light time-scales, which is not the case for the simulations presented in this dissertation. The fluid physics at the very high frequency electromagnetic time scales is not presently of interest for the applications concerned in this dissertation.

In most plasma fluid models commonly used in the plasma physics community, a parallel electric field cannot exist unless there is a resistivity included in the equation system. In this context, parallel and perpendicular are in relation to the magnetic field. To elaborate, Ohm's law, $\mathbf{E} = -\mathbf{u} \times \mathbf{B}$, does not allow for a parallel electric field unless an $\eta\mathbf{J}$ term is also included. This implies that in most reduced fluid models, a parallel electric field can exist only if there is a parallel current and resistivity. In the two-fluid plasma model, including Ampere's law with the displacement currents allows a parallel electric field to be present even if there is no parallel current. Also, the two-fluid model here is idealized, i.e. no explicit resistivity, nevertheless, a parallel electric field can exist and can be captured in the system. Yet another advantage of including the displacement current term comes from its potential application to coil boundary conditions or similar scenarios. In reduced fluid models, coil boundaries are included by specifying a $\mathbf{B}_{coil}$ and a $\mathbf{B}_{plasma}$ separately. In order to couple the two magnetic fields, a region of high resistivity, η, needs to be assumed between the coil boundary and the plasma configuration. The impact of doing this on the physics of the system could potentially introduce artificial artifacts because there needs to be a region of low density plasma between the coil and the plasma configuration that carries a current with high resistivity. For the two-fluid plasma model, retaining the displacement current term allows for the coil boundary conditions to be introduced within the plasma configuration self-consistently using Maxwell's equations.

Including the speed of light and consequently the displacement currents allows for non-neutral effects to be captured in the plasma. Single fluid MHD and even Hall-MHD cannot capture non-neutral effects. Hall-MHD includes the Hall term, the diamagnetic drift term,

and in some cases electron inertia, however, an inherent assumption of Hall-MHD is neutrality. The neutrality assumption does not allow local charge separation as a result of which potentially large local electric fields are missed. Non-neutral effects are important in plasmas and the full two-fluid plasma model is required to capture them. Even with an ideal full two-fluid plasma model (i.e. no explicit resistivity or transport), an anomalous resistivity can be observed. As will be presented later in this dissertation, the field-reversed configuration (FRC) is an example where such anomalous resistivity has been observed in experiments.

2.5 Non-Ideal Full Two-Fluid Plasma Model

A number of problems are in the collisionless regime and can be solved with the ideal two-fluid plasma model. Examples of these are the electromagnetic plasma shock, collisionless magnetic reconnection, etc. In order to have a physically relevant model when the system is in a collisional regime, it is important to take effects of transport into account through the inclusion of friction forces, thermal forces, viscosity, thermal equilibration, and heat flux in the fluid equations. Re-writing Eqs.(2.2-2.4) with transport coefficients based on Braginskii's derivations[17], the non-ideal two-fluid equations are

$$\frac{\partial \rho_e}{\partial t} + \nabla \cdot (\rho_e \mathbf{u}_e) = 0 \tag{2.38}$$

$$\frac{\partial \rho_e \mathbf{u}_e}{\partial t} + \nabla \cdot (\rho_e \mathbf{u}_e \mathbf{u}_e + p_e \mathbf{I}) = \frac{\rho_e q_e}{m_e}(\mathbf{E} + \mathbf{u}_e \times \mathbf{B}) - \nabla \cdot \overleftrightarrow{\mathbf{\Pi}}_e + \mathbf{R}_{ei} \tag{2.39}$$

$$\frac{\partial \epsilon_e}{\partial t} + \nabla \cdot ((\epsilon_e + p_e)\mathbf{u}_e) = \frac{\rho_e q_e}{m_e}\mathbf{u}_e \cdot \mathbf{E} - \nabla \cdot \mathbf{q}_e - \overleftrightarrow{\mathbf{\Pi}}_e : \nabla \mathbf{u}_e + Q_e \tag{2.40}$$

with subscript e for electrons and

$$\frac{\partial \rho_i}{\partial t} + \nabla \cdot (\rho_i \mathbf{u}_i) = 0 \tag{2.41}$$

$$\frac{\partial \rho_i \mathbf{u}_i}{\partial t} + \nabla \cdot (\rho_i \mathbf{u}_i \mathbf{u}_i + p_i \mathbf{I}) = \frac{\rho_i q_i}{m_i}(\mathbf{E} + \mathbf{u}_i \times \mathbf{B}) - \nabla \cdot \overleftrightarrow{\mathbf{\Pi}}_i - \mathbf{R}_{ei} \tag{2.42}$$

$$\frac{\partial \epsilon_i}{\partial t} + \nabla \cdot ((\epsilon_i + p_i)\mathbf{u}_i) = \frac{\rho_i q_i}{m_i}\mathbf{u}_i \cdot \mathbf{E} - \nabla \cdot \mathbf{q}_i - \overleftrightarrow{\mathbf{\Pi}}_i : \nabla \mathbf{u}_i + Q_i \tag{2.43}$$

with subscript i for ions. Here, $\mathbf{R}_{ei}$ is the momentum transfer term, $\overleftrightarrow{\mathbf{\Pi}}$ is the viscous stress tensor, $\mathbf{q}$ is the heat flux, and Q is the thermal equilibration between the species. The $\overleftrightarrow{\mathbf{\Pi}} : \nabla\mathbf{u}$ term represents the heat generated due to viscosity. In this dissertation, gyroviscosity is neglected for simplicity therefore, an absence of magnetic field is assumed for computing the viscous stress tensor. Each of these transport terms are identical to those described in Ref.[17]. These transport coefficients are expanded in Appendix A in order to understand their implementation in WARPX.

When transport is included in the system, the electron and ion collision frequencies must be additionally taken into account. In regimes where the collisionality is small, the explicit time-step is set by the electron or ion plasma frequencies, the electron or ion cyclotron frequencies, or the speed of light. However, in regimes of high collisionality, the electron and ion collision frequencies could restrict the explicit time-step. The collision frequency is given by

$$\nu_e = \frac{n_e q_e^4 \ln\Lambda}{12\pi^{3/2}\epsilon_0^2 m_e^{1/2}(kT_e)^{3/2}} \tag{2.44}$$

$$\nu_i = \frac{n_i q_i^4 \ln\Lambda}{12\pi^{3/2}\epsilon_0^2 m_i^{1/2}(kT_i)^{3/2}} \tag{2.45}$$

where subscripts $e,\ i$ represent electrons and ion respectively, n represents the number density of the species and $\ln\Lambda = \ln(12\pi n\lambda_D^3)$ where λ_D is the Debye length of the species. Often, $\ln\Lambda = 10$ is used since it is representative of the plasma parameter for most configurations. In addition to the two-fluid time-scales described in Eq. (2.16), the explicit time-scale is further affected by the collision frequencies when using transport coefficients

$$\Delta t = min\left(\Delta t_{ITF}, \frac{0.1}{\nu_e}, \frac{0.1}{\nu_i}\right). \tag{2.46}$$

where Δt_{ITF} is the minimum time-step from the ideal two-fluid plasma model.

Collisions distort the trajectory of a particle and alter its velocity. In a magnetic field, collisions disturb the regular gyro-motion of the particles and renew their trajectories at each collision time. In regions of low magnetic field, $\omega_{ce}/\nu_e \ll 1$ whereas in regions of large

magnetic field, $\omega_{ce}/\nu_e \gg 1$. In a strong magnetic field, the magnetic field has a large effect on the transport properties in the transverse direction. Along the magnetic field, however, particles move freely and the transport in the longitudinal direction is similar to the case of $\mathbf{B} = 0$.

The ions are more massive than the electrons, and this difference in the masses makes the individual species reach equilibrium separately before the two species come to equilibrium with each other. This is given by the ratio $\tau_{ee} : \tau_{ii} : \tau_{ei} = 1 : \sqrt{m_i/m_e} : m_i/m_e$ for $T_e = T_i$.

A brief overview of the transport terms is provided here based on the detailed discussion provided by Braginskii in Ref.[17].

2.5.1 Friction Force

The friction force term, $\mathbf{R}_U$, accounts for the inter-species collisions of electrons with ions where both ions and electrons lose momentum as a result of the friction force. However, since the ions are much more massive than the electrons, the electrons lose more momentum to the ions in the process. This results in a frictional force that is exerted on the electrons by the relatively stationary ions, and it is equal and opposite to the force exerted by the electrons on the ions. This term basically accounts for the $\eta\mathbf{j}$ electrical resistivity force. If electron-electron collisions occur more frequently than the electron-ion collisions, the $\mathbf{u}_\parallel$ term contribution to the friction force would decrease and become negligible. In the presence of a strong magnetic field, the longitudinal resistivity is smaller than the transverse resistivity, $\eta_\parallel < \eta_\perp$ since the coefficient of friction is smaller in the direction of longitudinal current compared to the direction of transverse current.

2.5.2 Thermal Force

The thermal force term, $\mathbf{R}_T$, arises when there is an electron temperature gradient in the presence of a strong magnetic field. The temperature gradient creates an anisotropy in the friction forces due to electron gyration around the magnetic field lines and the collisions of the electrons with ions generates a thermal force.

2.5.3 Viscosity

The viscosity for each species $\mathbf{\Pi}_s$ included in this dissertation is assumed in the absence of a magnetic field, therefore, it excludes gyroviscosity. It corresponds to a random-walk diffusion of momentum with a frequency of ν. In the absence of a magnetic field the rate-of-strain tensor obtained is symmetric because of the inclusion of velocity gradients along with their transpose for all terms of the tensor, and the inclusion of velocity divergence for the diagonal terms of the tensor. This results in a symmetric viscous stress tensor. For ion and electron fluids with similar temperatures and densities, the diffusivity is given by $D \sim \eta/\rho$, where D is the diffusivity, $\eta \sim 1/\sqrt{m}$ is the viscosity coefficient, m is the mass, and ρ is the mass density. The electron diffusivity exceeds the ion diffusivity by $\sqrt{m_i/m_e}$ for similar temperatures. However, the ion viscosity is greater than the electron viscosity by the same factor of $\sqrt{m_i/m_e}$. This implies that the viscosity of a plasma is primarily governed by the ions. Ions have large momentum owing to their large masses, therefore, the diffusion of momentum i.e. viscosity, is dominated by the ions.

2.5.4 Viscous Heating

In the presence of diffusion, such as viscosity, there is dissipation of energy in the form of heat[17]. In a plasma this is sometimes called gyrorelaxational heating. The viscous stress tensor and the velocity gradient of each of the electron and ion species are reduced to a scalar viscous heating term in the energy equation, $\overleftrightarrow{\mathbf{\Pi}} : \nabla \mathbf{u}$.

2.5.5 Heat Flux

The heat flux of electrons and ions, $\mathbf{q}_e$ and $\mathbf{q}_i$, is included in the model and corresponds to a random-walk heat diffusion with a frequency ν. The first component of the electron heat flux is $\mathbf{q}_U^e$ and it describes the terms in the heat flux that are proportional to the relative velocity between the species. The current along a magnetic field is primarily carried by electrons owing to their higher velocities. Since the energy fluxes are not balanced due to the acceleration of electrons, heat flow is present such that the flux is from the fast electrons to the slow electrons. Across a magnetic field, the friction force of the ions on electrons

accelerates the electrons over half a rotation while retarding them over the other half. The difference in the electron energies in the accelerated and retarded regions accounts for heat flux across field lines.

In addition to the current driven heat flux term, thermal heat conduction within the electrons provides the second component of electron heat flux, $\mathbf{q}_T^e$. There is a corresponding term for the ion heat flux, $\mathbf{q}_T^i$, Motion across a magnetic field line is displaced by approximately a Larmor radius between collisions rather than the mean free path in the longitudinal direction. In the direction parallel to the magnetic field, the electron thermal conductivity is greater than the ion thermal conductivity by a factor of $\sqrt{m_i/m_e}$ assuming similar species temperatures. However, in the transverse direction, the ion thermal conductivity is greater than the electron thermal conductivity by the same factor.

2.5.6 Heat Generation

The thermal equilibration term is the result of heat generation due to collisions of electrons with ions. For electrons, heat is generated as a result of the frictional force exerted by the ions on the electrons. This corresponds to an Ohmic heating term, $\mathbf{R}_U \cdot \mathbf{u}$ and a Joule heating term, $\mathbf{R}_T \cdot \mathbf{u}$. The fraction of the Ohmic heating and Joule heating acquired by the ions goes as a ratio of the mass $\sim m_e/m_i$ and consequently, can be neglected for the massive ions.

The collisions of the electrons with the ions transfer energy as a function of the mass ratio. This results in an energy exchange term, Q_Δ, that acts as a thermal equilibration term and is a function of species temperatures as well. If the ions and electrons have the same temperature, this term is 0. In the absence of all other transport terms, this thermal equilibration term drives the electrons and ions to reach an equilibrium at the same temperature.

2.5.7 Applicability

The transport terms described and used in the dissertation are applicable under the following conditions. Firstly, the ions are required to be more massive than the electrons to satisfy

the conditions under which the transport terms have been approximated. Secondly, the time variation must be slower than the collision time, i.e. the plasma quantities must not change significantly within a collision time. Thirdly, spatial variations are expected to be slow enough to satisfy the following conditions: the characteristic scale length over which quantities change significantly parallel to the magnetic field is expected to be larger than the mean free path, and the characteristic scale length over which quantities change significantly perpendicular to the magnetic field is expected to be larger than the Larmor radius. Also, the presence of instabilities in a plasma can limit applicability of the transport terms. The small spatial scale phenomena such as the behavior within a shock or small temporal scale phenomena such as dynamics within a Debye sphere violate the applicability of this model and are not being explored in this dissertation. The instabilities explored in this dissertation satisfy the temporal and spatial scales required for the applicability of this model. Therefore, since the scales are within the regime of applicability, this model can be used to study two-fluid instabilities in a plasma.

Chapter 3

NUMERICAL METHODS AND IMPLEMENTATION DETAILS

3.1 High-Resolution Wave Propagation Method

The high resolution wave propagation method can be applied to balance laws of the form Eq. (1.1). This section describes the wave propagation algorithm and the higher order corrections that are implemented to increase the order of accuracy from first to second. LeVeque describes this method in greater detail in [18, 19, 6]. This method is described in detail for two-fluid plasma equations in Ref.[1]. The wave propagation scheme belongs to the class of *Godunov methods* which rely on the solution of Riemann problems. The essential idea is as follows. The domain is discretized into cells and the solution in each cell is assumed to be represented by averages. At each cell interface the solution is reconstructed and will, in general, be discontinuous. This discontinuity is used as an initial condition for a Riemann problem. Figure 3.1 illustrates the discontinuity in the conserved variable at the cell interface, while the flux is continuous. The solution of the Riemann problem gives the conserved variables at the interface which are then used to compute numerical fluxes. Once the fluxes are known the solution in each cell is updated by tallying how much flux flows into the cell.

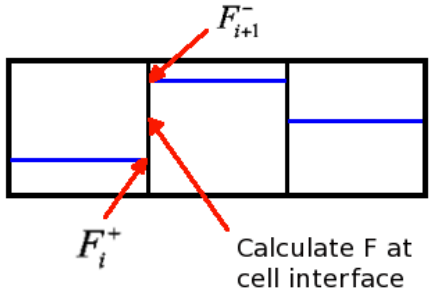

Figure 3.1: The conserved variables are allowed to be discontinuous at each cell interface while the flux through the interface is continuous.

In one dimension, second order accuracy can be achieved by performing a linear reconstruction of the waves needed to compute the numerical fluxes at the cell interface. In multiple dimensions high resolution *transverse corrections* are included which account for flow that is transverse to the coordinate axes. After solving the Riemann problem at each cell interface to determine the positive- and negative-going fluctuations, a second transverse Riemann problem is solved to compute the transverse fluctuations as detailed in Ref.[6]. With these transverse corrections the method is formally second order accurate and is stable to a CFL number of unity.

3.1.1 First Order Scheme

The scheme is detailed in one spatial dimension here and it can be extended to account for multiple dimensions. In one dimension, a homogeneous hyperbolic equation is written as

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} = 0, \tag{3.1}$$

where $\mathbf{Q}$ represents the conserved variable and $\mathbf{F}$ represents the flux in the X direction. The domain that this equation is discretized on is defined within the boundaries, $[x_a, x_b]$. The cells are introduced as $I_i = [x_{i-1/2}, x_{i+1/2}]$, for each cell interval, where $x_{i-1/2}$ and $x_{i+1/2}$ are the coordinates at the left and right edges of each cell. The cell center is defined as (x_i), where $x_i \equiv (x_{i-1/2} + x_{i+1/2})/2$. Now, taking the conservation law defined in Eq. (3.1), and integrating it over cell I_i from time t_n to t_{n+1} gives the update formula

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x}\left([\mathcal{F}]_{i+1/2}^{n+1/2} - [\mathcal{F}]_{i-1/2}^{n+1/2}\right) \tag{3.2}$$

where Q_i^n represents the value of the conserved variable based on the cell average,

$$Q_i^n \approx \frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} \mathbf{Q}(x,t)dx. \tag{3.3}$$

Here $\Delta x \equiv x_{i+1/2} - x_{i-1/2}$, $\Delta t \equiv t_{n+1} - t_n$ and the *numerical flux* at each of the cell interfaces, $[\mathcal{F}]$, is defined as

$$[\mathcal{F}]_{i-1/2}^{n+1/2} \approx \frac{1}{\Delta t} \int_{t_n}^{t_{n+1}} \mathbf{F}\big(Q(x_{i-1/2}, t), x_{i-1/2}\big) dt. \tag{3.4}$$

It is seen that the expression in Eq. (3.2) resembles the central difference formula. This equation is a general update formula for finite volume schemes. The numerical flux can be updated by using several different approaches and the choice of the flux update forms the basis for the various finite volume schemes. This method approximates the value of the conserved variable in a given cell as the cell average. Therefore, for cells sharing a given interface, the value at that interface will be discontinuous in general. A Riemann problem needs to be solved at each cell edge to determine the numerical flux at each interface.

For the homogeneous, hyperbolic equation described by:

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} = 0, \tag{3.5}$$

the Riemann problem is an initial value problem described by the initial conditions $\mathbf{Q}(x < 0, 0) = \mathbf{Q}_l$ and $\mathbf{Q}(x > 0, 0) = \mathbf{Q}_r$, where $\mathbf{Q}_{l,r}$ are constant vectors. If a hyperbolic equation system is linear then the Riemann problem has exact solutions. If it is a nonlinear equation set, then valid solutions around $x = 0$ (i.e. at the interface) can be obtained for short time intervals by introducing a linearization. As mentioned previously, the linear hyperbolic equation system described by Eq. (3.5) can be written in the form

$$\frac{\partial \mathbf{Q}}{\partial t} + \mathbf{A}\frac{\partial \mathbf{Q}}{\partial x} = 0, \tag{3.6}$$

where the flux Jacobian, $\mathbf{A}$, is constant for the linear system described here. This is a hyperbolic system, so it is known that all the eigenvalues of the flux Jacobian are real and the eigenvectors are assumed to be linearly independent, and in this case, orthonormal. In order to obtain the wave equations for the wave propagation method, Eq. (3.5) can be

multiplied with the left eigenvectors, l^p, to obtain

$$\frac{\partial w^p}{\partial t} + s^p \frac{\partial w^p}{\partial x} = 0, \tag{3.7}$$

where $w^p \equiv l^p \cdot \mathbf{Q}$. To solve the Riemann problem for such linear systems exactly, $w^p(x,t)$ needs to be determined. This is done by obtaining the initial condition for $w_0^p(x,0)$, where $w_0^p(x) = l^p \cdot \mathbf{Q}(x,0)$. Once this is done, $\mathbf{Q}(x,t) = \sum_p w^p r^p$ is solved to obtain the solutions. r^p, l^p and s^p represent the right eigenvectors, the left eigenvectors and the eigenvalues of $\mathbf{A}$.

The wave propagation method involves solving the Riemann problem at each cell interface and this solution is used to arrive at the following approximation to the numerical fluxes by accounting for the right- and left-going fluctuations.

$$[\mathcal{F}]_{i-1/2} = \frac{1}{2}\big(\mathbf{F}_i + \mathbf{F}_{i-1}\big) - \frac{1}{2}\big(\mathcal{A}^+ \Delta Q_{i-1/2} - \mathcal{A}^- \Delta Q_{i-1/2}\big). \tag{3.8}$$

Taking this expression and plugging it in the update formula defined by Eq. (3.2), the following equation is obtained:

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x}\left[\mathcal{A}^+ \Delta Q_{i-1/2} + \mathcal{A}^- \Delta Q_{i+1/2}\right]. \tag{3.9}$$

The $\mathcal{A}^{\pm}\Delta Q_{i-1/2}$ terms here are called the *fluctuations* and they are described by

$$\mathcal{A}^- \Delta Q_{i-1/2} = \sum_{p:s^p_{i-1/2}<0} \mathcal{Z}^p_{i-1/2} + \frac{1}{2} Z_{i-1/2} \tag{3.10}$$

$$\mathcal{A}^+ \Delta Q_{i-1/2} = \sum_{p:s^p_{i-1/2}>0} \mathcal{Z}^p_{i-1/2} + \frac{1}{2} Z_{i-1/2} \tag{3.11}$$

where $p : s^p_{i-1/2} < 0$ represents all the negative eigenvalues, to account for the negative-going fluctuations from the $i - 1/2$ interface. Likewise, $p : s^p_{i-1/2} > 0$ represents all the positive eigenvalues at the $i - 1/2$ interface.

$$\mathcal{Z}^p_{i-1/2} = l^p_{i-1/2} \cdot \big(\mathbf{F}_i - \mathbf{F}_{i-1}\big) r^p_{i-1/2} \tag{3.12}$$

and

$$Z_{i-1/2} = \sum_{p:s^p_{i-1/2}=0} \mathcal{Z}^p_{i-1/2}. \tag{3.13}$$

The above expression is the *F-Wave* method as developed by LeVeque[6]. If using the *Q-Wave* method, then $\mathcal{Z}^p_{i-1/2}$ will have the jump in the conserved variable, $\mathbf{Q}$, instead of the jump in flux, $\mathbf{F}$. Most of the simulations performed in this dissertation use the *Q-Wave* method with Roe-averaged fluxes[20]. The approximate Riemann solver of Roe is conservative and less computationally expensive than solving an exact Riemann problem even for non-linear problems with discontinuous solutions. Treating the fluctuations in Eq. (3.9), the identity,

$$\mathcal{A}^- \Delta Q_{i-1/2} + \mathcal{A}^+ \Delta Q_{i-1/2} = \sum_p \mathcal{Z}^p_{i-1/2} = \mathbf{F}_i - \mathbf{F}_{i-1} \tag{3.14}$$

follows from the definition of $\mathcal{Z}^p_{i-1/2}$ described by Eq. (3.12) (in this case for the *F-Waves*). If using *Q-Waves*, then the $\mathbf{F}$ is replaced with $\mathbf{Q}$ in the flux difference expression defined by Eq. (3.14). At each cell interface, the right and left eigenvectors, $r^p_{i-1/2}$, $l^p_{i-1/2}$ and the eigenvalues $s^p_{i-1/2}$ are computed from the flux Jacobian. The eigensystem used for linear systems is constant throughout, however, for nonlinear systems special treatments, i.e. an appropriate averaging, must be performed (such as Roe averaging). If Roe averages are not used, the scheme still provides the appropriate solution and continues to remain conservative when *F-Waves* are used as long as the method chosen to obtain the eigensystem is consistent throughout. To elaborate, using F-Waves, one can consistently choose arithmetic averages or just the left or right value for the conserved variables to arrive at the values for the fluctuations, and the result will continue to remain conservative as long as the same treatment is maintained throughout the domain and through all times. With Q-Waves however, an appropriate averaging scheme such as Roe averaging must be chosen for the solution to remain conservative. If Roe averages are used then a conservative solution can be obtained using either *F-Waves* or *Q-Waves*. The method described in this section can be applied to a variety of hyperbolic problems including those that have spatially dependent

conserved variables and flux functions.

3.1.2 High Resolution Corrections

The algorithm detailed in the previous section is only first order accurate and to increase the order of accuracy from first to second, certain corrections need to be performed. This is done by taking a Taylor series expansion of the conserved variables. The second order terms from this expansion are retained. The new, second order accurate algorithm is now described by

$$\begin{aligned} Q_i^{n+1} = Q_i^n &- \frac{\Delta t}{\Delta x}\left[\mathcal{A}^+\Delta Q_{i-1/2} + \mathcal{A}^-\Delta Q_{i+1/2}\right] \\ &- \frac{\Delta t}{\Delta x}\left([\tilde{\mathcal{F}}]_{i+1/2} - [\tilde{\mathcal{F}}]_{i-1/2}\right), \end{aligned} \tag{3.15}$$

where the new term introduced here, $[\tilde{\mathcal{F}}]_{i-1/2}$, is the *correction flux* given by

$$[\tilde{\mathcal{F}}]_{i-1/2} = \frac{1}{2}\sum_p \text{sign}(s^p_{i-1/2})\left(1 - \frac{\Delta t}{\Delta x}|s^p_{i-1/2}|\right)\mathcal{Z}^p_{i-1/2}. \tag{3.16}$$

This correction increases the order of the high resolution wave propagation method and makes it equivalent to the standard Lax-Wendroff algorithm[6]. As a result of this increased accuracy to second order in Eq. (3.15), the algorithm can experience severe oscillations near discontinuities similar to the Lax-Wendroff algorithm.

3.1.3 Limiters

Limiters can be applied in regions with sharp discontinuities. This limits the formal order of accuracy of the algorithm to first in these regions by replacing $\mathcal{Z}^p_{i-1/2}$ in Eq. (3.16) by a *limited* wave $\tilde{\mathcal{Z}}^p_{i-1/2} = \mathcal{Z}^p_{i-1/2}\phi(\theta^p_{i-1/2})$. Here, $\phi(\theta)$ is the limiter function that is chosen

$$\theta^p_{i-1/2} \equiv \frac{\mathcal{Z}^p_{I-1/2}\cdot\mathcal{Z}^p_{i-1/2}}{\mathcal{Z}^p_{i-1/2}\cdot\mathcal{Z}^p_{i-1/2}} \tag{3.17}$$

with $I = i - 1$ if $s^p_{i-1/2} > 0$ and $I = i + 1$ if $s^p_{i-1/2} < 0$. Some of the limiters that have been used with the wave propagation algorithm include the min-mod limiter,

$$\phi(\theta) = \text{minmod}(1, \theta), \tag{3.18}$$

the Superbee limiter,

$$\phi(\theta) = \max(0, \min(1, 2\theta), \min(2, \theta)), \tag{3.19}$$

the Van Leer limiter,

$$\phi(\theta) = \frac{\theta + |\theta|}{1 + |\theta|}, \tag{3.20}$$

and the monotonized centered limiter,

$$\phi(\theta) = \max(0, \min((1 + \theta)/2, 2, 2\theta)). \tag{3.21}$$

To extend these corrections to multi-dimensions, high resolution *transverse corrections* need to be performed in a similar manner to account for flow that is transverse to the coordinates.

3.1.4 Source Term Handling

The source terms are handled by performing a source term splitting method which involves solving the homogeneous equation separately from the ordinary differential equation (ODE). The ODE is the part that accounts for the source terms and it is described by:

$$\frac{d\mathbf{Q}}{dt} = \mathbf{S}. \tag{3.22}$$

For purposes of achieving higher order accuracy, i.e. second order accuracy in this case, the ODE described by Eq. (3.22) is first advanced with a time step of $\Delta t/2$. Then the homogeneous equation is solved over a full time step of Δt following which, the ODE is advanced again by another half time step of $\Delta t/2$. The results from each time advance are

used as initial conditions for the time steps that follow. To solve the ODE, any ODE scheme can be used and the one chosen here is that of the fourth order Runge-Kutta scheme. The source term splitting method detailed here is called *Strang splitting*[21].

For linear ODEs, taking the source Jacobian can determine the solution type. If the eigenvalues of the Jacobian are real, then the solution grows or decays for positive and negative eigenvalues respectively. For imaginary eigenvalues, however, the solution is oscillatory and the frequency of these harmonics is governed by the magnitude of the eigenvalues. The real component of the eigenvalues can lead to growing or decaying oscillations.

If there is a hyperbolic, homogeneous part to the equation system, then disturbances propagate as waves with finite speeds in the medium. Capturing these wave-propagations can play a significant role in understanding the physics described by the equation set. The source terms need to be treated appropriately because they could significantly affect the solution.

The source terms of the two-fluid system are particularly challenging as they represent undamped oscillations, i.e., the Jacobian of the source terms has purely imaginary eigenvalues. Such sources add physical dispersion to the system which can be difficult to resolve. For a discussion relevant to the two-fluid system see Ref.[1].

3.2 Runge-Kutta Discontinuous Galerkin Method (RKDG)

The RKDG method is a finite element method as opposed to the finite volume presented in Sec. 3.1. The RKDG method achieves higher spatial order by expanding the solution in polynomial basis functions. The balance law described in Eq. (1.1) is multiplied by a set of basis functions and is integrated over the element. The conserved variable is defined as a linear combination of the basis functions. In this paper tensor products of the Legendre polynomials are chosen as basis functions. This allows the construction of methods of arbitrary spatial order. Riemann problems are solved at each interface to compute the interface fluxes needed in the algorithm. Figure 3.2 illustrates a polynomial discretization of the solution within each cell. Just like the wave propagation method, the conserved variable is allowed to be discontinuous at each cell interface, but the flux is continuous across the interface. In general very simple approximate solvers can be used. In contrast,

the wave propagation method needs a much more accurate Riemann solver.

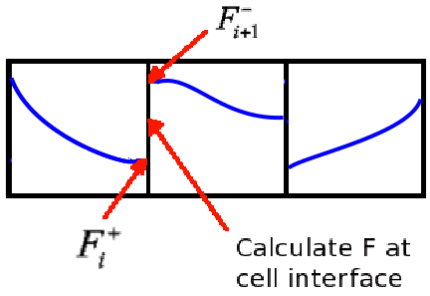

Figure 3.2: The conserved variables are allowed to be discontinuous at each cell interface while the flux through the interface is continuous.

An overview of the development of discontinuous Galerkin (DG) methods is provided by Cockburn, Karniadakis and Shu in Ref.[22]. DG methods were originally developed in the framework of neutron transport equations by Reed and Hill[23] for solving linear hyperbolic equations. The DG method was then applied to non-linear advection-dominated hyperbolic systems by Cockburn and Shu[24] who used the high-order total variation diminishing (TVD) Runge-Kutta time integration that was developed by Shu[25] with DG and developed local projection slope-limiters to balance the spurious oscillations in regions of sharp gradients. This was extended to higher spatial orders and multiple dimensions for applications to non-linear systems such as the Euler equations. Next came the evolution of the DG method for applications to advection-diffusion equation systems[26], such as the Navier-Stokes equations. One method to incorporate the second-order diffusion terms was implemented by Bassi and Rebay[27]. The diffusion terms cannot be directly applied to the weak formulation in the DG method without introducing inconsistencies, so auxiliary variables are introduced to represent the gradients of the conserved variables that appear in the diffusion term. These auxiliary variables are coupled to the hyperbolic system and are solved using the DG method without a time advance. Oden, Babuŝka and Baumann[28] provide an alternative approach to solve convection-diffusion problems that overcomes the disadvantage of introducing auxiliary variables and extra equations by using a method that resembles hybrid and interior penalty methods. This DG method involves defining discon-

tinuous approximations in partitions of the domain where both solution values and fluxes are discontinuous across the interfaces. This method is free of the penalty parameter used by Percell and Wheeler[29] that was introduced to extend the global element method[30, 31] to solve time-dependent diffusion problems where the choice of the penalty parameter could lead to bad conditioning of matrices or loss of conservation.

The DG method has also been applied to Maxwell's equations[32, 33] where the electromagnetic oscillations need to be appropriately resolved and to single-fluid MHD equations[34] where the plasma waves need to be appropriately resolved. The application of DG methods to nonlinear dispersive equations, specifically the Korteweg-de Vries equation, is explored in Ref.[35] where DG methods are shown to be advantageous in the presence of rapid oscillations and are capable of simultaneously capturing oscillations and discontinuous fronts in the solution. Zhang and Shu have compared the discontinuous Galerkin method to spectral finite volume methods in Ref.[36] for linear one-dimensional hyperbolic equations and state that the spectral finite volume method has larger errors than the DG method on the same mesh.

A 3^{rd} order total variation diminishing Runge-Kutta method is used for the time integration as discussed in Refs.[26]. This makes the RKDG method an explicit finite element method. The time step is restricted for numerical stability, CFL$\leq 1/(2p-1)$, where p is the spatial order that is determined based on the selection of the polynomial basis. The DG method presented and used in this dissertation is a modal DG scheme instead of a nodal DG scheme.

3.2.1 The Base Scheme

The discontinuous Galerkin (DG) scheme uses basis functions to represent the conserved quantities, so it assumes that the solution in each cell is a piecewise-polynomial. The basis-functions, v_r, for $r = 0, 1, \ldots$ are locally defined within each cell. Any function can be represented with the polynomial basis functions, and applying this idea to the conserved

quantities within each cell, it is seen that

$$\mathbf{Q} = \sum_{r=0}^{\infty} \mathbf{Q}_r v_r, \tag{3.23}$$

where $\mathbf{Q}_r$ are the expansion coefficients. The number of terms that are retained in this expansion is a representation of the spatial order of the solution. Orthogonal basis functions are selected,

$$\int_{I_i} v_r v_m dV = \Delta V C_r \delta_{rm}, \tag{3.24}$$

where ΔV is the volume of the cell (in 1-dimension it is Δx), C_r are normalization constants and δ_{rm} is the Kronecker-delta symbol. Using this orthogonality equation described by Eq. (3.24), and multiplying the orthogonal basis functions, v_m, with Eq. (3.23) gives,

$$\mathbf{Q}_r = \frac{1}{C_r \Delta V} \int_{I_i} \mathbf{Q} v_r dV, \tag{3.25}$$

which are the expansion coefficients that are used to project any function onto the basis function. Now applying this to the balance law described in Chapter 1, the equation is multiplied by v_r and integrated over the cell to get

$$\frac{\partial}{\partial t} \int_{I_i} v_r \mathbf{Q} dV + \int_{I_i} v_r \nabla \cdot \mathbf{F} dV = \int_{I_i} v_r \mathbf{S} dV. \tag{3.26}$$

Use integration by parts on the second term to get

$$\int_{I_i} v_r \nabla \cdot \mathbf{F} dV = \int_{\partial I_i} v_r \mathbf{F} \cdot \mathbf{n} dS - \int_{I_i} \mathbf{F} \cdot \nabla v_r dV. \tag{3.27}$$

The conserved variables are then expanded into basis functions using Eq. (3.23) to get

$$C_r \frac{d\mathbf{Q}_r}{dt} + \frac{1}{\Delta V} \int_{\partial I_i} v_r \mathbf{F} \cdot n dS - \frac{1}{\Delta V} \int_{I_i} \mathbf{F} \cdot \nabla v_r dV = \frac{1}{\Delta V} \int_{I_i} v_r \mathbf{S} dV. \tag{3.28}$$

for $r = 0, 1, \ldots$, where the second term on the left-hand-side of Eq. (3.28) represents the interface fluxes that are used to solve the Riemann problem at every cell interface. Rear-

ranging the time-dependent and spatially-dependent variables, Eq. (3.28) is written as

$$\frac{d\mathbf{Q}_r}{dt} = \mathcal{L}_r(\mathbf{Q}) \tag{3.29}$$

for $r = 0, 1, \ldots$, where $\mathcal{L}_r(\mathbf{Q})$ is the operator containing all the spatially dependent terms of the equations (flux and source computations),

$$\mathcal{L}_r(\mathbf{Q}) = -\frac{1}{\Delta V}\int_{\partial I_i} v_r \mathbf{F}\cdot n dS + \frac{1}{\Delta V}\int_{I_i} \mathbf{F}\cdot\nabla v_r dV + \frac{1}{\Delta V}\int_{I_i} v_r \mathbf{S} dV. \tag{3.30}$$

After computing $\mathcal{L}_r(\mathbf{Q})$, the ODE described by Eq. (3.29) needs to be solved for the expansion coefficients, Q_r using any standard ODE solver. The TVD Runge-Kutta time integration scheme is used here (second and third orders) which makes this DG method specifically a *Runge-Kutta Discontinuous Galerkin* (RKDG) scheme.

3.2.2 Runge-Kutta Time Integration

The 2^{nd} order total variation bounded (TVB) Runge Kutta method used in this dissertation is described by

$$\mathbf{Q}^1 = \mathbf{Q}^n + \Delta t \mathcal{L}_r(\mathbf{Q}^n) \tag{3.31}$$

$$\mathbf{Q}^{n+1} = \frac{1}{2}\mathbf{Q}^1 + \frac{1}{2}\mathbf{Q}^n + \frac{1}{2}\Delta t \mathcal{L}_r(\mathbf{Q}^1). \tag{3.32}$$

The TVD 3^{rd} order Runge-Kutta time integration[26] is described by

$$\mathbf{Q}^1 = \mathbf{Q}^n + \Delta t \mathcal{L}_r(\mathbf{Q}^n) \tag{3.33}$$

$$\mathbf{Q}^2 = \frac{3}{4}\mathbf{Q}^n + \frac{1}{4}(\mathbf{Q}^1 + \Delta t \mathcal{L}_r(\mathbf{Q}^1)) \tag{3.34}$$

$$\mathbf{Q}^{n+1} = \frac{1}{3}\mathbf{Q}^n + \frac{2}{3}(\mathbf{Q}^2 + \Delta t \mathcal{L}_r(\mathbf{Q}^2). \tag{3.35}$$

For the majority of applications in this dissertation the standard two-step 2^{nd} order Runge-Kutta scheme provides accurate solutions with computational efficiency.

Strong stability preserving Runge-Kutta (SSPRK) methods have been explored in recent

years and optimal SSP methods have been shown to provide higher temporal accuracy for linear and nonlinear problems with the ability to use larger time steps[37, 38]. While SSP time integration schemes could be easily extended for use with the DG method used in this dissertation, there is no benefit gained by using the higher temporal accuracy for the plasma problems explored here because of the relatively smooth temporal evolution. This is verified by exploring the two-fluid model with the 2-dimensional axisymmetric Z-pinch in following chapters using SSPRK methods to show no qualitative differences from the 3^{rd} order TVD RK scheme or even the 2^{nd} order TVB RK scheme. A 2^{nd} order TVB or a 3^{rd} order TVD Runge-Kutta method is well suited for time integration for the problems explored in this dissertation in terms of both accuracy and computational effort.

3.2.3 Selection of Basis Functions

The Legendre polynomials are commonly selected as basis functions for this algorithm because they are orthogonal. In 1-dimension, the basis functions would be

$$v_r(x) = P_r(\eta(x)), \tag{3.36}$$

where the interval, I_i, needs to be mapped to lie within the interval, $[-1, 1]$, because the Legendre polynomials are only defined within this interval. This mapping is done using

$$\eta(x) \equiv \frac{x - x_i}{\Delta x/2} \tag{3.37}$$

for x_i and Δx as described previously. The first few Legendre polynomials are

$$\{v_r\} = \{v_0, v_x, v_y\} \tag{3.38}$$

$$= \{1, \frac{x - x_i}{\Delta x/2}, \frac{y - y_i}{\Delta y/2}\} \tag{3.39}$$

for second order spatial accuracy. For third order spatial accuracy,

$$v_r = \{v_0, v_x, v_y, v_{xy}, v_{x^2}, v_{y^2}, v_{x^2y}, v_{xy^2}, v_{x^2y^2}\} \tag{3.40}$$

$$= \{1, v_x, v_y, v_x v_y, v_x^2 - \frac{1}{3}, v_y^2 - \frac{1}{3}, \left(v_x^2 - \frac{1}{3}\right) v_y, \left(v_y^2 - \frac{1}{3}\right) v_x, \left(v_x^2 - \frac{1}{3}\right)\left(v_y^2 - \frac{1}{3}\right)\} \tag{3.41}$$

The Legendre polynomials have a convenient property that makes them useful for the RKDG algorithm, they are orthogonal. They satisfy the property,

$$\int_{-1}^{1} P_n(x) P_m(x) dx = \frac{2}{2m+1}\delta_{mn}, \tag{3.42}$$

where the coefficients C_r are found to be $C_r = 1/(2r+1)$. Following the mapping, the value of the basis functions can be obtained at the cell interfaces,

$$v_r(x_{i\pm1/2}) = P_r(\pm1) = (\pm1)^r. \tag{3.43}$$

To handle the second and third terms on the right-hand-side of Eq. (3.30), the Legendre-Gauss quadrature (also known as Gaussian quadrature) is used so that these integrals can be numerically evaluated. Applying this to a 1-dimensional function, $Q(x)$, over interval I_i gives

$$\int_{I_i} Q(x) dx = \frac{1}{2}\int_{-1}^{1} Q(x(\eta)) d\eta = \frac{1}{2}\sum_j w_j \bar{Q}(\eta_j), \tag{3.44}$$

where $x(\eta) = \eta\Delta x/2 + x_i$, $\bar{Q}(\eta) \equiv Q(x(\eta))$ and w_j and η_j are weights and abscissa of the chosen Gaussian quadrature scheme. The choice of the Gaussian quadrature depends on the order of the basis polynomial used, i.e., it depends on the highest value of r selected in Eq. (3.23).

Using the Legendre polynomial basis functions in Eq. (3.30) for a 1-dimensional system,

$\mathcal{L}_r$ is now written as

$$\mathcal{L}_r(\mathbf{Q}) = -\frac{\mathbf{F}_{i+1/2} - (-1)^r \mathbf{F}_{i-1/2}}{\Delta x} + \frac{1}{\Delta x}\int_{-1}^{1} \frac{dP_r(\eta)}{d\eta}\bar{\mathbf{F}}d\eta + \frac{1}{2}\int_{-1}^{1} P_r \bar{\mathbf{S}} d\eta, \tag{3.45}$$

where $\bar{\mathbf{F}}(\eta) \equiv \mathbf{F}(Q(x(\eta), t))$ and $\bar{\mathbf{S}}(\eta) \equiv \mathbf{S}(Q(x(\eta), t))$. The interface fluxes are computed using the same method as the wave propagation method and the integrals in Eq. (3.45) are computed using Gaussian quadrature. This is easily extended to multiple dimensions.

3.2.4 General Implementation Details

A fully 3-dimensional RKDG method has been implemented in WARPX. The 3-dimensional implementation is a trivial extension of the 2-dimensional implementation using the appropriate number of polynomial basis function coefficients. The polynomial order chosen for the basis functions determines the spatial accuracy of the scheme and this corresponds to p^d coefficients for each variable where p is the polynomial order chosen and d is the dimensions.

Lax-Friedrich fluxes are used for the DG algorithm in order to provide additional dissipation to the dispersive numerical method. A Riemann flux can implemented in suitable situations, but this however, can be dispersive for a number of problems leading to negative density and negative pressure errors. Lax-Friedrich fluxes prove to be more robust and also preserve divergence better for Maxwell's equations. The Lax-Friedrich flux at each cell interface for a 1-dimensional scheme is calculated as

$$\mathbf{F} \cdot n = \frac{1}{2}(F_i^+ - F_{i+1}^-) \cdot n - \frac{1}{2}|\lambda|(Q_i^+ - Q_{i+1}^-) \cdot n \tag{3.46}$$

where $|\lambda|$ is the maximum characteristic speed (eigenvalue) in the equation system based on the average terms of the conserved variables at the cell centers i and $i+1$. Superscripts $+$ and $-$ represent values in the upper and lower edges of each cell. The higher spatial order of RKDG method makes it suitable for equation systems containing disparate speeds such as the two-fluid plasma model. Using Lax-Friedrich fluxes with such disparate speeds still provides a robust solution and a higher order accuracy than the wave propagation method for the same resolution as will be shown in later chapters of this dissertation.

An *effective resolution*, R, is defined for the DG method such that $R = hp^d$, h being the grid resolution, p being the spatial order specified based on the polynomial basis, and d being the number of dimensions. The *effective resolution* is specified for the DG method and not for the high-resolution wave propagation method because the degrees of freedom of the DG method are higher. In other words, the high-resolution wave propagation method solves n equations per grid cell, where n is the number of unknowns in the equation system. For the DG method however, np^d equations are solved within each grid cell. Therefore, the degrees of freedom of the DG method are np^d per grid cell or nhp^d over the entire domain.

The DG method implemented in this dissertation is formally higher order accurate than the polynomial basis chosen. For example, the 2^{nd} order wave propagation method is actually less than 2^{nd} order accurate whereas the 2^{nd} order RKDG method often provides formally greater than 2^{nd} order accuracy. This is explored with benchmark applications in Chapter 4. The reason for higher accuracy of the DG algorithm is explained by elaborating on the expansion coefficients defined in Eq. (3.23). In 2-dimensions, a 2^{nd} order accurate expansion is described by

$$Q(x, y) = Q_0 v_0 + Q_x v_x + Q_y v_y + Q_{xy} v_x v_y, \tag{3.47}$$

instead of

$$Q(x, y) = Q_0 v_0 + Q_x v_x + Q_y v_y. \tag{3.48}$$

The bilinear term in Eq. (3.47) is responsible for providing higher than 2^{nd} order accuracy. This principle is extended to all spatial orders and all dimensions to provide higher than the predicted accuracy. This is also a major difference between the RKDG method implemented in this dissertation and the one implemented by Loverich in Ref.[2].

3.2.5 Boundary Conditions

Ghost cells are used to specify the boundary conditions for both the wave propagation and the RKDG methods. Using ghost cells allows for the application of Dirichlet boundary

conditions where the variables carry a fixed edge value and Neumann boundary condition where the gradient at the boundary is specified.

For axisymmetric problems, the axis boundary condition is implemented at $r = 0$. Appropriate boundary conditions at the axis are found by assuming the variables are analytical about the axis and performing a series expansion about $r = 0$. Radial and azimuthal vector components are set to 0 at the axis. Scalar variables and axial vector components have no gradient at the axis. To implement the axis boundary condition using ghost cells, the scalar variables and axial vector components are copied into the ghost cells while the radial and azimuthal vector components are copied over to the ghost cells with a negative sign.

For problems with conducting wall boundary conditions, the normal velocity, the normal magnetic field, and the tangential electric fields go to 0 at a conducting wall. To implement this boundary condition using ghost cells, all variables are copied from the adjacent domain cells to the ghost cells while reversing the signs of the normal velocity, normal magnetic field and tangential electric fields. The remaining variables are extrapolated from the domain.

The coefficients of the RKDG method must be treated appropriately at the boundaries. The physical boundary conditions are implemented with consideration to the polynomial basis functions of the RKDG method. Specifically, to implement a 0 value at a boundary, the coefficients of all even basis functions in the ghost cells are set to the negative of the adjacent domain cells, and the coefficients of all odd basis functions in the ghost cells are set to the same values as the adjacent domain cells. The implementation of the zero gradient at the boundary uses the same procedure with opposite signs for the coefficients of the even and odd basis functions.

0 value at boundary:

$$\mathbf{Q}(\text{ghost cell}) = -\mathbf{Q}(\text{boundary}) \qquad \text{for even order coefficients 0,2,4,...} \tag{3.49}$$

$$\mathbf{Q}(\text{ghost cell}) = \mathbf{Q}(\text{boundary}) \qquad \text{for odd order coefficients 1,3,5,...} \tag{3.50}$$

0 gradient at boundary:

$$\mathbf{Q}(\text{ghost cell}) = \mathbf{Q}(\text{boundary}) \qquad \text{for even order coefficients 0,2,4,...} \tag{3.51}$$

$$\mathbf{Q}(\text{ghost cell}) = -\mathbf{Q}(\text{boundary}) \qquad \text{for odd order coefficients 1,3,5,...} \tag{3.52}$$

For the error correction potentials of the perfectly hyperbolic Maxwell's equations, special care needs to be taken for each of the boundary conditions. For the axis boundary condition, the gradient of both the error correction potentials is assumed zero across the axis. For the conducting wall boundary condition, the $\nabla \cdot \mathbf{E}$ correction potential, Φ, has 0 value at the boundary while the $\nabla \cdot \mathbf{B}$ correction potential, Ψ, has 0 gradient at the boundary. For an outflow (copy-out) boundary condition where the gradient of all variables is assumed 0 at the boundary, the error correction potentials are treated such that their value is 0 at the boundary (negated in the ghost cells). This is done to ensure that the error leaves the domain and is not reflected back. A similar implementation is needed for periodic boundary conditions where all the variables enter the domain from the opposite sides, however, the value of the error correction potentials should be set to 0 at the boundary so that they exit the domain without being re-introduced. Depending on whether the value or the gradient of the correction potentials is 0 at the boundary, Eq. (3.49-3.51) is used to appropriately treat the higher order coefficients of the error correction potentials.

3.2.6 Limiters

The DG scheme can produce large oscillations in the solution with the presence of sharp gradients. Like with the case of the Wave Propagation algorithm, limiters can be applied to the DG method as well around regions of discontinuities. For the wave propagation method, the limiters are applied to the waves, for the DG method however, they are applied to either the conserved variables or the characteristic variables depending on the limiter used. Conserved variables refer to the quantity $\mathbf{Q}$ that is being solved for. Characteristic variables refer to the strength of each wave. For a given expression,

$$\mathbf{Q} = \sum_{p=1}^{m} \omega^p r^p \tag{3.53}$$

ω^p refers to the characteristic variables and is constant along any p-characteristic. Here, r^p refers to the right eigenvectors and $\omega^p = l^p \mathbf{Q}$ where l^p refers to the left eigenvectors. There are two types of limiters that are investigated with this algorithm. The first is the characteristics-based limiter according to which the conserved variables are transformed

to characteristic variables. For $\mathbf{Q}_r^i$ defined as the expansion coefficients of the conserved variable in cell i, $a^p \equiv l^p \mathbf{Q}_1^i$, $a_+^p \equiv l^p(\mathbf{Q}_0^{i+1} - \mathbf{Q}_0^i)$ and $a_-^p \equiv l^p(\mathbf{Q}_0^i - \mathbf{Q}_0^{i-1})$. A modified minmod limiter is used here where the linear terms are checked for oscillations through a comparison with the forward and backward differences of the average term and the high order terms are set to zero if the linear terms need to be limited[39]. The coefficient of the linear term is modified using the minmod limiter as

$$\mathbf{Q}_1^i = \sum_p r^p \,\mathrm{mm}(a^p, a_+^p, a_-^p) \tag{3.54}$$

where r^p and l^p are the right and left eigenvectors of the flux Jacobian computed from cell averages. $\mathrm{mm}(a, b, c)$ is a modified min-mod function defined as

$$\mathrm{mm}(a, b, c) = a \quad \text{if } |a| < M\Delta x^2 \tag{3.55}$$

$$= \mathrm{m}(a, b, c) \quad \text{otherwise,} \tag{3.56}$$

where M is a constant and the function, $\mathrm{m}(a, b, c)$ is defined by

$$\mathrm{m}(a, b, c) = \max(a, b, c) \quad \text{if} sgn(a) = sgn(b) = sgn(c) = + \tag{3.57}$$

$$= \min(a, b, c) \quad \text{if} sgn(a) = sgn(b) = sgn(c) = - \tag{3.58}$$

$$= 0 \quad \text{otherwise.} \tag{3.59}$$

Once the limiting is done, the solution of the characteristic variables is transformed back to that of the conserved variables. Just as with the wave propagation algorithm, the limiter limits the order of the algorithm to a lower one in regions where it is applied.

Component-based limiters can also be applied to the DG algorithm. These are faster than characteristic-based limiters because they do not require computing the Jacobian and performing an eigen-decomposition. They involve directly applying the limiters to the conserved variables without transforming them. However, they are not TVDM so oscillations can develop.

Developing high-order limiters is a challenging research problem for the DG method.

One prospective high-order limiter explored here is described by Krivodonova in Ref.[40]. The high-order limiter described by Krivodonova starts limiting from the highest coefficient and stops when the first coefficient that does not need to be limited is encountered. This allows the maximum possible order to be retained in the solution within each cell. The high-order limiter can be applied with either component-based or characteristic-based limiting. This dissertation employs both component-based and characteristic-based high-order limiters which are useful for problems that do not have a large number of shocks. When a large number of shocks are present the solution is reduced to 1^{st} order in regions around shocks eliminating the advantages of using a high-order scheme. The implementation of the high-order limiters is described here using either component-based or characteristic-based limiting. This is a slight variation of the high-order limiting presented in Ref.[40] because the limiting performed here is dimension-by-dimension limiting highest to lowest orders in the x-, y-, and z-directions respectively instead of simultaneously limiting the expansion coefficients in all dimensions. An example for a 3-dimensional high-order limiting is presented here for 2^{nd} order. WARPX implements such limiting in a general way allowing arbitrary spatial order. Here $\mathbf{Q_{abc}}$ refers to the expansion coefficients for each of the directions a being x-direction coefficient, b being y-direction coefficient and c being z-direction coefficient.

- Step 1: First limit in x-direction only (compare with backward and forward difference for cells adjacent in the x-direction)
  - limit $\mathbf{Q}_{222}$, if not changed, goto y-direction
  - limit $\mathbf{Q}_{122}$, if not changed, goto y-direction
  - limit $\mathbf{Q}_{022}$, if not changed, goto y-direction
  - limit $\mathbf{Q}_{112}$ and $\mathbf{Q}_{121}$, if not changed, goto y-direction
  - limit $\mathbf{Q}_{012}$ and $\mathbf{Q}_{021}$, if not changed, goto y-direction
  - limit $\mathbf{Q}_{002}$ and $\mathbf{Q}_{020}$, if not changed, goto y-direction
  - limit $\mathbf{Q}_{111}$, if not changed, goto y-direction
  - limit $\mathbf{Q}_{011}$, if not changed, goto y-direction

  - limit $\mathbf{Q}_{001}$ and $\mathbf{Q}_{010}$, if not changed, goto y-direction

- Step 2: Next limit in y-direction only (compare with backward and forward difference for cells adjacent in the y-direction)

  - limit $\mathbf{Q}_{222}$, if not changed, goto z-direction
  - limit $\mathbf{Q}_{212}$, if not changed, goto z-direction
  - limit $\mathbf{Q}_{202}$, if not changed, goto z-direction
  - limit $\mathbf{Q}_{112}$ and $\mathbf{Q}_{211}$, if not changed, goto z-direction
  - limit $\mathbf{Q}_{102}$ and $\mathbf{Q}_{201}$, if not changed, goto z-direction
  - limit $\mathbf{Q}_{002}$ and $\mathbf{Q}_{200}$, if not changed, goto z-direction
  - limit $\mathbf{Q}_{111}$, if not changed, goto z-direction
  - limit $\mathbf{Q}_{101}$, if not changed, goto z-direction
  - limit $\mathbf{Q}_{100}$ and $\mathbf{Q}_{001}$, if not changed, goto z-direction

- Step 3: Next limit in z-direction only (compare with backward and forward difference for cells adjacent in the z-direction)

  - limit $\mathbf{Q}_{222}$, if not changed, stop
  - limit $\mathbf{Q}_{221}$, if not changed, stop
  - limit $\mathbf{Q}_{220}$, if not changed, stop
  - limit $\mathbf{Q}_{211}$ and $\mathbf{Q}_{121}$, if not changed, stop
  - limit $\mathbf{Q}_{201}$ and $\mathbf{Q}_{210}$, if not changed, stop
  - limit $\mathbf{Q}_{200}$ and $\mathbf{Q}_{020}$, if not changed, stop
  - limit $\mathbf{Q}_{111}$, if not changed, stop
  - limit $\mathbf{Q}_{110}$, if not changed, stop
  - limit $\mathbf{Q}_{100}$ and $\mathbf{Q}_{010}$, if not changed, stop

3.2.7 Axisymmteric Problems

The numerical method and WARPX have been developed for a Cartesian mesh and can be manipulated for axisymmetric problems without changing the infrastructure significantly. An axisymmetric problem can be formulated in the same manner as a problem in a regular Cartesian geometry. However, in evaluating the advection terms of the hyperbolic equations containing the divergence of the fluxes, additional terms need to be accounted from the divergence theorem for cylindrical coordinates. Axisymmetric problems can be applied in 1-dimension with radial derivatives of the fluxes or in 2-dimensions with radial and axial derivatives of the fluxes. The divergence theorem states that for a vector,

$$\nabla \cdot \mathbf{A} = \frac{1}{r}\frac{\partial}{\partial r}(rA_r) + \frac{1}{r}\frac{\partial A_\Phi}{\partial \Phi} + \frac{\partial A_z}{\partial z}. \tag{3.60}$$

Here the azimuthal derivative is neglected and the radial derivative term is written as

$$\frac{1}{r}\frac{\partial}{\partial r}(rA_r) = \frac{\partial A_r}{\partial r} + \frac{A_r}{r} \tag{3.61}$$

where the second term on the right-hand-side needs to be included as a geometric source term to provide a solution of an axisymmetric problem in Cartesian geometry. Likewise, ignoring all azimuthal derivatives, such a term also appears in the curl operator for cylindrical geometry

$$(\nabla \times \mathbf{A})_z = \frac{1}{r}\frac{\partial}{\partial r}(rA_\Phi) \tag{3.62}$$

which can be written as

$$\frac{1}{r}\frac{\partial}{\partial r}(rA_\Phi) = \frac{\partial A_\Phi}{\partial r} + \frac{A_\Phi}{r} \tag{3.63}$$

with the second term on the right-hand-side being the geometric source term for the curl operator. Additionally, the geometric source terms of the tensor fields in the fluxes also

need to be taken into account

$$(\nabla \cdot \mathbf{T})_r = \frac{1}{r}\frac{\partial}{\partial r}(rT_{rr}) + \frac{1}{r}\frac{\partial T_{\Phi r}}{\partial \Phi} + \frac{\partial T_{zr}}{\partial z} + \frac{T_{\Phi\Phi}}{r} \tag{3.64}$$

$$(\nabla \cdot \mathbf{T})_\Phi = \frac{1}{r}\frac{\partial}{\partial r}(rT_{r\Phi}) + \frac{1}{r}\frac{\partial T_{\Phi\Phi}}{\partial \Phi} + \frac{\partial T_{z\Phi}}{\partial z} + \frac{T_{\Phi r}}{r} \tag{3.65}$$

$$(\nabla \cdot \mathbf{T})_z = \frac{1}{r}\frac{\partial}{\partial r}(rT_{rz}) + \frac{1}{r}\frac{\partial T_{\Phi z}}{\partial \Phi} + \frac{\partial T_{zz}}{\partial z}. \tag{3.66}$$

Such terms appear in the momentum equation fluxes for the two-fluid plasma model. To include the geometric source terms, the balance law described in Eq. (1.1) is formulated as

$$\frac{\partial \mathbf{Q}}{\partial t} + \nabla \cdot \mathbf{F} = \mathbf{S} + \mathbf{S_G} \tag{3.67}$$

for any numerical method, with $\mathbf{S_G}$ containing the geometric sources. This implementation for axisymmetric problems is also applied to the wave propagation method.

3.2.8 Auxiliary Variables

Auxiliary variables are introduced to allow an arbitrary number of variables and equations to be added to the system. This allows the inclusion of non-conservative or non-hyperbolic variables that can be written in a flux-source form as

$$\mathbf{Q_{aux}} + \nabla \cdot \mathbf{F} = \mathbf{S} \tag{3.68}$$

without a time-derivative term. In doing this, an arbitrary spatial order discontinuous Galerkin method is used for the auxiliary variables. Most often the spatial order chosen for the auxiliary variables is the same as the conserved variables. In WARPX, auxiliary variables have flux, source and Riemann solver functions just like the conserved variables. For the conserved equations, auxiliary variables are passed from the input file and are included in all routines as `qaux`. When an auxiliary equation system is called to compute or update the auxiliary variables, it is passed into the right-hand-side (flux-source) solver of the discontinuous Galerkin method as if it were a conserved variable. The discontinuous

Galerkin method is then solved using the same method described in Sec. 3.2.1

$$\int_{I_i} v_r \mathbf{Q_{aux}} dV + \int_{I_i} v_r \nabla \cdot \mathbf{F} dV = \int_{I_i} v_r \mathbf{S} dV \tag{3.69}$$

where the spatially-dependent flux and source terms are multiplied with basis functions and integrated over each cell. In doing so, $\mathbf{Q_{aux}}$ has the same number of higher order coefficients as $\mathbf{Q}$.

For the auxiliary equation solvers, the conserved variables are passed using `qaux` as if they were auxiliary variables to allow their use in the auxiliary equations. This again is specified in the input files, examples of which are provided in Appendix C. This allows the use of the same base class in WARPX to compute the right-hand-side of conserved and auxiliary equations. In addition, writing it in this manner allows for a choice of any time-integration scheme to solve the ODE in Eq. (3.29) for updating the conserved variables while allowing the option of no time-integration for Eq. (3.68) to compute auxiliary variables. `wxdgrhscalc1dbase` and `wxdgrhscalc1dbasegengeom` are examples of discontinuous Galerkin right-hand-side base classes in 1-D for a Cartesian mesh and a structured, general geometry mesh. When using conserved variables the derived class used is `wxdgrhscalc1d` for a Cartesian mesh and `wxdgrhscalc1dgengeom` for general geometry, and auxiliary variables use the derived class, `wxdgauxsolver1d`. Similar classes exist in 2-D and 3-D.

3.2.9 Implementation Details for Hall-MHD

The Hall-MHD equation system in Sec. 2.3.3 is not completely described by the balance law form in Eq. (1.1). The equation system is no longer purely hyperbolic with source terms and the numerical method needs to be modified to account for the second order derivatives in Faraday's law, Eq. (2.10), that result from the generalized Ohm's law, Eq. (2.20), and the reduced Ampere's law, Eq. (2.21). Auxiliary variables are introduced namely electron current, J_e, and electric field, E, that are treated differently from the conserved variables as they do not require any time integration. Two methods for the treatment of auxiliary variables are explored. The first method involves a simple central differencing to compute the auxiliary variables from the conserved variables using Eq. (2.20) and Eq. (2.21) at the

beginning of each time step. These auxiliary variables are then used in the fluxes and sources to update the conserved variables.

A more accurate implementation of the auxiliary variables is performed using a full discontinuous Galerkin polynomial basis function expansion of the auxiliary variables just like with the conserved variables. The fluxes and sources of the auxiliary variables are treated in the same manner as the conserved variables by solving a Riemann problem at each interface and the only difference lies in the time integration. The auxiliary variables are calculated from the conserved variables at each time step and do not require any time integration scheme, while the conserved variables are updated using a Runge-Kutta time integration. The advantage of this full DG implementation over the central difference implementation for the auxiliary variables is the presence of higher order coefficients in the auxiliary variables consistent with the higher order DG coefficients for conserved variables. Additionally, for improved accuracy, the auxiliary variables are updated between each Runge-Kutta stage of the conserved variable update. The Hall-MHD auxiliary variables are written in flux-source form as

$$\int_{I_i} v_r \mathbf{J_e} dV - \int_{I_i} v_r \frac{1}{\mu_0} (\nabla \times \mathbf{B}) dV = -\int_{I_i} v_r \mathbf{J}_i dV \tag{3.70}$$

$$\int_{I_i} v_r n_e q_e \mathbf{E} dV - \int_{I_i} v_r (\nabla \cdot p_e \mathbf{I}) dV = -\int_{I_i} v_r (\mathbf{J}_e \times \mathbf{B}) dV \tag{3.71}$$

where the equations are multiplied with basis functions and integrated over each cell using the discontinuous Galerkin algorithm in the same manner as the conserved equations.

If using hyper-resistivity to eliminate the grid scale dependence of the whistler wave, additional auxiliary variables are introduced. Eq. (2.34) describes a hyper-resistivity that is used in this dissertation. The additional auxiliary variable is

$$\int_{I_i} v_r \mathbf{H_r} dV - \int_{I_i} v_r (\nabla \mathbf{J}) dV = 0 \tag{3.72}$$

such that $\mathbf{J}$ is the total current. The hyper-resistivity auxiliary variable, H_r, then appears

as a flux in the generalized Ohm's law

$$\int_{I_i} v_r n_e q_e \mathbf{E} dV - \int_{I_i} v_r (\nabla \cdot p_e \mathbf{I}) dV - \int_{I_i} v_r (\nu \nabla \cdot \mathbf{H_r}) dV = - \int_{I_i} v_r (\mathbf{J}_e \times \mathbf{B}) dV. \quad (3.73)$$

The boundary condition implementation for the auxiliary variables for Hall-MHD is similar to the conserved variable boundary condition implementation. For conducting wall and axis boundary conditions, the method described in Sec. 3.2.5 is used when the full DG expansion is used for the auxiliary variables. When using hyper-resistivity, the 9 additional variables introduced by taking the gradient of **J** are treated such that they have 0 value at the boundaries. This approximation can be implemented in a more sophisticated manner as needed for future applications.

For problems with large dynamics in the form of sharp gradients and discontinuous solutions, limiters are applied to the auxiliary variables the same way they are to the conserved variables. Only the component-based limiting is used for the auxiliary variables because the flux Jacobian is not specified for the auxiliary system the same way that it is specified for the conserved system.

3.2.10 Implementation Details for Braginskii Transport Terms

A central differencing implementation of the Braginskii[17] transport terms described in Sec. 2.5 has been previously implemented in WARPX to work with the finite volume method. As a part of this dissertation, the transport terms are implemented to use auxiliary variables with the discontinuous Galerkin method allowing solutions of higher order accuracy by using the finite element framework. Refer to Appendix A for a more detailed description of the transport coefficients. The 10 auxiliary variables and equations introduced for transport

include

$$\begin{aligned} gT_e - \nabla T_e = 0, \quad & gT_i - \nabla T_i = 0 \\ \mathbf{gu}_e - \nabla \mathbf{u}_e = 0, \quad & \mathbf{gu}_i - \nabla \mathbf{u}_i = 0 \\ du_e - \nabla \cdot \mathbf{u}_e = 0, \quad & du_i - \nabla \cdot \mathbf{u}_i = 0 \end{aligned} \tag{3.74}$$

$$\overleftrightarrow{\mathbf{\Pi}}_e = \eta_0 \left(\mathbf{gu}_e + \mathbf{gu}_e^T - \frac{2}{3} du_e \mathbf{I} \right) \tag{3.75}$$

$$\overleftrightarrow{\mathbf{\Pi}}_i = \eta_0 \left(\mathbf{gu}_i + \mathbf{gu}_i^T - \frac{2}{3} du_i \mathbf{I} \right) \tag{3.76}$$

$$\mathbf{q}_e = \mathbf{q}_U^e + \mathbf{q}_T^e = f(\mathbf{u}_e, \mathbf{u}_i, \mathbf{B}, T_e, n_e, gT_e) \tag{3.77}$$

$$\mathbf{q}_i = \mathbf{q}_T^i = f(\mathbf{B}, T_i, n_i, gT_i). \tag{3.78}$$

Each of the auxiliary variables in Eq. (3.74) has an associated flux term as shown. For Eqs.(3.75-3.78), the auxiliary variables of electron and ion viscosity, and electron and ion heat flux are written such that they use source terms only and have no flux terms. These auxiliary variables are multiplied with basis functions and integrated over each cell as described in Sec. 3.2.8 and are solved using the discontinuous Galerkin method described. Here $\mathbf{gu}$ and $\overleftrightarrow{\mathbf{\Pi}}$ have 9 auxiliary variables each and are tensors represented by

$$\mathbf{gu} = \begin{bmatrix} gu_{xx} & gu_{xy} & gu_{xz} \\ gu_{yx} & gu_{yy} & gu_{yz} \\ gu_{zx} & gu_{zy} & gu_{zz} \end{bmatrix}, \qquad \overleftrightarrow{\mathbf{\Pi}} = \begin{bmatrix} \Pi_{xx} & \Pi_{xy} & \Pi_{xz} \\ \Pi_{yx} & \Pi_{yy} & \Pi_{yz} \\ \Pi_{zx} & \Pi_{zy} & \Pi_{zz} \end{bmatrix}. \tag{3.79}$$

Once the auxiliary variables are computed, they are sent in to update the conserved variables with the transport terms. The transport equations defined in Sec. 2.5 use $\nabla \cdot \mathbf{q}_e$, $\nabla \cdot \mathbf{q}_i$, $\nabla \cdot \overleftrightarrow{\mathbf{\Pi}}_e$, and $\nabla \cdot \overleftrightarrow{\mathbf{\Pi}}_i$ as additional flux terms for the conserved equation system. The momentum transfer term, $\mathbf{R}_{ei}$, the electron and ion thermal equilibration terms, Q_e and Q_i, and the viscous heating terms, $\overleftrightarrow{\mathbf{\Pi}}_e : \nabla \mathbf{u}_e$ and $\overleftrightarrow{\mathbf{\Pi}}_i : \nabla \mathbf{u}_i$, are included as source terms for the conserved equations.

For axisymmetric problems, auxiliary variable fluxes $\nabla \cdot \mathbf{u}_e$ and $\nabla \cdot \mathbf{u}_i$, and conserved variable transport fluxes $\nabla \cdot \mathbf{q}_e$, $\nabla \cdot \mathbf{q}_i$, $\nabla \cdot \overleftrightarrow{\mathbf{\Pi}}_e$, and $\nabla \cdot \overleftrightarrow{\mathbf{\Pi}}_i$ will have geometric source terms that need to be accounted for in the manner described in Sec. 3.2.7.

Boundary conditions for the auxiliary transport variables need to be defined. Periodic and outflow boundary conditions are applied in the same manner as the conserved variables. For axis boundary conditions when solving axisymmetric problems, if a value has 0 gradient at the axis, then its derivative (which may be an auxiliary variable) is negated across the axis in the ghost cell. For 0 value and 0 gradient of auxiliary variable at axis, the boundary conditions are implemented as described in Eq. (3.49) and Eq. (3.51). At the axis, the boundary conditions are defined in Table 3.1, where q_x, q_y, and q_z are the heat flux in the x, y and z directions respectively.

$\frac{\partial T}{\partial x} = 0$	$\frac{\partial T}{\partial y} = 0$	$\frac{\partial T}{\partial z} = 0$ gradient
$gu_{xx} = 0$ gradient	$gu_{yy} = 0$ gradient	$gu_{zz} = 0$ gradient
$gu_{xy} = 0$	$gu_{yx} = 0$	$gu_{xz} = 0$
$gu_{zx} = 0$	$gu_{yz} = 0$	$gu_{zy} = 0$
$du = 0$ gradient		
$\Pi_{xx} = 0$ gradient	$\Pi_{yy} = 0$ gradient	$\Pi_{zz} = 0$ gradient
$\Pi_{xy} = 0$	$\Pi_{yx} = 0$	$\Pi_{xz} = 0$
$\Pi_{zx} = 0$	$\Pi_{yz} = 0$	$\Pi_{zy} = 0$
$q_x = 0$	$q_y = 0$	$q_z = 0$ gradient

Table 3.1: Axis boundary conditions for auxiliary variables introduced to include the transport coefficients defined by Braginskii for the two-fluid plasma model. These boundary conditions are applied to each of the electron and ion fluids.

For a conducting wall, the auxiliary variable boundary conditions are the same as the axis boundary conditions with the exception of $\frac{\partial T}{\partial y} = 0$ gradient and $q_y = 0$ gradient. In addition, the presence of viscosity requires a no-slip boundary condition at the wall, so the normal and tangential fluid velocities are 0 at the wall.

The Braginskii transport terms require a few additional specifications in WARPX to maintain numerical stability and prevent the transport terms from growing without bound in the limit that the applications are collisionless. Basically, if ν_e and ν_i appear in the denominator of any of the transport terms, examples of which are the viscosity coefficients, parallel thermal conductivities, etc., small values of ν_e and ν_i can make the transport coefficients become unreasonably large. To overcome this, an asymptote is set such that

if $\nu \ll (\omega_p, \omega_c)$ for each of the species, it is replaced by the next smallest frequency in computing the transport coefficients. An additional asymptote is specified for the thermal conductivities to ensure that the perpendicular and cross thermal conductivities never exceed the parallel thermal conductivity.

3.3 Implicit Discontinuous Galerkin Method

Implicit time integration is explored for the discontinuous Galerkin method for several reasons. Firstly, it does not require a stringent CFL-stability limit like the case of explicit methods. Secondly, for the two-fluid plasma model, the time step does not need to be restricted by the speed of light or the electron plasma frequency for the algorithm to be stable. Accuracy considerations alone can be used to determine the maximum time step used. This also becomes significant in problems where divergence error correction is used and the error correction speeds could exceed the speed of light.

Implicit DG schemes have been explored by Wang in Refs.[41] and [42] for the unsteady Euler equations. These can be extended to the two-fluid plasma model. Wang describes first- and second-order backwards differencing schemes (BDF1 and BDF2) and an implicit fourth-order Runge-Kutta scheme (IRK4). In this dissertation, BDF1, BDF2, and Crank-Nicolson (CN2) schemes are implemented in WARPX. Separating the time-dependent and spatial-dependent parts, the hyperbolic equation can be written as

$$M\frac{\partial \mathbf{Q}}{\partial t} + \mathbf{R_p}(\mathbf{Q}) = 0 \tag{3.80}$$

with M being the mass matrix. The implicit formulation for the ODE described is

$$\frac{M}{\Delta t}(\theta \mathbf{Q}_h^{n+1} - \theta \mathbf{Q}_h^{n}) + \mathbf{R_p}(\mathbf{Q}_h^{n+1}) = \mathbf{R_e}(\mathbf{Q}_h^{n+1}) \tag{3.81}$$

where $\theta = 1$ corresponds to BDF1, and $\theta = 0.5$ corresponds to CN2. The formulation for BDF2 is given by

$$\frac{M}{\Delta t}(\frac{3}{2}\mathbf{Q}_h^{n+1} - 2\mathbf{Q}_h^{n} - \frac{1}{2}\mathbf{Q}_h^{n-1}) + \mathbf{R_p}(\mathbf{Q}_h^{n+1}) = \mathbf{R_e}(\mathbf{Q}_h^{n+1}) \tag{3.82}$$

where

$$\mathbf{Q}_h = \sum_{r=1}^{r} \mathbf{Q}_r(t) v_r(x) \tag{3.83}$$

and $\mathbf{R_e}(\mathbf{Q}_h^{n+1})$ is the unsteady residual.

A numerical Jacobian is computed that allows for a Newton-type iteration scheme for the implicit advance. This numerical Jacobian is described by $J'(\mathbf{Q}_k)$ such that

- $\mathbf{Q}_{k+1} = \mathbf{Q}_k - J'(\mathbf{Q}_k)^{-1} J(\mathbf{Q}_k)$, $k = 0, 1, 2, ...$
- Approximately solve: $J'(\mathbf{Q}_k)\Delta\mathbf{Q}_k = -J(\mathbf{Q}_k)$
- Update $\mathbf{Q}_{k+1} = \mathbf{Q}_k + \Delta\mathbf{Q}_k$
- Jacobian matrix, $J'(\mathbf{Q}_k)$, approximated with finite differences

In order to set up the implicit DG implementation, the time-dependent and spatial-dependent contributions are written in ODE form similar to the RKDG method. The difference lies in the need to invert the mass matrix for the implicit scheme. The mass matrix is precomputed and stored ahead of time since it remains unchanged. The inversion of a large mass matrix can be computationally expensive especially for large domain sizes in multiple dimensions. The implicit time advance requires large matrix inversions but is stable for large time steps.

3.3.1 Implicit-DG Implementation Details

After exploring several standard solvers and preconditioners in the PETSc (Portable Extension Toolkit for Scientific Computation) package[11], an incomplete LU (ILU) left preconditioner is chosen with the GMRES linear solver to perform a cubic back-tracking line search technique for the non-linear solve. PETSc's Scalable Nonlinear Equations Solvers (SNES) are used for the implicit advance. This choice of solvers is motivated by the desire for accurate solutions with minimum computational effort and ease of implementation. Of the standard PETSc preconditioners, the ILU preconditioner is most effective for the

non-symmetric, block tridiagonal Jacobian matrix of the 1-D two-fluid plasma model and provides greater than a 60% speed-up compared to using no preconditioner. The ILU preconditioner employs Cholesky-like formulas to ensure that the preconditioning matrix only has nonzero values in locations where the Jacobian matrix has nonzero values.

A numerical Jacobian is computed using sparse matrix methods. The memory for the numerical Jacobian is allocated once in the beginning of the simulation using

```
MatCreateMPIAIJ(comm.getMpiComm(), numUnknowns, numUnknowns,
  totNumUnknowns, totNumUnknowns, nr, PETSC_NULL,
  nr_off, PETSC_NULL, &jacobian);
```

where `numUnknowns` is the product of the number of cells and the block size for the grid of the local processor. `totNumUnknowns` is the product of the number of cells and the block size of the entire global domain. The block size is $m_e p^d$ where m_e is the number of equations, p is the spatial order of the scheme and d is the number of dimensions. nr is the number of non-zeros per row that is specified as $3 \times block\ size$ in 1-dimension, $5 \times block\ size$ in 2-dimensions, and $7 \times block\ size$ in 3-dimensions. nr_off is the number of off-diagonal non-zeros per row which is 0 in 1-dimension, $2 \times block\ size$ in 2-dimensions, and $4 \times block\ size$ in 3-dimensions. It is important to pre-allocate the memory for maximum computational efficiency or else the simulation takes a very long time with dynamic allocation.

Once the memory is allocated for the numerical Jacobian, a matrix coloring is performed in order to efficiently use sparse matrix methods. A 1-dimensional matrix coloring is performed using

```
for (unsigned i=ilr; i<iup; ++i) {
  for (unsigned k=0; k<blockSize; ++k) {
    unsigned jr = i*blockSize + k;
    unsigned imin = ( i > 0 )     ? i-1 : 0;
    unsigned imax = ( i < imx-1 ) ? i+1 : imx-1;
    // set non-zero elements to 1.0
    for (unsigned jc = imin*blockSize; jc < (imax+1)*blockSize; ++jc) {
      MatSetValue(color, jr, jc, 1.0, INSERT_VALUES);
    }
  }
}
```

```
MatAssemblyBegin(color, MAT_FINAL_ASSEMBLY);
MatAssemblyEnd(color, MAT_FINAL_ASSEMBLY);
```

where `ilr` is the lower index of the local processor domain, `iup` is the upper index of the local processor domain, and `imx` is the upper index of the global domain. The matrix `color` is the same size as the Jacobian matrix. This assigns a value of 1 in regions where the Jacobian contains a value and 0 in all other regions. In 1-dimension, this is a block tri-diagonal matrix as is seen from the left-hand-side plot of Figure 3.3. In 2-dimensions, the matrix coloring is described by

```
for (unsigned j=ilry; j<iupy; ++j) {
  for (unsigned i=ilrx; i<iupx; ++i) {
    for (unsigned k=0; k<blockSize; ++k) {
      unsigned i2d = i+j*imaxx;
      unsigned jr = i2d*blockSize + k;
      unsigned imin = ( i > 0 )     ? i2d-1 : 0+j*imaxx;
      unsigned imax = ( i < imaxx-1 ) ? i2d+1 : (imaxx-1)+j*imaxx;
      // set non-zero elements to 1.0
      for (unsigned jc = imin*blockSize;
                    jc < (imax+1)*blockSize;
                    ++jc) {
        MatSetValue(color, jr, jc, 1.0, INSERT_VALUES);
      }
      // terms to the left of the tri-diagonal part
      if (i2d > imaxx-1)
        for (unsigned jc = (i2d-imaxx)*blockSize;
                      jc < (i2d-imaxx)*blockSize+blockSize;
                      ++jc)
          MatSetValue(color, jr, jc, 1.0, INSERT_VALUES);

      // terms to the right of the tri-diagonal part
      if (i2d < (imaxx*imaxy)-imaxx)
        for (unsigned jc = (i2d+imaxx)*blockSize;
                      jc < (i2d+imaxx)*blockSize+blockSize;
                      ++jc)
          MatSetValue(color, jr, jc, 1.0, INSERT_VALUES);
    }
  }
}
MatAssemblyBegin(color, MAT_FINAL_ASSEMBLY);
MatAssemblyEnd(color, MAT_FINAL_ASSEMBLY);
```

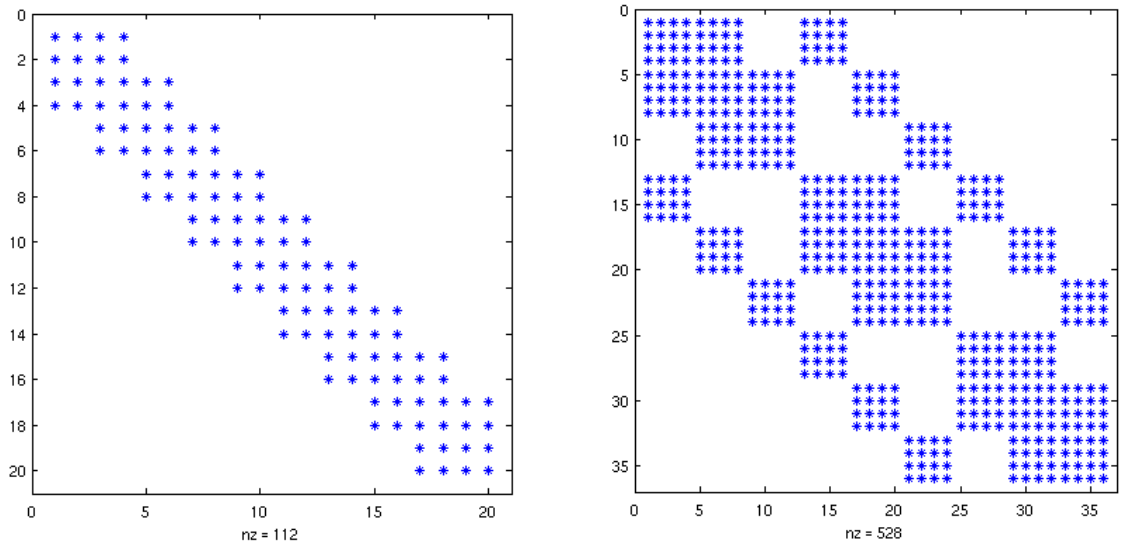

Figure 3.3: Matrix coloring plots for advection equation with DG spatial order 2. Left plot shows matrix coloring for Implicit-DG in 1-D for a grid of 10×10 cells. Right plot shows matrix coloring for 2-D Implicit-DG for a grid of 3×3 cells.

which forms a block tri-diagonal coloring matrix with 2 additional bands on either side of the tri-diagonal bands as is seen from the right-hand side plot of Figure 3.3. This implementation can be extended accordingly for 3-dimensions.

Following a matrix coloring, the numerical Jacobian is computed using

```
SNESSetJacobian(solver, jacobian, jacobian,
      SNESDefaultComputeJacobianColor, matFDColoring);
```

which computes a default PETSc numerical Jacobian, where the 2^{nd} `jacobian` in the function call can be substituted with a user-specified pre-conditioner if necessary. The non-linear solver PETSc function used is

```
SNESSolve(solver, PETSC_NULL, solVector);
```

where `solVector` is the $\mathbf{Q}$ solution vector. This calls

```
evaluateFunction(SNES snes, Vec x, Vec f)
```

to compute the right-hand-side for the discontinuous Galerkin method using an iterative Newton scheme to compute a $\mathbf{Q_{iter}}$. A tolerance of 10^{-6} is specified for most of the iterative problems in this dissertation. This function takes in arguments that allow the user to choose between BDF1, BDF2, and CN2 from the input file.

The choice of SNES solvers and pre-conditioners can be specified from the command-line during runtime using the commands

```
-ksp_type SOLVER
-pc_type PC
```

where `SOLVER` and `PC` can be one of the numerous options available within PETSc or they could be external packages that can be linked to PETSc, or they can be user specified custom routines. To make the semi-implicit two-fluid solvers efficient, physics-based preconditioners[43], p-multigrid methods[44], and similar optimization schemes can be explored. A brief description of the solvers and preconditioners tested with the two-fluid model is presented in Chapter 6 to explain the choice of the GMRES solver and the ILU preconditioner specifically for the two-fluid plasma model. For two-fluid simulations using parallel processing with PETSc, a benefit is only seen if there are at least $20,000$ unknowns per processor.

3.4 General Geometry with Discontinuous Galerkin Method

A 3-dimensional implementation of a structured general geometry discontinuous Galerkin method is detailed here. Similar to Eqs.(3.23-3.29), in 3-D the discontinuous Galerkin method is written as,

$$\underset{A}{\frac{\partial}{\partial t}\int_{I_i} v_{rst}\mathbf{Q}dV} \quad \underset{B}{+\int_{I_i} v_{rst}\frac{\partial \mathbf{F}}{\partial x}dV} \quad \underset{C}{+\int_{I_i} v_{rst}\frac{\partial \mathbf{G}}{\partial y}dV} \quad \underset{D}{+\int_{I_i} v_{rst}\frac{\partial \mathbf{H}}{\partial z}dV} \quad = \underset{E.}{\int_{I_i} v_{rst}\mathbf{S}dV} \tag{3.84}$$

where the hyperbolic balance law is multiplied with basis function v_{rst} and integrated over each cell. Term A results in a mass matrix. Terms B, C, and D apply integration by parts

to obtain a volume and a surface integral term as described in Eq. (3.27),

$$\int_{I_i} v_{rst}\frac{\partial \mathbf{F}}{\partial x}dV = \int_{\partial I_i} \mathbf{F} v_{rst} dS - \int_{I_i} \mathbf{F}\frac{\partial v_{rst}}{\partial x}dV \tag{3.85}$$

$$B \qquad\qquad B_1 \qquad\qquad B_2$$

$$\int_{I_i} v_{rst}\frac{\partial \mathbf{G}}{\partial y}dV = \int_{\partial I_i} \mathbf{G} v_{rst} dS - \int_{I_i} \mathbf{G}\frac{\partial v_{rst}}{\partial y}dV \tag{3.86}$$

$$C \qquad\qquad C_1 \qquad\qquad C_2$$

$$\int_{I_i} v_{rst}\frac{\partial \mathbf{H}}{\partial z}dV = \int_{\partial I_i} \mathbf{H} v_{rst} dS - \int_{I_i} \mathbf{H}\frac{\partial v_{rst}}{\partial z}dV \tag{3.87}$$

$$D \qquad\qquad D_1 \qquad\qquad D_2.$$

General geometry algorithm assumptions include planar hexahedral faces to avoid dealing with the complications of volumes and surface areas for curved faces. This maps an arbitrary hexahedral (not including triangular dipyramids and pentagonal pyramids) in physical space to a logical cube in computational space. Figure 3.4 shows an example of such a mapping in 3-D. Likewise, for the 2-D implementation, all edges are assumed to be straight, mapping between an arbitrary quadrilateral in physical space and a logical rectangle in computational space. A physical (x,y,z) is mapped to a computational (η,ξ,ζ) local

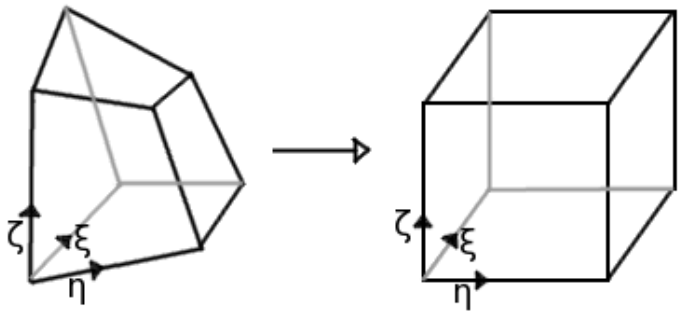

Figure 3.4: An example of the mapping for a 3-D case is displayed here where an arbitrary hexahedral with straight faces in physical space is mapped to a logical cube in computational space. General geometry is implemented for structured grids only.

coordinate within each cell such that the lowest (x,y,z) vertex in each cell determines the

(η,ξ,ζ) direction mapping for that cell based on the edges adjacent to that vertex. For each cell, η, ξ, and ζ are each defined between $[-1, 1]$ such that the lowest vertex in the cell has coordinates (-1,-1,-1) and the highest vertex has coordinates (1,1,1) in logical space.

An explanation of each of the volume and surface integrals along with a derivation of the grid metrics is summarized in the following sections.

3.4.1 Linear Mapping of Elements for 3-D General Geometry

A linear mapping is used in 3-D based on Figure 3.4. The linear mapping is defined using

$$x = a_1 + a_2\eta + a_3\xi + a_4\zeta + a_5\eta\xi + a_6\xi\zeta + a_7\eta\zeta + a_8\eta\xi\zeta \tag{3.88}$$

$$y = b_1 + b_2\eta + b_3\xi + b_4\zeta + b_5\eta\xi + b_6\xi\zeta + b_7\eta\zeta + b_8\eta\xi\zeta \tag{3.89}$$

$$z = c_1 + c_2\eta + c_3\xi + c_4\zeta + c_5\eta\xi + c_6\xi\zeta + c_7\eta\zeta + c_8\eta\xi\zeta. \tag{3.90}$$

Substituting $[-1, 1]$ for η, ξ, and ζ in Eqs.(3.88-3.90) for each of the 8 vertices of the hexahedral element, $x = Aa$, $y = Bb$, $z = Cc$, the following matrix is obtained

$$A = B = C = \begin{bmatrix} 1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 \\ 1 & 1 & -1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 & 1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 \end{bmatrix} \tag{3.91}$$

which is inverted to obtain each of the coefficients $a = A^{-1}x$, $b = B^{-1}y$, and $c = C^{-1}z$. Once the coefficients are computed, the grid metrics derivatives, $\frac{\partial x}{\partial \eta}$, $\frac{\partial x}{\partial \xi}$, $\frac{\partial x}{\partial \zeta}$, $\frac{\partial y}{\partial \eta}$, etc. are calculated from the mappings in Eqs.(3.88-3.90). These are computed at each quadrature point based on the values of η, ξ,and ζ. To obtain the derivatives $\frac{\partial \eta}{\partial x}$, $\frac{\partial \eta}{\partial y}$, $\frac{\partial \eta}{\partial z}$, $\frac{\partial \xi}{\partial x}$, etc. the

following matrix is constructed

$$\begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} = \begin{bmatrix} x_\eta & x_\xi & x_\zeta \\ y_\eta & y_\xi & y_\zeta \\ z_\eta & z_\xi & z_\zeta \end{bmatrix} \begin{bmatrix} d\eta \\ d\xi \\ d\zeta \end{bmatrix} \tag{3.92}$$

where, $x_\eta = \frac{\partial x}{\partial \eta}$, $x_\xi = \frac{\partial x}{\partial \xi}$, etc. This matrix is inverted to obtain

$$\begin{bmatrix} d\eta \\ d\xi \\ d\zeta \end{bmatrix} = \begin{bmatrix} \eta_x & \eta_y & \eta_z \\ \xi_x & \xi_y & \xi_z \\ \zeta_x & \zeta_y & \zeta_z \end{bmatrix} \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} \tag{3.93}$$

where $\eta_x = \frac{\partial \eta}{\partial x}$, $\eta_y = \frac{\partial \eta}{\partial y}$, etc.

3.4.2 3-D Volume and Volume Jacobian

The volume of each cell is computed using an efficient method[45] part of which is depicted in Figure 3.5. This method involves the sum of three 3×3 determinants to compute the

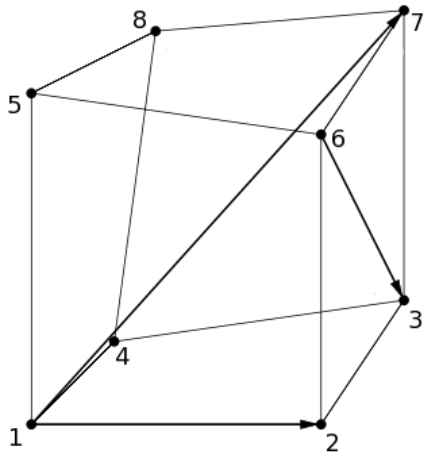

Figure 3.5: This figure is borrowed and modified from Ref.[45] to show one of the three 3×3 determinants leading to an efficient hexahedral volume computation.

volume using the formula

$$6V = |(\mathbf{x}_7 - \mathbf{x}_1), (\mathbf{x}_2 - \mathbf{x}_1), (\mathbf{x}_3 - \mathbf{x}_6)| \tag{3.94}$$

$$+ |(\mathbf{x}_7 - \mathbf{x}_1), (\mathbf{x}_5 - \mathbf{x}_1), (\mathbf{x}_6 - \mathbf{x}_8)| \tag{3.95}$$

$$+ |(\mathbf{x}_7 - \mathbf{x}_1), (\mathbf{x}_4 - \mathbf{x}_1), (\mathbf{x}_8 - \mathbf{x}_3)| \tag{3.96}$$

where V is the volume of the cell and $\mathbf{x}_n$ refers to the (x,y,z) vector at vertex n from Figure 3.5.

In the finite volume method, this volume is used in the computation for the integration. For the discontinuous Galerkin method, this volume is only used to determine the time step as a volume/area ratio. The volume integrals are computed using Jacobians that take the grid metrics into account since quadrature points are used in the interior and surface locations of each cell. The volume integral terms have $dV = dxdydz = Jd\eta d\xi d\zeta$, where J is the volume Jacobian containing the grid mapping information.

$$\delta x = \frac{\partial x}{\partial \eta}\delta\eta + \frac{\partial x}{\partial \xi}\delta\xi + \frac{\partial x}{\partial \zeta}\delta\zeta \tag{3.97}$$

$$\delta y = \frac{\partial y}{\partial \eta}\delta\eta + \frac{\partial y}{\partial \xi}\delta\xi + \frac{\partial y}{\partial \zeta}\delta\zeta \tag{3.98}$$

$$\delta z = \frac{\partial z}{\partial \eta}\delta\eta + \frac{\partial z}{\partial \xi}\delta\xi + \frac{\partial z}{\partial \zeta}\delta\zeta \tag{3.99}$$

leads to

$$dV = \begin{vmatrix} x_\eta d\eta & x_\xi d\xi & x_\zeta d\zeta \\ y_\eta d\eta & y_\xi d\xi & y_\zeta d\zeta \\ z_\eta d\eta & z_\xi d\xi & z_\zeta d\zeta \end{vmatrix} = |J| d\eta d\xi d\zeta \tag{3.100}$$

where $|J| = x_\eta y_\xi z_\zeta - x_\eta y_\zeta z_\xi + x_\xi y_\zeta z_\eta - x_\xi y_\eta z_\zeta + x_\zeta y_\eta z_\xi - x_\zeta y_\xi z_\zeta$.

3.4.3 3-*D Surface Areas and Surface Jacobians*

The surface areas of each cell are computed using standard vector manipulations. There are 3 surface areas belonging to each cell and these are specified in WARPX as *left* for the

YZ face at $\eta = -1$, *back* for the XZ face at $\xi = -1$ and *bottom* for XY face at $\zeta = -1$. Each cell owns its lower surface areas in WARPX. For each face, the surface area is computed using $0.5|d_1 \times d_2|$ where d_1 and d_2 are the diagonals for that face.

The surface Jacobians of each cell are computed to calculate surface integrals that result from an integration by parts for each of the flux integrals. Figure 3.6 shows the faces that are owned by a given cell. For face 1, $\delta\eta = 0$; for face 2, $\delta\xi = 0$, and for face 3, $\delta\zeta = 0$. The same principle applies to compute surface Jacobians as surface areas. For the YZ face

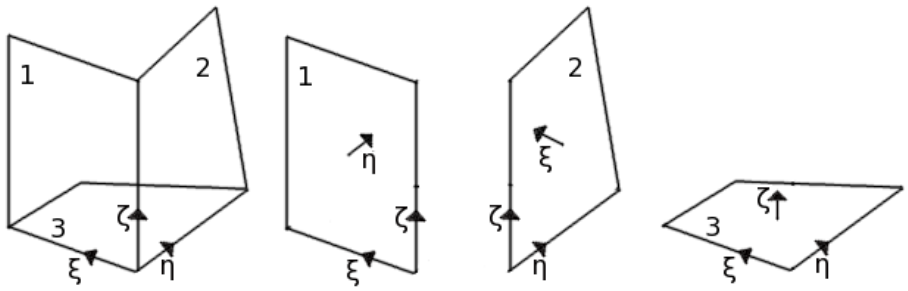

Figure 3.6: A given cell owns the faces adjacent to its lowest vertex, i.e. vertex 1 from Figure 3.5. The surface Jacobians of each cell are computed based on this depiction.

at $\eta = -1$, $\delta\eta = 0$,

$$\delta x = \frac{\partial x}{\partial \xi}\delta\xi + \frac{\partial x}{\partial \zeta}\delta\zeta \tag{3.101}$$

$$\delta y = \frac{\partial y}{\partial \xi}\delta\xi + \frac{\partial y}{\partial \zeta}\delta\zeta \tag{3.102}$$

$$\delta z = \frac{\partial z}{\partial \xi}\delta\xi + \frac{\partial z}{\partial \zeta}\delta\zeta. \tag{3.103}$$

This gives a *left Jacobian*, J_{S1}

$$dS_1 = |(x_\xi d\xi\hat{\mathbf{e}}_x + y_\xi d\xi\hat{\mathbf{e}}_y + z_\xi d\xi\hat{\mathbf{e}}_z) \times (x_\zeta d\zeta\hat{\mathbf{e}}_x + y_\zeta d\zeta\hat{\mathbf{e}}_y + z_\zeta d\zeta\hat{\mathbf{e}}_z)| \tag{3.104}$$

$$= \sqrt{(y_\xi z_\zeta - z_\xi y_\zeta)^2 + (z_\xi x_\zeta - x_\xi z_\zeta)^2 + (x_\xi y_\zeta - y_\xi x_\zeta)^2}d\xi d\zeta \tag{3.105}$$

$$= J_{S1}d\xi d\zeta \tag{3.106}$$

A similar implementation is performed to obtain the *back Jacobian*, J_{S2}, for the XZ face at

$\xi = -1$ where $\delta\xi = 0$ and to obtain the *bottom Jacobian*, J_{S3}, for the XY face at $\zeta = -1$ where $\delta\zeta = 0$.

3.4.4 3-D General Geometry Mass Matrix

Expansion coefficients are applied to term A in Eq. (3.84) like in Eq. (3.23) using basis functions v_{mnp} to obtain

$$\frac{\partial}{\partial t}\int_{I_i} v_{rst}\mathbf{Q}dV = \frac{d}{dt}\int_{I_i} v_{rst}\sum_{mnp}\mathbf{Q}_{mnp}v_{mnp}dV \tag{3.107}$$

$$= \sum_{mnp}\int_{I_i} v_{rst}v_{mnp}dV\frac{d\mathbf{Q}_{mnp}}{dt} \tag{3.108}$$

where $v_{rst} = v_{rst}(x(\eta,\xi,\zeta), y(\eta,\xi,\zeta), z(\eta,\xi,\zeta))$ and likewise for v_{mnp}. This gives

$$\frac{\partial}{\partial t}\int_{I_i} v_{rst}\mathbf{Q}dV = \sum_{mnp}\frac{d\mathbf{Q}_{mnp}}{dt}\int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} P_r(\eta)P_s(\xi)P_t(\zeta)P_m(\eta)P_n(\xi)P_p(\zeta)Jd\eta d\xi d\zeta \tag{3.109}$$

$$= M\frac{d\mathbf{Q}}{dt} \tag{3.110}$$

where J is the volume Jacobian defined in Sec. 3.4.2, P refers to the basis functions which are Legendre polynomials in this implementation, and M is the mass matrix that is inverted for each cell. Applying Gaussian quadrature, the mass matrix is defined as

$$M = \sum_j w_j^\eta \sum_k w_k^\xi \sum_l w_l^\zeta P_r(\eta_j)P_s(\xi_k)P_t(\zeta_l)P_m(\eta_j)P_n(\xi_k)P_p(\zeta_l)J(\eta_j,\xi_k,\zeta_l) \tag{3.111}$$

where the size of the mass matrix is $R \times R$, $R = m_e p^d$, m_e is number of conserved variables, p is the spatial order of the scheme, and d is the dimensions. The mass matrix is defined for each cell in the domain and its inverse is computed using LAPACK matrix inversion routines. All grid metrics are only computed once at the beginning of the simulation.

In the process of making the mass matrix general for arbitrary spatial order, the indexing in WARPX is slightly complicated. The mass matrix is defined using an indexer of type *ind* such that a row index and a column index are defined as a function of the Legendre

polynomial indices m,n,p,r,s,t, where *ind* is an indexer defined in the specified range,

```
WxIndexer<> ind = WxIndexer<>(WxRange(0,_spatialOrder,
  0,_spatialOrder, 0,_spatialOrder));
rowInd = ind.index(m,n,p);
colInd = ind.index(r,s,t);
```

Once the sum is computed over the basis functions and integrals defined in Eq. (3.111), a mass matrix indexer, $indMass$, is used defined in the specified range to store the given sum in the appropriate location within the symmetric mass matrix.

```
WxIndexer<> indMass = WxIndexer<>(WxRange
  (0, _spatialOrder*_spatialOrder*_spatialOrder,
   0,_spatialOrder*_spatialOrder*_spatialOrder));
massItr[indMass.index(rowInd,colInd)] = sum;
```

Any quantity can be projected onto basis functions using the mass matrix. The initial condition is applied using such a projection

$$\mathbf{Q}_{mnp} = \frac{\int_{I_i} P_m(\eta)P_n(\xi)P_p(\zeta)J(\eta,\xi,\zeta)\overline{\mathbf{Q}}d\eta d\xi d\zeta}{M}. \tag{3.112}$$

3.4.5 3-D General Geometry Rotation Matrix

Similar to the surface area computation, the unit normal vectors for each face are obtained using a cross product of the two diagonals, such that

$$\hat{n} = \frac{d_1 \times d_2}{|d_1 \times d_2|} \tag{3.113}$$

and they are defined for the same faces of a given cell that the surface areas are defined for. Care is taken to ensure that the right-hand rule is maintained in determining the directions of the unit normals when performing the cross product. For face 1 in Figure 3.6, the unit normal is in the η direction and the tangents are in ξ and ζ directions respectively. For face 2, the unit normal is in the ξ direction and the tangents are in η and ζ directions respectively and for face 3 the unit normal is in the ζ direction and the tangents are in η and ξ directions respectively. Unit tangent vectors are computed using the edges adjacent

to the vertex owned by the cell (vertex 1 in Figure 3.5), while maintaining the right-hand rule.

To rotate the data from physical coordinates to the local coordinates, the rotation matrix used is

$$R = \begin{bmatrix} n_x & n_y & n_z \\ t_x^1 & t_y^1 & t_z^1 \\ t_x^2 & t_y^2 & t_z^2 \end{bmatrix} \tag{3.114}$$

where n is the unit normal vector and t^1 and t^2 are unit tangent vectors in each of the other 2 directions defined within each cell using the lowest vertex and its adjacent edges. A rotation from local to physical coordinates is performed using an inverse of the above matrix, where the inverse and transpose are the same.

3.4.6 3-D General Geometry Surface Integrals

To compute the surface integrals, terms B_1, C_1, and D_1, in Eqs.(3.85-3.87), a Gaussian quadrature is used for each surface such that

$$\begin{aligned} B_1 + C_1 + D_1 = &\sum_k w_k^\xi \sum_l w_l^\zeta \left(\left[\overline{\mathbf{F}} \cdot \mathbf{n} v_{rst} J_{S1}\right]_{\eta_{ix+\frac{1}{2}},\xi,\zeta} - \left[\overline{\mathbf{F}} \cdot \mathbf{n} v_{rst} J_{S1}\right]_{\eta_{ix-\frac{1}{2}},\xi,\zeta} \right) \\ &+ \sum_j w_j^\eta \sum_l w_l^\zeta \left(\left[\overline{\mathbf{G}} \cdot \mathbf{n} v_{rst} J_{S2}\right]_{\eta,\xi_{iy+\frac{1}{2}},\zeta} - \left[\overline{\mathbf{G}} \cdot \mathbf{n} v_{rst} J_{S2}\right]_{\eta,\xi_{iy-\frac{1}{2}},\zeta} \right) \\ &+ \sum_j w_j^\eta \sum_k w_k^\xi \left(\left[\overline{\mathbf{H}} \cdot \mathbf{n} v_{rst} J_{S3}\right]_{\eta,\xi,\zeta_{iz+\frac{1}{2}}} - \left[\overline{\mathbf{H}} \cdot \mathbf{n} v_{rst} J_{S3}\right]_{\eta,\xi,\zeta_{iz-\frac{1}{2}}} \right) \end{aligned}$$

using the appropriate surface Jacobians for a given direction. The conserved variables are expanded to determine their value at surface quadrature locations of each cell. These surface conserved quantities are then rotated to the local coordinate system. If using auxiliary variables, these are also rotated to the local system. The edge fluxes are computed in the local coordinate system. This gives the normal and tangential fluxes in each local direction. WARPX uses an edge flux routine for general geometry which computes the left and right surface fluxes in the local coordinate system.

```
_eqnSet.edgefluxgengeom(0, xm, &qlLocal[0], &qauxlLocal[0], &flLocal[0]);
_eqnSet.edgefluxgengeom(0, xm, &qrLocal[0], &qauxrLocal[0], &frLocal[0]);
```

This function is independent of the local direction, since the local fluxes are specified as normals and tangents for each direction being solved. The Riemann problem is then solved in the local coordinate system. The edge fluxes and fluctuations at each surface quadrature location are then rotated back to the physical coordinate system and use the appropriate surface Jacobians to update the solution of conserved variables.

3.4.7 3-D General Geometry Volume Integrals

Terms B_2, C_2, and D_2 in Eqs.(3.85-3.87) constitute the volume integrals of the fluxes and term E in Eq. (3.84) is a volume integral of the sources. Gaussian quadrature is used to approximate these volume integrals in much the same way as the surface integrals, for quadrature locations at (η_j,ξ_k,ζ_l). The derivatives of the basis functions are written as

$$\frac{\partial v_{rst}}{\partial x} = \frac{\partial v_{rst}}{\partial \eta}\frac{\partial \eta}{\partial x} + \frac{\partial v_{rst}}{\partial \xi}\frac{\partial \xi}{\partial x} + \frac{\partial v_{rst}}{\partial \zeta}\frac{\partial \zeta}{\partial x} \tag{3.115}$$

$$\frac{\partial v_{rst}}{\partial \eta} = \frac{dP_r(\eta)}{d\eta}P_s(\xi)P_t(\zeta) \tag{3.116}$$

$$\frac{\partial v_{rst}}{\partial \xi} = P_r\frac{dP_s(\xi)}{d\xi}P_t(\zeta) \tag{3.117}$$

$$\frac{\partial v_{rst}}{\partial \zeta} = P_r(\eta)P_s(\xi)\frac{P_t(\zeta)}{d\zeta}. \tag{3.118}$$

$\frac{\partial v_{rst}}{\partial y}$ and $\frac{\partial v_{rst}}{\partial z}$ can be written in a similar manner. Using the grid metric derivatives $\frac{\partial \eta}{\partial x}$, $\frac{\partial \eta}{\partial y}$, $\frac{\partial \eta}{\partial z}$, etc., defined in Sec. 3.4.1, the volume integrals of the fluxes are computed at the quadrature locations using the volume Jacobians and these update the conserved variables. For the volume integrals of the fluxes, no rotation needs to be performed and the conserved variables in physical coordinates are used to compute the flux terms in physical coordinates using a separate flux calculation function in WARPX where the directions (0,1,2) need to be specified so that the fluxes are computed in the actual x-, y-, and z-directions.

```
_eqnSet.flux(0, xm, _ql, _qauxl, _fl); // X-flux
_eqnSet.flux(1, xm, _ql, _qauxl, _fr); // Y-flux
_eqnSet.flux(2, xm, _ql, _qauxl, _df); // Z-flux
```

For the sources, Gaussian quadrature is used to multiply the set of basis functions and the volume Jacobian with the source term computed at the quadrature location. This is then used to update the values of the conserved variables with the source terms. Again, no rotation needs to be performed for inclusion of the source terms as long as the conserved variables used are in the physical coordinate system.

Separating the spatial- and temporal-dependent terms, the formulation obtained based on Eqs.(3.84-3.87) is

$$\frac{d\mathbf{Q}}{dt} = M^{-1}\mathcal{L}_r(\mathbf{Q}) \tag{3.119}$$

$$\mathcal{L}_r(\mathbf{Q}) = -(B_1 + C_1 + D_1) + B_2 + C_2 + D_2 + E \tag{3.120}$$

which can be used with any time integration scheme for the discontinuous Galerkin method.

3.4.8 General Geometry Test Cases

A number of test cases are performed using Euler and Maxwell's equations in skewed grids, star-shaped grids and circular grids in 2-dimensions. For the Euler equations, a pulse is initialized such that it uses the initial condition shown in Fig. 3.7. This initial condition is applied to fluid density and pressure while all momenta are initialized to zero. The same initial condition is used for all the general geometry configurations presented in Figs. 3.8 to 3.12 where the benchmark solution is for a rectangular Cartesian grid shown in Fig. 3.8. All simulations for the general geometry test cases are performed at a grid resolution of 20×20 cells. Periodic boundary conditions are used.

Figure 3.9 shows a solution of the density pulse in a skewed grid after it has propagated 1/4 of a period. The skewness in the grid is described using

$$v_x = x + \alpha\Delta x \cos \pi b_1(x - c_1) \sin 2\pi b_2(y - c_2) \tag{3.121}$$

$$v_y = y + \alpha\Delta y \sin 2\pi b_1(x - c_1) \cos \pi b_2(y - c_2) \tag{3.122}$$

where (v_x, v_y) represent the vertices, (x, y) represent logical cell values that span the domain size $[X_L, X_U] \times [Y_L, Y_U]$. $b_1 = 1/(X_U - X_L)$, $b_2 = 1/(Y_U - Y_L)$, $c_1 = (X_U + X_L)/2$,

$c_2 = (Y_U + Y_L)/2$, and α is the skewness coefficient.

The star-shaped grid in Fig. 3.10 defines the vertices using the following routine

```
r = if(x*x+y*y>1.e-10, sqrt(x*x+y*y), 1.e-10)
d = if(sqrt(x*x)>sqrt(y*y),sqrt(x*x),sqrt(y*y))
x1 = r1*d*x/r
y1 = r1*d*y/r
x1_eff = if(sqrt(x1*x1)>1.e-10, x1, 1.e-10)
theta = atan(sqrt(y1*y1) / x1_eff)
r2 = 1 + 0.2*cos(6*theta)
x2 = x1*r2
y2 = y1*r2
w = d*d
v_x = w*x2 + (1.0-w)*x/sqrt(2.0)
v_y = w*y2 + (1.0-w)*y/sqrt(2.0)
```

The circular grid[46] in Fig. 3.11 defines the vertices using the following routine

```
r = rad
s = r/sqrt(2.0)
ymax = s*sqrt(2.0-x*x)
fr = 1.0001
ystretch = s*(y-y0)
xmax = s*sqrt(2.0-y*y)
xstretch = s*(x-x0)
ycirc = if(x*y>0,(y-x)/(fr-abs(x))*(ymax-s*abs(x))+s*x,
                 (y+x)/(fr-abs(x))*(ymax-s*abs(x))-s*x)
xcirc = if(x*y>0,(x-y)/(fr-abs(y))*(xmax-s*abs(y))+s*y,
                 (x+y)/(fr-abs(y))*(xmax-s*abs(y))-s*y)
v_x = if(1.0>=abs(x/y), xstretch, xcirc)
v_y = if(1.0>=abs(x/y), ycirc, ystretch)
```

where `rad` is the user-specified radius of the circular grid.

The circular grid in Fig. 3.12 defines the vertices using the following routine

```
x1 = 2.0*abs(x-0.5)
y1 = 2.0*abs(y-0.5)
d1 = sqrt((1.0+1.0/tan(0.5*pi*x1))^2 - cos(0.25*pi*x1)^2) - sin(0.25*pi*x1)
d2 = sqrt((1.0+1.0/tan(0.5*pi*y1))^2 - cos(0.25*pi*y1)^2) - sin(0.25*pi*y1)
f1 = -(cos(0.5*pi*x1)+ sqrt((1.0+sin(pi*x1))/(cos(.25*pi*x1))^2
     - (sin(0.5*pi*x1))^2))
```

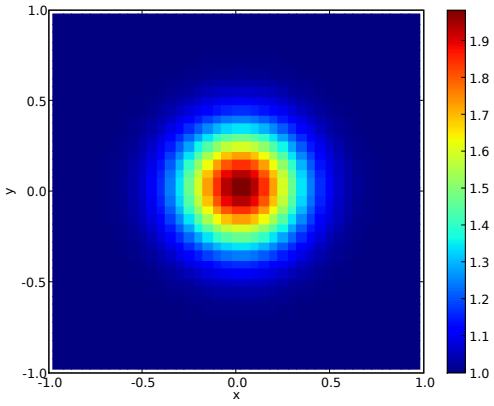

Figure 3.7: Euler pulse initial condition to test propagation using several general geometry configurations.

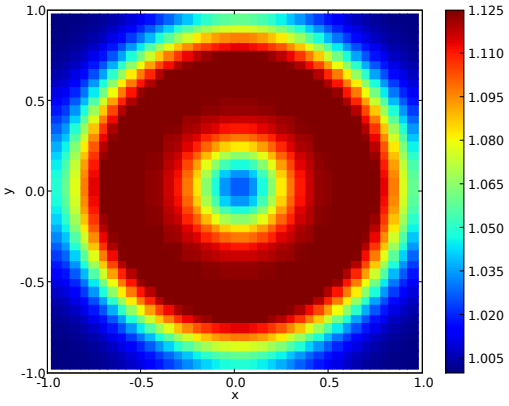

Figure 3.8: Euler pulse plot of fluid density after 0.25 transit times for rectangular Cartesian grid.

```
f2 = -(cos(0.5*pi*y1)+ sqrt((1.0+sin(pi*y1))/(cos(.25*pi*y1))^2
     - (sin(0.5*pi*y1))^2))
xp = if(x1==1.0, cos(0.25*pi*y1),
        if(y1==1.0, sin(0.25*pi*x1),
           if(x1==0.0 and y1==0.0, 0.0,
              if(x1==0.0, 0.0,
                 if(y1==0.0, (1.0+1.0/tan(0.5*pi*x1))-d1,
                             (sqrt(4.*d1^2*(1.0-f1/d1^2-f2/d2^2)
                             - (f1-f2)^2/d2^2)-(2.0*d1+d1/d2^2*(f1-f2)))/
                               (2.0*(1.0+d1^2/d2^2)))))))
yp = if(x1==1.0, sin(0.25*pi*y1),
        if(y1==1.0, cos(0.25*pi*x1),
           if(x1==0.0 and y1==0.0, 0.0,
              if(x1==0.0, (1.0+1.0/tan(0.5*pi*y1))-d2,
                 if(y1==0.0, 0.0, xp*d1/d2 + 0.5*(f1-f2)/d2)))))
v_x = xp*sign(x-0.5)
v_y = yp*sign(y-0.5)
```

All test cases show that the Euler pulse propagation is captured appropriately. Figure 3.11 has relatively uniform sized cells throughout the domain, but the diagonal elements can become heavily skewed. Figure 3.12 has highly deformed cells near the boundary, so this makes the solution less accurate as it approaches the boundary. Similar test cases are performed for Maxwell's equations to ensure that the curl operators obey the right-hand-

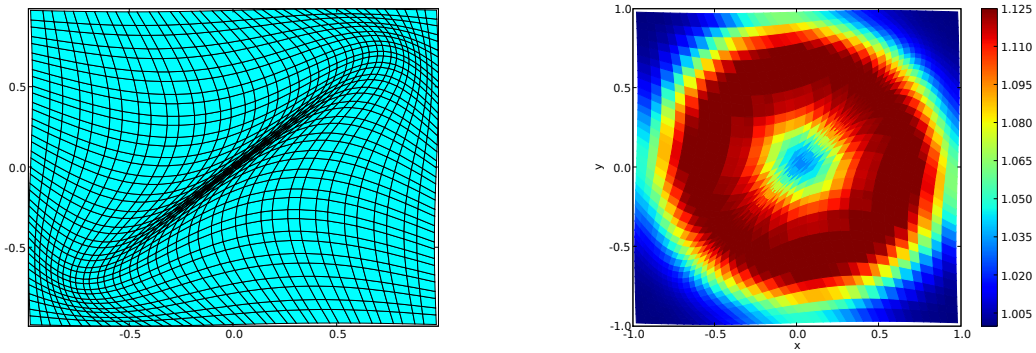

Figure 3.9: Left plot shows the skewed grid. Right plot shows the Euler pulse fluid density after 0.25 transit times for skewed grid with $\alpha = 3$.

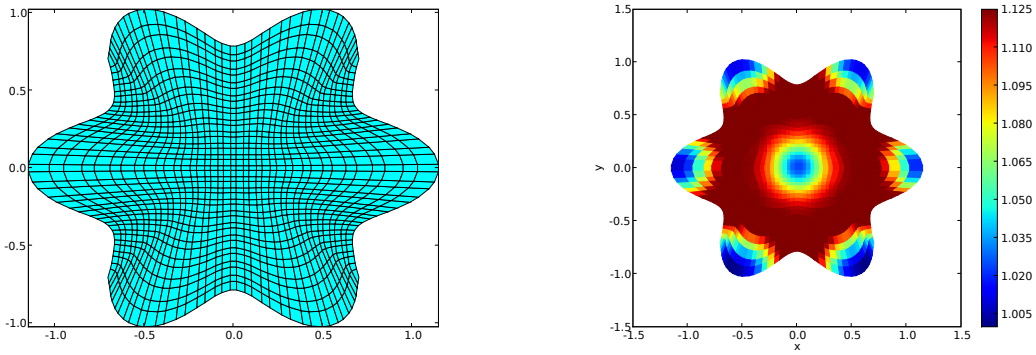

Figure 3.10: Left plot shows the star-shaped grid. Right plot shows the Euler pulse fluid density after 0.25 transit times for star-shaped grid.

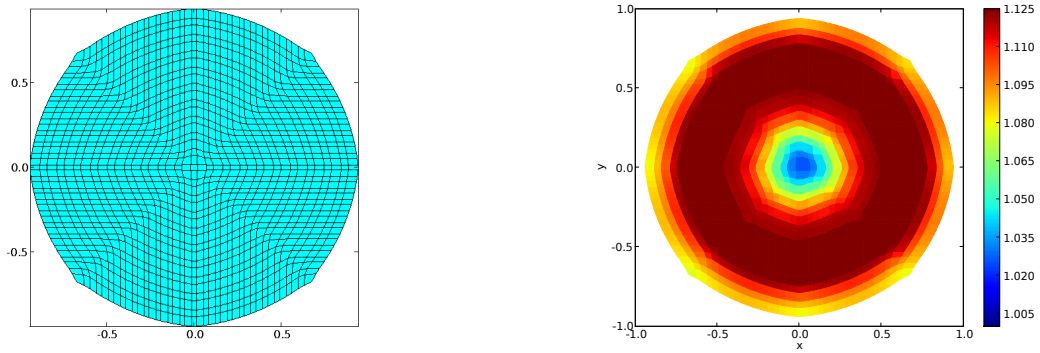

Figure 3.11: Left plot shows the circular grid. Right plot shows the Euler pulse fluid density after 0.25 transit times for circular grid.

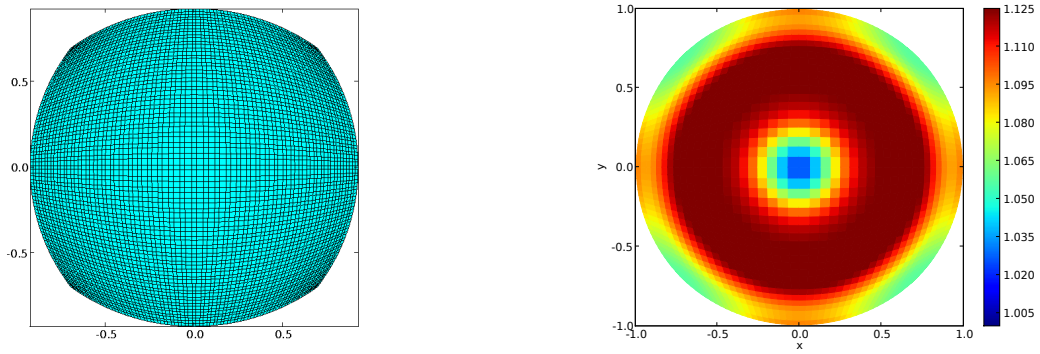

Figure 3.12: Left plot shows another circular grid. Right plot shows the Euler pulse fluid density after 0.25 transit times for circular grid 2.

rule for the general geometry algorithm. Following the 2-dimensional general geometry test cases, 3-dimensional Euler and Maxwell pulses are initiated in a cylindrical domain and are tested for the same initial conditions. The cylindrical grid is rotated in all 3 orientations to ensure that the propagation in all directions is correct. A single-block cylindrical grid that is used to model some 3-dimensional two-fluid plasma configurations is described in Fig. 3.13 where the physical coordinates are mapped to the computational domain using a functional mapping. In the computational domain, the domain and cells are logical cubes. The heavily skewed cells on each of the diagonals as seen in Fig. 3.13 could pose a problem in producing grid effects within the solution or could make implementation of limiters challenging. Yet another 3-D grid is described in Fig. 3.14 where the heavily skewed elements near the boundaries could pose a challenge if the solution reaches or interacts with the boundary.

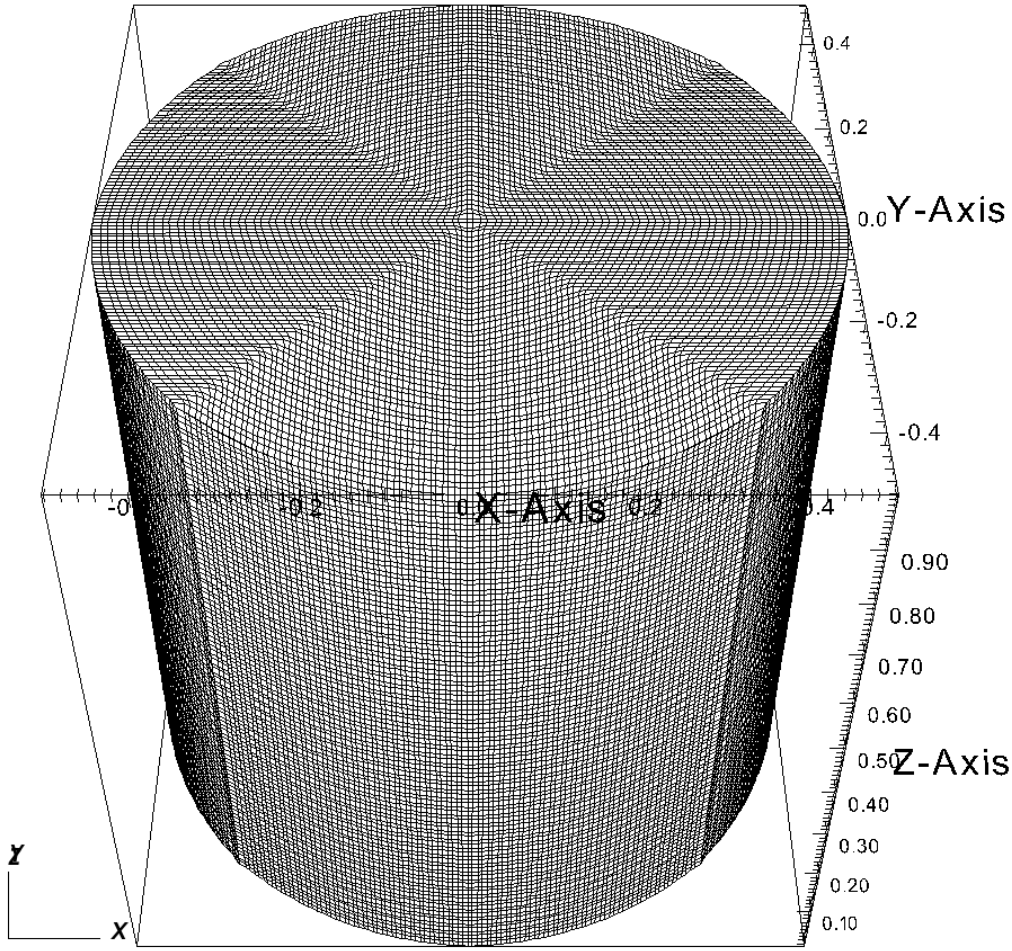

Figure 3.13: A single-block 3-dimensional cylindrical grid that is used for 3-D two-fluid plasma configurations in this dissertation. This grid is mapped to a logical cube in computational space. Note distorted cells along diagonals.

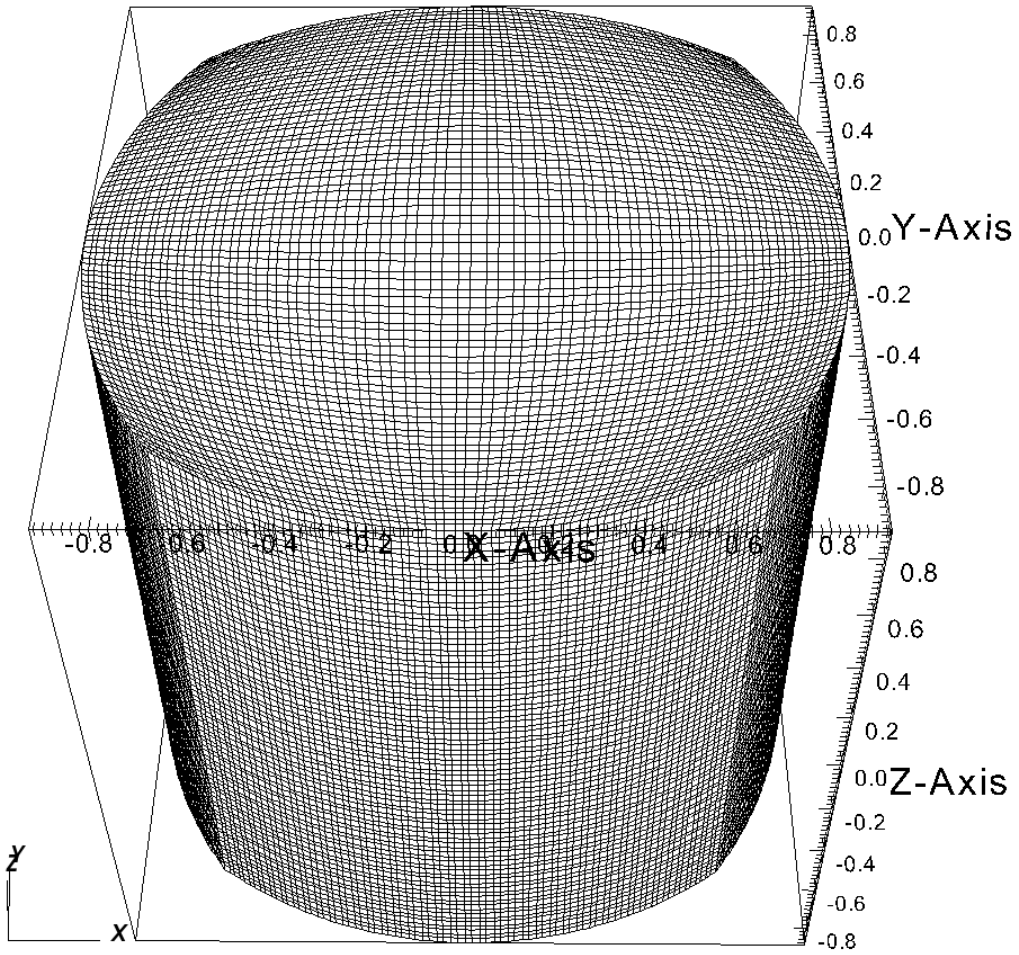

Figure 3.14: A single-block 3-dimensional cylindrical grid that is used for 3-D two-fluid plasma configurations in this dissertation. This grid is mapped to a logical cube in computational space. Note distorted near boundaries.

Chapter 4

COMPARISONS BETWEEN RKDG AND WAVE PROPAGATION METHODS

The plasma fluid equations studied in this dissertation differ from previously published work due to the presence of dispersive source terms in the non-linear balance laws. The appropriate treatment of advection and dispersion terms and the complications that arise from the presence of significant dispersion in plasma models provide a unique set of comparisons between the wave propagation and the RKDG methods. There is very little literature regarding the numerical solution of hyperbolic equation systems that contain dispersive source terms without the presence of any explicit dissipation[1, 2]. Hakim, Loverich and Shumlak[1] implement the high resolution wave propagation method for the two-fluid plasma model studied in this paper. Loverich and Shumlak[2] implement the Runge-Kutta discontinuous Galerkin method for the two-fluid plasma model. Neither of these references address the effect of the physical dispersive source terms of the two-fluid plasma model on the accuracy of the numerical methods. A comparison of the wave propagation and discontinuous Galerkin methods to determine which of the two algorithms provides more accurate solutions of the two-fluid plasma model is the main focus of this paper.

The high-resolution wave propagation method and the RKDG method are compared for benchmark applications of the linear advection equation, Euler equations, and Maxwell's equations. The Euler and Maxwell's equations are then combined into the two-fluid plasma model and are compared for two-fluid applications. Details of these comparisons are presented in Ref.[47] while this chapter presents a summary of the comparisons. The two-fluid model has dispersive source terms which makes it unique compared to traditional fluid models that often have diffusive source terms.

4.1 Linear Advection Equation

The linear advection of a one-dimensional Gaussian pulse $q(x,0) = e^{-10(x-1.5)^2}$ is used to compare the spatial order and computational effort for the wave propagation and RKDG methods. This benchmark problem does not contain any source terms and is only included to provide a reference to compare other applications that may not have the dispersive sources of the two-fluid plasma model. The linear advection equation is

$$\frac{\partial q}{\partial t} + \frac{\partial q}{\partial x} = 0. \tag{4.1}$$

Periodic boundary conditions on a domain $1 < x < 5$ are used. After propagating the pulse one period though the domain the l_2 norm of the error is computed by comparing to the exact solution. The effective order of the method is computed by measuring the dependence of the l_2-norm on the grid spacing. In this test the time-step is kept constant. The l_2-norm is calculated by

$$||\Delta q||_2 = \sqrt{\frac{1}{n}\sum_{x=1}^{n}(q-\tilde{q})^2}, \tag{4.2}$$

where q is the numerical solution, $\tilde{q}$ is the analytical solution. The summation occurs over every grid or quadrature point.

Figure 4.1 shows the measured l_2-norms of the solutions obtained for different grid resolutions with a time step that is lower than that of the 8^{th} order RKDG method with 500 cells ($\Delta t = 0.0003$ is used). The slopes measured from the linear regions of the plot are as shown in the second column of Table 4.1. The 8^{th} order RKDG solution converges to the analytical solution rapidly so the linear behavior is only noted at lower resolutions. The table shows that the computed order of convergence exceeds the formal order of the methods for this linear problem with a fixed Δt. The fixed time step isolates the effect of the spatial order. Figure 4.2 shows the l_2-norms of the solutions obtained for different grid resolutions with a variable time step that is chosen based on the maximum allowable CFL number for the methods at each resolution. The slopes measured from the linear regions of

Method	Order (fixed Δt)	Order (maximum Δt)
WAVE	1.9	1.9
RKDG 2^{nd} order	2.7	2.0
RKDG 3^{rd} order	3.4	3.2
RKDG 4^{th} order	4.2	3.0
RKDG 8^{th} order	8.0	3.1

Table 4.1: Slopes of l_2-norm vs Δx to determine spatial order of accuracy of the methods for the linear advection equation.

the plot are as shown in the third column of Table 4.1. The table shows that the computed order of convergence for the higher-order RKDG methods is restricted to approximately 3.5 for variable Δt. This restriction is due to the temporal order of 3 chosen for all solutions of the RKDG method. Figure 4.3 shows the computational effort of the two methods for this problem. As expected, the wave propagation method requires the least computational effort and higher-order RKDG methods require more computational effort. The effort is linear with the number of grid points.

4.2 Euler Equations with Dispersive Source Terms

Due to the complexity of the full two-fluid plasma system it is difficult to investigate the effects of dispersion on the full non-linear physics. In this section a simpler model is introduced that allows for dispersion to be included with the Euler equations in the form of dispersive source terms. This models the quasineutral ion cyclotron waves, which are dispersive waves, in a uniform plasma with a magnetic field that is constant in space and time. The momentum equation includes the force from a uniform transverse magnetic field, which produces dispersive effects.

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + u\frac{\partial}{\partial x}\mathbf{u}\right) + \frac{\partial p}{\partial x} = nq\mathbf{u}\times\mathbf{B} \tag{4.3}$$

$$= \rho\omega_c \mathbf{u}\times\hat{b} \tag{4.4}$$

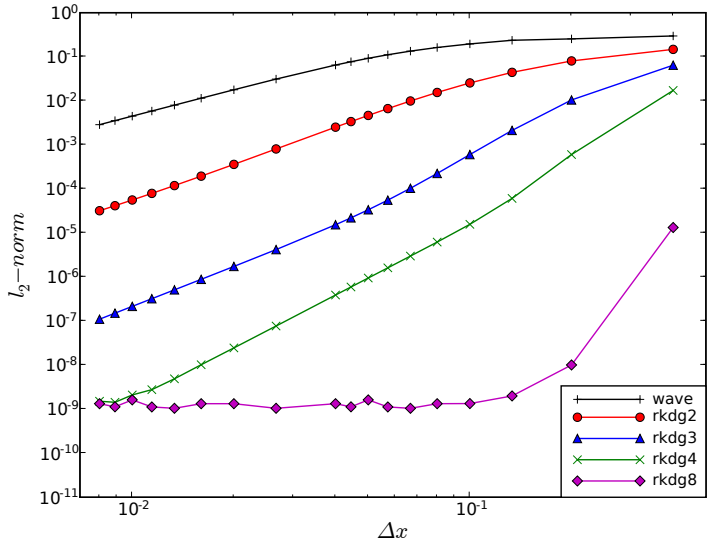

Figure 4.1: Log-log plots of l_2-norms of solution as a function of Δx for the linear advection problem using the same time step to isolate the effect of the spatial order for all the numerical methods - the wave propagation method, 2^{nd}, 3^{rd}, 4^{th}, and 8^{th} order RKDG. The slopes of the lines are tabulated in Table 4.1 which shows that the numerical order for this problem exceeds the formal order of the methods. The RKDG methods are also seen to be much more accurate for the same cell spacing than the wave propagation method.

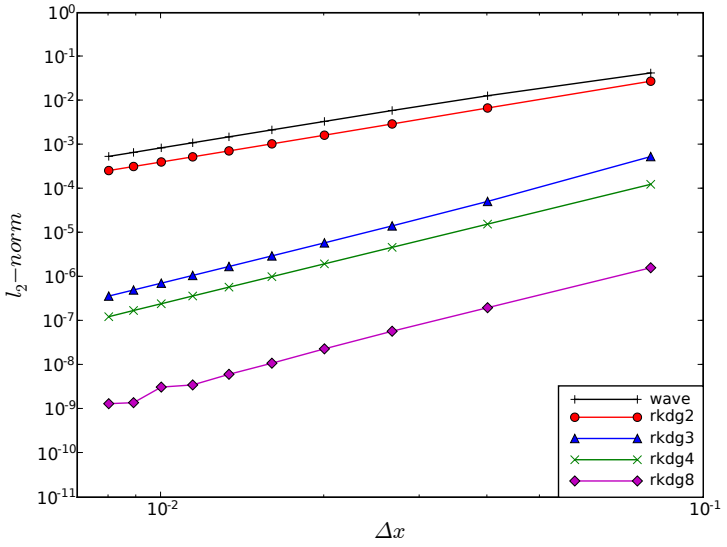

Figure 4.2: Log-log plots of l_2-norms of solution as a function of Δx for the linear advection problem using the maximum allowable CFL number to set the time step for each of the numerical methods - the wave propagation method, 2^{nd} order RKDG, 3^{rd} order RKDG, 4^{th} order RKDG, and 8^{th} order RKDG. The slopes of the lines are tabulated in Table 4.1 which shows that the numerical order of convergence for the higher order RKDG methods is restricted by the temporal order of 3 that is chosen for all the RKDG solutions.

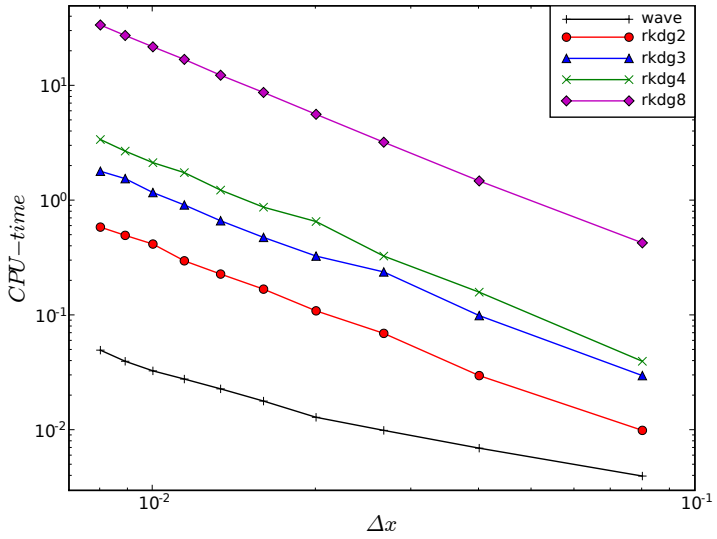

Figure 4.3: Log-log plots of computational time (sec) as a function of Δx for the linear advection problem solved by various numerical methods - the wave propagation method, 2^{nd}, 3^{rd}, 4^{th} and 8^{th} order RKDG. The wave propagation method takes less computational effort than RKDG methods for the same grid resolution.

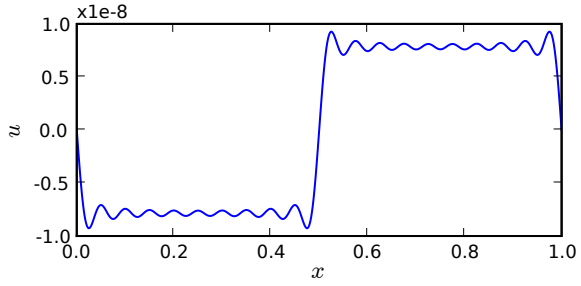

Figure 4.4: Initial condition with $N = 9$ that is used to approximate the step function. This initial condition is used for all the results obtained in this section.

where ρ is the mass density, $\mathbf{u} = (u, v)$ is the fluid velocity, p is the pressure, $\mathbf{B}$ is a uniform magnetic field, $\hat{b}$ is the unit vector of $\mathbf{B}$ and ω_c is the cyclotron frequency. The eigenvalues of the source Jacobian are 0,0, and $\pm i\omega_c$. The non-zero imaginary eigenvalues indicates that the system has undamped, nonpropagating oscillations which combined with the sound wave leads to dispersive waves

$$\omega_n = \pm \left(k_n^2 c_s^2 + \omega_c^2\right)^{1/2} \tag{4.5}$$

where $c_s \equiv \sqrt{\gamma p_0/\rho_0}$ is the speed of sound and k_n is the wave number.

To initialize the simulation the fluid is perturbed with a velocity

$$u_1(x) = u_1^0 \sum_{n=0}^{N} \frac{i}{2n+1} e^{ik_n x} \tag{4.6}$$

with $k_n = 2\pi(2n+1)$ and ω_n computed from Eq. (4.5) and u_1^0 is a constant. As $N \to \infty$, Eq. (4.6) represents a step function for the interval $[0, 1]$. With the perturbation of Eq. (4.6) the exact solution for the linearized velocity $u(x, t)$ is given by

$$\tilde{u}(x, t) = -\sum_{n=0}^{\infty} \frac{u_1^0}{2n+1} \sin(k_n x + \omega_n t). \tag{4.7}$$

The test problem is initialized using the exact solution for all perturbed variables with the velocity given by Eq. (4.7). Figure 4.4 shows the initial condition for $u_1^0 = 10^{-8}$, $N = 9$

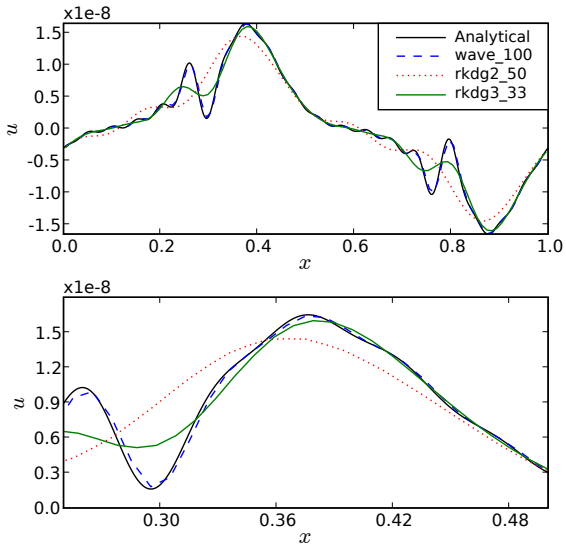

Figure 4.5: Velocity at $t = 3$ for an effective resolution of 100 cells for the wave propagation and RKDG methods, i.e., 100 cells for wave propagation, 50 for 2^{nd} order RKDG and 33 for 3^{rd} order RKDG. $c_s = \sqrt{2}$ and $\omega_c = 10$. The bottom plot has an expanded scale to show the details of the solution.

using $\gamma = 2$, $\omega_c = 10$, $\rho_0 = p_0 = 1$ and $\omega_c = 10$. For these values the sound speed is given by $c_s = \sqrt{2}$. Periodic boundary conditions are applied on a domain $0 < x < 1$. A CFL number of 1 is used for the wave propagation method and $1/(2p-1)$ is used for the RKDG method. The temporal order of the RKDG method is 3^{rd} order for this problem. Limiters are not applied in either method and the solutions at $t = 3$ are compared to the exact solution.

Figures 4.5 and 4.6 compare the analytical solution to the wave propagation method and to the RKDG method. The number of grid elements, h, is adjusted with the spatial order, p, of the RKDG method so the effective resolution, hp, remains constant. Accuracy is measured by taking an l_2-norm. These figures along with Table 4.2 show that the wave propagation method is more accurate than the 2^{nd}, 3^{rd} and 5^{th} order RKDG methods while all methods have the same effective resolution. The 8^{th} order RKDG solution with only 12 cells, however, is more accurate than the wave propagation method.

The computational time required to advance the solution from $t = 0$ to $t = 1$ for each

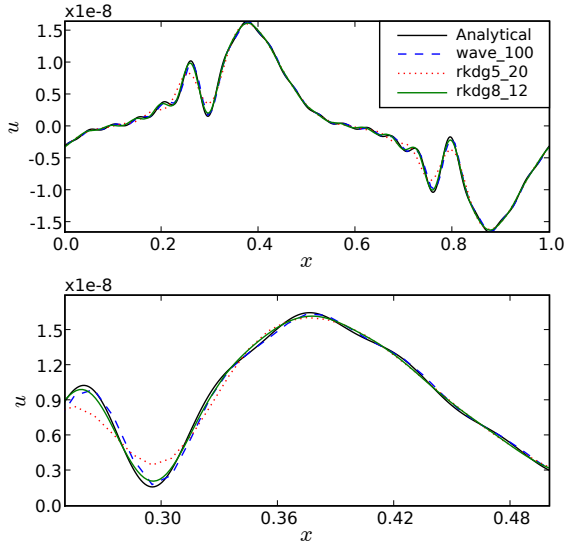

Figure 4.6: Velocity at $t = 3$ with 100 cells for wave propagation, 20 for 5^{th} order RKDG and 12 for 8^{th} order RKDG. $c_s = \sqrt{2}$ and $\omega_c = 10$. The bottom plot has an expanded scale to show the details of the solution.

Method	l_2-norm	Computational time to $t = 1$s
WAVE_100	2.6×10^{-9}	0.02
RKDG2_50	2.2×10^{-8}	0.03
RKDG3_33	1.2×10^{-8}	0.04
RKDG5_20	6.9×10^{-9}	0.07
RKDG8_12	2.3×10^{-9}	0.19

Table 4.2: l_2-norm of velocity to quantify accuracy for each method, and computational time required to advance the solution to $t = 1$s to quantify computational effort for the dispersive Euler system using $\omega_c = 10$.

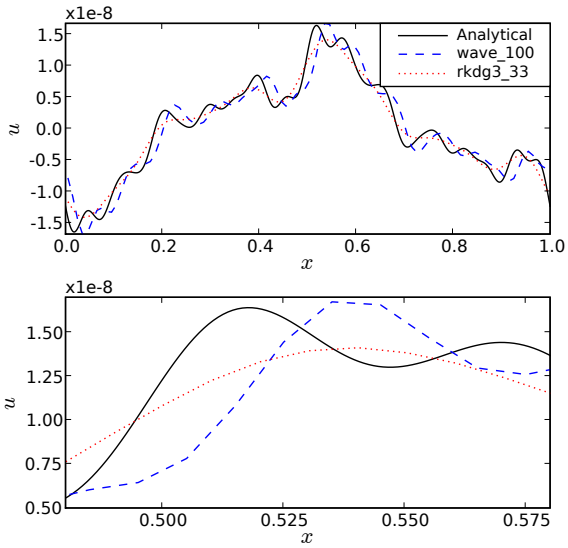

Figure 4.7: Velocity at $t = 3$ with 100 cells for wave propagation and 33 for 3^{rd} order RKDG. These are for $c_s = \sqrt{2}$ and $\omega_c = 50$. Using a larger ω_c leads to phase errors for the wave propagation method while the lower order RKDG method for the same effective resolution is diffusive but has no phase error. The bottom plot has an expanded scale.

method is presented in Table 4.2. Each method has a different CFL stability limit, and each is operated at its maximum CFL value. The solution of the wave propagation method is more accurate as compared to the RKDG solutions while using less computational effort for low ω_c. However, when ω_c is increased, the wave propagation method exhibits phase errors in the solution. Figure 4.7 shows that the 3^{rd} order RKDG solution using 33 cells is more diffusive than the wave propagation solution at 100 cells, but the RKDG solutions do not have phase errors even at lower orders. Figure 4.8 displays results for the wave propagation method with 100 cells, the 8^{th} order RKDG with 12 cells and the 16^{th} order RKDG with 6 cells. Increasing the grid resolution of the wave propagation method from 100 cells to 500 cells reduces the phase error, as shown in Fig. 4.9, and going to even higher resolution eliminates it.

The computational time required to advance the solution from $t = 0$ to $t = 1$ for each method for $\omega_c = 50$ is presented in Table 4.3. Table 4.3 shows that the 16^{th} order RKDG

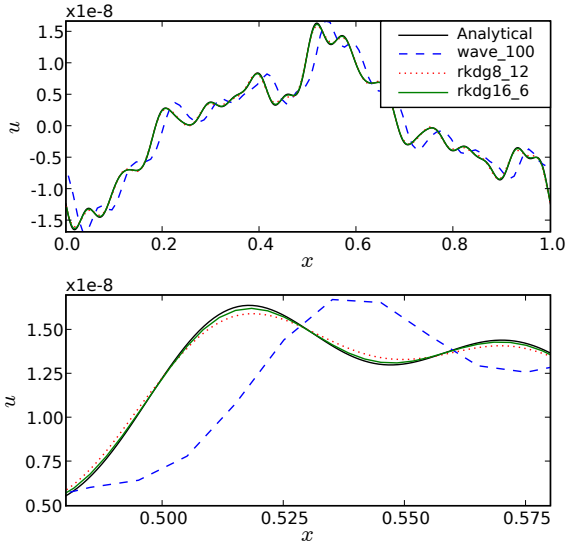

Figure 4.8: Velocity at $t = 3$ with 100 cells for wave propagation, 12 for 8^{th} order RKDG and 6 for 16^{th} order RKDG. These are for $c_s = \sqrt{2}$ and $\omega_c = 50$. Using a larger ω_c leads to phase errors for the wave propagation method. The bottom plot has an expanded scale.

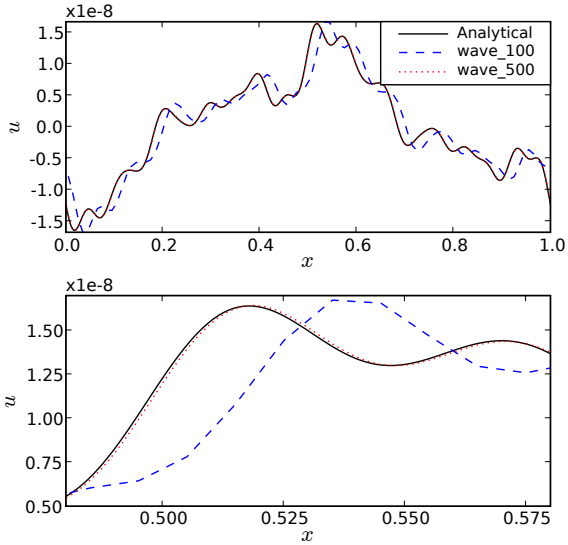

Figure 4.9: Velocity at $t = 3$ with 100 cells as compared to 500 cells for the wave propagation method. These are for $c_s = \sqrt{2}$ and $\omega_c = 50$. The bottom plot has an expanded scale to highlight the small phase error that is present even with 500 cells with large ω_c.

Method	l_2-norm	Computational time to $t = 1$s
WAVE_100	2.2×10^{-8}	0.02
WAVE_500	2.2×10^{-9}	0.30
RKDG3_33	1.3×10^{-8}	0.04
RKDG8_12	2.3×10^{-9}	0.19
RKDG16_6	6.6×10^{-10}	0.30

Table 4.3: l_2-norm of velocity to quantify accuracy for each method, and computational time required to advance the solution to $t = 1$s to quantify computational effort for the dispersive Euler system using $\omega_c = 50$.

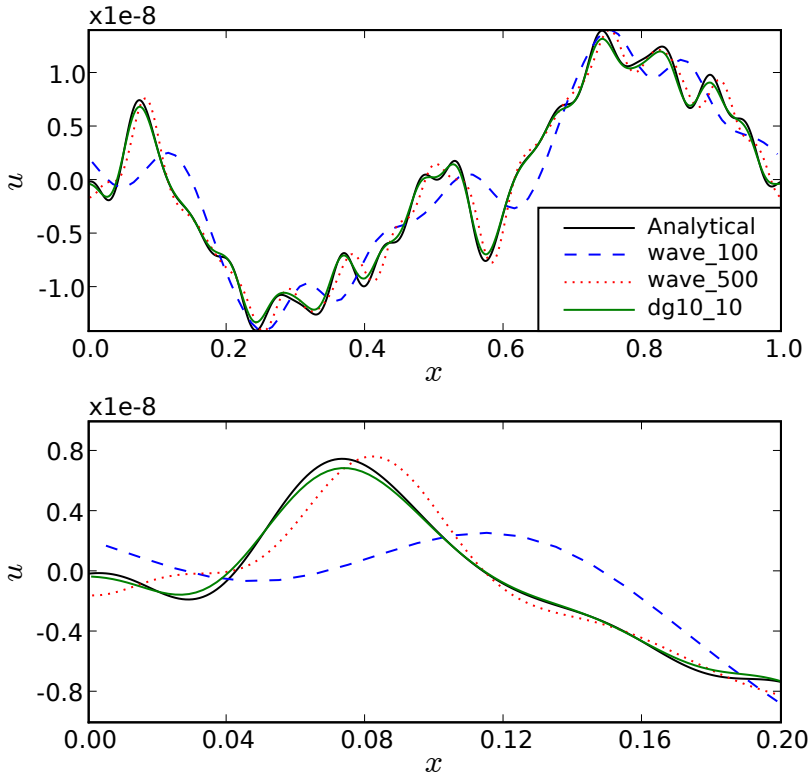

Figure 4.10: Velocity at $t = 3$ with 100 cells and 500 cells for the wave propagation method as compared to 10 cells for the 10^{th} order RKDG method. These are for $c_s = \sqrt{2}$ and $\omega_c = 100$. The bottom plot has an expanded scale to highlight the phase error that is present even with 500 cells when a higher cyclotron frequency is used.

Method	l_2-norm	Computational time to $t = 1$
WAVE_100	2.7×10^{-9}	0.1
WAVE_500	1.0×10^{-9}	0.4
RKDG10_10	3.5×10^{-10}	0.18

Table 4.4: l_2-norm of velocity to quantify accuracy for each method, and computational time required to advance the solution to $t = 1$ to quantify computational effort for the dispersive Euler system using $\omega_c = 100$.

solution with only 6 cells is more accurate than the 500 cell wave propagation method. The 16^{th} order RKDG method with only 6 cells uses the same computational effort as the wave propagation method with 500 cells. Table 4.3 also shows that the wave propagation method converges to order 1.9 which is consistent with the algorithm's ability to provide up to a 2^{nd} order accuracy. For large ω_c, the RKDG method provides a more accurate solution even when it is run at a lower effective resolution using high spatial order. For comparable accuracy with large ω_c, the RKDG method uses less computational effort as compared to the wave propagation method. The phase errors in the wave propagation method are caused by the large source terms compared to the advection terms that result from increasing ω_c. This hypothesis is supported by measuring larger phase errors when the source term is increased by setting $\omega_c = 100$ as compared to the $\omega_c = 50$ solution in Fig. 4.9. The $\omega_c = 100$ results are shown in Fig. 4.10 and Table 4.4. In particular, the error for the 500 cell solution is larger for the $\omega_c = 100$ solution than for the $\omega_c = 50$ solution. When the magnitude of the source term becomes large compared to the advection terms in the equation system, the wave propagation method produces phase errors. Increasing the source term strength for a given resolution increases the error. Table 4.4 shows that a 10^{th} order RKDG method with 10 cells has higher accuracy and uses less computational effort than a 500 cell wave propagation method when $\omega_c = 100$. Hence, the proper handling of source terms becomes critical.

The wave propagation method uses the source term splitting described in Sec. 3.1, and this splitting leads to the phase errors. The characteristic oscillation period caused by

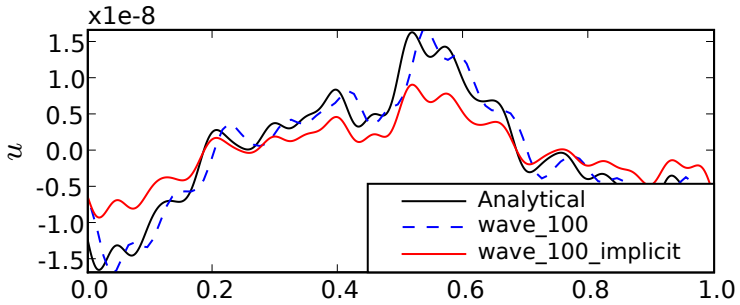

Figure 4.11: Velocity at $t = 3$ with 100 cells for the wave propagation method using source splitting versus using an unsplit implicit source term update. These are for $c_s = \sqrt{2}$ and $\omega_c = 50$. The implicit source update is included to prove that source splitting is responsible for the phase errors. However, the implicit solution is subject to severe diffusion.

the source terms is $\tau_c = 2\pi/\omega_c$ for the solution. The characteristic time for information to propagate is $\tau_s = \Delta x/c_s$. For the oscillation to be well resolved, τ_c must be sufficiently larger than τ_s. For the case of the wave propagation method with 100 cells, this requirement is violated because $\tau_c = 0.126$ while $\tau_s = 0.071$, which is not sufficient to resolve τ_s. The lack of proper sampling leads to the phase error seen in Figs. 4.8 and 4.10. The characteristic frequency of the sources introduces ω_c^{-1} time-scales that must be resolved in addition to the other time-scales in the system. The explicit time-step must be sufficiently small for proper sampling of the source frequency such that $\Delta t < \omega_c^{-1}$. To further support that the source splitting causes the phase errors, an unsplit implicit source term update is implemented for the wave propagation method using 100 cells with $\omega_c = 50$. The implicit source term update is described by

$$\mathbf{Q}(t+\Delta t) = \left(\mathbf{I} - \frac{\Delta t}{2}\mathbf{J_S}\right)^{-1}\left[\mathbf{Q}(t) - \Delta t\mathcal{L}(\mathbf{Q}(t)) + \frac{\Delta t}{2}\mathbf{J_S}\mathbf{Q}(t)\right] \tag{4.8}$$

where $\mathcal{L}$ represents the flux update. Figure 4.11 shows that while the implicit source term solution is more diffusive, the phase errors are eliminated. Exploring unsplit source term handling for the wave propagation method for equation systems with purely dispersive source terms, without the diffusive nature of an implicit source term update, could make the wave propagation method more robust to such phase error problems.

A von Neumann analysis is performed to specifically quantify the stability condition for the source term update for the wave propagation method. It is noted that as long as the condition

$$\Delta t \leq \frac{2\sqrt{2}}{\omega_c} \tag{4.9}$$

is satisfied, the wave propagation method with a Runge-Kutta source advance is stable in the presence of large ω_c. If a Δt is chosen such that the stability condition in Eq. (4.9) is satisfied, and the time-step accounts for the additional ω_c^{-1} time-scale, the wave propagation solution with 100 cells becomes diffusive. The wave propagation method generally becomes diffusive with CFL numbers less than 1. This presents a numerical difficulty in resolving the physical dispersions accurately while minimizing diffusive errors and requires a higher grid resolution for the wave propagation method. Using higher order spatial representations with the RKDG method solves this problem with less computational effort and greater accuracy in the presence of large source terms.

4.3 Maxwell's Equations

Maxwell's equations are used to compare the wave propagation method to the discontinuous Galerkin method in two dimensions. The reason for using this equation system for the comparisons is because these equations form a part of the two-fluid plasma model that described in Chapter 2. To satisfy the $\nabla \cdot \mathbf{E}$ and $\nabla \cdot \mathbf{B}$ divergence constraints, a hyperbolic form of Maxwell's equations is used[1, 14].

There are no source terms for this equation system. The problem involves initializing a two-dimensional circular pulse in a rectangular domain with conducting wall boundary conditions on all four boundaries. The pulse is initialized only in the magnetic field profile, B_z while all other quantities are initialized to zero. The initial condition is set using a Gaussian profile and is offset from the center of the domain to allow for asymmetries in the solution. The initial condition is shown in Fig. 4.12.

A converged wave propagation solution at a resolution of 1000×1000 cells is used for comparisons with several spatial orders for the RKDG method and for several grid

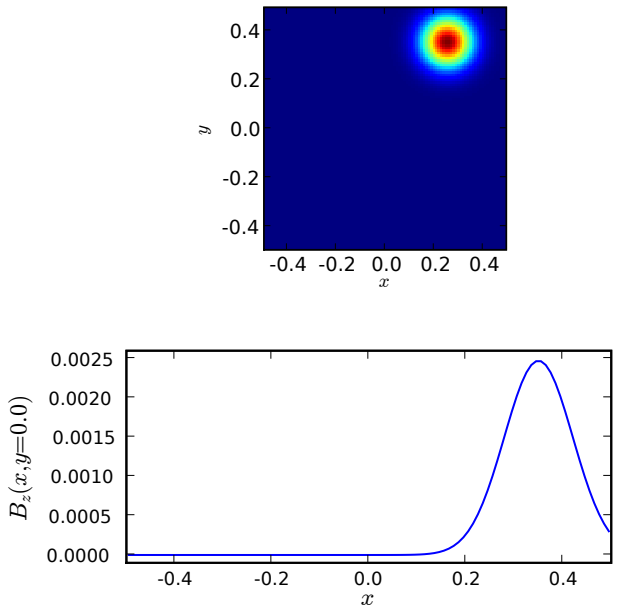

Figure 4.12: Initial conditions for out-of-plane magnetic field, B_z, used for Maxwell's equations circular pulse.

resolutions for both the RKDG and the wave propagation methods. The cross-section of the magnetic field, B_z, along the x-direction for $y = 0$ is compared among the methods and with the highly resolved solution to determine accuracy. Figure 4.13 shows the wave propagation solution compared at several grid resolutions. Figures 4.14 and 4.15 show the wave propagation solution with 100 cells compared to solutions obtained using several spatial orders with RKDG. Simulations are run to $t = 0.8$ where time is normalized by the speed of light transit time across the domain.

The wave propagation solution with 100×100 cells differs significantly from the converged solution. Therefore, a higher resolution is needed for the wave propagation method. With 300×300 cells the wave propagation method takes more computational effort than all the RKDG solutions explored in Figs. 4.14 and 4.15. In Fig. 4.14 all solutions have the same effective grid resolution. In Fig. 4.15, the RKDG solutions have lower effective grid resolution than the 100×100 cells wave propagation method and still provide a more accurate solution. The computational time required to advance the solution from $t = 0$ to $t = 0.2$

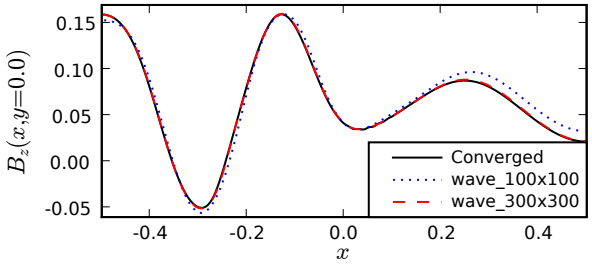

Figure 4.13: Cross-section of the magnetic field at time, $t = 0.8$ with $c = 1$, for the Maxwell's equations circular pulse using the wave propagation method with 100×100 cells and 300×300 cells.

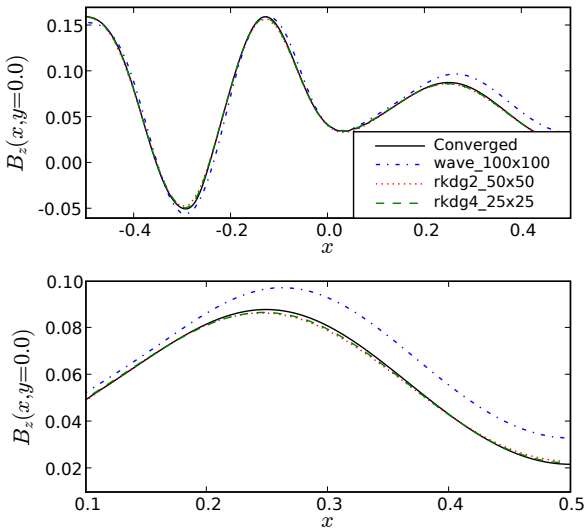

Figure 4.14: Cross-section of the magnetic field at time, $t = 0.8$ with $c = 1$ for the Maxwell's equations circular pulse. The bottom plot has an expanded scale to show the details of the solution. The wave propagation solution has a resolution of 100×100 cells, the 2^{nd} order RKDG uses 50×50 cells and the 4^{th} order RKDG uses 25×25 cells.

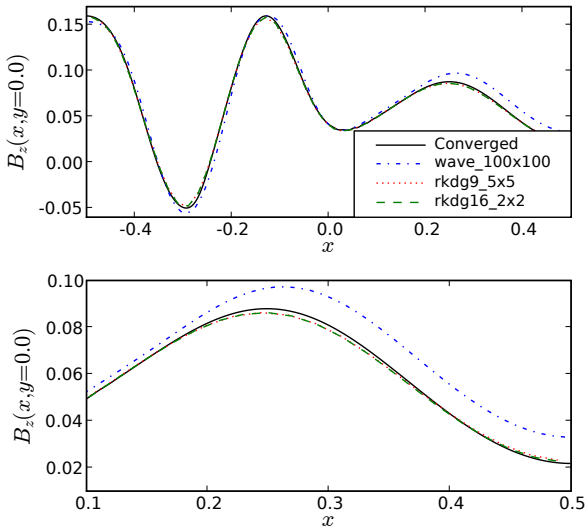

Figure 4.15: Cross-section of the magnetic field at time, t=0.8 with $c = 1$ for the Maxwell's equations circular pulse. The bottom plot has an expanded scale to show the details of the solution. The wave propagation solution has a resolution of 100×100 cells, the 9^{th} order RKDG uses 5×5 cells and the 16^{th} order RKDG uses 2×2 cells.

Method	l_2-norm	Computational time to $t = 0.2$s
WAVE_100x100	7.7×10^{-2}	0.74
WAVE_300x300	1.3×10^{-2}	23.8
RKDG2_50x50	1.6×10^{-2}	2.9
RKDG4_25x25	8.6×10^{-3}	6.9
RKDG9_5x5	8.2×10^{-3}	8.2
RKDG16_2x2	8.9×10^{-3}	24.2

Table 4.5: l_2-norm of B_z to quantify accuracy for each method, and computational time required to advance the solution to $t = 0.2$s to quantify computational effort for Maxwell's equation circular pulse.

for each method is presented in Table 4.5. Table 4.5 shows that the RKDG method is more accurate for this problem when using the same effective resolution as the wave propagation method as well as when using a lower effective resolution compared to the wave propagation method.

Since the 300×300 cell wave propagation solution provides a more accurate result than the 100×100 cell wave propagation solution, the 300×300 cell result is used to compare the computational effort of the two methods. Each method in Table 4.5 has a different CFL stability limit, and each is operated at its maximum CFL value. The wave propagation method, with a grid resolution of 300×300 cells in Fig. 4.13 takes 8 times more computational effort than the 2^{nd} order RKDG solution with 50×50 cells in Fig. 4.14, 3.5 times more computational effort than the 4^{th} order RKDG solution with 25×25 cells in Fig. 4.14, 3 times more computational effort than the 9^{th} order RKDG solution with only 5×5 cells in Fig. 4.15, and approximately the same computational effort as the 16^{th} order RKDG solution with only 2×2 cells in Fig. 4.15 to get a solution that is less accurate than the higher-order RKDG solutions. Even at a very low grid resolution of only 2×2 cells, the 16^{th} order RKDG solution provides a more accurate solution than the 100×100 and the 300×300 cell results for the wave propagation method. It is more computationally efficient to use the RKDG method for this problem.

4.4 Full Two-Fluid Plasma Model

Using the equations described in Chapter 2, comparisons of the two-fluid plasma model are presented here for a 1-dimensional axisymmetric pulse, and a 1- and 2-dimensional axisymmetric Z-pinch.

For axisymmetric problems, the divergence and curl operators in cylindrical coordinates have terms proportional to $1/r$. These additional terms are incorporated into the source terms as geometric sources as described in Sec. 3.2.7. The boundary conditions are treated in the manner described in Sec. 3.2.5.

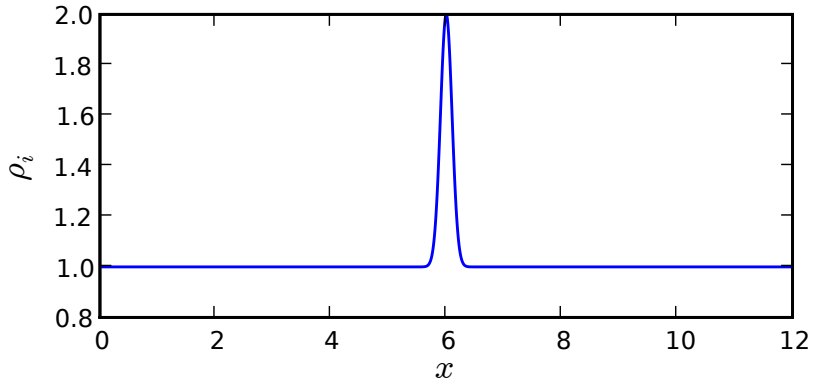

Figure 4.16: Initial ion mass density for the two-fluid plasma soliton. Ion pressure, electron mass density, and electron pressure have the same profile.

Method	l_2-norm	Computational time to $t = 40$
WAVE_512	1.0×10^{-2}	19.6
WAVE_1024	3.6×10^{-3}	43.2
RKDG2_256	4.5×10^{-3}	20.8
RKDG3_171	8.3×10^{-4}	9.96
RKDG4_128	4.6×10^{-4}	19.5
RKDG5_100	3.5×10^{-4}	10.3

Table 4.6: l_2-norm of ion mass density to quantify accuracy for each method and computational time required to advance the solution to $t = 40$ to quantify computational effort for the two-fluid soliton.

4.4.1 Two-Fluid Plasma Soliton in 1-Dimension

The two-fluid plasma model, is applied to one-dimensional soliton propagation[48] where a pulse is initialized in the ion and electron densities and pressures as shown in Fig. 4.16. The fluid pressures are initialized using fluid temperatures such that $T_i = T_e = 0.01$ and $\mathbf{B}_z = 1$. All other fluid and field variables are initialized to zero. The ion-to-electron mass ratio is 25. The ratio of the speed of light to the electron sound speed, $c/c_{se} = 2$, and the ratio of the speed of light to the ion sound speed, $c/c_{si} = 10$. The speed of light is chosen such that it is the fastest speed in the system. Periodic boundary conditions are used.

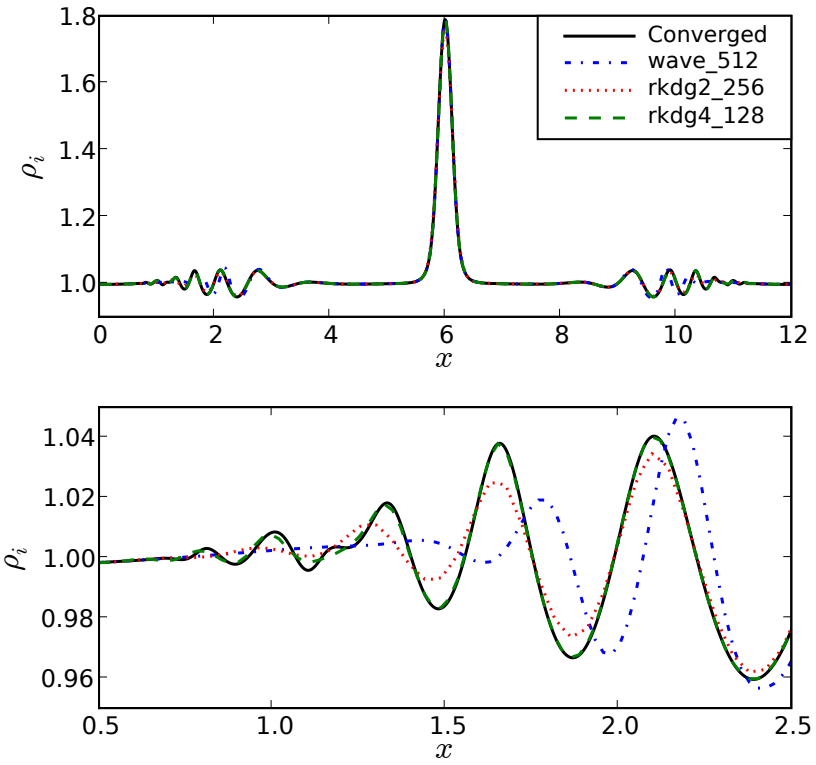

Figure 4.17: The ion mass density, ρ_i, is compared for the two-fluid plasma soliton using wave propagation and RKDG methods. The solution is shown at $t = 40$ with $c = 1$. The wave propagation method uses 512 cells, RKDG 2^{nd} order uses 256 cells, and RKDG 4^{th} order uses 128 cells so all methods have the same effective resolution. The bottom plot has an expanded scale to show the details of the solution. The wave propagation method has the largest phase errors.

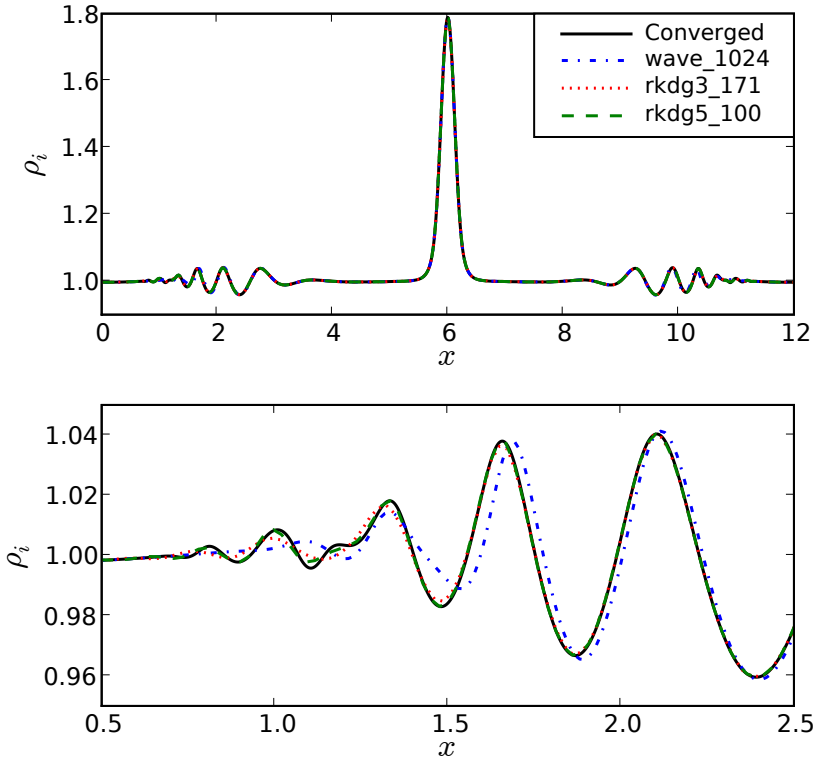

Figure 4.18: The ion mass density, ρ_i, is compared for the two-fluid plasma soliton using wave propagation and RKDG methods. The solution is shown at $t = 40$ with $c = 1$. The RKDG 3^{rd} order solution with 171 cells and the RKDG 5^{th} order solution with 100 cells provide a more accurate solution than the 1024 cell wave propagation method at double the effective resolution. The bottom plot has an expanded scale to show the details of the solution.

The wave propagation solution at a resolution of 5000 cells is chosen as the converged solution and is used to compare the wave propagation method to the RKDG method. This converged solution is compared to the 3^{rd} order RKDG using 1000 cells to verify that both methods converge to the same solution. Simulations are run to $t = 40$ where time is normalized by the speed of light transit time across the domain. One full period of the ion soliton occurs at $t = 100$. Figure 4.17 and Table 4.6 show that for the same effective resolution of 512 cells, i.e. wave propagation with 512 cells, 2^{nd} order RKDG with 256 cells and 4^{th} order RKDG with 128 cells, the RKDG method provides a more accurate solution than the wave propagation method. Figure 4.18 shows that even when the resolution of the wave propagation method is doubled to 1024 cells, the solution is less accurate than the approximate effective resolution of 512 cells using 3^{rd}, 4^{th}, and 5^{th} order RKDG methods. Phase errors in the wave propagation method solution are evident in the expanded scale plots of Figs. 4.17 and 4.18. These phase errors occur in the waves that are propagating away from the initial pulse and result from source splitting just as with the quasineutral ion cyclotron waves explored in Sec. 4.2.

The computational time required to advance the solution from $t = 0$ to $t = 40$ for each method is presented in Table 4.6. Each method has different CFL stability limit, and each is operated at its maximum CFL value. When using 1024 cells, the wave propagation method takes 2 times the computational effort as compared to the 4^{th} order RKDG method using 128 cells, and 4 times the computational effort as compared to the 5^{th} order RKDG method using 100 cells. There does not seem to be an obvious trend in the CPU times for the RKDG method. This nonmonotonic variability in the CPU times is attributed to the stability condition for the RKDG methods described in Ref.[26]. CFL$\leq 1/(2p-1)$ is valid for spatial order, p, as long as the temporal order is $p+1$. For all DG spatial orders presented here, a 3^{rd} order Runge-Kutta time integration scheme is used because 2^{nd} order Runge-Kutta time integration is unstable for DG when $p > 2$. In order to use 3^{rd} order Runge-Kutta time integration with $p > 2$, the maximum allowable CFL number is more restrictive as described in Ref.[26]. For this problem, the RKDG method provides a more efficient solution when computational effort is taken into account as shown in Table 4.6. While increasing the grid resolution eliminates the phase errors in the wave propagation method, the RKDG

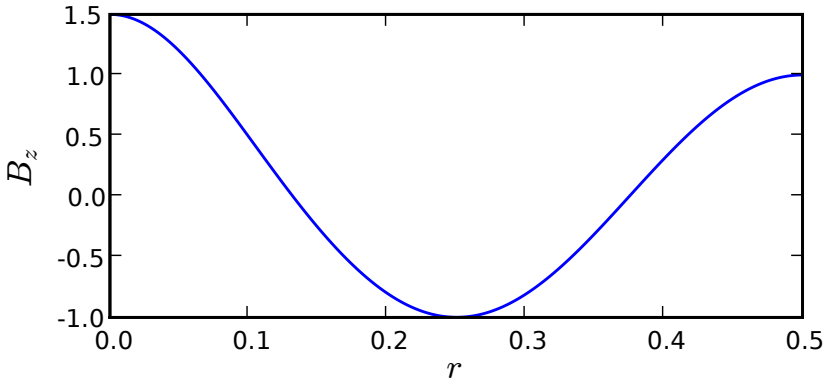

Figure 4.19: Initial condition for axial magnetic field for the axisymmetric two-fluid pulse.

method is more computationally efficient for this problem. Two-dimensional applications of a two-fluid soliton produce similar results.

4.4.2 Axisymmetric Two-Fluid Plasma Pulse in 1-Dimension

The two-fluid equations are applied to an axisymmetric one-dimensional problem where a pulse is initialized in the axial magnetic field, B_z, as shown in Fig. 4.19. The electron and ion densities and pressures are initially constant throughout the domain with all other fluid and electromagnetic terms initialized to 0. The ion-to-electron mass ratio is 2.5 to minimize negative pressure errors and allow for faster evolution. The ratio of the speed of light to the electron sound speed, $c/c_{se} = 35$, and the ratio of the speed of light to the ion sound speed, $c/c_{si} = 55$. Axis boundary conditions are used on the left edge of the domain while conducting wall boundary conditions are used on the right edge.

The wave propagation solution at a resolution of $10,000$ cells is chosen as the converged solution and is used to compare the wave propagation method to the RKDG method. This converged solution is compared to the 3^{rd} order RKDG using 1000 cells and it is verified that both methods converge to the same solution. The fluid azimuthal velocity, v_ϕ, shows the largest variation among the methods. For this reason, the electron azimuthal velocity is chosen for the comparisons. Simulations are run to $t = 0.8$ where time is normalized

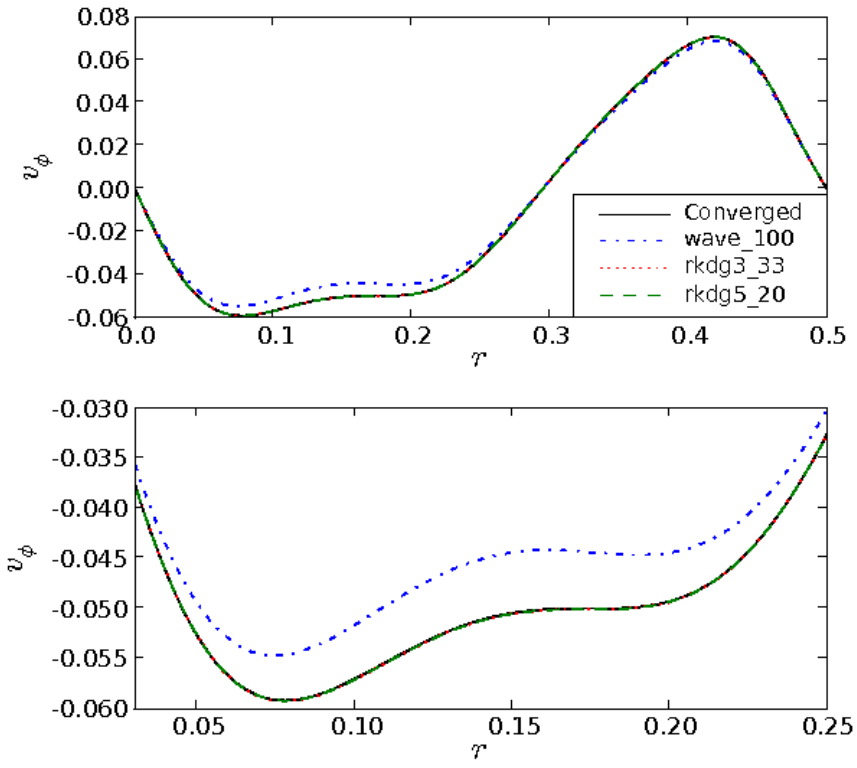

Figure 4.20: The electron fluid azimuthal velocity, v_ϕ, is compared for the axisymmetric two-fluid pulse using wave propagation and RKDG methods. The solution is shown at $t = 0.8$ with $c = 1$. The wave propagation method uses 100 cells, RKDG 3^{rd} order uses 33 cells and RKDG 5^{th} order uses 20 cells so all methods have approximately the same effective resolution. The bottom plot has an expanded scale to show the details of the solution. At the same effective resolution, the wave propagation method performs poorest.

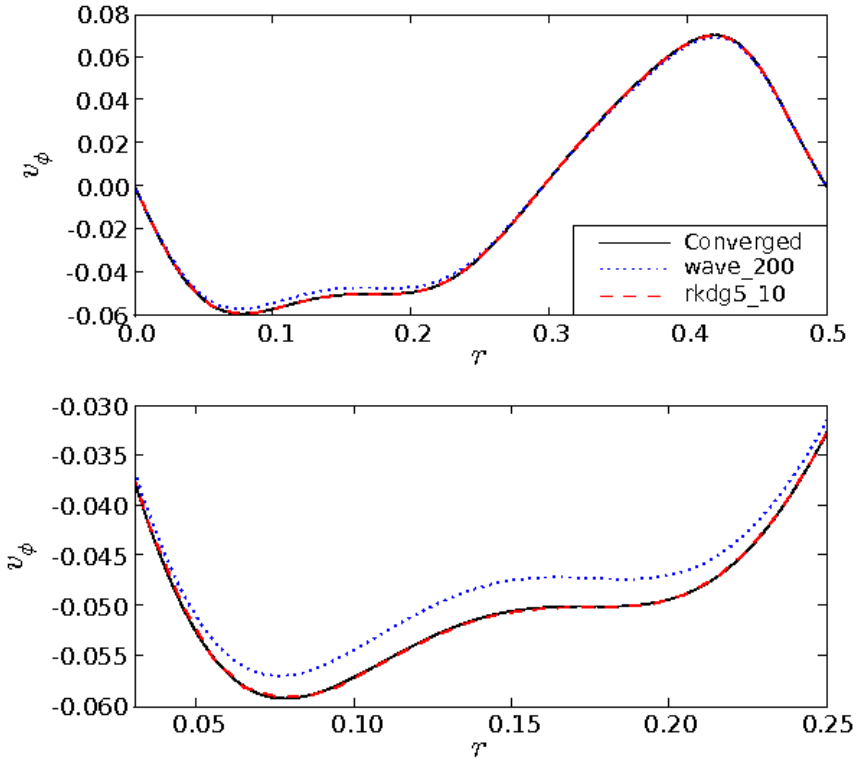

Figure 4.21: The electron fluid azimuthal velocity, v_ϕ, is compared for the axisymmetric two-fluid pulse using wave propagation and RKDG methods. The solution is shown at $t = 0.8$ with $c = 1$. The RKDG 5^{th} order solution uses 10 cells with an effective resolution that is 1/4 that of the wave propagation method which uses 200 cells. The bottom plot has an expanded scale to show the details of the solution. Even with a lower effective grid resolution that is 1/4 the resolution of the wave propagation method, the RKDG method provides a more accurate solution than the wave propagation method.

Method	l_2-norm	Computational time to $t = 0.4$s
WAVE_100	3.3×10^{-3}	0.04
WAVE_200	1.7×10^{-3}	0.14
RKDG3_33	4.2×10^{-5}	0.07
RKDG5_20	3.6×10^{-5}	0.09
RKDG5_10	9.9×10^{-5}	0.03

Table 4.7: l_2-norm of electron azimuthal velocity to quantify accuracy for each method, and computational time required to advance the solution to $t = 0.4$s to quantify computational effort for the two-fluid pulse.

by the speed of light transit time across the domain. Figure 4.20 and Table 4.7 show that for the same effective resolution of 100 cells, i.e. wave propagation with 100 cells, 3^{rd} order RKDG with 33 cells and 5^{th} order RKDG with 20 cells, the RKDG method provides a more accurate solution than the wave propagation method. Figure 4.21 shows that even when the resolution of the wave propagation method is doubled to 200 cells, the solution is less accurate than the effective resolution of 100 cells using 3^{rd} and 5^{th} order RKDG methods. Even the 5^{th} order RKDG method using only 10 cells, i.e. 1/4 the effective resolution of the 200 cell wave propagation method, provides a more accurate solution than the wave propagation method with 200 cells as is seen from Table 4.7. The computational time required to advance the solution from $t = 0$ to $t = 0.4$ for each method is presented in Table 4.7. Each method has different CFL stability limit, and each is operated at its maximum CFL value. When using 200 cells, the wave propagation method takes 2 times the computational effort as compared to the 3^{rd} order RKDG method using 33 cells, and 1.6 times the computational effort as compared to the 5^{th} order RKDG method using 20 cells. The 5^{th} order RKDG method using only 10 cells uses approximately 1/5 the computational effort of the 200 cell wave propagation method and still provides a more accurate solution.

For this problem, the RKDG method provides a more efficient solution when computational effort is taken into account as shown in Table 4.7. It is seen that the wave propagation solutions are slightly diffusive for the fluid variables. This can be attributed to the two-fluid plasma model containing disparate wave speeds in the system, namely the fluid speeds of

sound and the speed of light. As a result, information propagating at slower characteristic speeds in the system can be diffused. Such diffusion is not seen in the RKDG solutions because the higher spatial and temporal order of the RKDG method provides more accurate solutions even at lower grid resolutions.

4.4.3 Axisymmetric Z-Pinch Equilibrium in 1-Dimension

The two numerical methods are compared for a steady-state application of the two-fluid plasma model - the one-dimensional, axisymmetric Z-pinch equilibrium problem. The simulation is initialized using

$$p_0 = \frac{J_0^2}{1-\alpha}\left(\frac{1}{4}R_p^2 - 12R_p^4 + \frac{4}{3}128R_p^6\right) \tag{4.10}$$

$$J_{zi} = 0 \tag{4.11}$$

$$J_{ze} = \begin{cases} J_0\left(1-64r^2\right) & \text{if } r < R_p \\ 0 & \text{otherwise} \end{cases} \tag{4.12}$$

$$B_\phi = \begin{cases} J_0\left(\frac{1}{2}r - 16r^3\right) & \text{if } r < R_p \\ J_0\left(\frac{1}{2}R_p - 16R_p^3\right)\frac{R_p}{r} & \text{otherwise} \end{cases} \tag{4.13}$$

$$p = \begin{cases} p_0 - J_0^2\left(\frac{1}{4}r^2 - 12r^4 + \frac{4}{3}128r^6\right) & \text{if } r < R_p \\ p_0 - J_0^2\left(\frac{1}{4}R_p^2 - 12R_p^4 + \frac{4}{3}128R_p^6\right) & \text{otherwise} \end{cases} \tag{4.14}$$

where R_p is the pinch radius, $\rho_i = (m_i/m_e)\rho_e = m_i p/p_0$ and $p_i = p_e = \frac{1}{2}p$. In these simulations, $R_p = \frac{1}{8}$, $\alpha = \frac{1}{10}$ and $J_0 = \frac{1}{10}$. The ion-to-electron mass ratio is 25. All remaining variables are initialized to 0. The electron and ion density profiles shift radially a small amount to produce a radial electric field as the initialization adjusts to find an equilibrium.

Axis boundary conditions are used on the left edge of the domain, and conducting wall boundary conditions are used on the right edge.

The wave propagation method is compared to the 2^{nd}, 3^{rd}, 5^{th} and 8^{th} order RKDG methods. The methods are compared for their abilities to maintain equilibrium. The results shown in Fig. 4.22 are at a characteristic transit time of 50 on a domain $r = 0$

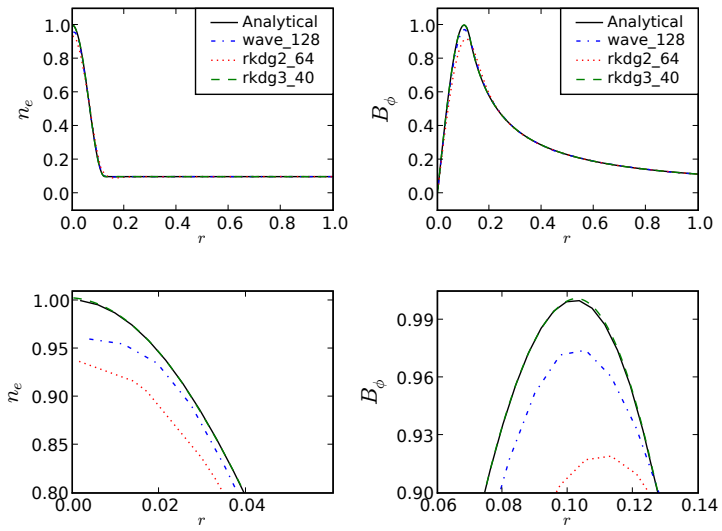

Figure 4.22: Electron number density and azimuthal magnetic field as functions of radius after 50 characteristic transit times are shown for the two-fluid axisymmetric Z-pinch equilibrium. The wave propagation method uses 128 cells, RKDG 2^{nd} order uses 64 cells and RKDG 3^{rd} order uses 40 cells so all methods have approximately the same effective resolution. The solution of the wave propagation method is diffusive compared to the 3^{rd} order RKDG and the 2^{nd} order RKDG is even more diffusive. All values normalized to the initial peak values. Lower plots have an expanded scale to show the details of the solution.

to $r = 1 = 16\mathrm{r}_{Li}$ with a conducting wall on the right boundary where r_{Li} is the ion Larmor radius. The parameters used here are $\gamma = 5/3$, speed of light $c = 1.0$, ion and electron charge-to-mass ratios of $q_i/m_i = 10$, $q_e/m_e = 250$, ion-to-electron mass ratio of $m_i/m_e = 25$, ion Larmor radius-to-domain length of $r_{Li}/x_0 = 1/16$, and ion skin depth-to-domain length of $\delta_i/x_0 = 1/10$. At a characteristic transit time of 20, the 2^{nd} order RKDG and wave propagation solutions are diffusive while the 3^{rd} order RKDG solution maintains equilibrium. The higher-order RKDG solutions, i.e. solutions greater than 2^{nd} order, hold equilibrium better than the wave propagation method for the same effective resolution.

When the grid resolution of the wave propagation method is doubled from 128 cells to 256 cells, it requires more computational effort than the 3^{rd} order RKDG method with 40 cells while still being more diffusive than the 3^{rd} order RKDG solution. Investigating this

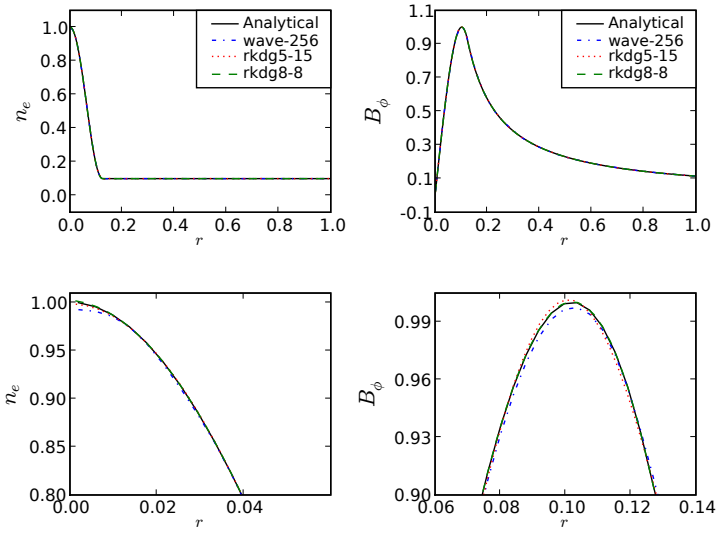

Figure 4.23: Electron number density and azimuthal magnetic field as functions of radius after 50 characteristic transit times are shown for the two-fluid axisymmetric Z-pinch equilibrium. The wave propagation method uses 256 cells, RKDG 5^{th} order uses 15 cells and RKDG 8^{th} order uses 8 cells so the effective resolutions of the RKDG solutions are about 1/3 and 1/4 that of the wave propagation solution. The low resolution, high spatial order RKDG solution holds equilibrium better than the wave propagation at 256 cells. All values normalized to the initial peak values. Lower plots have an expanded scale to show the details of the solution.

further in Fig. 4.23 with 5^{th} order RKDG using only 15 cells and 8^{th} order RKDG using only 8 cells it is seen that even at a resolution of 256 cells, the wave propagation method is diffusive as compared to the higher-order RKDG solutions with 1/3 and 1/4 the effective grid resolution. The RKDG method at higher spatial orders holds equilibrium better than the wave propagation method. It is seen from Table 4.8 that the 3^{rd} order RKDG solution provides the most accurate solution for the same effective grid resolution as the 128 cell wave propagation method. The computational time required to advance the solution for 50 characteristic transit times for each method is presented in Table 4.8. Each method has a different CFL stability limit, and each is operated at its maximum CFL value.

The full two-fluid plasma model has imaginary eigenvalues for the source terms. Performing a source splitting for the wave propagation method causes the equilibrium decay.

Method	l_2-norm	Computational time to $t = 50$s
WAVE_128	8.8×10^{-2}	3.7
WAVE_256	2.4×10^{-2}	13.8
RKDG2_64	1.5×10^{-1}	3.8
RKDG3_40	4.8×10^{-3}	5.4
RKDG5_15	1.5×10^{-2}	6.7
RKDG8_8	1.6×10^{-2}	5.7

Table 4.8: l_2-norm of electron number density to quantify accuracy for each method, and computational time required to advance the solution to 50 characteristic transit times to quantify computational effort for the two-fluid 1-dimensional Z-pinch.

Bale et al.[18] discuss this problem and the treatment of the source terms. They determine that the source term handling through the process of splitting applied to the wave propagation method might not work well for solutions close to a steady-state. At equilibrium, the variation in fluxes must balance the source terms exactly and using split methods when the fluxes and sources are significant can introduce errors in the solution. Bale et al. suggest an alternate solution which involves only splitting the deviation from steady-state into waves. This is done by a process of distribution of the eigen-decomposition of the source terms into the neighboring cells based on the sign of the corresponding eigenvalues. This, however, is not applicable to the case of the two-fluid plasma model. The source Jacobian for this equation system has all imaginary eigenvalues, so this unsplit method is not suitable here.

4.4.4 Axisymmetric Z-Pinch Equilibrium in 2-Dimensions

A two-dimensional Z-pinch problem is investigated using the two-fluid plasma model. A perturbation is specified and evolved in time to see the resulting evolution of a two-fluid drift-turbulence instability. The wave propagation and RKDG methods are compared for their abilities to capture the physics appropriately. The computational expense is also compared for the two methods.

The axisymmetric Z-pinch simulations use the same initial conditions and perturbations

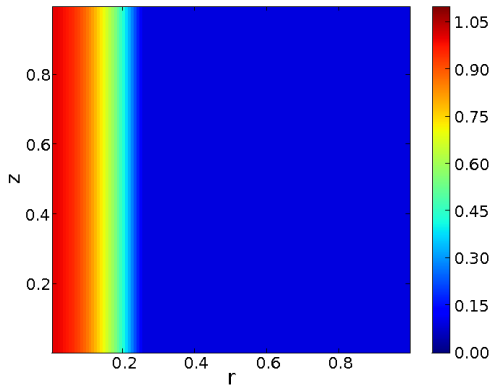

Figure 4.24: Initial conditions for ion density for a two-fluid Z-pinch equilibrium in 2-dimensions.

as in Ref.[3].

$$J_{zi} = 0 \tag{4.15}$$

$$J_{ze} = \begin{cases} J_0 & \text{if } r < R_p \\ 0 & \text{otherwise} \end{cases} \tag{4.16}$$

$$B_\phi = \begin{cases} \frac{1}{2} r \mu_0 J_0 \left[1 + \epsilon \sin(2\pi k z)\right] & \text{if } r < R_p \\ \frac{1}{2} \frac{R_p^2}{r} \mu_0 J_0 \left[1 + \epsilon \sin(2\pi k z)\right] & \text{otherwise} \end{cases} \tag{4.17}$$

$$p = \begin{cases} p_0 - \frac{1}{4}\mu_0 J_0^2 r^2 & \text{if } r < R_p \\ \alpha \mu_0 J_0^2 R_p^2 & \text{otherwise} \end{cases} \tag{4.18}$$

where R_p is the pinch radius, $p_0 = \frac{1}{4}(1+\alpha)\mu_0 J_0^2 R_p^2$, $\rho_i = (m_i/m_e)\rho_e = p/p_0$ and $p_i = p_e = \frac{1}{2}p$. In these simulations, $R_p = \frac{1}{4}$, $\alpha = \frac{1}{10}$, $J_0 = 1$, $\epsilon = 1/100$ and k denotes the wave number. The ion-to-electron mass ratio is 25. In two-dimensions, the initial ion density profile is shown in Fig. 4.24.

Axis boundary conditions are implemented on the left edge and conducting wall boundary conditions are implemented on the right edge, just like the one-dimensional case. For the RKDG method, the axis and conducting wall boundary conditions are treated in the manner described in Sec. 3.2.5. Periodic boundary conditions are implemented on the top

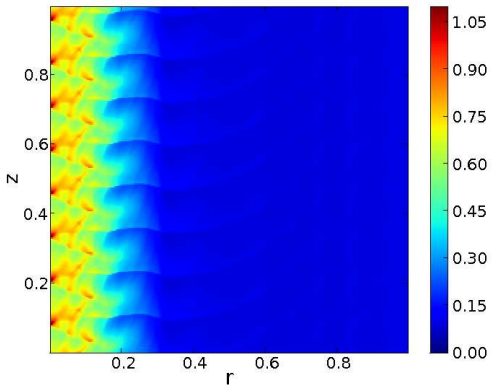

Figure 4.25: Ion density after 2 Alfvén transit times for the two-fluid axisymmetric Z-pinch in 2-dimensions using the wave propagation method with a grid resolution of 1000×1000 cells. This high resolution solution is used as a benchmark.

and bottom edges.

A sinusoidal perturbation as shown in Eq. (4.17) with $k = 8$ is applied to the azimuthal magnetic field, B_ϕ. The solution is then allowed to evolve to 2 Alfvèn transit times. A high resolution solution using a 1000×1000 cell wave propagation method is shown in Fig. 4.25 as a benchmark for the wave propagation and RKDG solutions. The high resolution solution is considered the converged solution for this problem and it is compared to a 500×500 cell RKDG solution, 1000×1000 cell RKDG solution, and a 2000×2000 cell wave propagation solution to ensure that both methods converge to the same solution. As the grid resolution is increased beyond the 1000×1000 cell wave propagation solution, the evolution of the instability does not change. The evolution for the wave propagation method with resolutions of 128×128 and 256×256 is shown in Fig. 4.26. This is compared to a 2^{nd} order RKDG solution with resolutions of 64×64 and 128×128 and a 3^{rd} order RKDG solution with a resolution of 128×128 as shown in Fig. 4.27. The plots shown are after 2 Alfvén transit times.

The drift parameter for this system is $v_e/v_{si} \approx 8$, where v_e is the electron drift velocity and v_{si} is the ion sound speed. The ratio of the pinch radius to the ion Larmor radius, r_p/r_{Li}, is approximately 3. Comparing Figs. 4.26 and 4.27 it is seen that both the wave

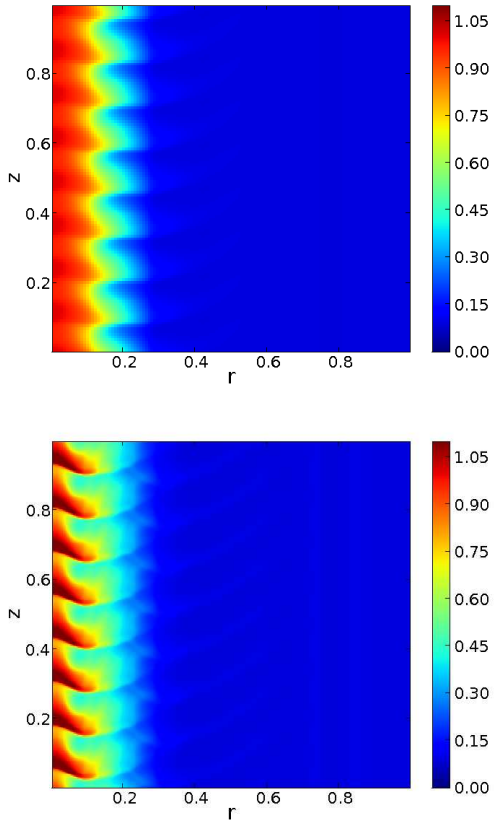

Figure 4.26: Ion density after 2 Alfvén transit times for the two-fluid axisymmetric Z-pinch in 2-dimensions using the wave propagation method with a grid resolution of 128×128 (top plot) and 256×256 (bottom plot).

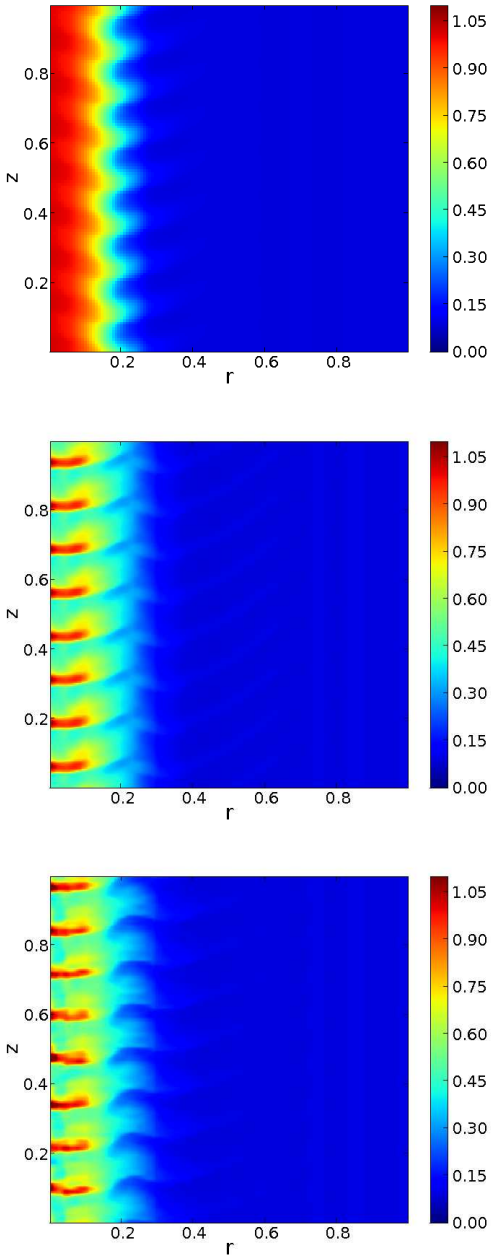

Figure 4.27: Ion density after 2 Alfvén transit times for the two-fluid axisymmetric Z-pinch in 2-dimensions using the 2^{nd} order RKDG method with a grid resolution of 64×64 (top plot), 2^{nd} order RKDG with 128×128 cells (middle plot), and 3^{rd} order RKDG with 128×128 cells (bottom plot). It is seen that the 3^{rd} order RKDG solution is very similar to the 2^{nd} order solution with 128×128 cells and is slightly closer to the high resolution solution.

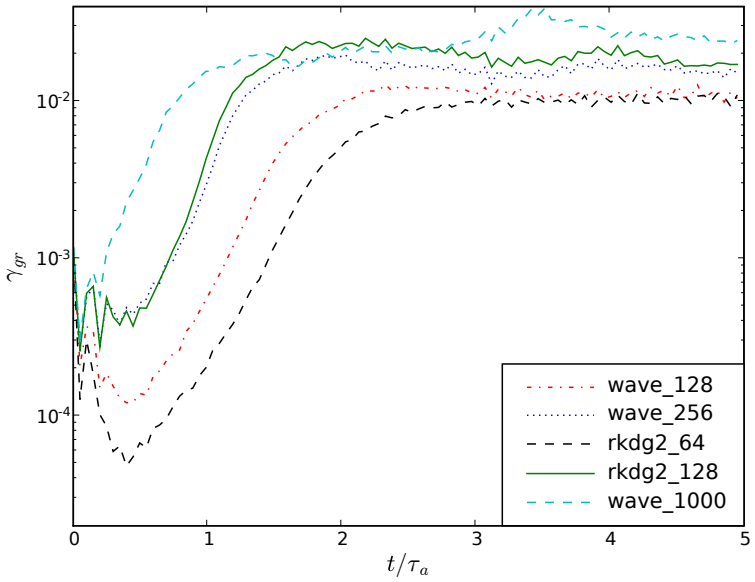

Figure 4.28: Growth rates for the perturbation in the magnetic field as a function of the time, t/τ_a, where τ_a is the Alfvén transit time. It is observed that increasing the resolution makes the instability growth rates approach that of the high resolution solution.

propagation method and the RKDG method (at 2^{nd} and 3^{rd} order) capture 8 wavelengths of the short-wavelength instability. This is better seen from the instability growth rates of Fig. 4.28 where the perturbation in magnetic field is measured against an unperturbed solution[3]. The growth rate of the instability is computed using

$$\gamma_{gr} = \iint |\Delta \mathbf{B}| \, 2\pi r dr dz \tag{4.19}$$

where $\Delta \mathbf{B}$ refers to the difference in magnetic field between the solutions of a perturbed equilibrium and an unperturbed equilibrium. The unperturbed equilibrium solution is needed to account for oscillations that occur in the system since the equilibrium is not a true two-fluid equilibrium initially.

Figure 4.28 shows that increasing the resolution causes the instability to set in sooner in time and have a greater slope. An increase in the resolution shows convergence towards the high resolution solution. The growth rates show that the 64×64 cell RKDG solution has the largest variation from the high resolution solution, while the 128×128 cell RKDG solution has the least. The instability is least to best resolved in the following order - 64×64 cell 2^{nd} order RKDG solution, 128×128 cell wave propagation solution, 256×256 cell wave propagation solution, 128×128 cell 2^{nd} order RKDG solution, 128×128 cell 3^{rd} order RKDG solution. For the same effective resolution of 256×256, the 2^{nd} order RKDG method takes about 1.3 times the computational effort of the wave propagation method. The wave propagation solution at a resolution of 256×256 takes about 9 times the computational effort of the 128×128 cell wave propagation solution. The 2^{nd} order RKDG solution at a resolution of 128×128 takes about 7 times the computational effort of the 64×64 cell RKDG solution. At the same effective resolution of 256×256, the growth rates of the RKDG and wave propagation solutions are comparable. The short wavelength instability is better resolved at higher grid resolutions.

Comparing the 2^{nd} and 3^{rd} order RKDG solutions at a resolution of 128×128 in Fig. 4.27, it is seen that the 3^{rd} order solution is very similar to the 2^{nd} order solution while only being slightly closer to the high resolution solution even though the 3^{rd} order solution uses the high-order limiters from Ref.[40]. The computational effort of the 3^{rd} order solution is about

10 times that of the 2^{nd} order solution for this resolution. Doubling the grid resolution while using 2^{nd} order for the RKDG method provides a more converged solution and only uses about 7 times greater computational effort. There might not be an advantage of going to higher-order RKDG solutions for this problem because the application of limiters locally reduces the solution to 1^{st} order when sharp gradients are present. The solutions clearly have sharp gradients throughout the domain.

Chapter 5

BENCHMARKING COLLISIONLESS AND COLLISIONAL TWO-FLUID PLASMA MODEL

5.1 Introduction

The full two-fluid plasma model is benchmarked to previously published results in this chapter following which it is used to obtain two-fluid results that are not captured by reduced fluid models.

Divergence errors are a common problem in plasma models and have been studied to a large extent for the MHD equations[49, 50]. Some of the methods previously explored include the 8-wave formulation for the MHD equations[51] where an eighth wave accounts for the divergence of the magnetic field. In problems with strong shocks, this scheme can produce incorrect jump conditions and consequently can lead to incorrect results demonstrated in Ref.[49]. Yet another scheme commonly used in MHD is the constrained transport[52], and central difference schemes which require a finite difference discretization on a staggered grid that maintains the $\nabla \cdot \mathbf{B}$ constraint. Additional methods include projection schemes[53], which may introduce errors in the presence of discontinuities in the solution, and mixed potential formulations of Maxwell's equations, which require proper treatment of the Lorentz gauge condition.

Divergence errors in MHD can be large in regions of shocks. However, for most applications of the two-fluid plasma model, shocks occur in the ion fluid and not in the magnetic field, and only occasionally in the electron fluid. The charge separation that results in the presence of shocks could lead to numerical errors due to the solution of the Riemann problem at cell interfaces and the application of limiters to reduce the order in the vicinity of the shocks. This could introduce numerical errors in the electric field that can affect the remaining variables. Therefore, the $\nabla \cdot \mathbf{B}$ constraint is usually not problematic around shocks. The $\nabla \cdot \mathbf{E}$ constraint however, can be rather large in regions of shocks. The $\nabla \cdot \mathbf{B}$

constraint becomes problematic in regions of strong magnetic field dynamics such as fast reconnection.

The divergence constraints are satisfied initially for all the problems investigated in this dissertation, but the numerical methods could introduce small errors in the divergence constraints that could grow with time. This could result from insufficient grid resolution, or insufficient spatial order potentially brought about by the use of limiters to truncate the order of the numerical method in regions of shocks. Two problems are investigated for divergence errors and the effect of the divergence corrections using the described perfectly hyperbolic technique. The first is the application of the full two-fluid plasma model to an electromagnetic plasma shock to explore the effect of $\nabla \cdot \mathbf{E}$ errors in the presence of shocks. For the electromagnetic plasma shock, higher spatial orders of the discontinuous Galerkin method are not necessarily advantageous since the limiters reduce the order of the solution to 1 in regions of sharp gradients. Therefore, a spatial order of 2 is most suitable for this problem. The second application uses the full two-fluid plasma model and Hall-MHD to simulate magnetic reconnection where the tearing and reconnecting magnetic field lines can lead to $\nabla \cdot \mathbf{B}$ errors even if the divergence constraints are maintained initially.

Following a study of divergence errors of the two-fluid plasma model, the differences between the ideal and non-ideal two-fluid model using the magnetic reconnection problem and axisymmetric Z-pinch are explored.

5.2 Ideal Two-Fluid Electromagnetic Shock and Divergence Errors

An electromagnetic shock is initialized using the conditions described in Eq. (5.1) on the left half and right half of the domain. A realistic ion-to-electron mass ratio of 1836 is chosen for the simulations. The simulations are run using a grid resolution of 2000 cells for a grid from $[0, 1]$ where the shock is initialized at $x = 0.5$. The normalized speed of light is 1 such that the ratio of the light speed to the ion sound speed is $c/c_{si} = 155$, the light speed to the electron sound speed is $c/c_{si} = 3.6$, and the ratio of the light speed to the Alfvén speed is $c/v_A = 80$. Simulations are run to 10 light transit times. The initial conditions are the

same as Ref.[54].

$$\begin{pmatrix} n_e \\ u_e \\ v_e \\ w_e \\ P_e \\ n_i \\ u_i \\ v_i \\ w_i \\ P_i \\ E_x \\ E_y \\ E_z \\ B_x \\ B_y \\ B_z \\ \Phi \\ \Psi \end{pmatrix} = \text{left of shock:} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0.5 \times 10^{-4} \\ 1 \\ 0 \\ 0 \\ 0 \\ 0.5 \times 10^{-4} \\ 0 \\ 0 \\ 0 \\ 0.75 \times 10^{-2} \\ 0 \\ 1 \times 10^{-2} \\ 0 \\ 0 \end{pmatrix}, \quad \text{right of shock:} \begin{pmatrix} 0.125 \\ 0 \\ 0 \\ 0 \\ 0.05 \times 10^{-4} \\ 0.125 \\ 0 \\ 0 \\ 0 \\ 0.05 \times 10^{-4} \\ 0 \\ 0 \\ 0 \\ 0.75 \times 10^{-2} \\ 0 \\ -1 \times 10^{-2} \\ 0 \\ 0 \end{pmatrix}. \tag{5.1}$$

$\nabla \cdot \mathbf{B}$ does not produce divergence errors for this problem since it is always 0 for the 1-dimensional case. However, in the presence of shocks, $\nabla \cdot \mathbf{E}$ errors become significant and need to be treated appropriately. For Hall-MHD, the charge neutrality condition minimizes the effect of numerical $\nabla \cdot \mathbf{E}$ errors so this divergence constraint usually only poses an issue for the full two-fluid plasma model in the presence of shocks.

Figure 5.1 shows the evolution of the ion and electron number densities after 0.125 Alfvén transit times (τ_a) for solutions without any divergence error corrections, with error correction coefficients of γ=χ=1 (such that the error correction speeds are γc=χc=c) and with error correction coefficients of γ=χ=2 (such that the error correction speeds are

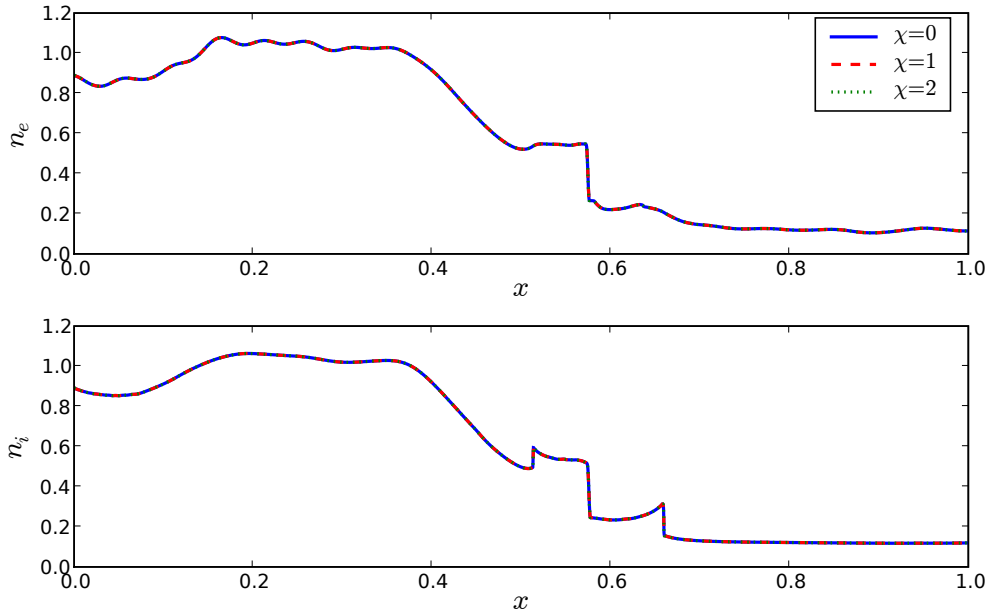

Figure 5.1: Electron (upper) and ion (lower) densities for the electromagnetic shock after $0.125\tau_a$ using $\nabla \cdot \mathbf{E}$ error correction coefficients χ = 0,1, and 2. The $\nabla \cdot \mathbf{E}$ error is not obvious from the fluid solution.

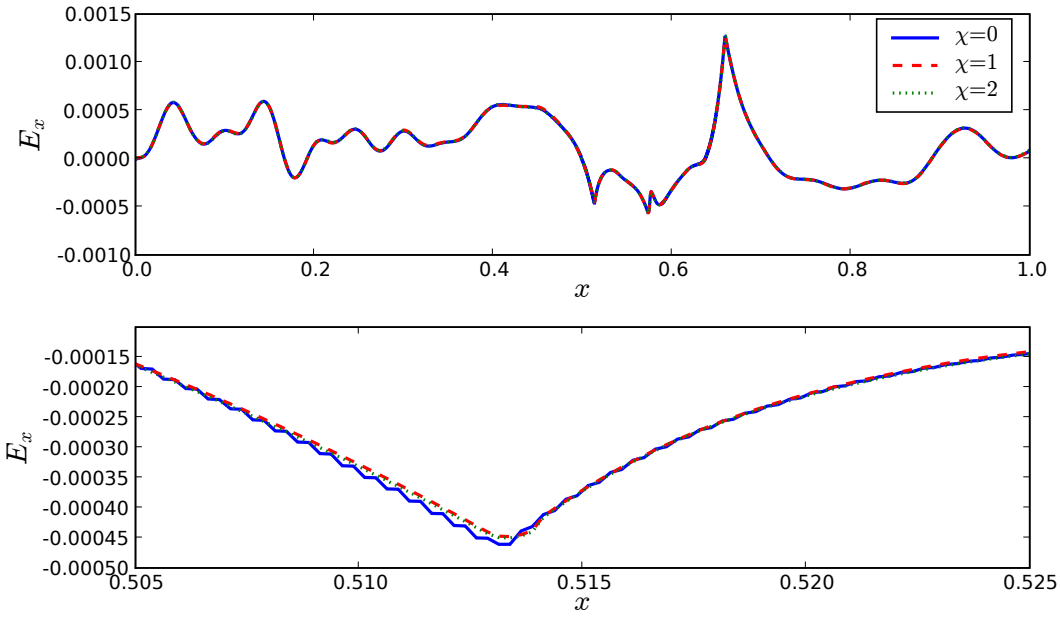

Figure 5.2: Electric field in the x-direction for the electromagnetic shock after $0.125\tau_a$ using $\nabla \cdot \mathbf{E}$ error correction coefficients $\chi = 0$,1, and 2. The bottom plot is at an expanded scale. Note $\chi > 0$ clearly decreases the numerical $\nabla \cdot \mathbf{E}$ errors that are seen in the $\chi = 0$ solution (blue line) and improves the solution by advecting the error out of the domain at a speed of χc.

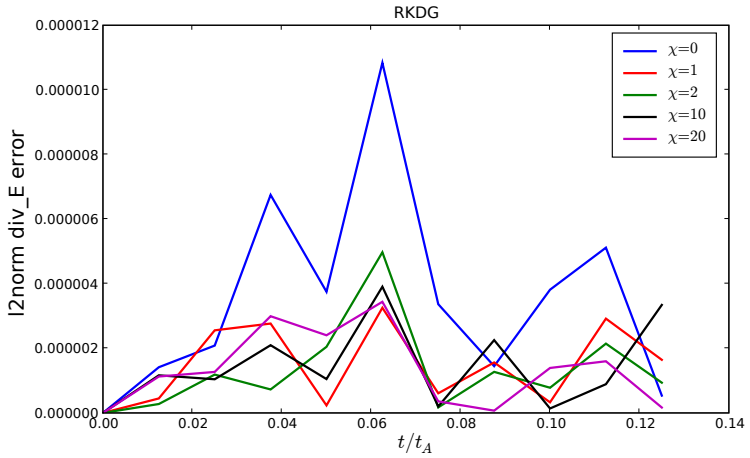

Figure 5.3: L_2-norm of $\nabla \cdot \mathbf{E}$ errors for RKDG method using values of $\chi = 0$,1,2,10,and 20. Note, $\chi = 1$ decreases the error, but $\chi > 1$ provides no additional benefit.

γc=χc=$2c$). Figure 5.2 shows the x-direction electric field along with an expanded scale of the plot to demonstrate the improvement in the solution when using $\nabla \cdot \mathbf{E}$ correction speed, χc. There is an improvement in going from $\chi = 0$ to $\chi = 1$ as is seen from the electric field after $0.125\tau_a$, but there is not much gained in going to higher values of χ with the DG algorithm.

An asymptotic analysis of the divergence errors by varying χ is performed in Fig. 5.3 for the RKDG method and in Fig. 5.4 for the wave propagation method. Fig 5.4 shows that the wave propagation method continues to correct for $\nabla \cdot \mathbf{E}$ errors when higher values of χ are used since higher values of χ produce lower l_2-norms. For the RKDG method, Fig 5.3 shows that the l_2-norm decreases when going from $\chi = 0$ to $\chi = 1$ but higher values of χ do not decrease the l_2-norm further. For the RKDG method, not much benefit is gained in using $\chi > 1$ for the two-fluid electromagnetic shock problem. Doing an asymptotic analysis of the grid resolution for the RKDG method in Fig. 5.5 shows that the $\nabla \cdot \mathbf{E}$ errors decrease with increasing grid resolution of the RKDG method. Overall, when no error corrections are used, the RKDG method has lower errors than the wave propagation method.

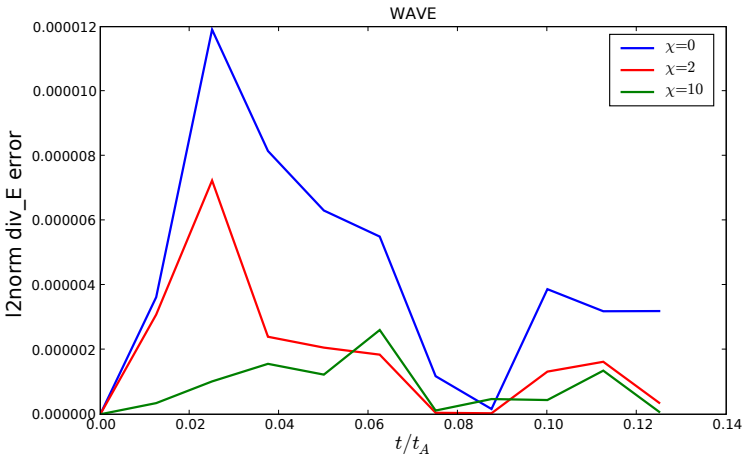

Figure 5.4: L_2-norm of $\nabla \cdot \mathbf{E}$ errors for wave propagation method using values of $\chi = 0$,2, and 10. Note, $\chi = 1$ decreases the error, and $\chi = 10$ decreases the error even further.

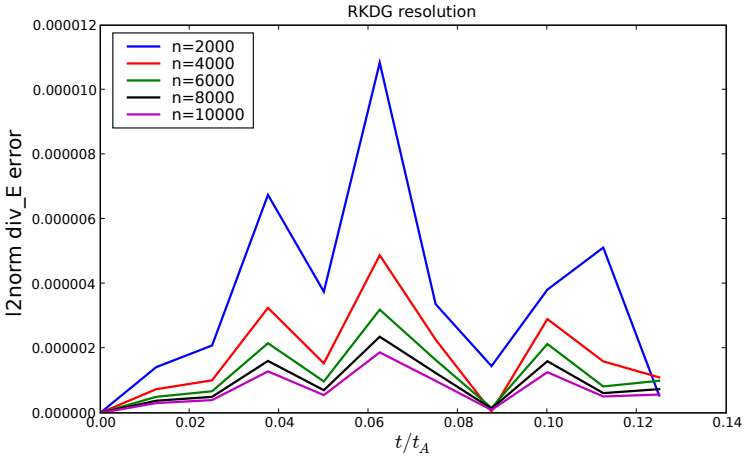

Figure 5.5: L_2-norm of $\nabla \cdot \mathbf{E}$ errors for different grid resolutions of the RKDG method using $\chi = 0$. Note, the error decreases with increasing grid resolution.

5.3 Ideal Two-Fluid Magnetic Reconnection Benchmark

The Geospace Environmental Modeling (GEM) reconnection challenge[55] is an effort to use various models to understand the mechanism of collisionless magnetic reconnection. Specifically the interest lies in exploring fast magnetic reconnection which is observed to occur in collisionless plasmas. This plays an important role in understanding space plasma physics phenomena such as solar flares, geomagnetic substorms, etc. Collisionless magnetic reconnection is observed using a number of plasma models for benchmarking and comparison purposes. The various models used include Hall-MHD with anisotropic pressure[56], resistive- and Hall-MHD[57, 58, 59], full particle[60, 61], and hybrid[62] models.

GEM challenge magnetic reconnection is simulated using the initial conditions similar to that described in Ref.[55]. The only difference in the initial conditions is the choice of a smaller value of B_0 to ensure that the normalized speed of light is the fastest speed in the system. The initial conditions are as follows for a domain ranging from $[-12.8, 12.8] \times [-6.4, 6.4]$,

$$B_x = B_0 \tanh \frac{y}{\lambda} - \psi_0 \frac{\pi}{l_y} \cos\left(2\pi \frac{x}{l_x}\right) \sin\left(\pi \frac{y}{l_y}\right) \tag{5.2}$$

$$B_y = \psi_0 \frac{2\pi}{l_x} \sin\left(2\pi \frac{x}{l_x}\right) \cos\left(\pi \frac{y}{l_y}\right) \tag{5.3}$$

$$j_z = \frac{B_0}{\mu_0 \lambda} \mathrm{sech}^2 \frac{y}{\lambda} \tag{5.4}$$

$$\rho_e = n_0 m_e \left(\mathrm{sech}^2 \frac{y}{\lambda} + \frac{1}{5}\right) \tag{5.5}$$

$$\rho_i = n_0 m_i \left(\mathrm{sech}^2 \frac{y}{\lambda} + \frac{1}{5}\right) \tag{5.6}$$

$$P_e = \frac{1}{12} \frac{B_0^2}{\mu_0} \frac{\rho_e}{m_e n_0 (\gamma - 1)} \tag{5.7}$$

$$P_i = \frac{5}{12} \frac{B_0^2}{\mu_0} \frac{\rho_i}{m_i n_0 (\gamma - 1)} \tag{5.8}$$

$$\tag{5.9}$$

where a perturbation, $\psi_0 = 0.1 B_0$, is applied in B_x and B_y. Other parameters include $n_0 = 1$, $B_0 = 0.1$, $l_x = 25.6$, $l_y = 12.8$, $m_i/m_e = 25$, and current sheet thickness, $\lambda = 0.5$.

Figure 5.6 shows the evolution of the ion density, the out-of-plane current, and the magnetic field vectors at $t = 0$, $15\omega_{ci}$, and $25\omega_{ci}$ using a grid resolution of 128×64. Figure 5.7 shows the evolution at a grid resolution of 256×128 and Fig. 5.7 shows the evolution at a grid resolution of 512×256. Note formation of an island in the 512×256 cell solution that is not seen in the lower resolution cases. The island formation could result from the magnetic field lines tearing and reconnecting and trapping some plasma in the center of the domain. If there were no asymmetries in the grid and the solution, the smaller island in the center of the domain would reach an equilibrium at the center of the domain. But the presence of even the slightest asymmetries makes the island move to one side or the other and merge with the larger magnetic islands. Such asymmetries could be brought about by grid effects, numerical error in approximating edge flux calculations, or the application of limiters to name a few.

For a benchmark case, several grid resolutions of the reconnection problem are used to compute the reconnected flux to confirm convergence. The reconnected flux is a measure of the net y-direction magnetic field and is computed using

$$\phi(t) = \frac{1}{2} \sum_{x} |B_y(x, y = 0, t)| \Delta x \tag{5.10}$$

and is normalized to be in the same scale as previous publications. These results are presented in Fig. 5.9. All resolutions, produce reconnection rates (slope of the reconnected flux vs time) that are in agreement with each other and with previously published results [55, 58]. The total amount of reconnected flux after $40/\omega_{ci}$ is the same for all resolutions. The high resolution solution at 512×256 starts to reconnect earlier in time. The slopes of the lines, i.e. the reconnection rates, do not vary significantly among the different grid resolutions.

A survey of the effect of mass ratio on magnetic reconnection rate shows that varying the ion-to-electron mass ratio to use $M = 25$, 50, and 100, does not affect the amount of reconnected flux or the reconnection rate as seen from Fig. 5.10. For this simulation realistic space plasma parameters are used to ensure that, for all mass ratios, the fluid speeds are significantly smaller than the light speed. This realistic problem is run to the same number

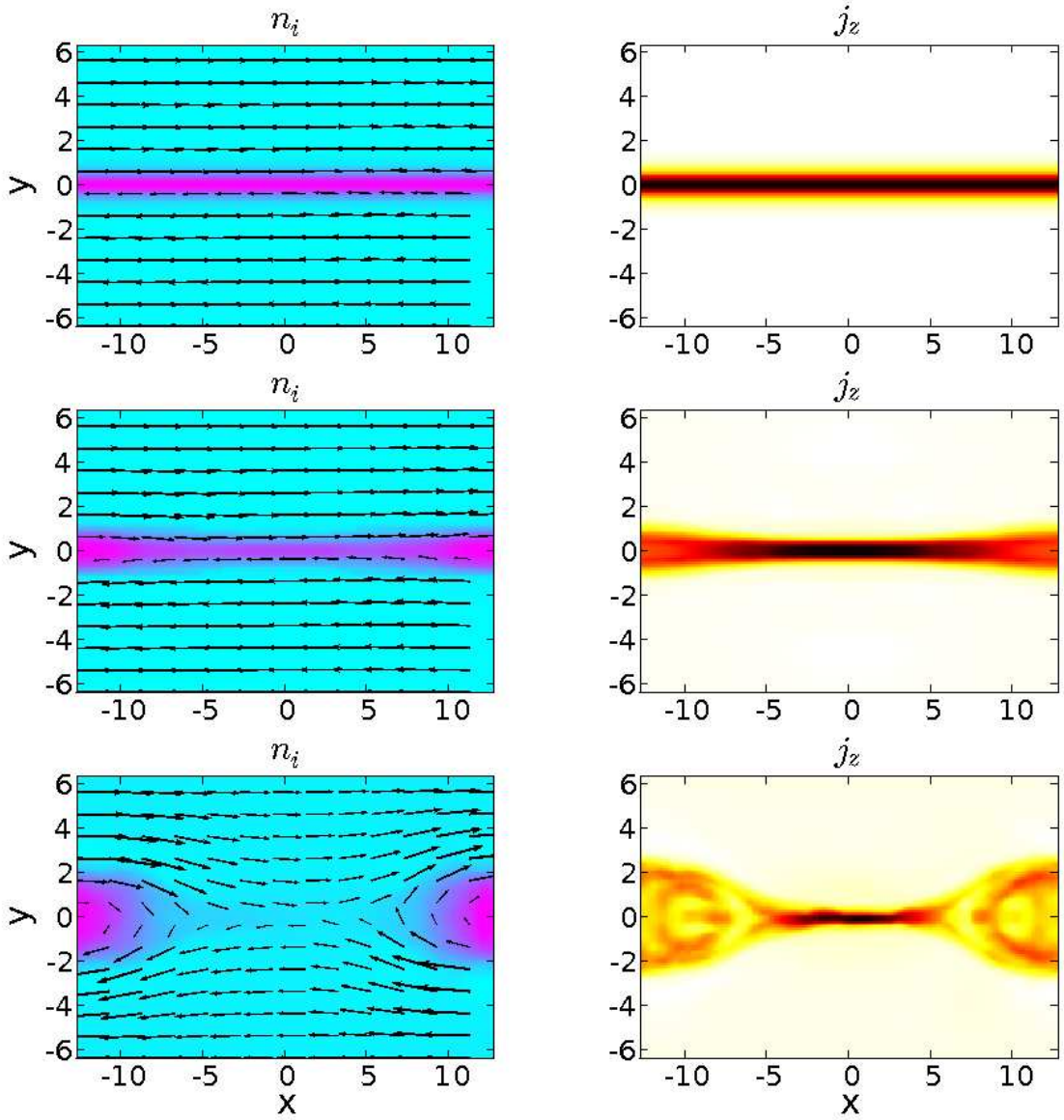

Figure 5.6: Solutions of GEM challenge magnetic reconnection at $t = 0, 15/\omega_{ci}, 25/\omega_{ci}$ from top to bottom using a grid resolution of 128×64. Left plots are of ion density with magnetic field vectors, and right plots are of total out-of-plane current.

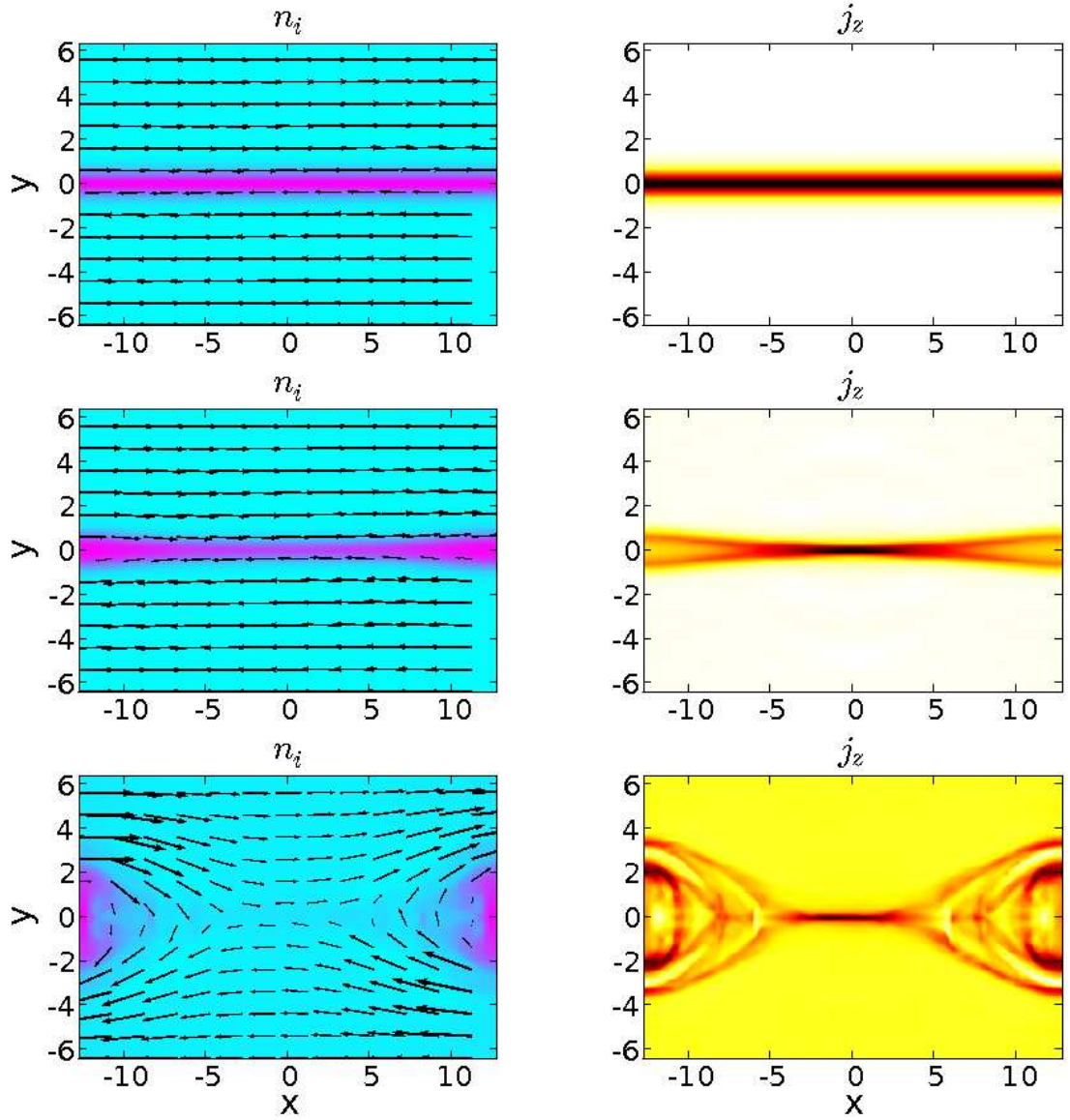

Figure 5.7: Solutions of GEM challenge magnetic reconnection at $t = 0, 15/\omega_{ci}, 25/\omega_{ci}$ from top to bottom using a grid resolution of 256×128. Left plots are of ion density with magnetic field vectors, and right plots are of total out-of-plane current.

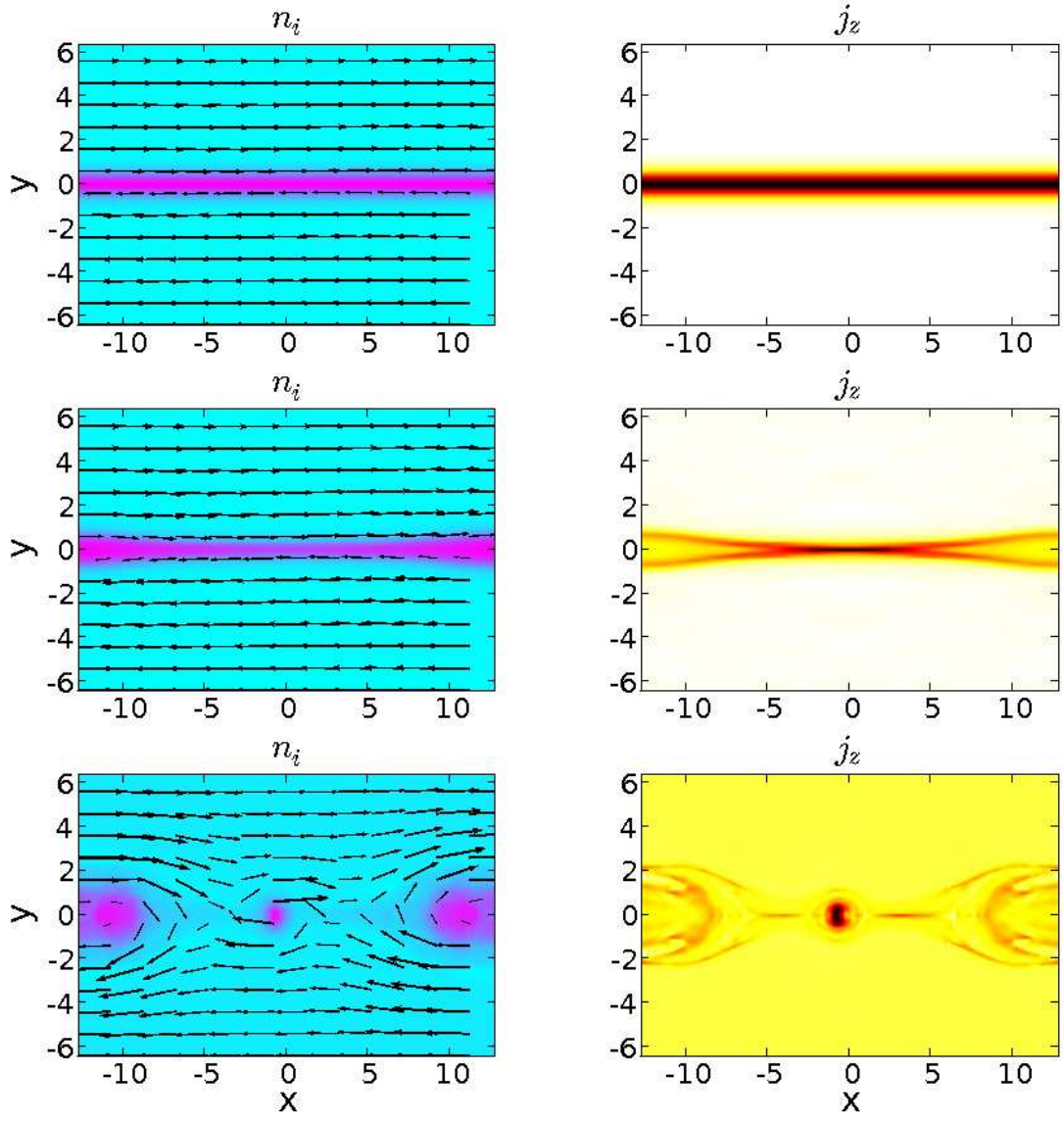

Figure 5.8: Solutions of GEM challenge magnetic reconnection at $t = 0, 15/\omega_{ci}, 25/\omega_{ci}$ from top to bottom using a grid resolution of 512×256. Left plots are ion density with magnetic field vectors, and right plots are total out-of-plane current. Note island formation in the higher resolution case that moves and merges to the left.

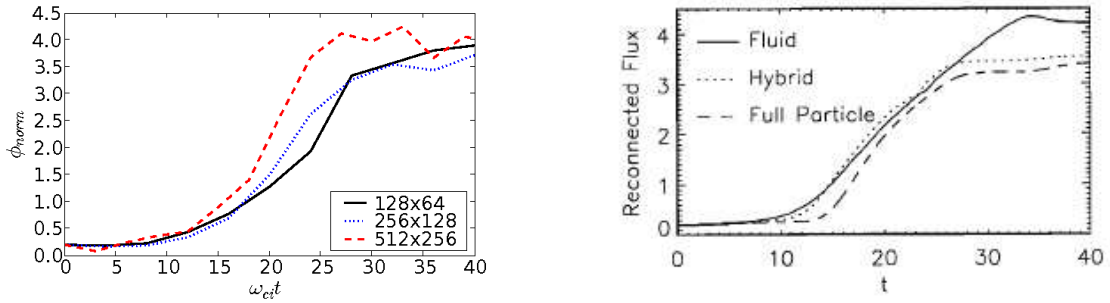

Figure 5.9: Left plot shows solutions of GEM challenge reconnected flux as a function of the ion cyclotron times ω_{ci}^{-1} for several grid resolutions. These results agree well with the right-hand-side plot GEM challenge results previously published in Ref.[55, 58].

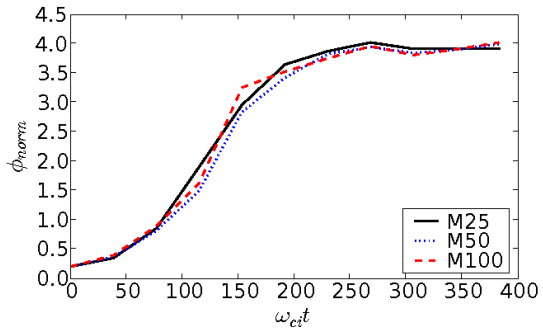

Figure 5.10: Solutions of realistic parameter regime magnetic reconnection problem reconnected flux as a function of the ion cyclotron times ω_{ci}^{-1} for several mass ratios, $M = 25, 50, 100$ using a larger perturbation for reduced computational cost.

of light transit times as the GEM challenge problem which results in $400/\omega_{ci}$. For this study, a higher magnetic field perturbation is used to trigger the reconnection process earlier in time to reduce computational cost. This problem differs from the artificial benchmark GEM challenge problem.

Shocks propagate in the ion fluid and several shocks in the ion fluid interact with each other while traveling in opposite directions. There are large turbulent flows that result in the ion momentum that appear as the islands form. The gradients in the electron fluid and the electromagnetic fields are not as strong. This system has no explicit dissipation added in the form of collisional effects so the only form of dissipation present is introduced

by the application of limiters in regions of sharp gradients. Fast, collisionless reconnection occurs when the fluids are able to break the frozen-in flux condition allowing the field line to tear and diffuse into the current layer. This is often done with the inclusion of resistivity in single-fluid MHD but that is an absolute minimum as single-fluid MHD does not capture fast magnetic reconnection accurately. The full two-fluid plasma model has the Hall term and the diamagnetic drift terms that allow the ion fluid to break the frozen-in flux condition similar to Hall-MHD. No artificial or physical dissipation terms are required for fast, collisionless reconnection to occur with the two-fluid plasma model. Additionally, the inclusion of electron mass and non-neutral effects through the displacement current allow the electron fluid to also break the frozen-in flux condition, a feature that is missed in most Hall-MHD models. The following section studies the magnetic reconnection solutions in the presence of collisional and transport effects with explicit dissipation using the equations described in Sec. 2.5.

A study of divergence error corrections, specifically $\nabla \cdot \mathbf{B}$ errors, for the GEM challenge magnetic reconnection problem is performed in Chapter 7 for the full two-fluid plasma model and for Hall-MHD.

5.4 Non-Ideal Two-Fluid Magnetic Reconnection

In order to explore magnetic reconnection with the inclusion of Braginskii's transport terms for the two-fluid plasma model, realistic astrophysical plasma parameters are used. The GEM challenge magnetic reconnection problem is an artificial problem that is primarily in the collisionless regime. Arbitrarily adding numerical resistivities, viscosities, and hyper-resistivities for purposes of numerical ease of computation may not be physically meaningful to this problem. In order to model fast reconnection with transport, simulations are performed using a realistic space plasma density of $n_0 = 1 \times 10^6 \mathrm{m}^{-3}$, a magnetic field of $B_0 = 1 \times 10^{-8}\mathrm{T}$, and other parameters are computed from these based on the initial conditions described in Sec. 5.3. The speed of light is artificially reduced to minimize computational cost while ensuring that it is the fastest speed in the system and the magnetic field perturbation is increased to trigger reconnection earlier in time. $M = 25$, $c/c_{se} = 9$, $c/c_{si} = 19$, $c/V_A = 19$, $ve/c_{si} \approx 1.3$, $l_x = 80\delta_i$, and $l_y = 40\delta_i$. The collision frequency

is over 10 orders of magnitude smaller than the plasma and cyclotron frequencies thus, the problem is primarily in the collisionless regime and the transport terms are computed self-consistently using Braginskii's transport terms described in Sec. 2.5. This allows physically relevant thermal and friction forces, viscosities, resistivities, heat fluxes, and thermal equilibration terms to be included in the model as they become relevant depending on the collision frequencies at any given time-step.

The Lundquist number for this regime is $S = \frac{\mu_0 l_y v_A}{\eta} \sim 10^{19}$, where η is the resistivity computed from the conserved variables. This is typical for astrophysical plasmas that often have $S > 10^{10}$[63]. In a number of reconnection simulations performed with Hall-MHD and single-fluid MHD, numerical resistivities and hyper-resistivities are chosen such that $S \sim 10^6 - 10^8$. Such lower Lundquist numbers are relevant for laboratory plasmas but these would be non-physical for most space plasmas in reconnection.

Figures 5.11 and 5.12 show solutions of ion density after $100/\omega_{ci}$ and $400/\omega_{ci}$ for the ideal and the non-ideal two-fluid plasma model. The densities appear to be very similar in both cases with subtle differences seen in the $400/\omega_{ci}$ solutions where at the same color-scale, the non-ideal solution looks smoother than the ideal solution. There are more significant differences observed in the out-of-plane currents in Fig. 5.13 where the ideal solution collapses to an X-point while the non-ideal solution has enough resistivity even in the low, relatively collisionless regime to allow for 2 Y-point solutions. The current sheet in the center of the domain for the non-ideal solution collapses to approximately the electron skin depth, δ_e which is in agreement with previous publications[64]. This is measured by computing the full-width half-maximum of the total current at the center of the domain. Also seen from Fig. 5.13 are smoother gradients with the non-ideal case as compared to the ideal solution. The smoother gradients result from the physical dissipative terms present through viscosity, resistivity, and heat flux which damp out some of the highly turbulent motions in the ion fluid.

A survey of the fluid and electromagnetic energies with the ideal and non-ideal two-fluid model in Fig. 5.14 shows that the magnetic energy is converted to fluid energy as the magnetic fields reconnect and reach the lowest energy state. There is not too much of a difference between the ideal and non-ideal cases, the ion fluid gains most of the energy that

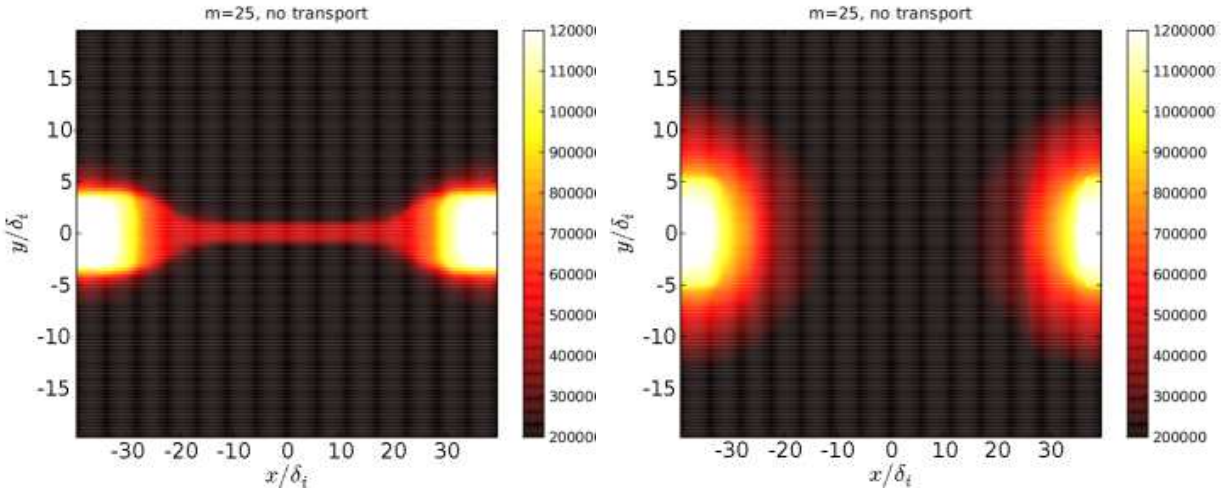

Figure 5.11: Left plot shows solutions of ion density magnetic reconnection after $100/\omega_{ci}$. Right plot shows solutions of ion density magnetic reconnection after $400/\omega_{ci}$. These are plots of the ideal two-fluid plasma model using realistic astrophysical plasma parameters of $n_0 = 1 \times 10^6$ and $B_0 = 1 \times 10^{-8}$.

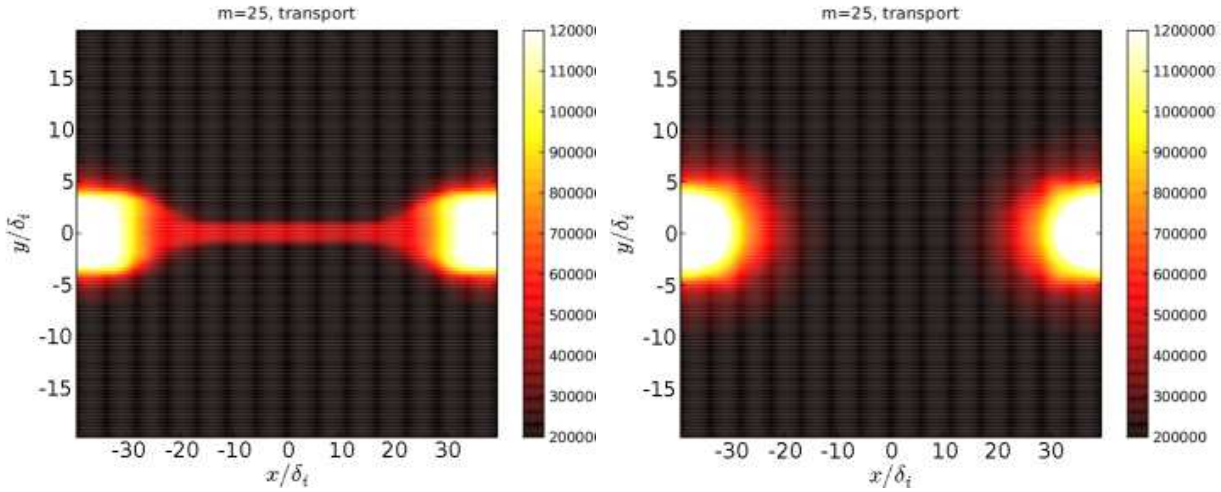

Figure 5.12: Left plot shows solutions of ion density magnetic reconnection after $100/\omega_{ci}$. Right plot shows solutions of ion density magnetic reconnection after $400/\omega_{ci}$. These are plots of the non-ideal two-fluid plasma model with Braginskii's transport coefficients using realistic astrophysical plasma parameters of $n_0 = 1 \times 10^6$ and $B_0 = 1 \times 10^{-8}$.

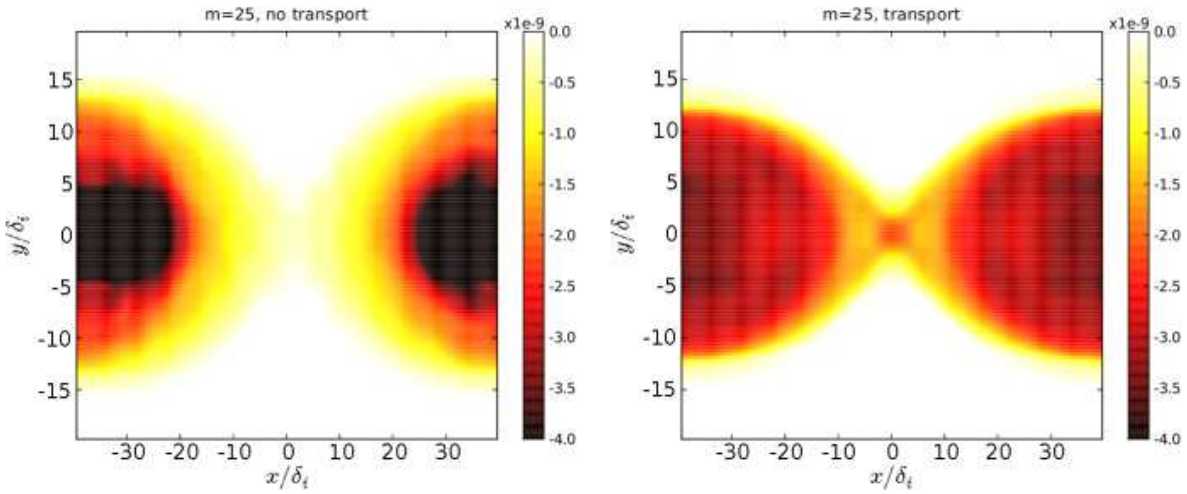

Figure 5.13: Left plot shows solutions of out-of-plane current for magnetic reconnection after $400/\omega_{ci}$ using the ideal two-fluid plasma model. Right plot shows solutions of out-of-plane current for the non-ideal two-fluid plasma model after $400/\omega_{ci}$.

is lost by the magnetic field. The electron fluid gains a little energy from the magnetic field. The electric field energy is very small in these cases.

Additional simulations are performed by varying the size of the current sheet, λ, with respect to the ion and electron skin depths, δ_i and δ_e. Magnetic reconnection is explored for several regimes using the two-fluid plasma model that has the capability to span the entire range from the single-fluid MHD limit to the full two-fluid limit. For $\lambda > \delta_i$, as studied previously in Figs. 5.11-5.13, both ions and electrons are magnetized and the MHD model could be sufficient if resistivity is included to break the frozen-in flux condition as long as $\lambda \gg \delta_i$. Hall-MHD is also applicable in this regime as is the full two-fluid plasma model. For the regime where $\delta_i > \lambda > \delta_e$, the ions are unmagnetized while the electrons are magnetized. In this regime, Hall-MHD is required as a minimum and can sufficiently capture reconnection. However, if δ_e scales become significant as is the case if the electrons become unmagnetized, then a resistivity is necessary for Hall-MHD to accurately capture reconnection. Using the two-fluid plasma model for $\delta_i > \lambda > \delta_e$, the solutions obtained do not significantly differ from the $\lambda > \delta_i$ regime. These regimes are explored to identify how the current layer collapses if the current sheet thickness is changed and to see how

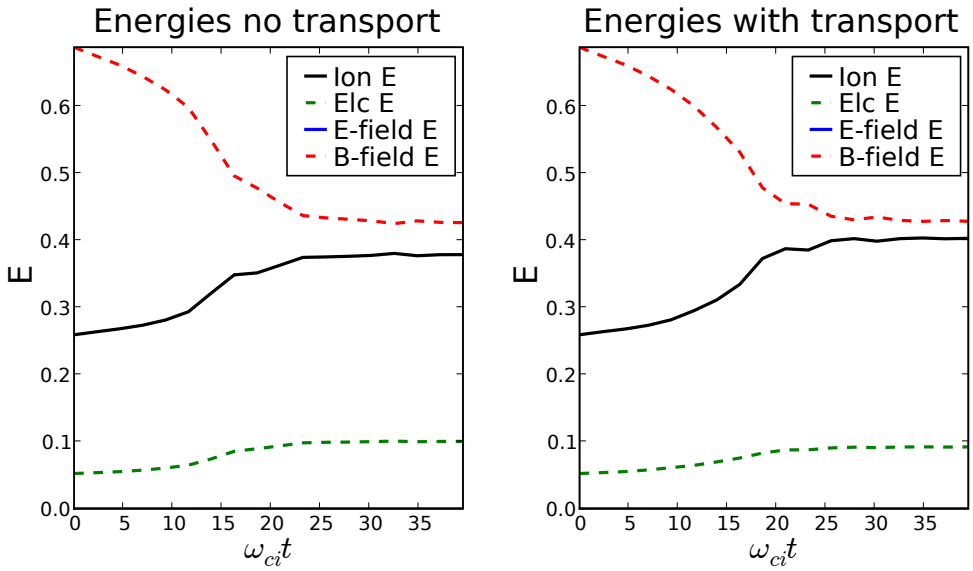

Figure 5.14: Realistic reconnection solutions of normalized total ion, total electron, electric field and magnetic field energies for the ideal and non-ideal two-fluid model. The total ion and electron energies are the sum of the internal and kinetic energies of each of the species. The electric field energy is very small. The ion fluid gains most of the energy and the electron fluid gains some energy from the magnetic field.

reconnection progresses as the regime is changed from magnetized ions and electrons to unmagnetized ions and electrons.

For $\lambda < \delta_e$, Hall-MHD requires the inclusion of resistivity to allow the electrons to become unmagnetized. The ideal full two-fluid plasma model self-consistently simulates reconnection when the electrons are unmagnetized. For simulations in this regime, the domain is reduced to keep the same ratio of current sheet thickness to domain size. For $\lambda < \delta_e$, the island in the ideal two-fluid solution starts to develop a velocity at around $250/\omega_{ci}$ following which the island accelerates to the right of the domain as seen in Fig. 5.15. This acceleration is not present when transport coefficients are included in the two-fluid plasma model as seen in Fig. 5.16 where it appears to reach a steady-state. Figure 5.17 shows a comparison of the electron density for the ideal and collisional cases at several ion cyclotron times which better shows the acceleration. Higher resolution simulations using 256×128 cells and 512×256 cells are performed for the ideal model to study whether this acceleration is an artifact of insufficient resolution and numerical errors. Figure 5.18 shows that this behavior is present even at higher resolutions, the main difference being the direction in which the island propagates. In fact, it appears that the 256×128 resolution solution and the 512×256 resolution solution have the same solution after $350/\omega_{ci}$ with the same preferred direction of acceleration of the island. This behavior is attributed to random turbulence in the fluids due to the absence of dissipative terms in the ideal model.

For $\lambda \sim \delta_e$, the non-ideal solution of the current layer does not collapse to δ_e anymore since the current sheet was initialized to be smaller than δ_e. The acceleration of the ideal solution is attributed to currents and magnetic fields that develop leading to a $\mathbf{J} \times \mathbf{B}$ force that pushes the solution to the right. Figures 5.19 and 5.20 show that the J_y and B_z that develop in the ideal solution, are an order of magnitude larger than the non-ideal solution. The larger magnitude of the J_y current and B_z field at the center of the magnetic island produces a force to the right for the ideal solution. This force acts on the center of the island and imparts momentum to both fluids at the center of the island. The fluid at the trailing edge tries to catch up due to the pressure differential that develops from the accelerating fluid at the center. Eventually the resulting low pressure in the trailing edge becomes so small that negative pressure errors result in the algorithm and the simulation crashes.

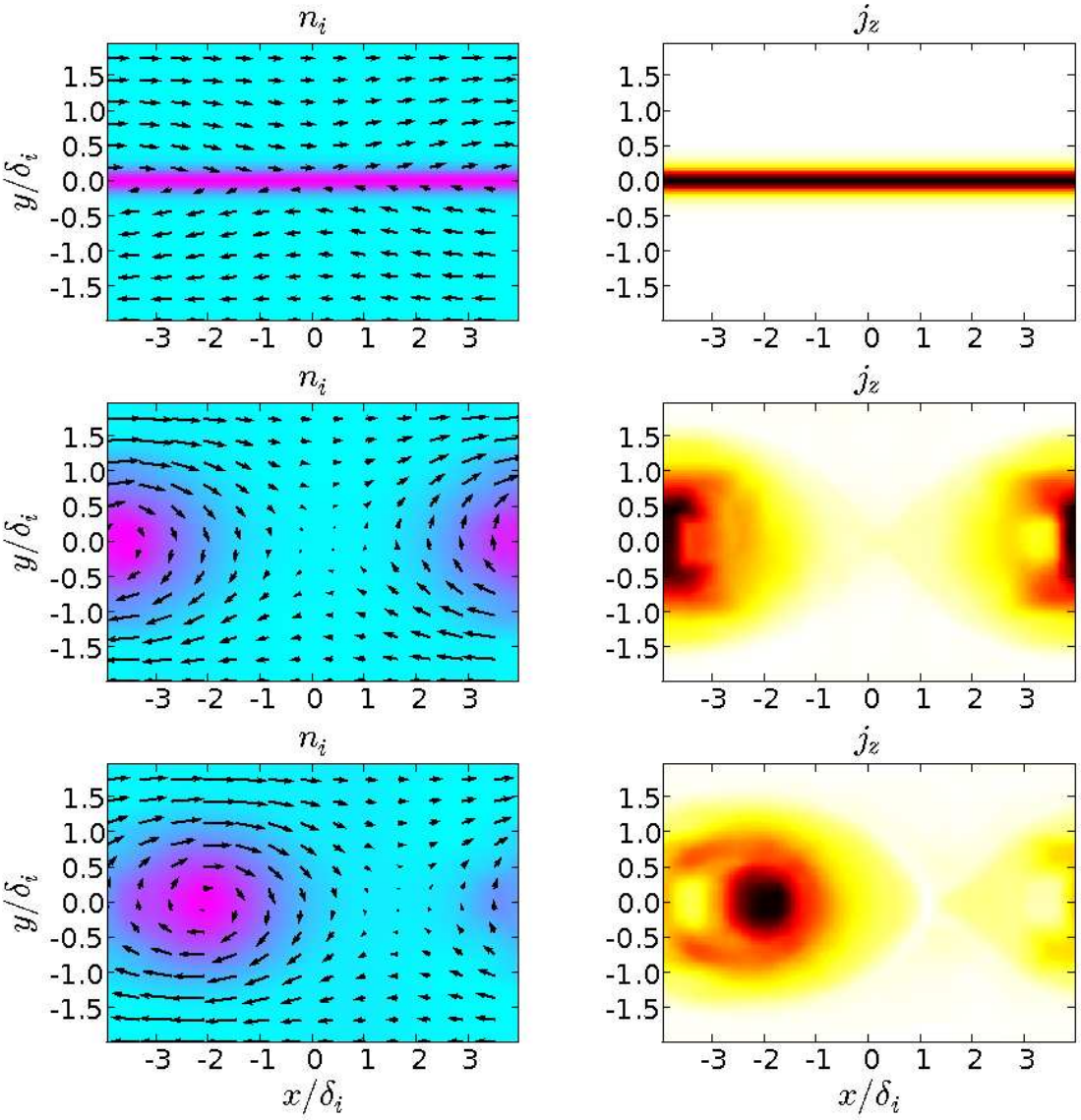

Figure 5.15: Solutions of realistic magnetic reconnection for $\delta_e > \lambda$ at $t = 0, 250/\omega_{ci}, 350/\omega_{ci}$ from top to bottom using a grid resolution of 128×64 with the ideal two-fluid model. Left column is ion density with magnetic field vectors and right column is total out-of-plane current. Note the magnetic island accelerates to the right of the domain.

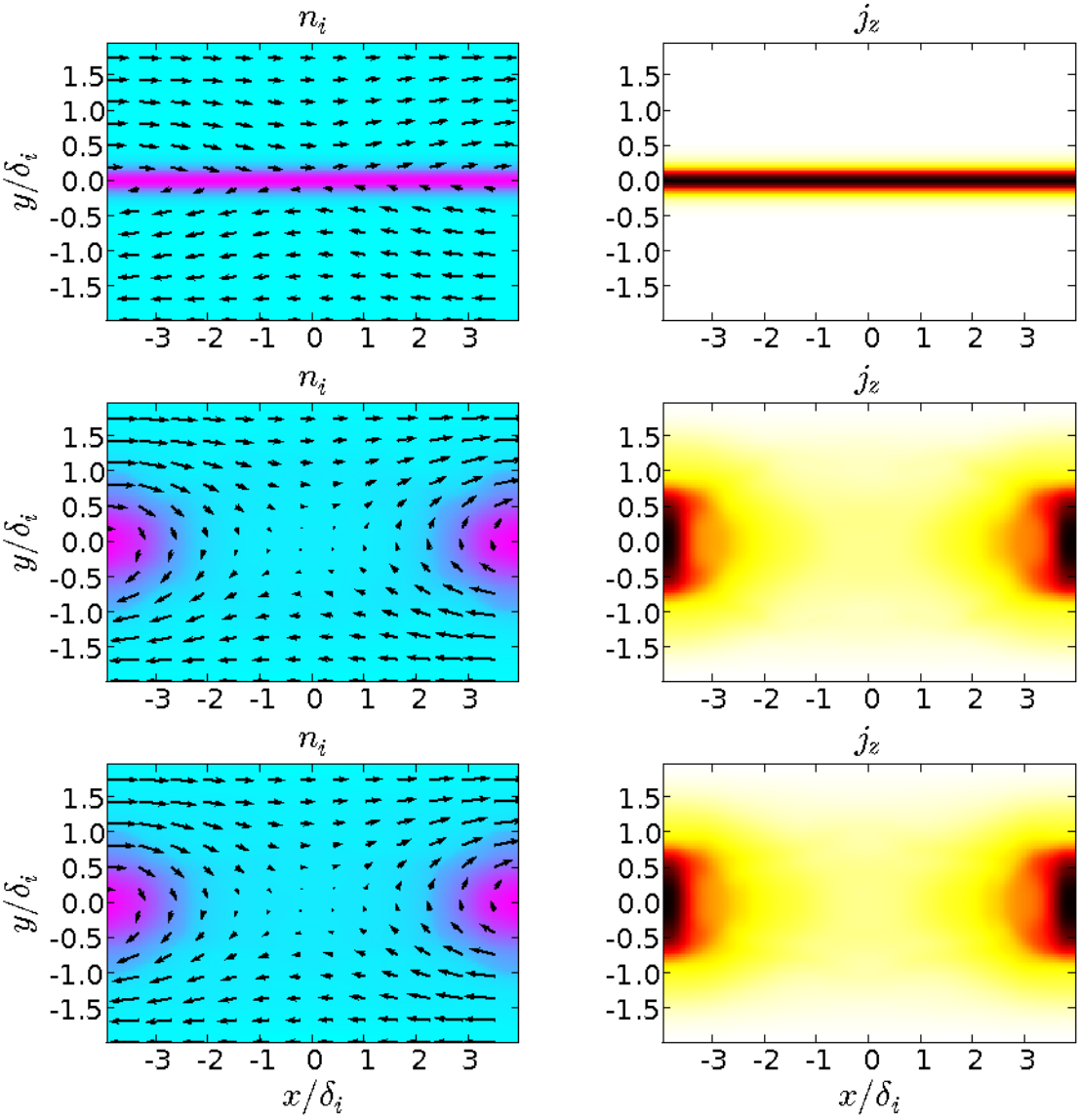

Figure 5.16: Solutions of realistic magnetic reconnection for $\delta_e > \lambda$ at $t = 0, 250/\omega_{ci}, 350/\omega_{ci}$ from top to bottom using a grid resolution of 128×64 with the non-ideal two-fluid model. Left column is ion density with magnetic field vectors and right column is total out-of-plane current. Note the solution reaches steady-state.

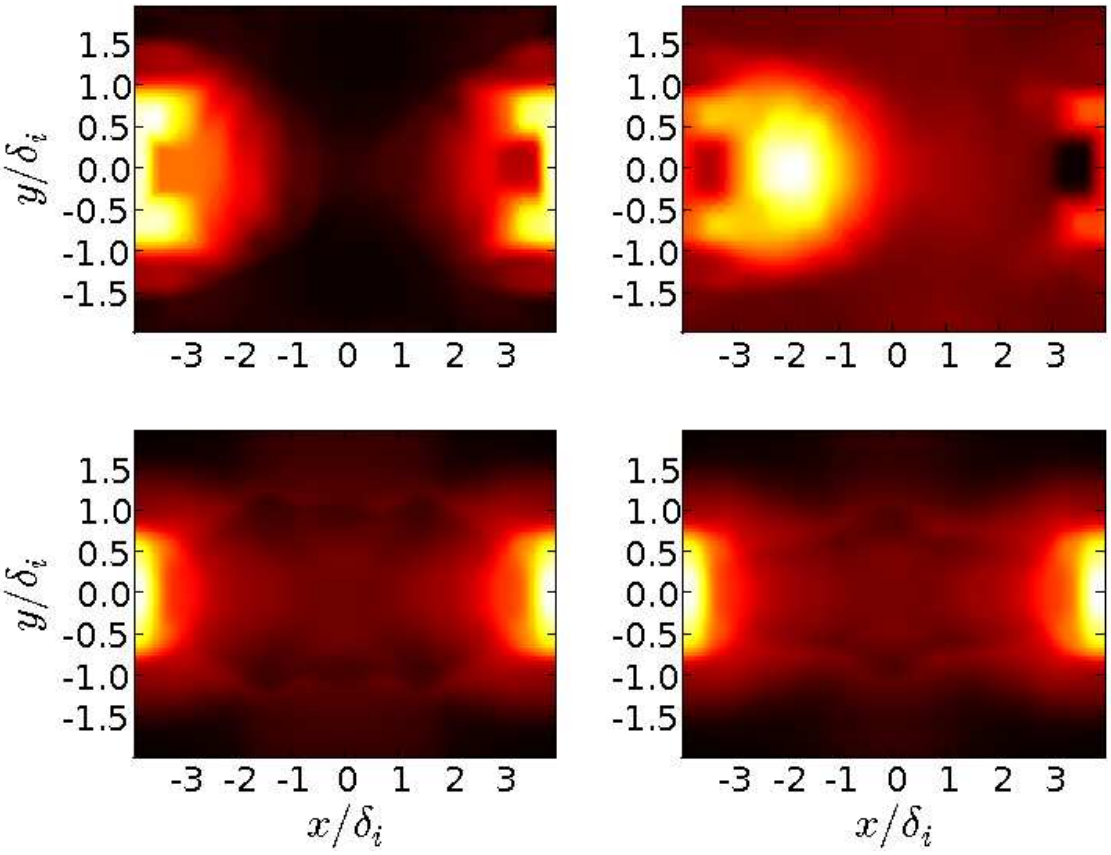

Figure 5.17: Solutions of realistic magnetic reconnection for $\delta_e > \lambda$ from left to right at $t = 250/\omega_{ci}, 350/\omega_{ci}$. Top plot is for the ideal solution of ρ_e using a grid resolution of 128×64. Bottom plot is for a non-ideal solution of ρ_e using a grid resolution of 128×64. Note the acceleration in the ideal solutions while the non-ideal solution reaches a steady-state.

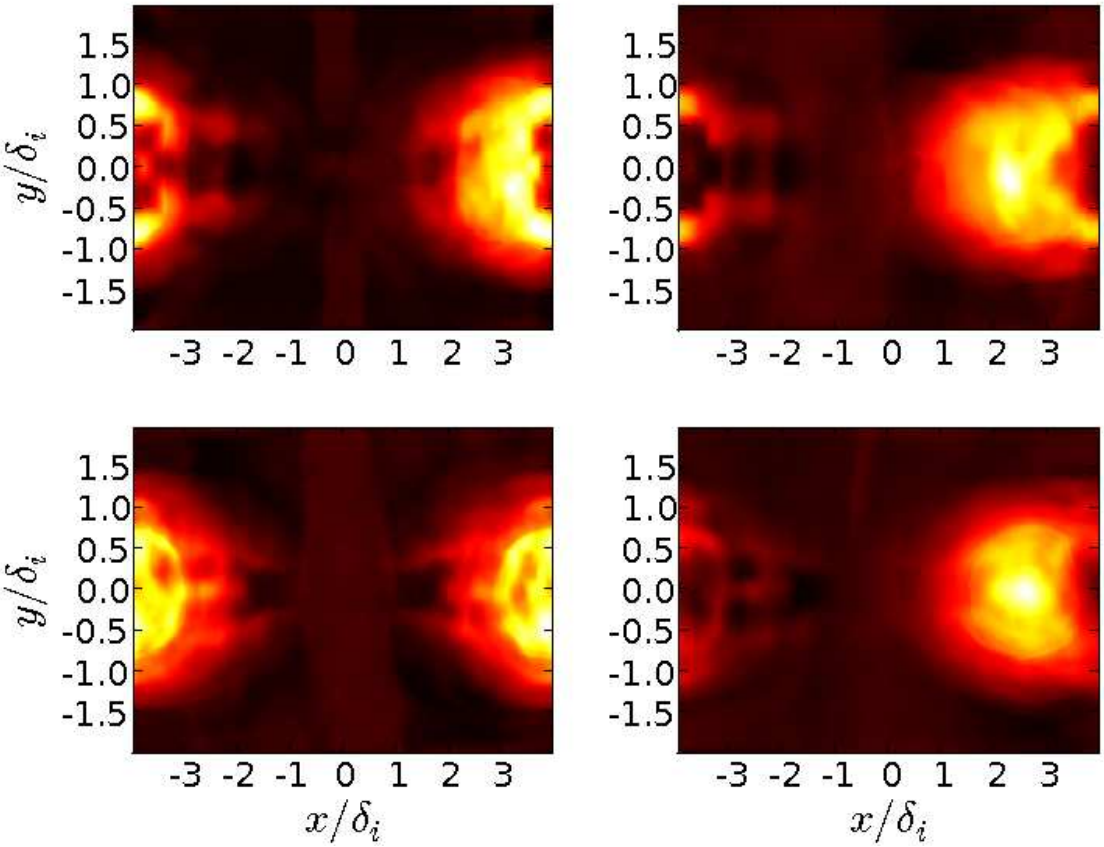

Figure 5.18: Solutions of realistic magnetic reconnection for $\delta_e > \lambda$ from left to right at $t = 250/\omega_{ci}, 350/\omega_{ci}$. Top plot is for the ideal solution of ρ_e using a grid resolution of 256×128. Bottom plot is for an ideal solution of ρ_e using a grid resolution of 512×128. Note the acceleration in the ideal solutions with a different preferred direction compared to the lower resolution solution in Fig. 5.17. Even the high resolution results show that the ideal solution gains acceleration and eventually becomes unstable.

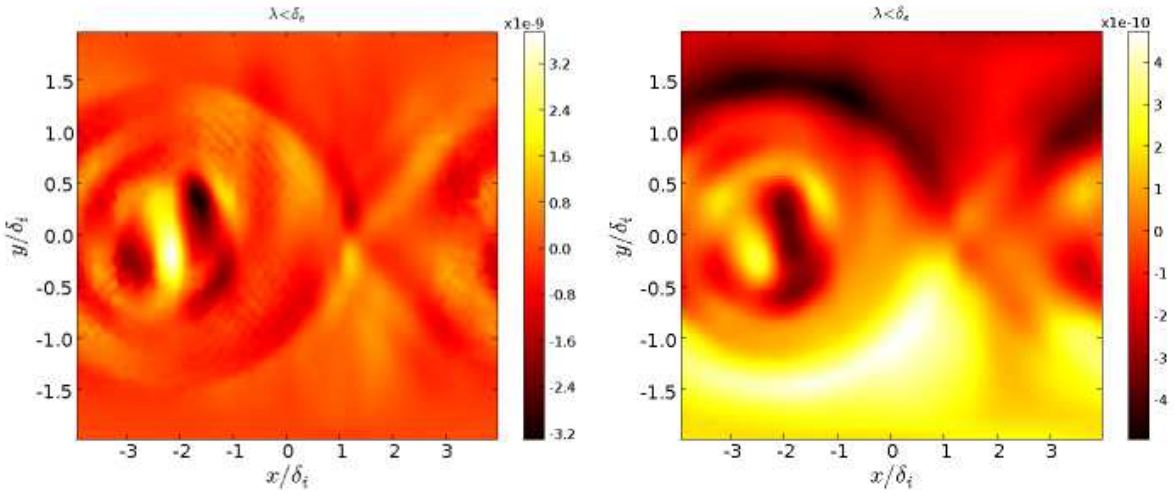

Figure 5.19: Solutions realistic magnetic reconnection for the ideal two-fluid plasma model. Left plot shows solutions of J_y and right plot shows solutions of B_z after $350/\omega_{ci}$.

The collisional case can better distribute the momentum and energy to reduce the sharp gradients that develop in the currents and magnetic fields. In this regime, both the electron and ion fluids have shocks that propagate through the domain. The presence of dissipative transport terms in the form of viscosity, resistivity, and heat flux better distributes such gradients in the domain and maintains the force balance that allows the solution to reach a steady-state. For the collisionless solution, the absence of explicit dissipation makes it unstable as there is no mechanism to dissipate the magnetic energy after the linear stage of the reconnection process. This is an important problem that requires full two-fluid physics to accurately capture the behavior in regimes where $\lambda \sim \delta_e$ because finite electron inertia needs to be present to resolve the physics accurately.

Large gradients are observed in the out-of-plane electric field and the out-of-plane current early in time. Figure 5.21 presents results of the out-of-plane currents, J_z, that shows turbulent behavior in the ideal solution very early in time at $50/\omega_{ci}$. The non-ideal solution does a better job of dissipating momentum and energy such that large gradients do not develop. Going to higher grid resolution will show the turbulence features more clearly and this can be done to study turbulence scales. The very thin current-sheet and spatial

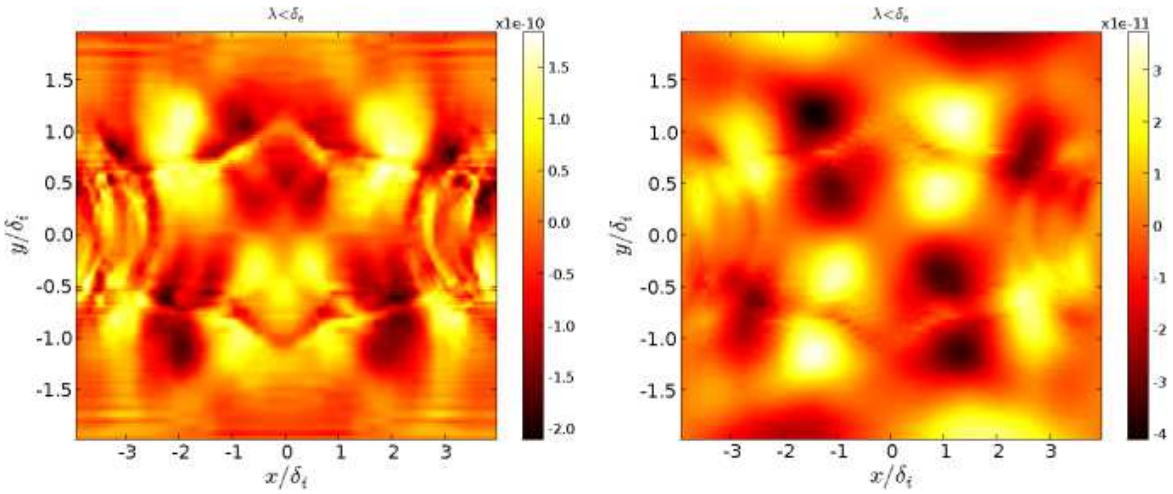

Figure 5.20: Solutions of realistic magnetic reconnection for the non-ideal two-fluid plasma model. Left plot shows solutions of J_y and right plot shows solutions of B_z after $350/\omega_{ci}$.

scales involved are synonymous to Kolmogorov micro-scales in fluid mechanics. Kolmogorov micro-scales are the smallest scale in turbulent flow beyond which there is only viscous dissipation of energy. While the ideal two-fluid model does not have viscosity or any explicit dissipation for that matter, the solution does change as the grid resolution is increased and smaller phenomena are resolved. The changes involve resolution of additional turbulent structures in the fluid. In fluid dynamics, the energy cascade describes the transfer of kinetic energy to smaller and smaller scales until it reaches the Kolmogorov scale[65]. This energy transfer occurs through inviscid processes. Hence, this description can explain the turbulent phenomena that are observed for the ideal two-fluid solution at such small spatial scales that eventually drives the solution unstable. While the Kolmogorov scales may not be the turbulent scales of the ideal two-fluid model due to absence of viscosity, the scales observed may lie somewhere on the energy cascade. In the non-ideal case, viscous dissipation occurs through all time and this damps out the large gradients that may develop and drive the turbulent behavior.

Figure 5.22 shows the out-of-plane electric field, E_z, after $100/\omega_{ci}$. The large gradients in the ideal two-fluid solution are observed in the electric fields and currents early in time

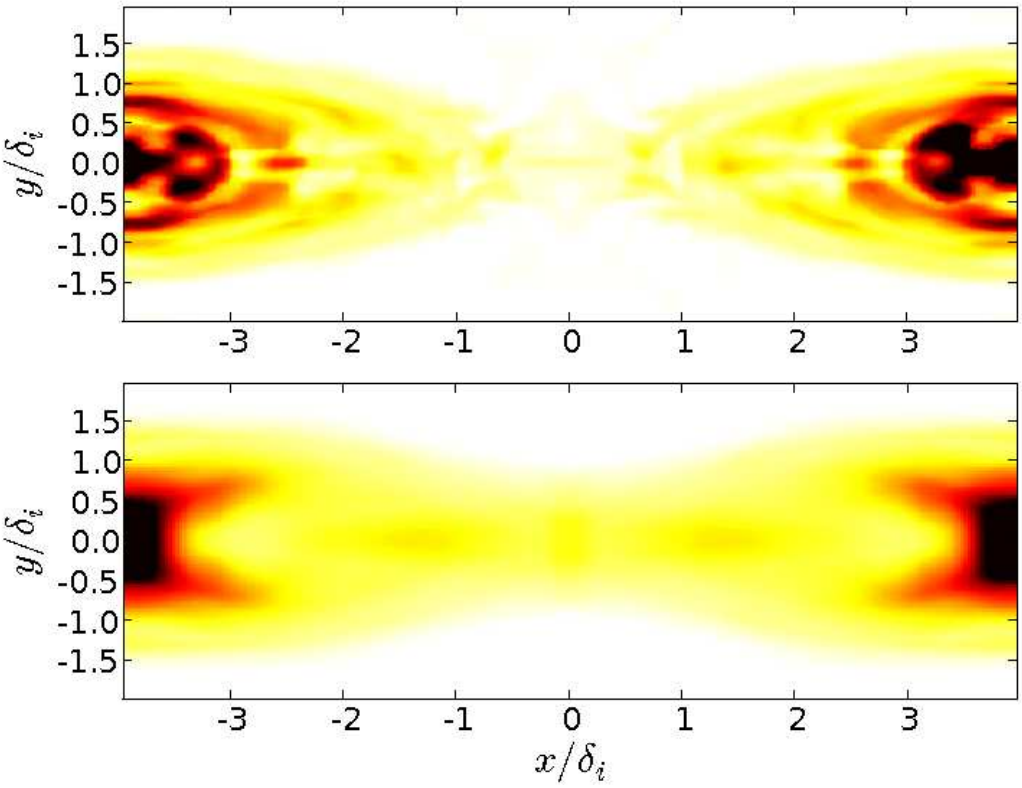

Figure 5.21: Solutions of realistic magnetic reconnection for $\delta_e > \lambda$ after only $50/\omega_{ci}$. Top plot is for the ideal solution of J_z. Bottom plot is for a non-ideal solution of J_z. Note the turbulence structures that develop in the ideal solution.

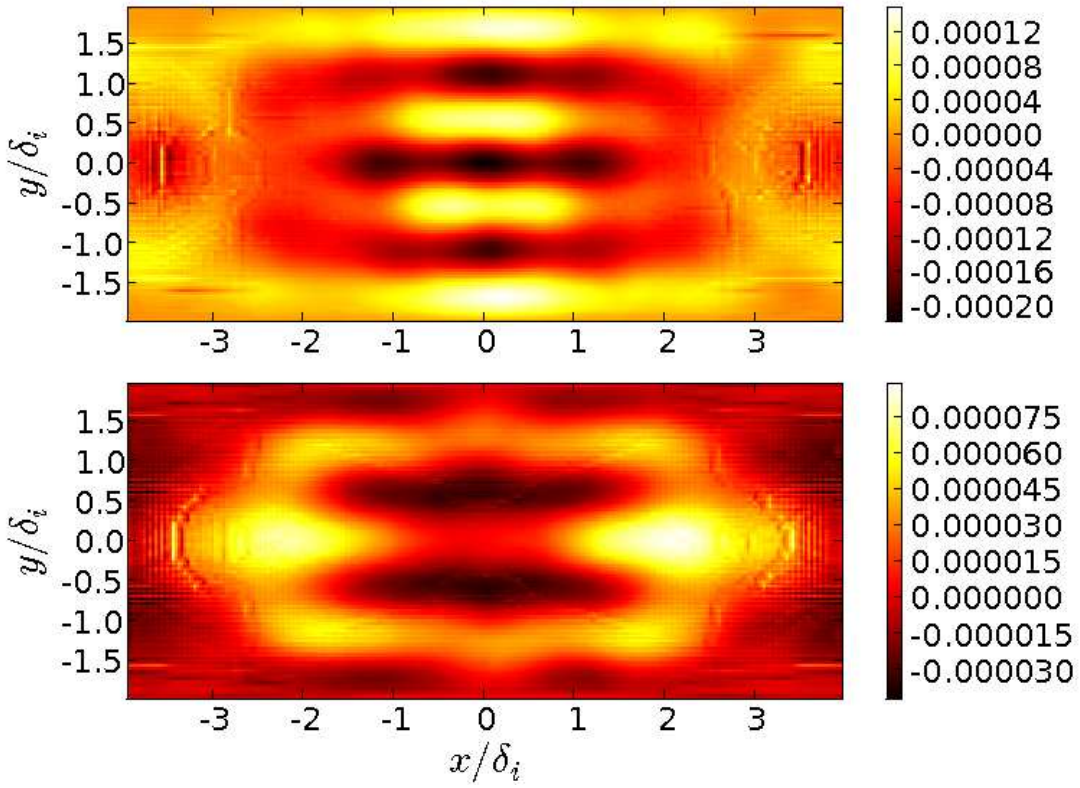

Figure 5.22: Solutions of realistic magnetic reconnection for $\delta_e > \lambda$ after $100/\omega_{ci}$. Top plot is for the ideal solution of E_z. Bottom plot is for a non-ideal solution of E_z. Note the gradients that develop in the ideal solution are an order of magnitude larger than the non-ideal solution.

and these eventually affect other variables and drive the solution unstable. Since there is no explicit dissipation, the gradients are allowed to grow to significantly larger values than they would with dissipation. The non-ideal E_z in Figure 5.22 has magnitudes and gradients that are an order of magnitude smaller than the ideal solution.

5.5 Non-ideal Two-Fluid Axisymmetric Z-pinch

The axisymmetric Z-pinch described in Chapter 4 is explored using Braginskii's transport terms with real experimental parameters based on the ZaP Flow Z-pinch experiment at the University of Washington. These parameters include $n_0 = 10^{22}\mathrm{m}^{-3}$ such that $\rho = n_0 p/p_0$, current $I_0 = 50\mathrm{kA}$ such that $J_0 = I_0/(\pi R_p^2)$, and $R_p = 0.01\mathrm{m}$. This results in a magnetic field of $B_0 = 1\mathrm{T}$ at R_p. The parameter regime uses an artificial mass ratio of $M = 25$, $c/c_{si} \approx 20$, $c/c_{se} \approx 4$, $c/v_A \approx 20$, $v_e/v_{thi} \approx 0.9$, and $R_P/r_{Li} \approx 4$. The collision frequencies are about 5 orders of magnitude smaller than the plasma and cyclotron frequencies initially. The Lundquist number for this regime is initially about 10^4. The initial condition for ion density, ion energy, azimuthal magnetic field, and axial current is presented in Fig. 5.23 and is the same for the ideal and non-ideal solutions of the two-fluid plasma model.

Figure 5.24 shows solutions of the ion density after $2.5\tau_a$ using the ideal and non-ideal two-fluid plasma model. It is seen that the solutions are very similar and the non-ideal solution serves to damp out some of the sharper gradients producing an overall smooth solution while still capturing the small-wavelength two-fluid drift-turbulence instability. The differences are better seen in Fig. 5.25 where the radial electric field is compared for the ideal and non-ideal two-fluid model. There are larger gradients in the ideal two-fluid solution as compared to the non-ideal two-fluid solution. The larger gradients are a consequence of larger charge separation in the ideal model that allows the development of larger local electric fields through the displacement currents in Ampere's law. The non-ideal two-fluid model has dissipative terms in the form of resistivity and viscosity that serve to smooth out the gradients and the local charge separation. Consequently, the local electric fields have smaller gradients. The collisionless case does not have an explicit mechanism to dissipate the electric and magnetic energies that develop in the system.

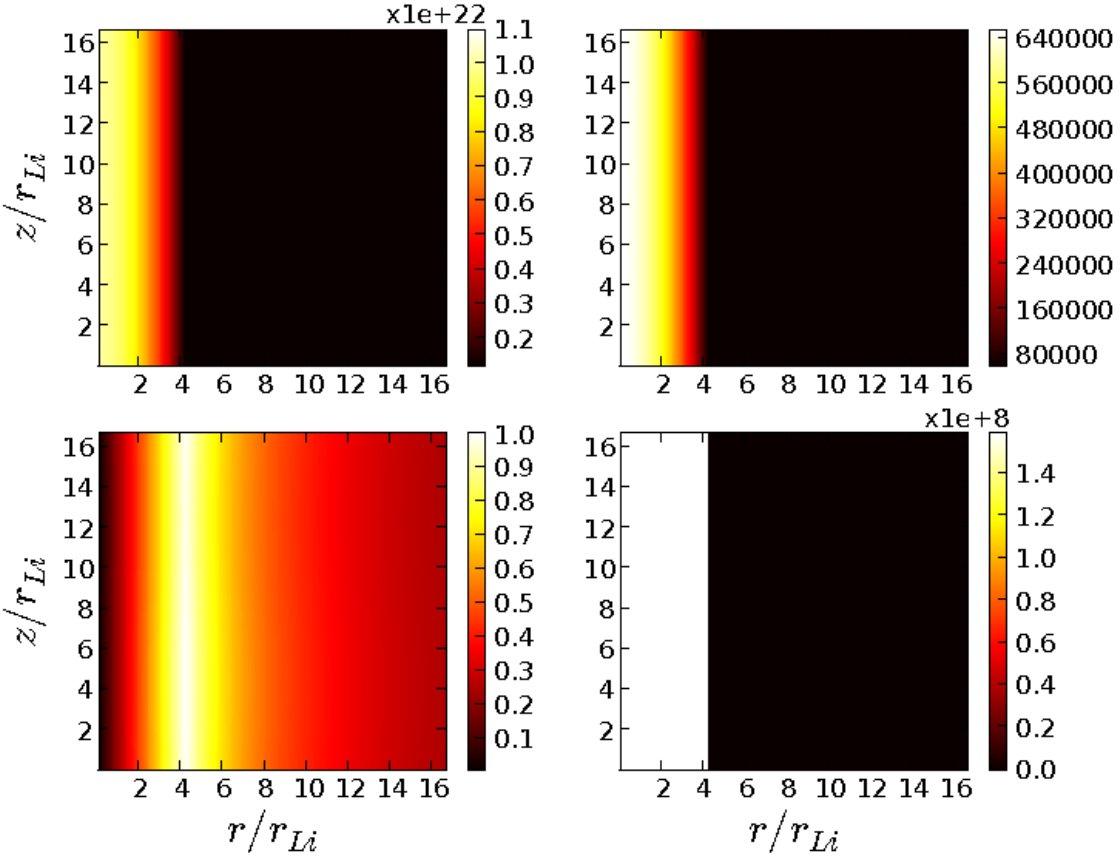

Figure 5.23: Initial condition for realistic Z-pinch simulations to compare the ideal and non-ideal two-fluid models. Top plots are ion density (left) and ion energy (right). Bottom plots are azimuthal magnetic field (left) and axial current (right).

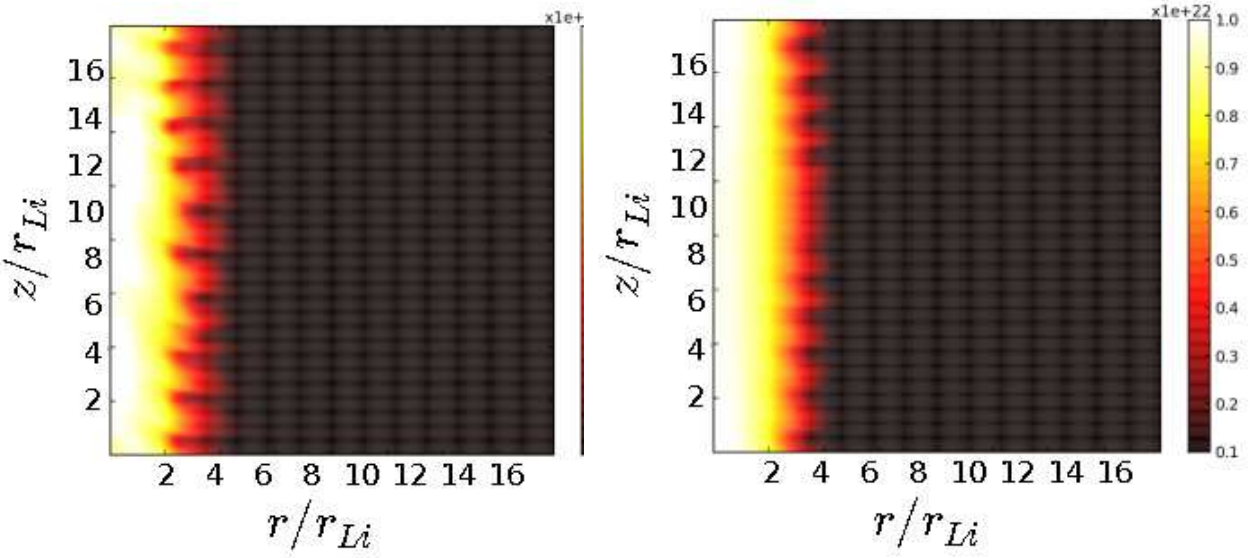

Figure 5.24: Left plot shows solutions of ion density for the axisymmetric Z-pinch after $2.5\tau_a$ using the ideal two-fluid plasma model. Right plot shows solutions of ion density for the axisymmetric Z-pinch after $2.5\tau_a$ using the non-ideal two-fluid plasma model.

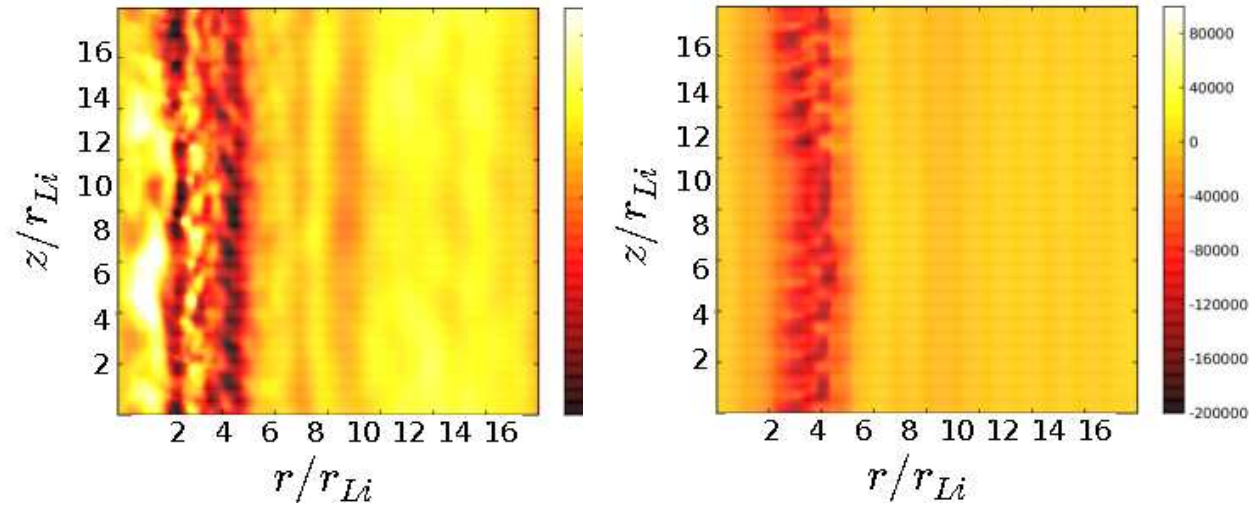

Figure 5.25: Left plot shows solutions of radial electric field for the axisymmetric Z-pinch after $2.5\tau_a$ using the ideal two-fluid plasma model. Right plot shows solutions of radial electric field for the axisymmetric Z-pinch after $2.5\tau_a$ using the ideal two-fluid plasma model. Note larger gradients in the ideal solution.

Chapter 6

A SEMI-IMPLICIT, IDEAL, FULL TWO-FLUID PLASMA MODEL

6.1 Motivation for a Semi-Implicit Method

The characteristic speeds of the two-fluid plasma model include the fluid speeds of sound and the speed of light. Characteristic frequencies include the ion and electron plasma frequencies, and the ion and electron cyclotron frequencies. There are high frequency and low frequency phenomena in plasma dynamics. For the high frequency phenomena, explicit schemes can have a restrictive time-step, which provides the motivation for implicit or semi-implicit schemes. For low frequency phenomena, explicit schemes perform better because implicit schemes can have a large system size with stiff equations. The two-fluid plasma model is suitable for capturing high and low frequency phenomena. This provides the motivation to introduce a semi-implicit two-fluid plasma model where the electron Euler equations and Maxwell's equations are solved implicitly, consequently removing the restrictions of the high frequency phenomena. The ion Euler equations are updated explicitly since physics considerations often require the ion fluid time-scales to be resolved.

Semi-implicit solvers have been implemented for Hall-MHD with special treatment for just the Hall term such that the remaining MHD terms are advanced at the single fluid time-scales. Examples of these are the implicit treatment of the Hall term in Ref.[66] and explicit sub-cycling of just the Hall term using smaller time-steps in Ref.[67] with explicit treatment of the remaining MHD terms at single-fluid time-scales. A semi-implicit Hall-MHD scheme with adaptive mesh refinement is presented in Ref.[68]. A robust, fully implicit Hall-MHD scheme is proposed in Ref.[69] which has the advantage of avoiding the splitting errors that occur in semi-implicit schemes. There are no previous publications describing an implicit or a semi-implicit full two-fluid plasma model.

A 2^{nd} order Runge-Kutta discontinuous Galerkin method[24] is used for the explicit time-advance of the ion Euler equations. An iterative Newton-based, Crank-Nicolson scheme

is used with the discontinuous Galerkin method[42] for the implicit time-advance of the electron Euler and Maxwell's equations. The semi-implicit scheme uses a splitting method that is described as follows.

$$Q_i^{n+1} = f_1(Q_e^n, Q_{EM}^n, Q_i^n) \tag{6.1}$$

is solved explicitly to advance the ion equations, Eq. (2.2-2.4), over a time-step of Δt. Here Q represents the conserved variables, subscripts i, e, and EM represent the ions, electrons, and electromagnetic equations respectively. Following the ion update,

$$(Q_e^{n+1}, Q_{EM}^{n+1}) = f_2(Q_e^n, Q_{EM}^n, Q_i^{n+1}) \tag{6.2}$$

is solved implicitly to advance the electron equations and Maxwell's equations, Eq. (2.2-2.13), over a time-step of Δt. The order of advance for the ions may not be so important because their dynamics are primarily determined by the electromagnetic fields. A 2^{nd} order splitting method[21] provides no benefit for this implementation. To perform a second order splitting, the implicit update for the electron Euler and Maxwell's equations would be done over $\Delta t/2$ following which the ion Euler equations would be advanced over Δt. Then the electron Euler and Maxwell's equations would be updated over another $\Delta t/2$ using the updated ion variables. Second order splitting is better suited for implementations such as a source term splitting where all conserved variables in the equation system are split in time in the same manner[6]. For the semi-implicit two-fluid plasma model, the splitting method described by Eqs.(6.1) and (6.2) provides the same accuracy as a Strang splitting implementation since the ion conserved variables are treated separately from the electron and electromagnetic conserved variables. A true leap-frog scheme might provide better accuracy, but not if the Δt is varied throughout the simulation. For all simulations presented in this dissertation, the Δt is computed from the characteristic speeds and frequencies for every time-step. While an efficient leap-frog scheme can be implemented for simulations requiring a fixed time-step or for equilibrium problems, problems with large dynamics such as moving shocks, rarefaction waves, contact discontinuities, etc. need to account for varying

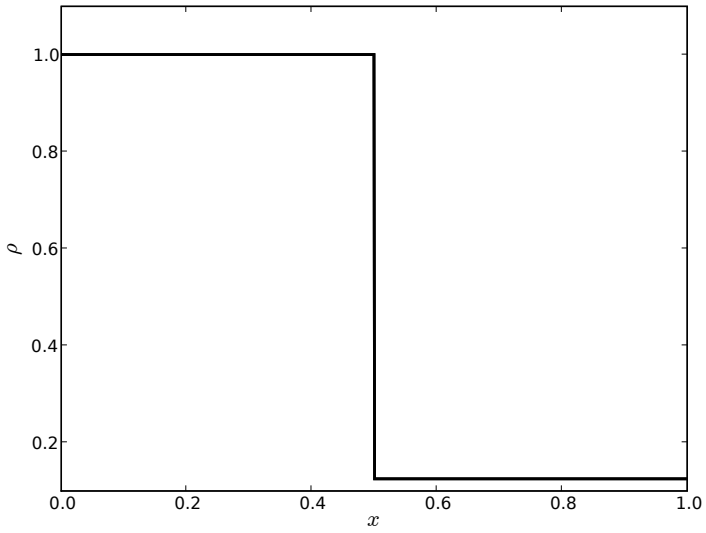

Figure 6.1: Total mass density initial condition for the two-fluid electromagnetic shock using the two-fluid plasma model with a grid resolution of 100 cells.

time-steps. A leap-frog implementation could be problematic if the Δt changes between the explicit and implicit updates. A GMRES solver is used with the ILU preconditioner from the PETSc package[11] as described in Sec. 3.3.1.

6.2 Iterative Solvers and Pre-conditioners for Two-Fluid Plasma model

PETSc's SNES solvers and pre-conditioners are explored for applications of a 1-dimensional electromagnetic plasma shock using the semi-implicit, ideal, full two-fluid plasma model described. A spatial order of 2 is chosen for all simulations presented here. The initial conditions used are the same as Ref.[54] for a normalized ion Larmor radius of 1 where a shock is initialized in electron and ion densities, electron and ion pressures, and the magnetic field. The initial conditions for total density are described by Figure 6.1.

Solving the two-fluid electromagnetic shock problem using 100 cells, the line search Newton method that employs cubic backtracking (SNES_LS) and trust region method (SNES_TR) give the same results using the same amount of computational effort. Table 6.1

Method	CPU time (s)
GMRES	15.24
Conjugate Residual	15.41
Transpose-Free Quasi-Minimal Residual (1)	15.57
Richardson	15.59
Conjugate Gradient Squared	15.60
BiConjugate Gradient STAB	15.75
BiConjugate Gradient	16.21
Transpose-Free Quasi-Minimal Residual (2)	16.21
Conjugate Gradient	19.24
Chebychev	110.54
Least Squares Method	> 1000

Table 6.1: Comparing the computational effort for various PETSc iterative solvers in CPU time (s) for a 1-D two-fluid electromagnetic plasma shock from t=0 to 10 light transit times using a resolution of 100 cells. GMRES is the fastest iterative solver for this problem.

shows the computational effort of each of the PETSc KSP solvers that are used iteratively for non-linear convergence. The CPU time shown here is the time taken to evolve the electromagnetic shock from t=0 to 10 where the shock structure is fully developed. It is seen that the GMRES method is the most computationally efficient for the semi-implicit two-fluid plasma model.

A similar comparison is performed for the various preconditioners offered by PETSc. To test the preconditioners, the non-linear solver chosen is GMRES. Table 6.2 shows the computational effort of each of the preconditioners tested. There are many additional preconditioners that can be added on as external packages such as Hypre[70], Euclid[71], etc. that constitute future work. A brief overview of each of the preconditioners tested is presented to justify the choice of the ILU preconditioners.

- ILU: A Gaussian elimination or Cholesky factorization is often used for matrix $\mathbf{A} = \mathbf{R} \times \mathbf{R}$ where $\mathbf{R}$ is not sparse. Incomplete LU preconditioner uses Cholesky-like formulas so that the Jacobian matrix $\mathbf{M} = \tilde{\mathbf{R}} \times \tilde{\mathbf{R}}$ where $\tilde{\mathbf{R}}$ only has non-zeros where matrix $\mathbf{A}$ has non-zeros. It works best for sparse, non-symmetric matrices like the block-diagonal matrices of the two-fluid plasma model using the discontinuous Galerkin

method.

- Block Jacobi: This is a generalization of the Jacobi preconditioner to block Jacobi form and is appropriate for block-diagonal matrices. Local effects to certain components are considered in this method, while connections to other components are ignored to improve efficiency.

- Successive Over-Relaxation: This is a classical iterative method that often just uses one step, i.e. only one sequential matrix.

$$\mathbf{Q}_i^k = \mathbf{Q}_i^{k-1} + w\frac{r_{ii}}{a_{ii}} \tag{6.3}$$

  Over-relaxation methods are when $w > 1$. This method is best suited for linear systems that occur in partial differential equations.

- Additive Schwarz Method: This uses a domain decomposition algorithm. An approximate solution $\mathbf{x}$ is chosen for system $\mathbf{Ax} = \mathbf{b}$. Then two or more overlapping sub-domains are considered and the approximate solutions in the sub-domains are recomputed until the desired precision is achieved.

- KSP: This uses an approximate linear solver as a preconditioner (e.g. GMRES, Conjugate Gradient, Chebychev, Richardson, BiConjugate Gradient, etc.).

- LU: This uses a direct solver LU factorization for the linear system in order to obtain a preconditioner.

- Jacobi: For a system of $\mathbf{Ax} = \mathbf{b}$, a small sub-matrix of $\mathbf{A}$ is diagonalized, and this is repeated until convergence is achieved. For a non-singular matrix, preconditioner $\mathbf{M}$ =diagonal($\mathbf{A}$).

- Incomplete Cholesky: This is appropriate for spare matrices and uses Cholesky-like formulas. The reason it is not applicable here is because it requires symmetric matrices in addition to requiring $\mathbf{A}$ to be positive definite and Hermitian.

Method	CPU time (s)
ILU	15.24
Block Jacobi	15.34
Successive Over-Relaxation (SOR)	15.35
Additive Schwarz Method	16.18
Linear Solver (KSP)	19.14
LU	20.80
Jacobi	24.44
No preconditioner	40.59
Incomplete Cholesky	> 200
Cholesky	> 200

Table 6.2: Comparing the computational effort for various PETSc preconditioners in CPU time (s) for a 1-D two-fluid electromagnetic plasma shock from t=0 to 10 light transit times using a resolution of 100 cells. GMRES solvers are used for all the preconditioners listed. ILU is the fastest preconditioner for this problem.

- Cholesky: This requires only half of the matrix to be represented explicitly. A Cholesky factorization is computed and separate lower-triangular and upper-triangular systems are solved. This relies on symmetry for fastest convergence as a result of which it is not an efficient preconditioner for the two-fluid plasma model using the discontinuous Galerkin method.

6.3 Application to 1-D Electromagnetic Plasma Shock

Results of a 1-dimensional electromagnetic plasma shock using the semi-implicit two-fluid plasma model are presented and benchmarked to Ref.[54]. In the results presented here, a realistic ion-to-electron mass ratio of 1836 and a realistic normalized speed of light-to-Alfvén speed ratio of $10,000$ are chosen. Often, simulations are performed using artificial mass ratios and artificial speed of light-to-Alfvén speed ratios to reduce computational effort. Such artificial parameters could alter the physics that is observed and this provides the motivation to explore realistic regimes and an implicit implementation.

For the realistic parameter regime, solutions of the fully explicit and semi-implicit two-fluid plasma model are presented in Figure 6.2 after 0.125 Alfvén transit times (t_A). A

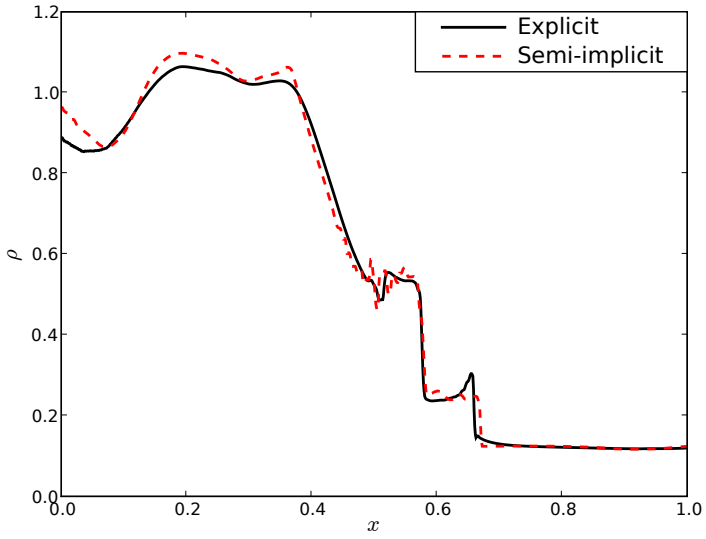

Figure 6.2: Total mass density after 0.125 Alfvén transit times with the explicit and semi-implicit two-fluid plasma model. Both time-integration schemes are able to capture the two-fluid physics described in Ref.[54], but use realistic normalized speed of light for the same grid resolution of 500 cells.

grid resolution of 500 cells is used with 2^{nd} spatial order for explicit and semi-implicit results. The time step of the explicit scheme is 6×10^{-6} s while the implicit time-step is 9.5×10^{-4} s. At a time of $0.125\tau_a$, the shock structure is completely developed. The CPU time to advance from t=$0\tau_a$ to $0.0125\tau_a$ (1/10 of the simulation time) for the implicit scheme (610 s) is approximately 0.6 the CPU time of the explicit scheme (1010 s) since the implicit time-step is over 2 orders of magnitude larger than the explicit time-step. While the fully explicit two-fluid plasma model uses less computational effort than the semi-implicit scheme for artificial parameter regimes, the large computational effort makes the fully explicit two-fluid model impractical for realistic parameter regimes. Implicit time-stepping schemes are better suited to overcome the restrictive time-step of the electromagnetic equations and electron equations for realistic parameters. The semi-implicit algorithm described here solves the full two-fluid plasma model while eliminating the disadvantages of the restrictive time-step. In order to capture the two-fluid physics, accuracy and physics considerations indicate that the simulations be run at ion time-scales as a minimum. Hence, solving the ion equations explicitly when running at ion time-scales provides a more computationally efficient solution as the Jacobian matrix size is significantly reduced as compared to a fully implicit two-fluid plasma model.

6.4 Application to 2-D Magnetic Reconnection

Some scaling studies are performed for the 2-dimensional GEM challenge magnetic reconnection problem using the semi-implicit two-fluid plasma model. Fig. 6.3 shows results of the scaling studies for the 2-D problem using the semi-implicit scheme. The scaling studies here include the advance time to go from $t = 0$ to $t = 1$ and for fairness, do not take into account the matrix coloring times. The work per processor is roughly 256 cells with 13×4 degrees of freedom per cell. This gives a total of $13,312$ variables per processor. In order to have an efficient parallel PETSC implementation, $10,000+$ unknowns per processor are required as a minimum[11]. Ideally, $20,000+$ unknowns per processor provide most efficient use of the algorithm and solvers. The explicit algorithm in WARPX is embarrassingly parallel and scales very well, but as expected, implicit algorithms do not usually scale as well as explicit algorithms. While the parallel semi-implicit implementation scales reasonably

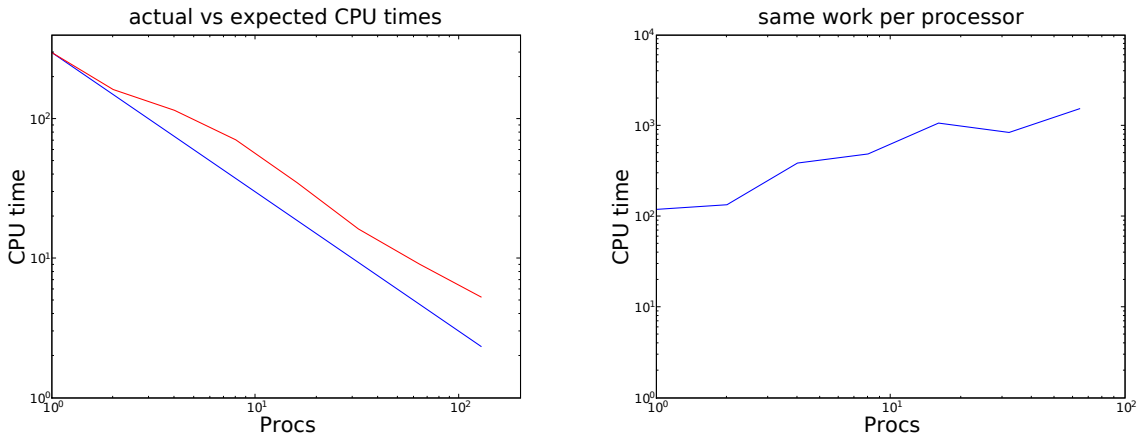

Figure 6.3: Left plot shows the ideal and actual scaling for a grid resolution of 128×64 as the number of processors are increased. Right plot shows the scaling when the problem size per processor is held constant. It is seen that the semi-implicit implementation scales reasonably well after it reaches about 16 processors.

well, the semi-implicit implementation for the artificial parameters of the GEM challenge problem takes 500 times longer than the explicit implementation to produce a comparable solution. However, for realistic parameter regimes where $c/v_A \sim 1000$ and $M = 1836$, the explicit method requires very small time-steps and this is where the semi-implicit implementation is expected to hold a strong advantage. In order to implement and test realistic parameter regimes using the semi-implicit two-fluid model, some future work is required to make the algorithm robust, and to find efficient ways to separate the linear and non-linear parts of the solution such that the implicit part of algorithm does not have to deal with solving a strongly nonlinear problem. The use of realistic parameters makes the equation system stiff and as a result, the Jacobian matrix becomes ill-conditioned.

Solutions are obtained for an ideal two-fluid plasma model without the need for numerical resistivity or viscosity but this semi-implicit implementation can be extended to include additional physics in the form of resistivity, viscosity, and heat conduction. The semi-implicit implementation of the full two-fluid plasma model with the discontinuous Galerkin method has been implemented in 2-dimensions as a part of this dissertation. The results have been benchmarked to explicit solutions for applications of magnetic reconnection GEM

challenge. However, the computational effort is at least an order of magnitude larger when using a semi-implicit scheme over an explicit scheme with realistic parameters. PETSc's SNES solvers may not continue to be as efficient as the solution becomes heavily nonlinear. Also, going from 1-D to 2-D drastically increases the matrix size. In addition to having more grid points and more dimensions to account for, the number of coefficients per cell also increases depending on the spatial order chosen. With such a large system size, the algorithm needs improvement in order to have an efficient semi-implicit scheme for the two-fluid plasma model. This can be improved in several ways and constitutes a major part of future work. Firstly, an approximate analytical Jacobian computation could significantly speed up the process especially when the problem is heavily nonlinear. This could allow fewer iterations prior to convergence. Secondly, more robust preconditioners can be explored to make the semi-implicit two-fluid solvers efficient. Some candidates are physics-based preconditioners[43], p-multigrid methods[44], and similar optimization schemes. An initial step towards exploring preconditioners could include using a lower spatial order discontinuous Galerkin solution as a preconditioning matrix and increasing the order as necessary.

The electromagnetic shock is not an easy problem to simulate using an implicit time-stepping scheme. In the presence of shocks, the solution is nonlinear and the application of limiters often makes the solution so strongly nonlinear that it is unable to converge. This results because a derivative of the solution is required to compute the Jacobian for the Newton iteration. In the presence of shocks in the solution, the Jacobian becomes very large and oscillatory in the location of the shocks and increases the condition number of the matrix often making it ill-conditioned. The discontinuous Galerkin method requires continuity in the fluxes although the conserved variables are allowed to be discontinuous. Application of limiters in shocks or regions of sharp gradients often represents a discontinuity in the flux leading to an undefined or at least a strongly nonlinear Jacobian. The heavily nonlinear nature of the system is one of the main reasons why a fully implicit two-fluid plasma model is rather difficult to solve in this manner. Even the semi-implicit two-fluid model is not as effective as may be expected for these reasons. The algorithm can be improved with knowledge of the problem and the physics to make it more linear. One potential method is the solution of the ion fluid using an explicit discontinuous Galerkin

method as described here while the electron fluid and electromagnetic fields can be solved using an implicit continuous finite element or finite volume method. This could potentially work for problems that have strong shocks in the ion fluid but not necessarily in the electron fluid and electromagnetic fields.

Chapter 7

COMPARISONS OF THE TWO-FLUID PLASMA MODEL WITH HALL-MHD

The full two-fluid plasma model and Hall-MHD described in Chapter 2 are compared for applications of the 1-dimensional electromagnetic plasma shock, the 2-dimensional Geospace Environment Modeling magnetic reconnection (GEM challenge), the 2-dimensional axisymmetric Z-pinch instabilities and the 2-dimensional Hill's vortex FRC. No explicit resistivity or viscosity is included in either of these fluid models. Some of these comparisons have been done in Ref.[72]. For all problems presented here unless otherwise stated, the electron energy equation is included in the Hall-MHD model[67] and the full discontinuous Galerkin polynomial expansion is used for all conserved and auxiliary variables with component-based limiting. The discontinuous Galerkin implementation for Hall-MHD's auxiliary variables, although slightly more accurate than the central differencing implementation, is more numerically challenging. This is because of the high-order coefficients that could lead to oscillations in the cell interfaces leading to negative density/pressure errors. Floor values are used in the 2-D simulations to maintain a numerically stable Hall-MHD simulation. Any time the value of the density or pressure falls below this minimum floor value, it is reset to the floor and this reduces the negative density/pressure errors. The two-fluid model does not require floor values for most problems, however, the DG implementation of Hall-MHD does. To provide a physical context, $c \sim 10^3 - 10^4 v_A$ and $v_W \sim 10 v_A$ in astrophysical plasmas, where v_W is the whistler wave speed. In a number of Hall-MHD simulations, the whistler wave speed is unphysically larger than the Alfvén speed due to the nature of the Hall-MHD dispersion relation where the whistler wave grows quadratically without bound as the wave number is increased.

7.1 1D Electromagnetic Plasma Shock

An electromagnetic plasma shock is initialized with a gradient in electron and ion densities, pressures and the z-direction magnetic field for the two-fluid plasma model. The initial conditions used are the same as described in Chapter 5. For Hall-MHD, the ion fluid, electron pressure and magnetic fields are initialized the same way as the two-fluid case. The simulations are performed using a Runge-Kutta discontinuous Galerkin method with 2^{nd} order in space and time using 256 cells. The MHD limit for this problem is at $r_{Li} = 0$.

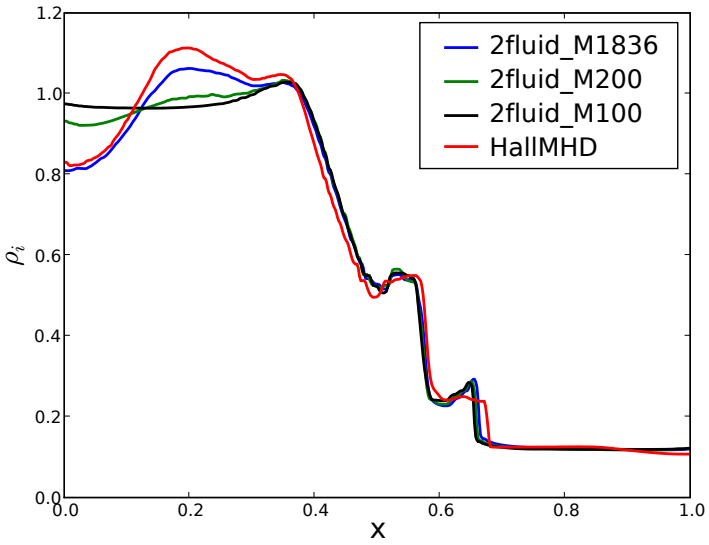

Figure 7.1: The two-fluid plasma model is compared to Hall-MHD after 0.2 Alfvén transit times for $r_{Li} = 7 \times 10^{-1}$. The two-fluid plasma model solution with several ion-to-electron mass ratios agrees well with the Hall-MHD solution.

Figures 7.1 and 7.2 show the two-fluid model and Hall-MHD solutions for different ion Larmor radii and different ion-to-electron mass ratios, M, of the two-fluid model. At an ion Larmor radius of 7×10^{-1} in Fig. 7.1, the two-fluid model using a realistic ion-to-electron mass ratio of $M = 1836$ agrees well with the Hall-MHD model that assumes massless electrons. Even at $M = 100$ the solutions agree well in regions of the rarefaction, contact discontinuity and the shock with differences arising in the whistler wave propagation on

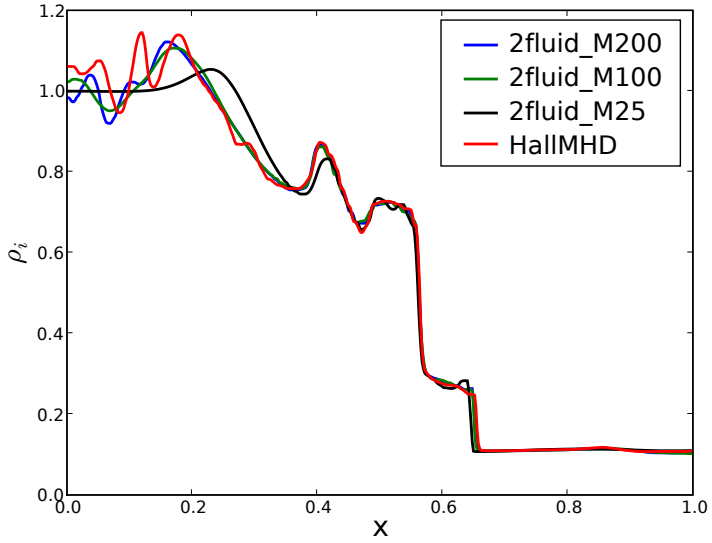

Figure 7.2: The two-fluid plasma model is compared to Hall-MHD after 0.2 Alfvén transit times for $r_{Li} = 7 \times 10^{-2}$. The Hall-MHD solution is similar to the two-fluid plasma model solution with ion-to-electron mass ratio, $mi/me = 183.6$. With realistic ion-to-electron mass ratio at this Larmor radius, the problem becomes stiff for the two-fluid model.

the left of the domain. At an ion Larmor radius of 7×10^{-2} in Fig. 7.2, the two-fluid model becomes stiff if the realistic ion-to-electron mass ratio is used because the electron plasma frequency needs to be resolved and can be more restrictive than the speed of light for setting the time step. However, artificially decreasing the ion-to-electron mass ratio of the two-fluid model still provides a comparable solution to Hall-MHD for $r_{Li} = 7 \times 10^{-2}$. Further decreasing the ion Larmor radius such that $r_{Li} = 7 \times 10^{-4}$ provides two-fluid and Hall-MHD solutions that approach the ideal-MHD solution consistent with theory. This is seen from Fig. 7.3. In this regime the two-fluid model is very stiff and Hall-MHD takes less computational effort.

The time step for the two-fluid model is restricted by the speed of light which is assumed to be $100v_A$ for this problem. Since the speed of light is assumed infinite in Hall-MHD, the whistler wave speed restricts the time step using the cutoff wave number described previously. Both the two-fluid model and Hall-MHD solutions have the same effective grid

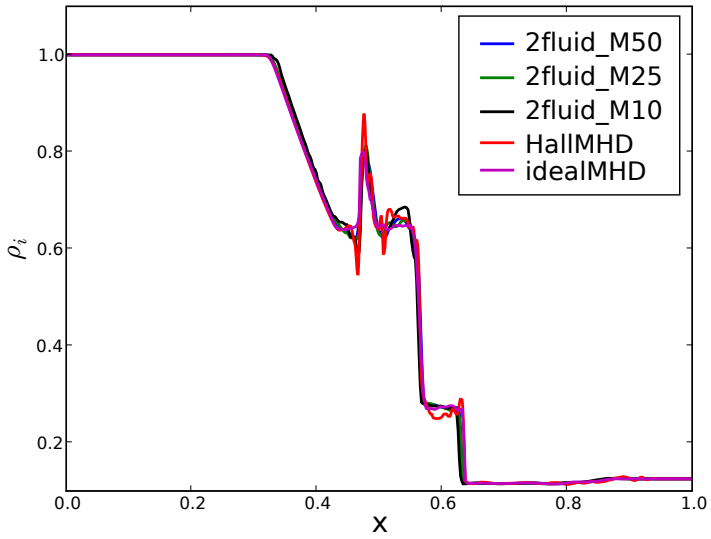

Figure 7.3: The Hall-MHD model is compared to ideal-MHD after 0.2 Alfvén transit times for $r_{Li} = 7 \times 10^{-4}$. At this Larmor radius the two-fluid model becomes too stiff and requires large computational effort to get a solution similar to that of Hall-MHD. The Hall-MHD solution here is compared to the ideal-MHD solution. The solutions have similar qualities but the characteristic speeds seem to vary. It is seen that decreasing the Larmor radius approaches the ideal-MHD limit.

resolution. The time step used for the two-fluid model with real ion-to-electron mass ratio for the $r_{Li} = 7 \times 10^{-1}$ case is approximately 100 times larger than the time step used for Hall-MHD leading to 135 times more computational effort to obtain a Hall-MHD solution as compared to the two-fluid model. For the $r_{Li} = 7 \times 10^{-2}$ case, Hall-MHD takes 14 times more computational effort than the two-fluid model to produce comparable results. Hall-MHD is extremely computationally intensive even though it provides results comparable to the two-fluid model when r_{Li} becomes significant. Hall-MHD is also able to provide results comparable to ideal-MHD when r_{Li} becomes very small with less computational effort than the two-fluid model which becomes stiff in this regime. However, in regimes where small spatial and temporal scales are of interest, the two-fluid model uses much less computational effort for this problem compared to Hall-MHD and provides comparable solutions even after artificially decreasing the ion-to-electron mass ratio and the speed of light.

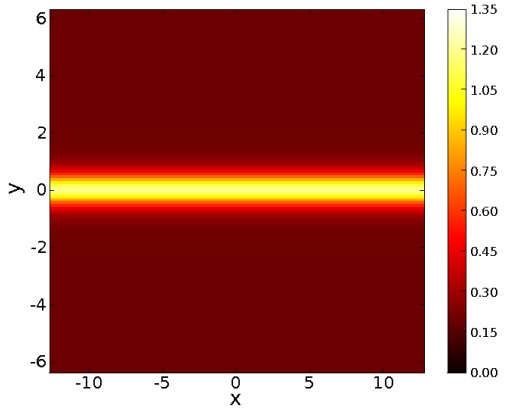

Figure 7.4: Initial condition of ion density for the magnetic reconnection GEM challenge problem.

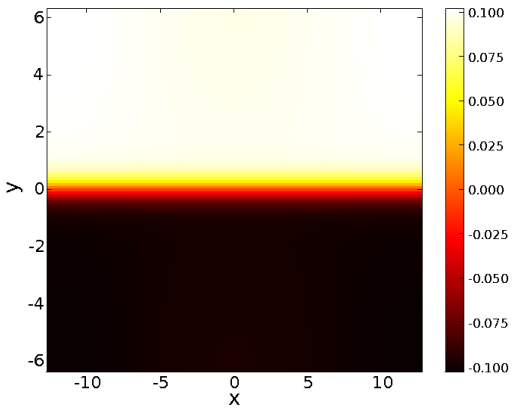

Figure 7.5: Initial condition of x-direction magnetic field for the magnetic reconnection GEM challenge problem.

7.2 2D GEM Challenge Collisionless Magnetic Reconnection

Magnetic reconnection has been explored using a number of fluid and particle codes due to its role in magnetosphere dynamics, space plasmas and laboratory plasmas. Shay et al.[58] determine that the inclusion of the Hall term is necessary to produce physically correct reconnection rates and Hall-MHD remains the minimum physical model needed to accurately capture magnetic reconnection.

A current sheet is initialized with a density profile of $\text{sech}^2(y)$ and an x-direction in-plane magnetic field profile of $\tanh(y)$. A small initial perturbation is applied to the in-plane magnetic fields in the x- and y-directions. The electron momentum in the z-direction is initialized for the two-fluid model such that it follows the density profile. The initial conditions are the same as described in Ref.[55] and are shown for the ion density and x-direction magnetic field in Fig 7.4 and 7.5. For Hall-MHD, the electron currents are calculated from the Hall current and the ion fluid velocity. The Hall current is obtained from the curl of the magnetic fields. All results are for a resolution of 128×64 cells using the 2^{nd} order RKDG method with periodic boundary conditions in the x-direction and conducting wall boundaries in the y-direction.

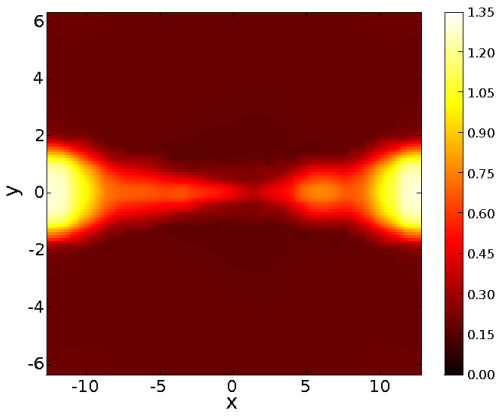

Figure 7.6: Solution of ion density for the two-fluid model is presented for the magnetic reconnection problem at $\omega_{ci}t = 20$. An island forms in the center of the domain that moves to the right and merges with the plasma. This solution does not use any $\nabla \cdot \mathbf{B}$ correction.

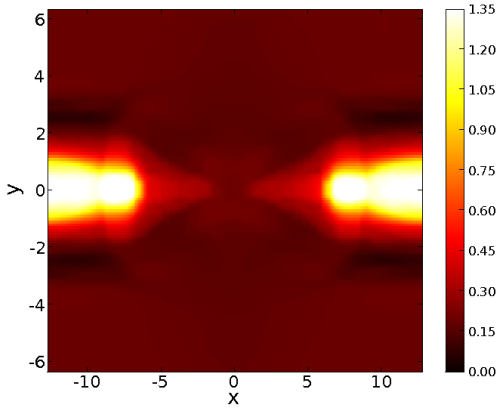

Figure 7.7: Solution of ion density for Hall-MHD is presented for the magnetic reconnection problem at $\omega_{ci}t = 20$. There is no noticeable island formation at the center of the domain but an island of small magnitude appears at high resolution. This solution does not use any $\nabla \cdot \mathbf{B}$ correction.

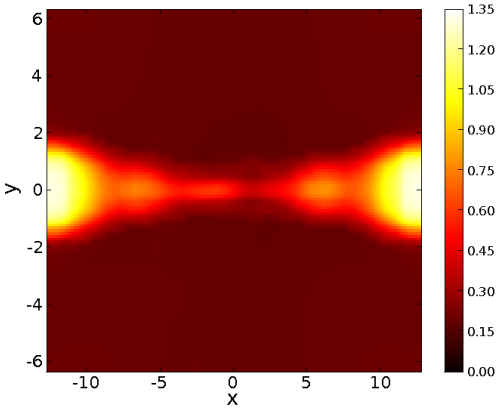

Figure 7.8: Solution of ion density for the two-fluid model is presented for the magnetic reconnection problem at $\omega_{ci}t = 20$ using $\nabla \cdot \mathbf{B}$ correction. Two islands form in the center of the domain that move in either direction and merge with the plasma.

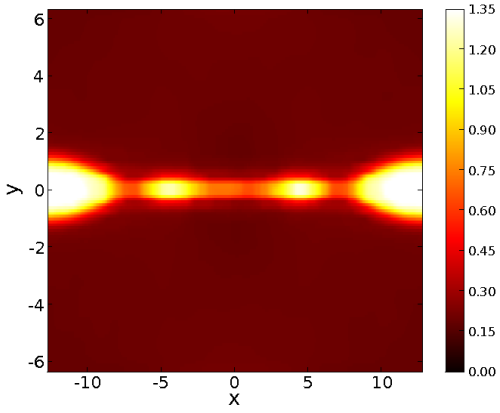

Figure 7.9: Solution of ion density for Hall-MHD is presented for the magnetic reconnection problem at $\omega_{ci}t = 20$ using $\nabla \cdot \mathbf{B}$ correction. There are noticeable islands that form at the center of the domain that move and merge with the plasma.

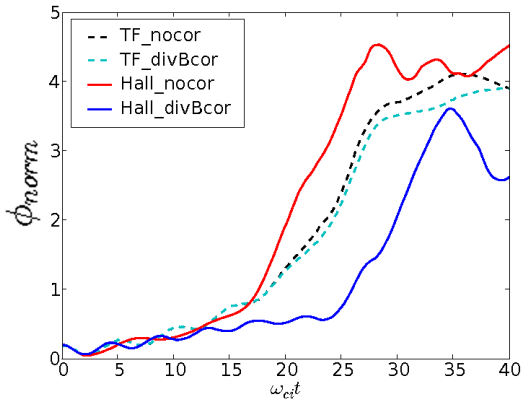

Figure 7.10: Reconnected magnetic flux is shown as a function of time, $\omega_{ci}t$, for the two-fluid model (dashed lines) and Hall-MHD (solid lines) with and without divergence corrections. The solid red line is the Hall-MHD solution without $\nabla \cdot \mathbf{B}$ correction, and the solid blue line is the Hall-MHD solution with $\nabla \cdot \mathbf{B}$ correction. The dashed black line is the full two-fluid solution without $\nabla \cdot \mathbf{B}$ correction and the dashed cyan line is the full two-fluid solution with $\nabla \cdot \mathbf{B}$ correction. The reconnection rates match well with previous literature[58]. Hall-MHD reconnects more flux than the two-fluid model before saturating when divergence corrections are not employed.

For the two-fluid model, $M = 25$ and $c \approx 10\, v_A$. Figures 7.6 and 7.7 show the ion density for reconnection at a time of $\omega_{ci}t = 20$ for the two-fluid model and Hall-MHD without divergence corrections. Figures 7.8 and 7.9 show the reconnection at a time of $\omega_{ci}t = 20$ for the two-fluid model and Hall-MHD with $\nabla \cdot \mathbf{B}$ corrections. Divergence corrections more accurately capture the islands that form and merge with the rest of the plasma, and provide better agreement between the full two-fluid and Hall-MHD models. These islands eventually move to either side of the domain and begin to merge with the plasma at a time of about $\omega_{ci}t = 33$.

The reconnected magnetic flux for the two-fluid model and Hall-MHD are shown in Fig. 7.10. The magnetic flux reconnection rates from this plot agree well with previously published results[58]. The Hall-MHD solution reconnects more flux than the two-fluid model before it saturates without $\nabla \cdot \mathbf{B}$ corrections. The reconnected flux and the reconnection rate of the two-fluid model does not change significantly with and without divergence corrections for this problem. However, the impact for Hall-MHD is more significant. The reconnection

begins later in time ($\omega_{ci}t = 25$) when $\nabla \cdot \mathbf{B}$ corrections are included for Hall-MHD as compared to $\omega_{ci}t = 15$ without Hall-MHD $\nabla \cdot \mathbf{B}$ corrections. Furthermore, with the inclusion of $\nabla \cdot \mathbf{B}$ corrections less flux is reconnected and the reconnection rate agrees better with the two-fluid plasma model since the errors in the magnetic field are reduced.

A study of divergence corrections is performed for the Hall-MHD and two-fluid plasma models to quantify the amount of error present and the correction applied. The values of the error correction coefficients were chosen such that the error correction speeds were the same as the speed of light. This ensured that the time-step was not affected in advecting the divergence errors out of the domain. Figures 7.11 and 7.12 quantify the $\nabla \cdot \mathbf{B}$ and $\nabla \cdot \mathbf{E}$ errors for the full two-fluid and Hall-MHD models for this problem. It is seen that the $\nabla \cdot \mathbf{B}$ errors are significantly reduced for both fluid models when using perfectly hyperbolic Maxwell's equations divergence corrections and this has a greater impact on the reconnection rates for Hall-MHD. The perfectly hyperbolic Maxwell's equations provide sufficient corrections for $\nabla \cdot \mathbf{B}$ errors. Also, for the two-fluid plasma model, higher grid resolutions without divergence corrections do not seem to significantly increase or decrease the $\nabla \cdot \mathbf{B}$ error as is seen from Fig. 7.11 but using the divergence corrections significantly reduces this error even at lower resolutions.

The $\nabla \cdot \mathbf{E}$ errors are not an issue for Hall-MHD due to the charge neutrality assumption. For the two-fluid plasma model, however, the DG method provides much lower $\nabla \cdot \mathbf{E}$ errors than the wave propagation method and increasing the grid resolution significantly reduces this error as is seen from Fig. 7.12. The perfectly hyperbolic Maxwell's equations do not yet work as effectively to correct for the $\nabla \cdot \mathbf{E}$ errors of the two-fluid plasma model as compared to the $\nabla \cdot \mathbf{B}$ errors for this problem.

For this problem, the very restrictive time step of Hall-MHD makes it require 15 times the computational effort of the two-fluid model for the same grid resolution and the same initial conditions. Since the reconnection rates of both models match previously published results, the two-fluid model provides the more computationally effective solution. No explicit resistivity or dissipation is added to these models. Therefore, reconnection could either be triggered by the inherent diffusion in the grid or from the use of flux limiters. However, it is also possible that the presence of the Hall term, diamagnetic term and electron inertia

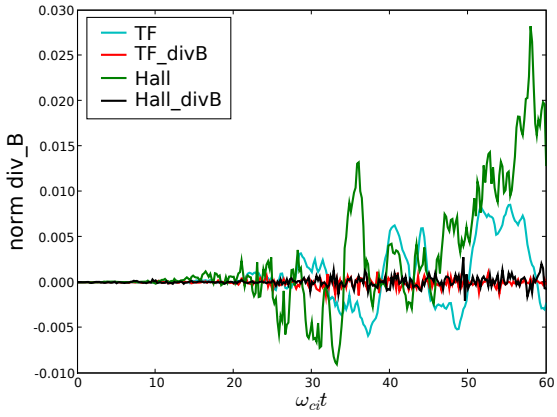

Figure 7.11: Quantifying the $\nabla \cdot \mathbf{B}$ errors for the full two-fluid plasma model and for Hall-MHD with and without $\nabla \cdot \mathbf{B}$ corrections. It is seen that the errors are significantly reduced for both fluid models when using the perfectly hyperbolic Maxwell's equations for divergence corrections.

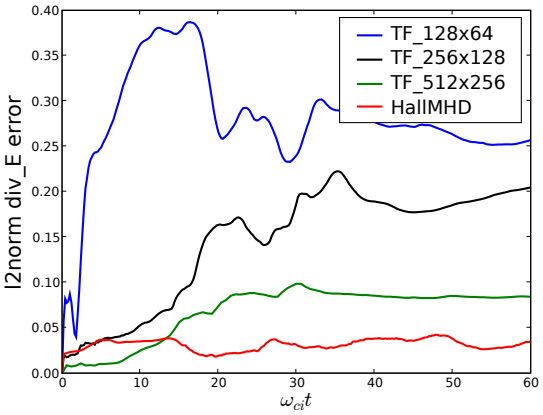

Figure 7.12: Quantifying the $\nabla \cdot \mathbf{E}$ errors for the full two-fluid plasma model and for Hall-MHD for different grid resolutions of the two-fluid plasma model without using any divergence corrections. Going to higher resolution significantly reduces $\nabla \cdot \mathbf{E}$ errors.

play a role to break the frozen-in flux condition.

7.3 2D Axisymmetric Z-pinch

The axisymmetric Z-pinch small-wavelength drift-turbulence instability presented here uses the same initial and boundary conditions that were simulated in Sec. 4.4.4 but with a single wavelength perturbation for all fluid models. Hall-MHD solutions are compared to the two-fluid solutions using a discontinuous Galerkin method with 2^{nd} order in space and time on a grid of 128×128 cells. For both models, the ratio of the pinch radius to the ion Larmor radius, r_p/r_{Li}, is approximately 3. The two-fluid model uses an ion-to-electron mass ratio of 25 and the speed of light is $c \approx 16 v_A$.

Figure 7.13 shows the formation of the short-wavelength drift-turbulence instability in the two-fluid plasma model and in Hall-MHD. The Hall-MHD solution has a much greater growth rate at this grid resolution compared to the two-fluid plasma model. At 1.5 Alfvén transit times, the solution for Hall-MHD is already in the non-linear regime as seen from

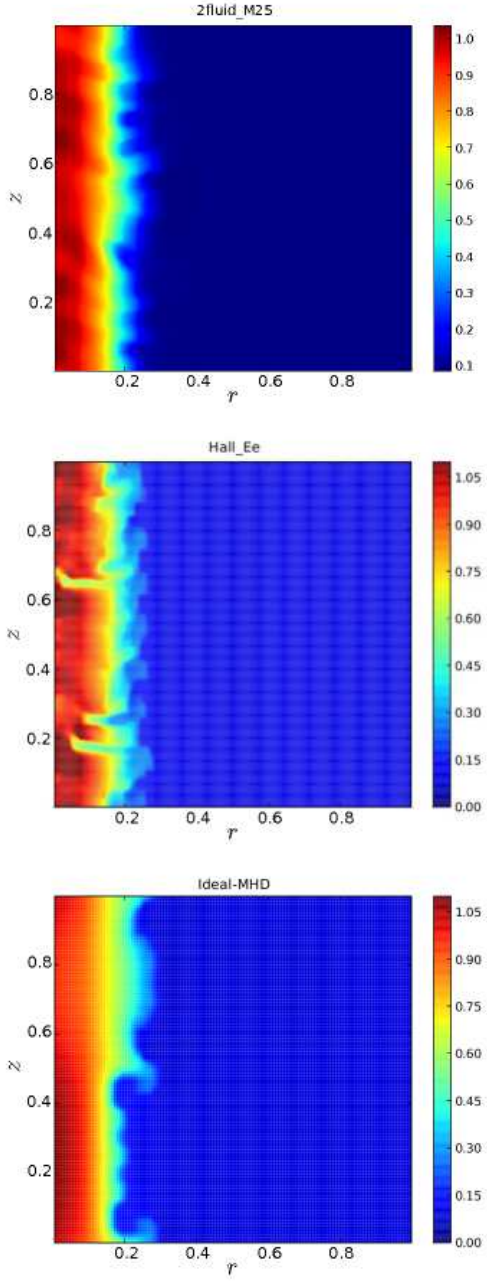

Figure 7.13: Ion density after 1.5 Alfvén transit times for the two-fluid axisymmetric Z-pinch (top plot), Hall-MHD solution (middle plot), and ideal-MHD solution (bottom plot). It is seen that the drift-turbulence instability grows for the two-fluid and Hall-MHD solutions with the small-scale instabilities that form. The ideal-MHD solution just forms a sausage instability.

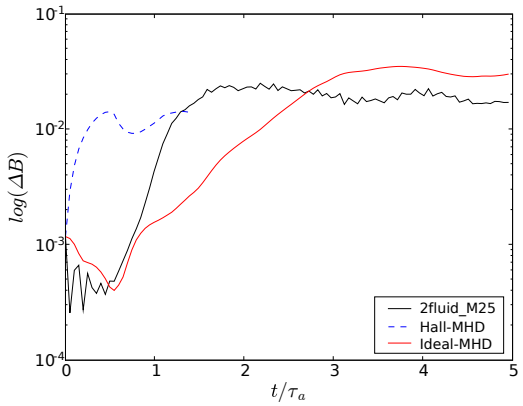

Figure 7.14: Growth rates for the perturbation in the magnetic field as a function of the time, t/τ_a, where τ_a is the Alfvén transit time for the two-fluid model, Hall-MHD and ideal-MHD. It is observed that the growth rates of Hall-MHD and two-fluid plasma model are larger than the ideal-MHD growth rate.

Fig. 7.14. Ideal-MHD does not capture this drift-turbulence instability and forms a sausage instability based on the initial perturbation. The growth rate of the drift-turbulence instability seen in the two-fluid model and Hall-MHD are larger than the growth rates predicted by ideal-MHD. The instability growth rate for the Hall-MHD model is higher than that for the two-fluid model and sets in earlier in time. This could be because no explicit dissipation or resistivity is present in the Hall-MHD model implemented here. As a result of this, increasing the grid resolution resolves waves of smaller and smaller wavelength due to the need to resolve the whistler wave. Adding a resistivity or a hyper-resistivity to the Hall-MHD equations might provide sufficient dissipation to allow the instability to set in at approximately the same time scales as the two-fluid model. The addition of a resistivity or a hyper-resistivity will be investigated as future work towards the subsequent dissertation.

Hall-MHD requires 35 times the computational effort of the two-fluid model for this problem. One difficulty in using the Hall-MHD model is the need for floor values in density and pressure without which the simulation results in negative density/pressure errors early in time. For this problem, the two-fluid model does not require such floor values. This could potentially be overcome by introducing dissipation in the form of resistivity or hyper-

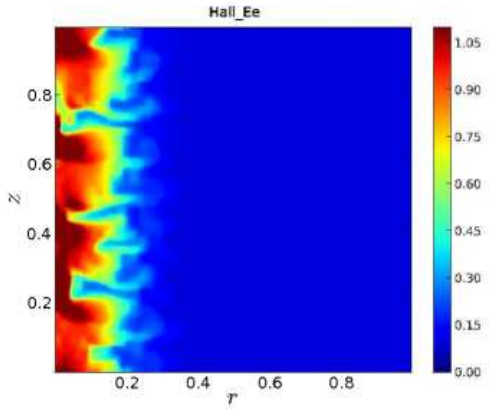

Figure 7.15: Ion density after 1.5 Alfvén transit times for Hall-MHD solution using hyper-resistivity $\nu = 6 \times 10^{-5}$. It is seen that the drift-turbulence instability looks closer to the full two-fluid solution with respect to the approximate number of modes.

resistivity in the Hall-MHD model.

7.3.1 Using Hyper-Resistivity

Simulations of the 2-D Z-pinch are performed using hyper-resistivity for Hall-MHD as described in Sec.2.3.3 and 3.2.9. The hyper-resistivity coefficient, ν, is chosen based on the scales of the desired instability wavelength. What this means is that when $\nu = 0$, the Hall-MHD model resolves waves of smaller and smaller wavelengths as the grid resolution is increased since the grid scales determined the resolution of the modes. Hyper-resistivity is applied using $\nu = 1/k^2$ where the wavenumber k is specified by the user and is maintained for all grid resolutions. The Z-pinch solution for ion density using $\nu = 6 \times 10^{-5}$ is shown in Fig. 7.15.

The instability growth rates are studied for several values of the hyper-resistivity coefficient. When using only a single wavelength perturbation for the Z-pinch, the small-scale instability still evolves and grows on top of the single wavelength perturbation. The growth rate of this instability is shown in Fig. 7.16 for several values of ν. In order to trigger the fastest growing mode in the simulation, an 8 wavelength perturbation is applied and the resulting growth rate of the instability is shown in Fig. 7.17 for several values of ν.

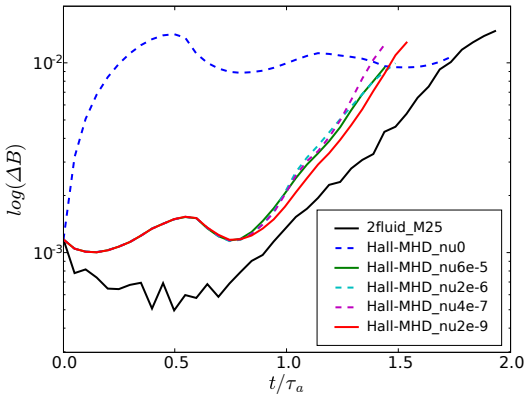

Figure 7.16: Growth rates for the perturbation in the magnetic field as a function of the time, t/τ_a, where τ_a is the Alfvén transit time for the two-fluid model when using a single wavelength perturbation. Notice that using ν for Hall-MHD approaches the two-fluid growth rate and convergence is seen for the Hall-MHD model for several values of ν.

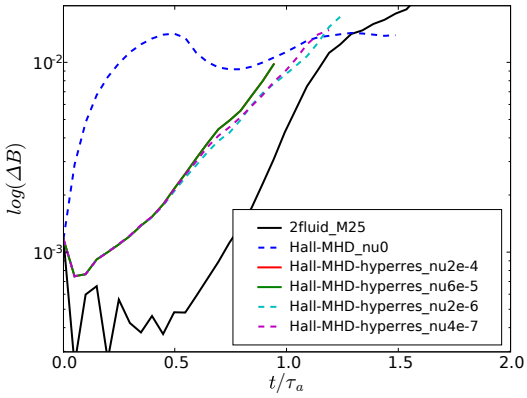

Figure 7.17: Growth rates for the perturbation in the magnetic field as a function of the time, t/τ_a, where τ_a is the Alfvén transit time for the two-fluid model when using an 8 wavelength perturbation. Using several values of ν for Hall-MHD shows convergence but this growth rate does not quite capture the fastest growing mode that is captured by the full two-fluid growth rate.

7.4 2D Hill's Vortex Field Reversed Configuration

A 2-dimensional Hill's vortex FRC is simulated using an approximate MHD initial condition shown in the top plot of Fig. 7.18 which uses Eqs.(7.1-7.9). A Hill's vortex profile is used to initialize the FRC following which it is allowed to relax to a two-fluid FRC equilibrium. The pseudocolor represents the ion mass density and the streamlines represent the magnetic fields. While the density, pressure, currents, and magnetic fields inside the separatrix are initially specified using the Hill's vortex FRC profile, the external field is approximated differently. There is a background density and pressure applied outside the separatrix with no current. In the two-fluid plasma model, the current would be initialized by specifying an ion and/or electron momentum. In the FRC initial conditions applied here, the ion and electron momenta are 0 and are allowed to evolve from the magnetic fields that appear as sources in the momentum equation. The external magnetic field is in the axial direction with a magnitude chosen such that the profile is in radial force balance at $z = 0$. The initial external magnetic field is zero in the region at $r < a$ and outside of the separatrix, a being the separatrix radius. Thus, there is a discontinuity both at the separatrix and along the line defined by $r = a$.

It is non-trivial to compute an external magnetic field profile that is in a true two-fluid equilibrium and does not have flux penetrating the boundaries. This would have to be done numerically by solving the Grad-Shafranov equation to describe a flux conserver MHD equilibrium and allow a two-fluid equilibrium to develop. If flux were to penetrate into the boundary, divergence of $\mathbf{B}$ errors could occur since conducting walls are assumed at the $r = 0.06$ boundary. If the $\nabla \cdot \mathbf{B}$ constraint is not satisfied initially, then it may produce large errors over the course of the simulation. Hence a simple radial equilibrium (at $z = 0$) profile is chosen such that only a z-direction magnetic field is specified initially outside the separatrix between the separatrix radius and the flux conserver boundary. This maintains $\nabla \cdot \mathbf{B}$ everywhere and allows an external radial field to develop as the solution evolves into a two-fluid equilibrium. There is a discontinuity initially in the external magnetic field since no external field is specified when $r < a$ where a is the separatrix radius, in this case 0.04. The algorithm is able to handle this discontinuity without any difficulty. Since there

is no axial equilibrium, large dynamics are expected to develop. The solution is expected to expand axially with some plasma leaving the separatrix as the external magnetic field lines evolve to have a radial component to confine the plasma both radially and axially. The first plot in Fig. 7.18 shows the initial condition where $b = 0.1$ is the separatrix length measured axially in each direction from $z = 0$. The two-fluid and Hall-MHD solutions use the same initial conditions for ρ_i, P_i, P_e, B_r, and B_z. J_ϕ is self-consistently computed from the reduced Ampere's law. The initial condition is described by

$$r_s = a\sqrt{1.0 - \frac{z^2}{b^2}} \quad \text{for} - b < z < b \tag{7.1}$$

$$x_s = \frac{r_s}{r_c} \tag{7.2}$$

$$B_e = \frac{B_0}{\left(1 - \frac{a^2}{r_c^2}\right)} \tag{7.3}$$

$$P_i = P_e = 0.5\frac{B_0^2}{2\mu_0}\left(\frac{r^2}{a^2}\right)\left(4 + \frac{a^2}{b^2}\right)\left(1 - \frac{r^2}{a^2} - \frac{z^2}{b^2}\right) \tag{7.4}$$

$$\rho_i = m_i\frac{P_i}{kT_i} \tag{7.5}$$

$$\rho_e = m_e\frac{P_e}{kT_e} \tag{7.6}$$

$$J_\phi = \begin{cases} -\frac{B_0}{\mu_0} r\left(\frac{4}{a^2} + \frac{1}{b^2}\right) & \text{if } \frac{r^2}{a^2} + \frac{z^2}{b^2} < 1 \\ 0 & \text{otherwise} \end{cases} \tag{7.7}$$

$$B_r = \begin{cases} -\frac{B_0 rz}{b^2} & \text{if } \frac{r^2}{a^2} + \frac{z^2}{b^2} < 1 \\ 0 & \text{otherwise} \end{cases} \tag{7.8}$$

$$B_z = \begin{cases} -B_0\left(1 - 2\frac{r^2}{a^2} - \frac{z^2}{b^2}\right) & \text{if } \frac{r^2}{a^2} + \frac{z^2}{b^2} < 1 \\ B_e & \text{if } r > a \\ 0 & \text{otherwise} \end{cases} \tag{7.9}$$

where $r_c = 0.06$ is the flux conserver radius and $B_0 = 0.5$.

The initial peak number density is chosen to be approximately $3 \times 10^{21}/\text{m}^3$, with $T_i = T_e = 100$ eV, deuterium ion mass, and the kinetic parameter, $s \approx 4$. For the two-fluid plasma model, $c \approx 22v_A$ and ion-to-electron mass ratios of 25, 50 and 100 are explored. Realistic mass ratio makes the simulation fail with a negative density/pressure error as it becomes

very stiff. For $M = 100$, the results do not significantly differ from the $M = 25$ case so it is sufficient to assume $M = 25$. For large mass ratios, the electron plasma frequency can be more restrictive than the light speed for setting a time step. For Hall-MHD, $v_W \approx 125 v_A$. The boundary conditions include an axis boundary at $r = 0$, a conducting wall at $r = 0.06$, and periodic boundaries in the axial direction.

Figures 7.18 and 7.19 show the magnetic field profile and the ion density using an approximate MHD equilibrium to initialize the two-fluid and Hall-MHD models which is initially only in radial equilibrium at the mid-plane and settles to a new two-fluid equilibrium. The solutions of the two-fluid model and Hall-MHD after $0.5\tau_a$ show that both models undergo an expansion in the axial direction as expected. As the external magnetic field lines evolve towards an equilibrium profile, the separatrix expands axially until an axial force-balance is achieved. After $2.5\tau_a$, and $5\tau_a$ the two-fluid and Hall-MHD profiles appear to reach an equilibrium. An asymptotic study is performed for the two-fluid plasma model using $M = 50$ and $M = 100$ to ensure that the two-fluid FRC equilibrium that is reached is not affected by the ion-to-electron mass ratio. These results are presented in Fig. 7.20. It is noted that changing the mass ratio does not significantly affect the FRC equilibrium solution of the two-fluid plasma model.

Figure 7.21 shows the conserved flux in the FRC between the null point and the axis for the two-fluid model with $M = 25$ and the Hall-MHD model. This closed flux integral is approximated by the summation

$$\phi = \sum_r B_z(r, z = 0) r dr \tag{7.10}$$

where the flux is computed at the midplane in the axial direction. The conserved flux from each of these models appears to be consistent and remains relatively constant over time. There is no explicit dissipation added to either of the fluid models. The total flux at each time is normalized using the total initial flux. The normalized flux for the two-fluid model deviates by about 2.5% over $5\tau_a$. As seen in Figs. 7.18 and 7.19, the FRC experiences motions in the radial direction and undergoes some axial asymmetries. Considering all these distortions of the FRC profile over $5\tau_a$, a deviation of only 2.5% for the closed flux

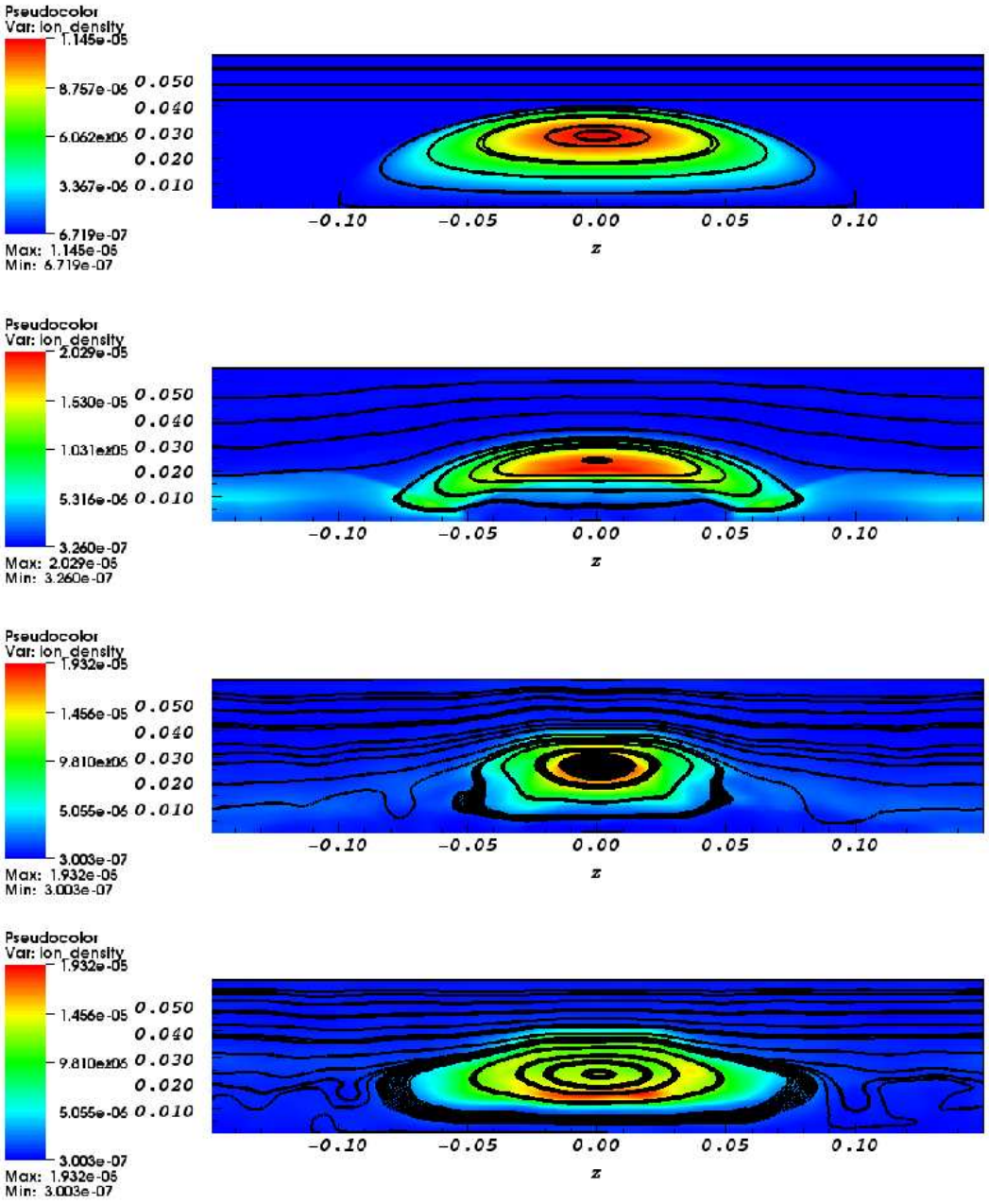

Figure 7.18: FRC evolution of ion mass density and magnetic field streamlines for the two-fluid model. Top plot is the initial condition which is in radial equilibrium with no radial magnetic field specified outside the separatrix. Second plot is the ion density and magnetic field after $0.5\tau_a$. Third plot is the ion density and magnetic field after $2.5\tau_a$. Last plot is the ion density and magnetic field after $5\tau_a$.

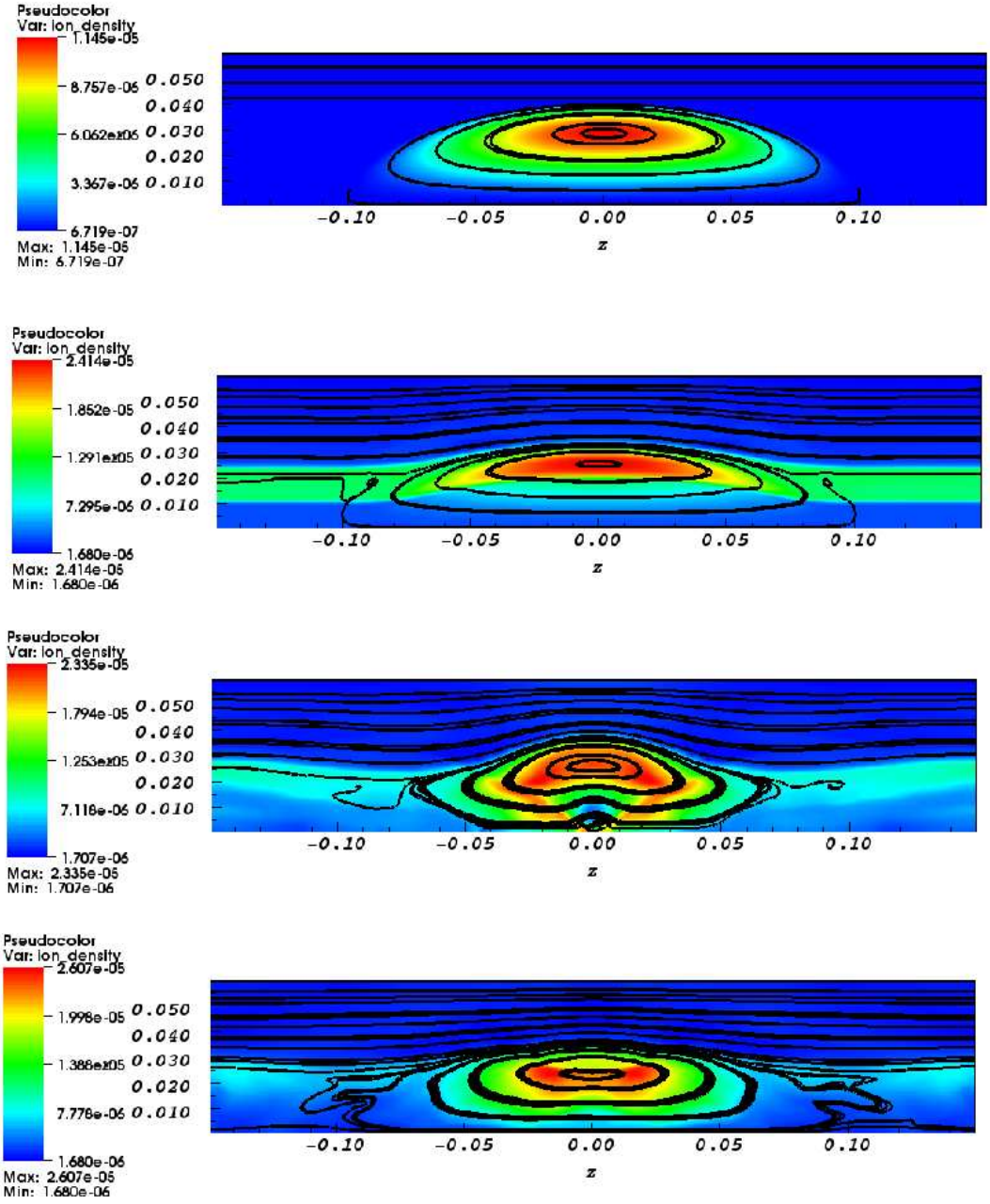

Figure 7.19: FRC evolution for Hall-MHD. Top plot is the initial condition for ion mass density and magnetic field streamlines which is in radial equilibrium with no radial magnetic field specified outside the separatrix. Second plot is the ion density and magnetic field after $0.5\tau_a$. Third plot is the ion density and magnetic field after $2.5\tau_a$. Last plot is the ion density and magnetic field after $5\tau_a$.

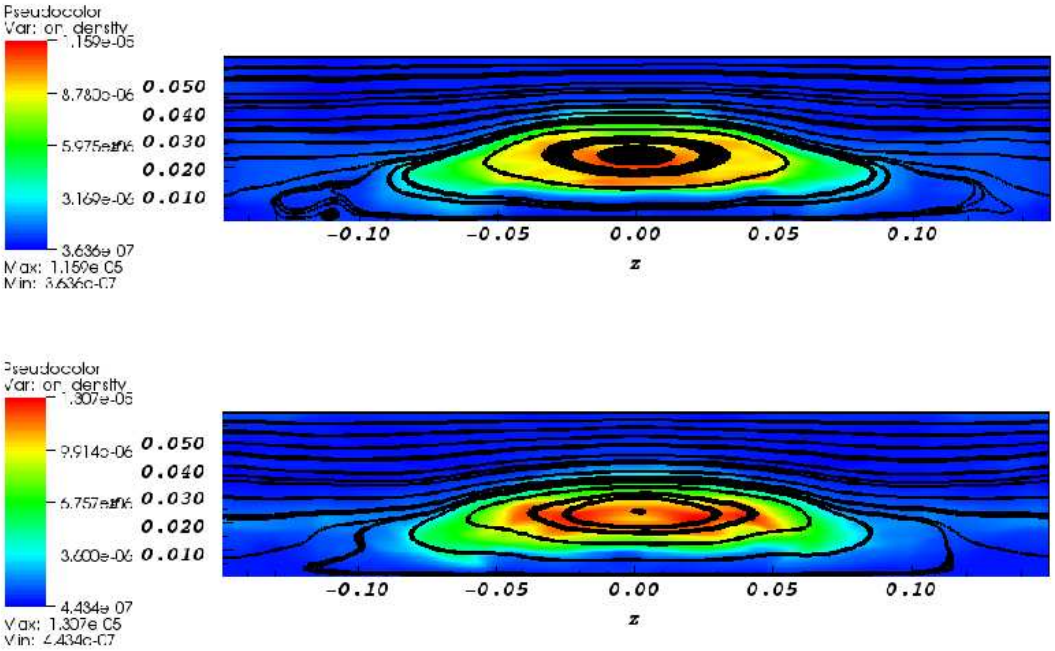

Figure 7.20: FRC evolution for several mass ratios using the two-fluid model. Top plot is the ion density and magnetic field after $5\tau_a$ using $M = 50$. Bottom plot is the ion density and magnetic field after $5\tau_a$ using $M = 100$.

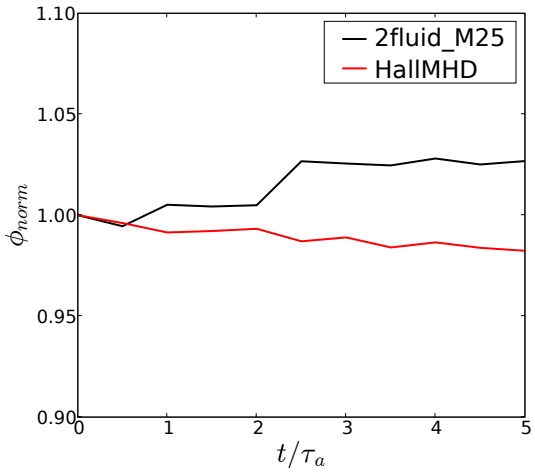

Figure 7.21: FRC Hill's Vortex conserved flux between the null-point and the axis for the two-fluid $M = 25$ and Hall-MHD models. Note that the closed flux is mostly constant except for a deviation of approximately 2.5% for the two-fluid closed flux over $5\tau_a$.

demonstrates that the algorithm is relatively stable and is capable of handling dynamics in the solution effectively. The presence of a discontinuity in the initial condition in B_z and the small variation in the closed flux confirm the robust nature of the numerical methods used.

For this problem, Hall-MHD requires 8-30 times the computational effort of the two-fluid model depending on the density/pressure floor value that is selected. For some problems, Hall-MHD requires a higher density/pressure floor. A higher density floor value leads to lower speeds i.e. Alfvén, sound, whistler and magnetosonic speeds. This results in higher time steps thus taking less computational effort. The two-fluid model appears to take less computational effort to provide comparable solutions to Hall-MHD for all the problems explored above that lie in the two-fluid regime when solved explicitly.

In addition to relaxing to a two-fluid equilibrium from the approximate MHD equilibrium profile initially specified, the solution develops a toroidal magnetic field. It is not uncommon to see toroidal fields and poloidal currents develop in such a profile even though it was initialized with 0 toroidal field. The development of a toroidal field in an FRC is strictly a two-fluid effect and cannot be captured by single fluid MHD. While the large toroidal fields

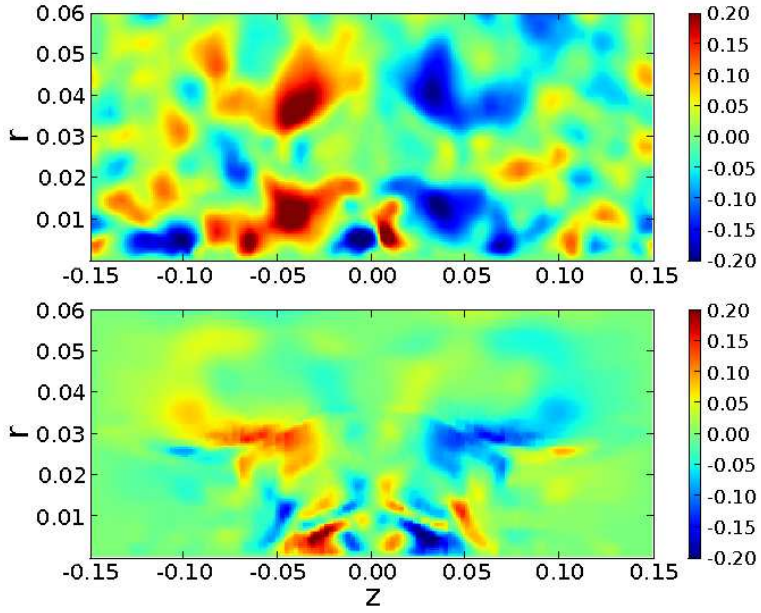

Figure 7.22: FRC profile of toroidal magnetic field after $2.5\tau_a$ in the two-fluid plasma model (top plot) and the Hall-MHD model (bottom plot).

observed here could be the result of the discontinuous initial condition leading to large dynamics, FRC experiments have observed the development of a toroidal magnetic field. Toroidal fields develop in both the two-fluid and Hall-MHD models. The toroidal magnetic field that develops is less than half the poloidal magnetic field at any given time during the evolution, often being an order of magnitude smaller than the poloidal magnetic field. Figure 7.22 shows the toroidal magnetic field in the two-fluid and Hall-MHD solutions after $2.5\tau_a$. The solutions are to the same scale, however, the two-fluid model develops a larger toroidal magnetic field than Hall-MHD after $2.5\tau_a$. The toroidal magnetic field plots show that the toroidal field within the separatrix is out-of-plane on one side and in-plane on the other side. Ref.[73] examines the generation of the substantial toroidal magnetic field inside the separatrix and states that this is due to stretching of the poloidal field by a sheared toroidal electron flow. It is interesting to note that after $5\tau_a$, the toroidal field for both the two-fluid model and the Hall-MHD model appears to reverse polarity and decreases in magnitude as compared to the solution at $2.5\tau_a$. Such toroidal field reversal has been

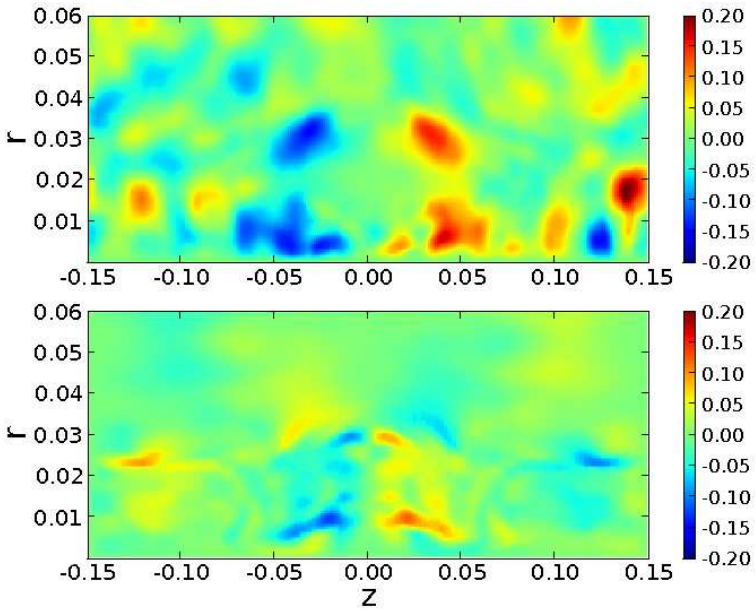

Figure 7.23: FRC profile of toroidal magnetic field after $5\tau_a$ in the two-fluid plasma model (top plot) and the Hall-MHD model (bottom plot).

seen in experiments such as the Translation, Confinement, Sustainment (TCS) device at the Redmond Plasma Physics Laboratory (RPPL)[74]. This is seen from Fig. 7.23 at which point the solution has reached a stable equilibrium.

Chapter 8

TWO-FLUID INSTABILITIES

The two-fluid drift-turbulence instability (DTI) is investigated in this chapter for applications of the Z-pinch and the Hill's vortex FRC. A large number of publications are available describing the lower hybrid drift instability (LHDI)[75, 76], evolution of LHDI in current sheet equilibria[77, 78], LHDI in Z-pinches[3], LHDI in field-reversed plasmas[79] and correlations between LHDI and anomalous resistivity[80, 4]. Ref.[81] analyzes LHDI in a thin current sheet using kinetic simulations. The nonlinear small-wavelength instability observed in two-fluid simulations of the Z-pinch and the FRC are studied in this chapter to determine if these are LHDI. Ref.[77] presents an analysis of LHDI and the evolution of LHDI-like modes from drift kink and drift sausage modes (DKI and DSI). This resembles the instability that is being observed here; the initial sausage and kink perturbations in the Z-pinch drive smaller-wavelength modes in the two-fluid regime. Refs.[75] and [82] mention that the fastest growing LHDI wavelength is governed by $kr_{Le} \sim 1$ and that LHDI is observed to develop in regions of strong density gradients. Additionally, LHDI requires $\mathbf{k} \cdot \mathbf{B} = 0$. While the instability being observed here appears to correspond with the LHDI, the wavelength of the instability appears to be on the order of $kr_{Li} \sim 1$. The following sections present an analysis of the two-fluid instability with respect to fastest growing modes and growth rates. Due to the wavelength of the instability not being on the order of $kr_{Le} \sim 1$, this drift-turbulence instability is not LHDI. Refs.[3] and [4] incorrectly refer to this instability as LHDI. All simulations in this chapter use the collisionless two-fluid model.

8.1 Axisymmetric Z-pinch Small-Wavelength Instability

The small-wavelength instability is investigated using the axisymmetric Z-pinch. The initial conditions and parameters used are the same as Sec. 5.5 except c is chosen to be an order of

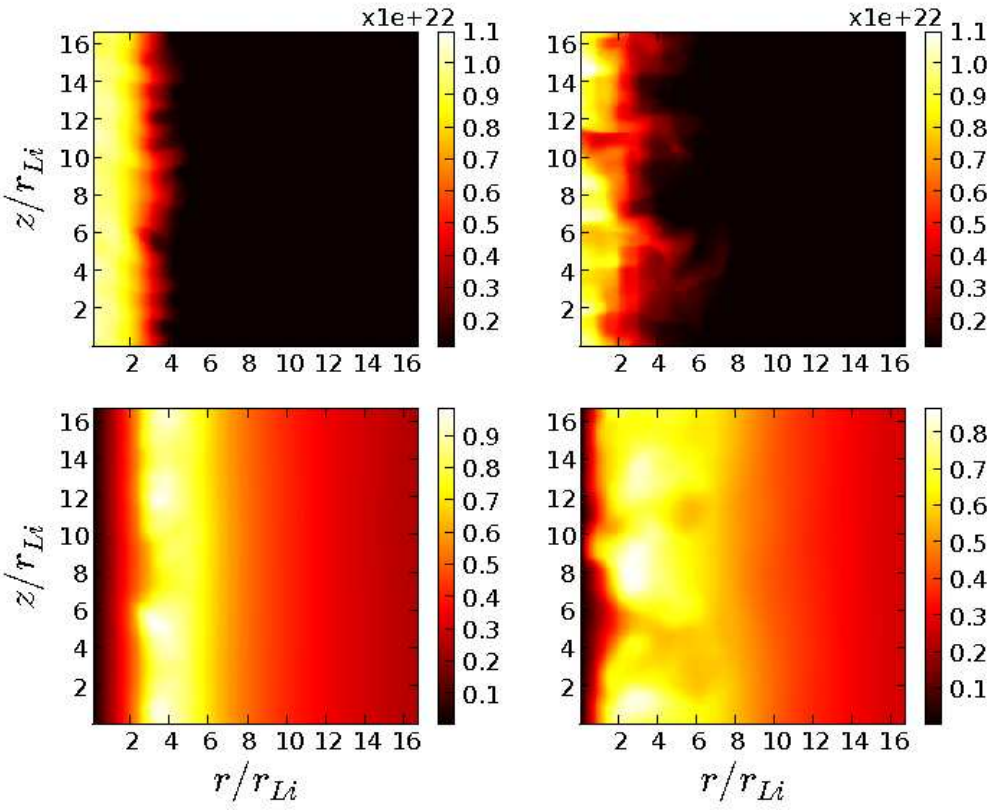

Figure 8.1: Two-fluid drift turbulence instability ion density for the axisymmetric Z-pinch after $2.5\tau_a$ (left plot) and $5\tau_a$ (right plot). The top plots are of ion density and the bottom plots are of the azimuthal magnetic field. The wavelengths of the instability are on the order of the ion Larmor radius.

magnitude higher. $M = 25$, $c/c_{si} \approx 200$, $c/c_{se} \approx 40$, $c/v_A \approx 195$, $v_d/v_{thi} = 0.9$, $R_p/r_{Li} \approx 4$ where $R_p = 0.25L$ and L is the domain length. Figure 8.1 shows the ion density and azimuthal magnetic field after $2.5\tau_a$ and $5\tau_a$. It is observed that the small wavelength modes that develop in the axisymmetric Z-pinch are on the order of the ion Larmor radius even though only a single wavelength sausage mode perturbation was initialized in the magnetic field.

Figure 8.2 shows plots of fluid kinetic and thermal energies after $4\tau_a$. Note that the thermal energies of both fluids are many orders of magnitude larger than the kinetic energies.

The higher ion kinetic energy is due to larger ion mass compared to the electrons. The thermal energies of the fluids appear to peak at the same locations and it seems that the ions and electrons share the same regions of high thermal energy. However, in the temperature plots in Fig. 8.3 the ion temperatures appear to be the inverse of the electron temperatures, i.e. hot ions exist in regions of cold electrons and vice versa. While the magnetic field heats the magnetized electrons, the ions are unmagnetized. Since a collisionless two-fluid model is used here, there are no transport terms that drive the fluid temperatures together.

Figure 8.3 shows additional interesting features in the fluid temperatures after $4\tau_a$. It appears that high electron temperatures correspond to high magnetic fields. Since the electrons are magnetized in this regime, the high magnetic field confines the electrons and causes them to heat up. The hot regions of electrons correspond generally to regions of lower electron kinetic energy in Fig. 8.2. Looking at both fluid temperatures, it is observed that both fluids undergo adiabatic heating in the regions where shocks occur in the ion fluid. Interestingly, The ions have high kinetic energy in regions of low ion temperature. Thus, the ions lose thermal energy to gain kinetic energy.

Furthermore, the velocity profiles of each of the fluids show the development of shear. Figure 8.4 shows the evolution of the ion axial velocity, the electron axial velocity, the axial electric field, the ion radial velocity, the electron radial velocity, and the radial electric field after $2\tau_a$ and Fig. 8.5 shows the evolution after $4\tau_a$. Initially the electrons carry all the current and the ion velocity is 0. Also, the initial electron velocity is only in the axial direction and the initial radial velocities of both fluids are 0. Figures 8.4 and 8.5 show that the magnitudes of the radial velocities that result from the evolution of the drift-turbulence instability are approximately the same order of magnitude as the axial velocities and all velocities peak in the region of the pinch radius where there is maximum density gradient. As the simulation progresses, it is observed that the ions and electrons develop a velocity shear that creates a similar profile in the radial electric field. The axial electric field also develops a profile that is consistent with the axial ion and electron velocities. The highly dynamic axial ion velocity in the region of the pinch radius results from a charge separation that occurs as the short-wavelength modes grow. The separation between positive and negative ion axial velocity corresponds to the wavelength of the instability. With the presence of velocity

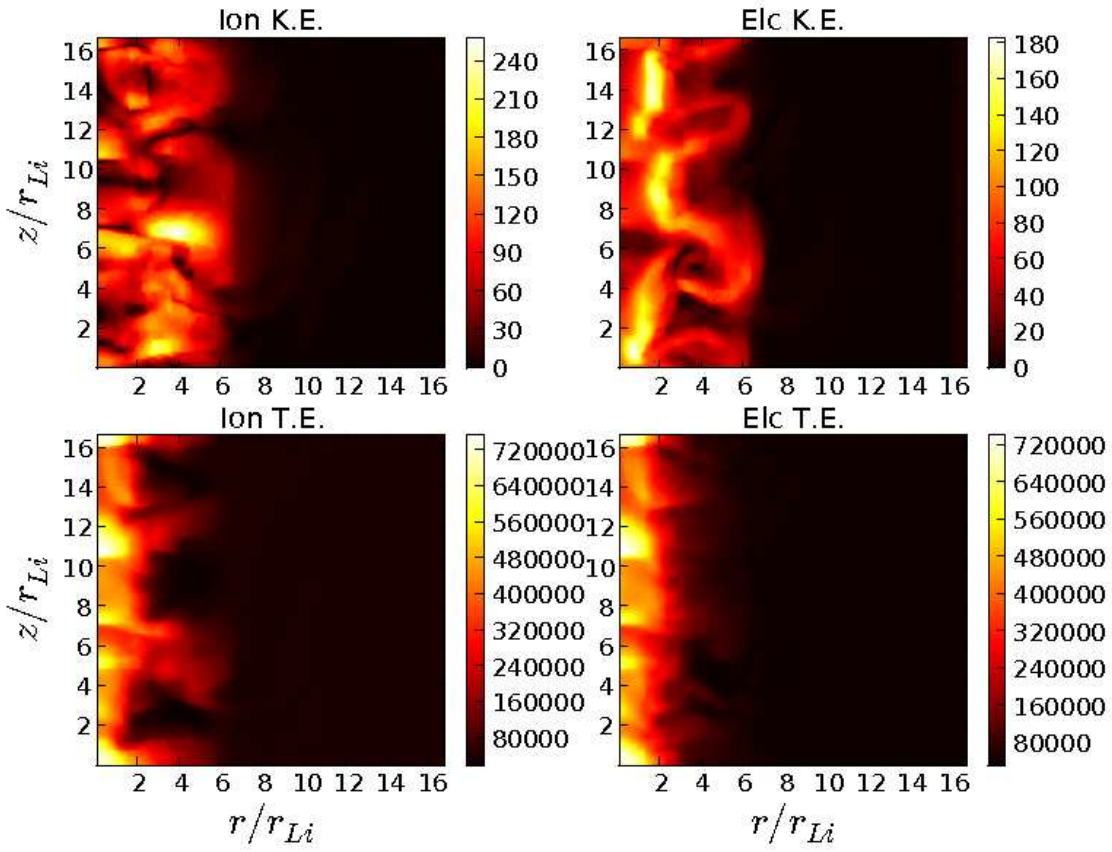

Figure 8.2: Two-fluid drift-turbulence instability for ion (left) and electron (right) kinetic energy in the top plots, and ion (left) and electron (right) thermal energy in the bottom plots after $4\tau_a$. Note fluid thermal energies are much larger than the kinetic energies. Also, both ions and electrons appear to share regions of high thermal energy.

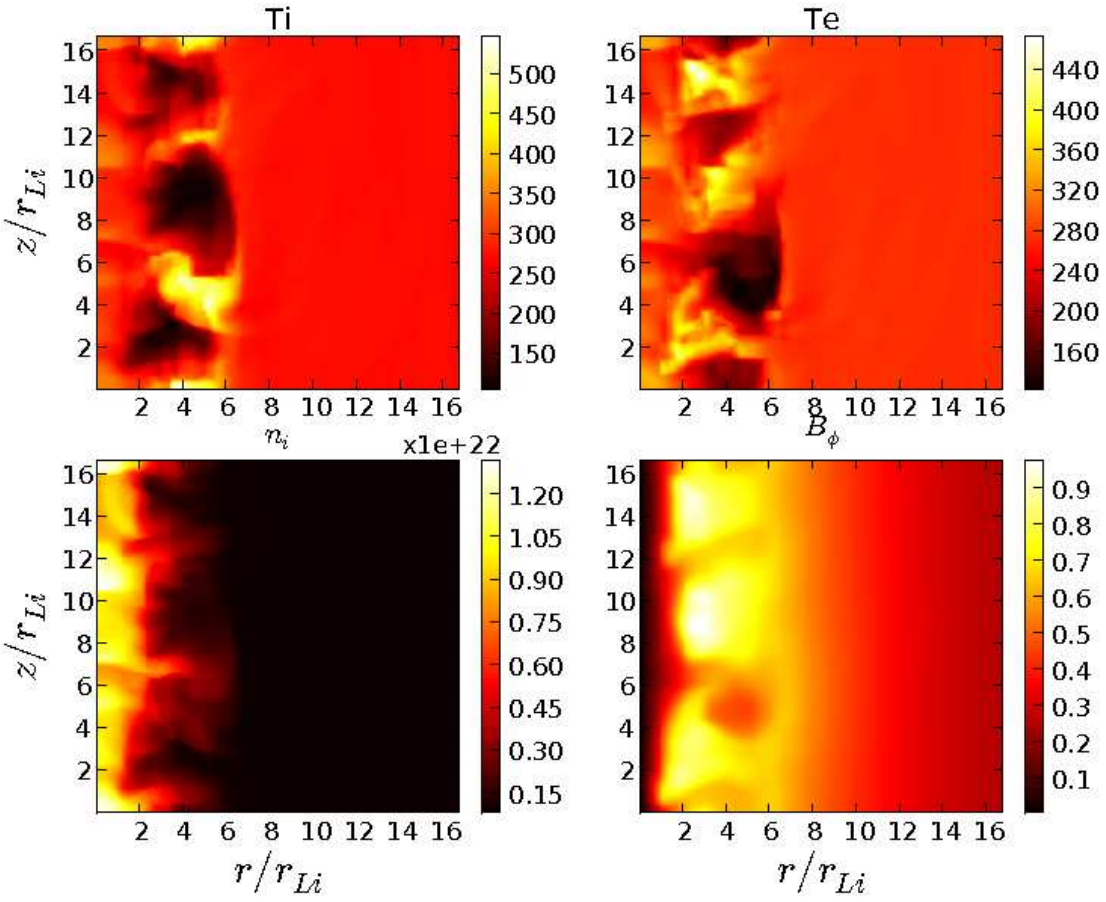

Figure 8.3: Two-fluid drift-turbulence instability for ion (left) and electron (right) temperatures in the top plots, ion number density (left) and azimuthal magnetic field (right) in the bottom plots after $4\tau_a$. Note hot ions in regions of cold electrons and vice versa. Also, hot electrons correspond to regions of large magnetic field.

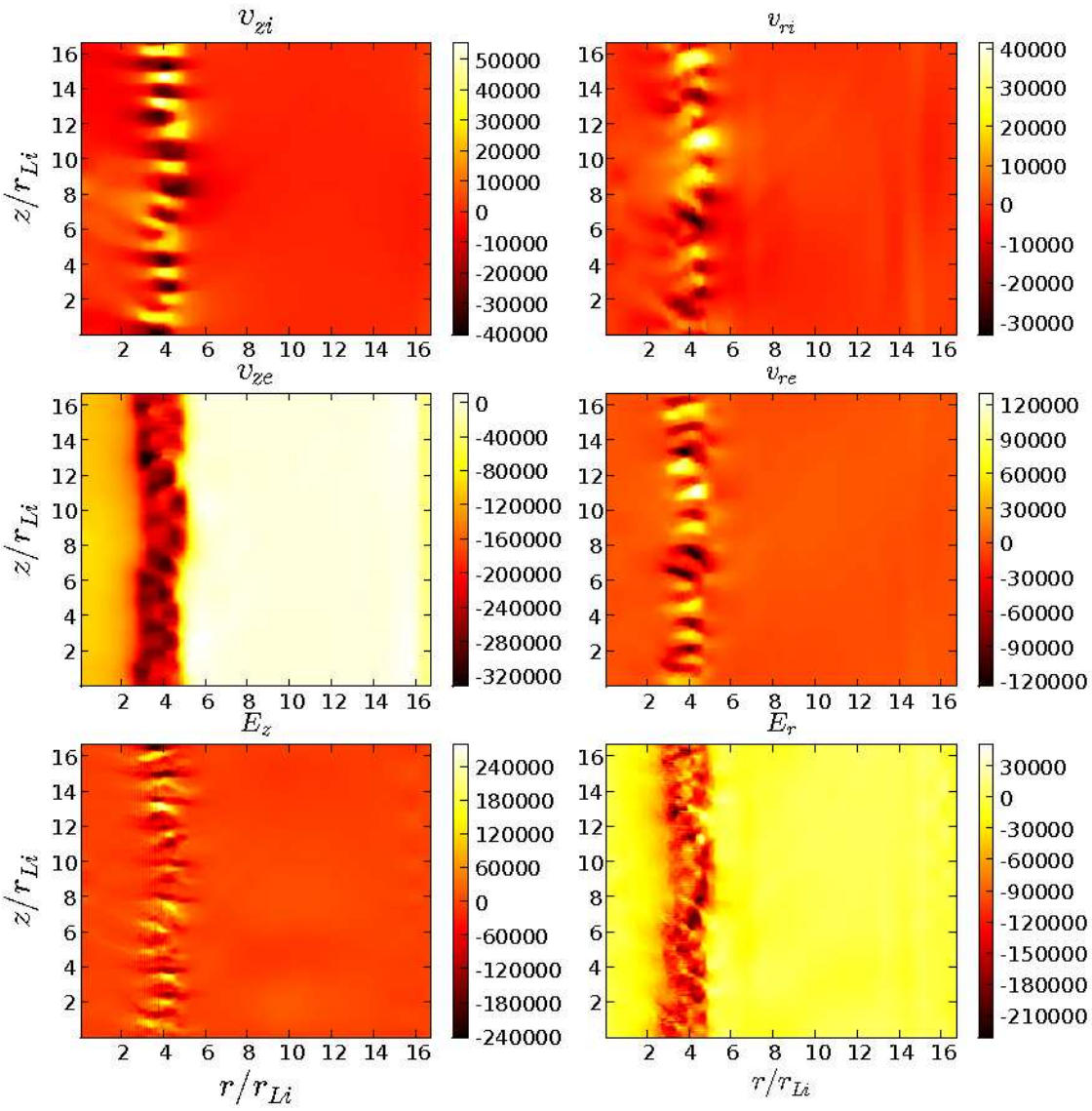

Figure 8.4: Two-fluid drift-turbulence instability for velocity profiles. Left: Ion axial velocity (top), electron axial velocity (middle), axial electric field (bottom) after $2\tau_a$. Right: Ion radial velocity (top), electron radial velocity (middle), radial electric field (bottom) after $2\tau_a$. Note the shear profile that develops in the ion radial velocity which was initialized to 0. This is also seen from the radial electric field.

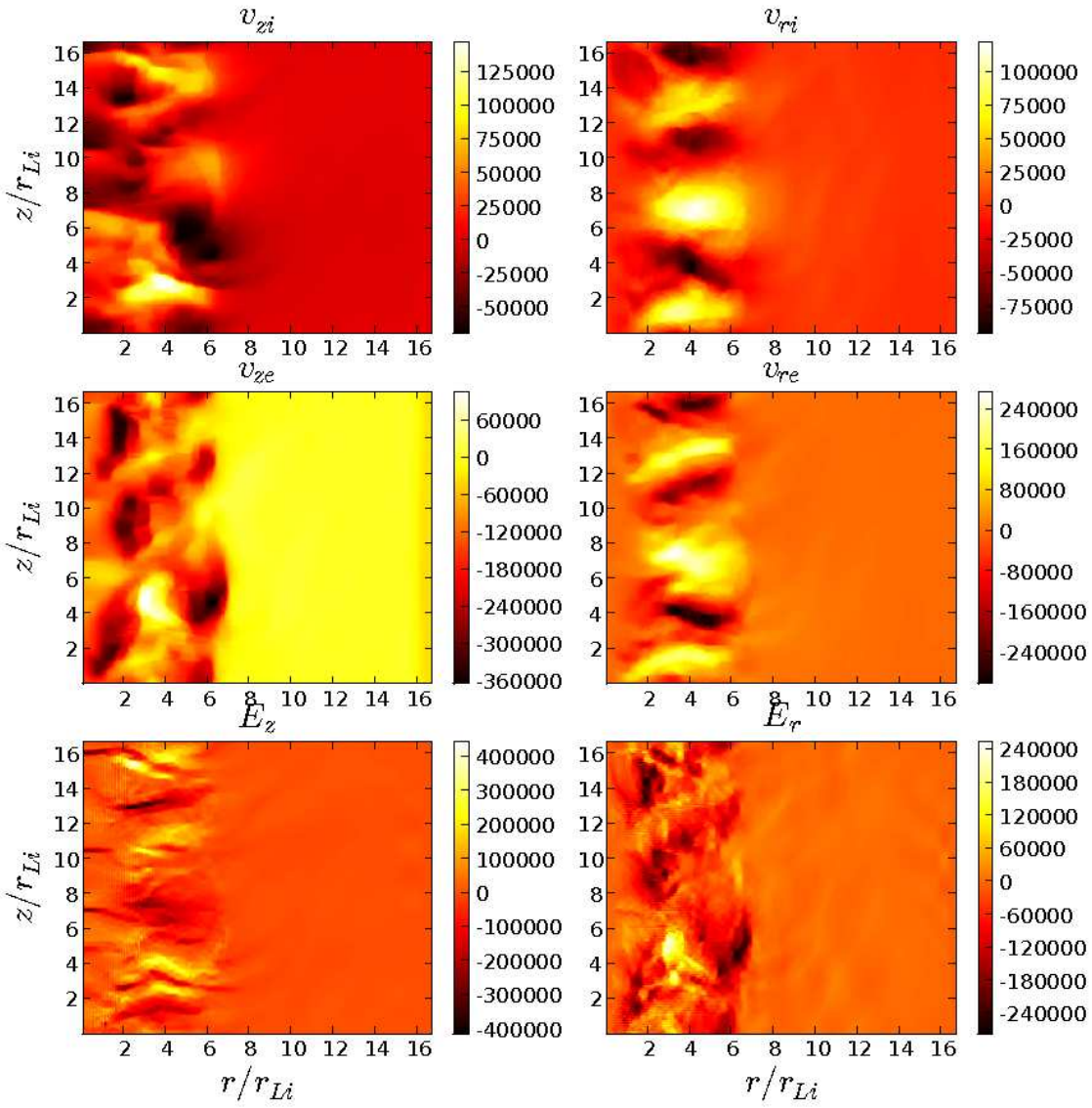

Figure 8.5: Two-fluid drift-turbulence instability for velocity profiles. Left: Ion axial velocity (top), electron axial velocity (middle), axial electric field (bottom) after $2\tau_a$. Right: Ion radial velocity (top), electron radial velocity (middle), radial electric field (bottom) after $4\tau_a$. Note the strong shear profile that develops in the ion radial velocity which was initialized to 0.

shear as seen from the radial velocity and electric field profiles, the instability develops a secondary Kelvin-Helmholtz type fluid instability (KHI) where vortices and shocks form in the ion fluids driving it turbulent and eventually unstable. KHI has been investigated using ideal- and Hall-MHD[83].

The simulation is in the appropriate regime for LHDI with unmagnetized ions and magnetized electrons, $T_e \sim T_i$, and $\mathbf{k} \cdot \mathbf{B} = 0$ so it is possible that LHDI initially triggers the instability. Also, the instability observed appears to grow from the region of maximum gradient in density which is consistent with the evolution of LHDI[82]. For the fastest growing LHDI modes, $kr_{Le} \sim 1$, but there are longer LHDI modes on the order of $k\sqrt{r_{Le}r_{Li}} \sim 1$[81]. The wavelength of the instabilities observed in the two-fluid Z-pinch, $kr_{Li} \sim 1$, are clearly larger than even the long-wavelength regime of LHDI. Ref.[84] investigates an instability that has a wavelength that is transverse to the magnetic field and is of the order or smaller than the ion Larmor radius. However, the perturbations in Ref.[84] are along the magnetic field unlike the Z-pinch results presented in this dissertation where the perturbations are perpendicular to the magnetic field. In order to observe LHDI modes ($kr_{Le} \sim 1$) in the present parameter regime, the grid resolution needs to be sufficiently high such that r_{Le} is resolved.

Ref.[85] discusses a drift kink instability where the instability wavelengths are $kr_{Li} \sim 1$ but the drift kink instability growth rate becomes weak as the ion-to-electron mass ratio is increased. The drift-kink instability plays a significant role in the solution in the presence of large, finite electron mass. Section 8.2 explores the effect of increasing mass ratios on the two-fluid drift-turbulence instability that is observed. There appears to be no significant impact of large ion-to-electron mass ratio on the development of the instability. The two-fluid drift turbulence instability observed here resembles what Refs.[86] and [87] refer to as LHDI. These modes are not on the order of $kr_{Le} \sim 1$ and are of longer wavelength, therefore the term LHDI is used loosely to describe the instability observed in these references. However, Ref.[87] presents an analysis to describe the instability modes observed as a coupling between the ion acoustic wave and the whistler wave. In Ref.[86], an obliquely propagating electromagnetic drift instability is analyzed and it is found that when the drift parameter is large, the backward propagating fast whistler wave reacts with the forward propagating

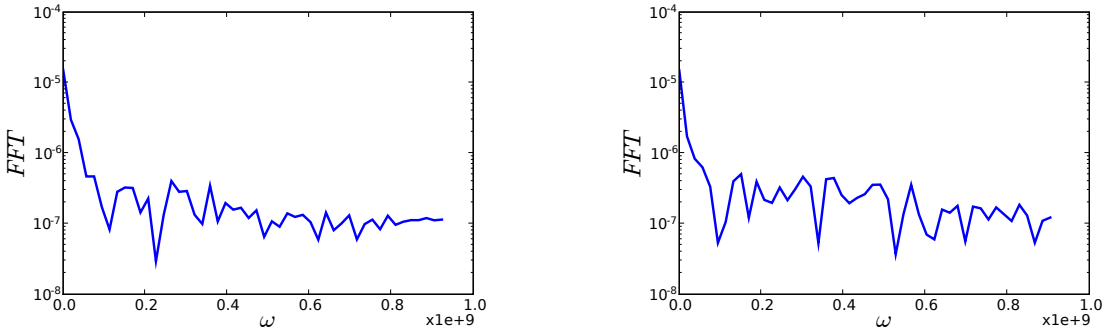

Figure 8.6: FFT-frequency plots for the axisymmetric Z-pinch instability. Left plot is for a resolution of 128×128 cells and right plot is for a resolution of 256×256 cells with a temporal resolution of 100 frames. $\omega_{LH} \approx 2 \times 10^8$

slow ion sound wave leading to a drift-turbulence instability. This instability resembles the two-fluid drift-turbulence instability observed in the Z-pinch and could account for the anomalous resistivity observed in experiments.

A fast Fourier transform (FFT) is performed on the solution to identify the fastest growing modes. First, the location of the maximum initial gradient in density is determined. The value of the density at this location is obtained for all time and an FFT is done to obtain a plot of wave number versus frequency. In doing this, the dominant frequencies are obtained for the given spatial and temporal resolution. This is done for a grid resolution of 128×128 cells and 256×256 cells using 100 outputs in time. Figure 8.6 shows plots of the FFT-frequency for each of the resolutions. The lower-hybrid frequency $\omega_{LH} = \sqrt{\omega_{ci}\omega_{ce}} \approx 2 \times 10^8$. It is seen that frequencies on the order of the lower-hybrid frequency are non-zero in the simulation but the lower-hybrid frequency does not appear to be dominant for the temporal resolution that is shown. Higher temporal resolution may show this frequency to be dominant. Additionally, it is possible to look at the growth rates of individual modes by doing an FFT as shown in Fig. 8.7. The location of the maximum gradient in the radial direction is chosen by looking at the initial ion density. Since the modes that develop are k_z modes, an FFT is performed for all points in z at the location of maximum gradient in the radial direction. Mode 1 corresponds to the initial sausage mode perturbation that is

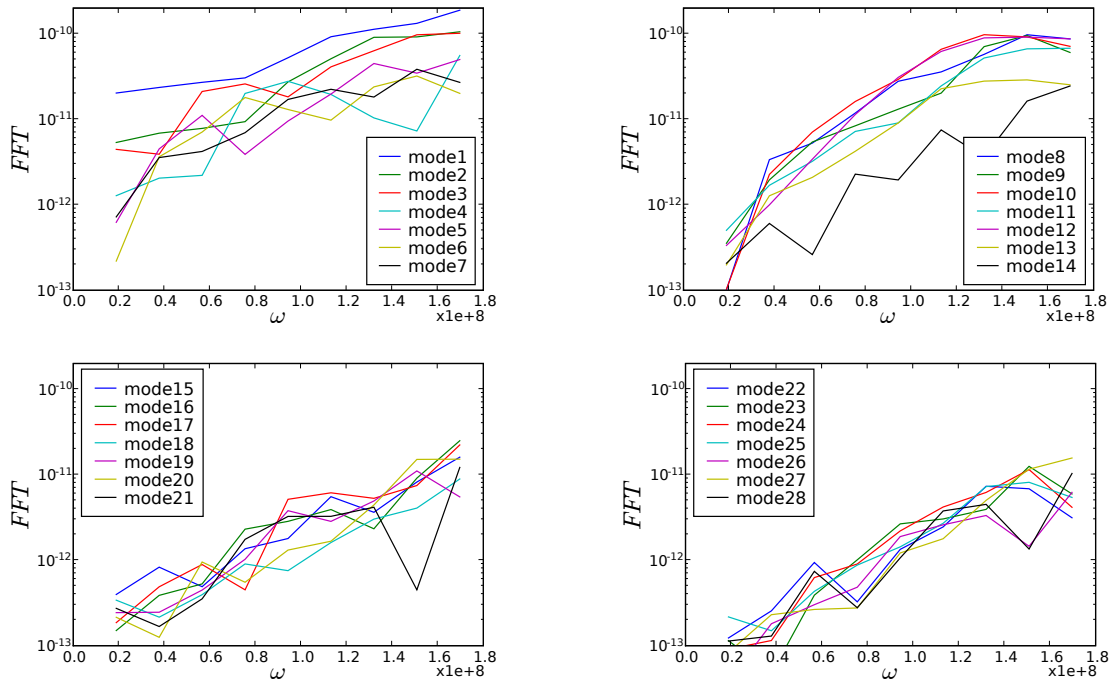

Figure 8.7: FFT plots for the axisymmetric Z-pinch instability showing growth rates of individual modes ranging from 1 to 28. A resolution of 128 × 128 cells is used. Mode 1 corresponds to the initial sausage mode perturbation. Modes 8-12 correspond to the fastest growing modes in the system.

applied. The FFT of modes 1-28 are plotted where the slope of each line corresponds to the growth rate of the mode. Modes 8-12 appear to have the largest slope indicating the largest growth rate. Specifically, modes 10 and 12 have the largest growth rates prior to nonlinear saturation.

The Mach number and the drift parameter at $2\tau_a$ and $4\tau_a$ are investigated in Figure 8.8. The ion Mach number at $2\tau_a$ shows that the ions are subsonic throughout and the drift parameter at this time has values of upto 3. After $4\tau_a$ the ions eventually become supersonic in some regions and the drift parameter reaches values of 4.

Refs.[88] and [75] study the evolution of LHDI in low drift velocity regimes ($v_d/v_{thi} << 1$) but also acknowledge that substantial resistivity and plasma heating can occur for large drift velocity regimes as well as small drift velocity regimes. The drift-turbulence instability

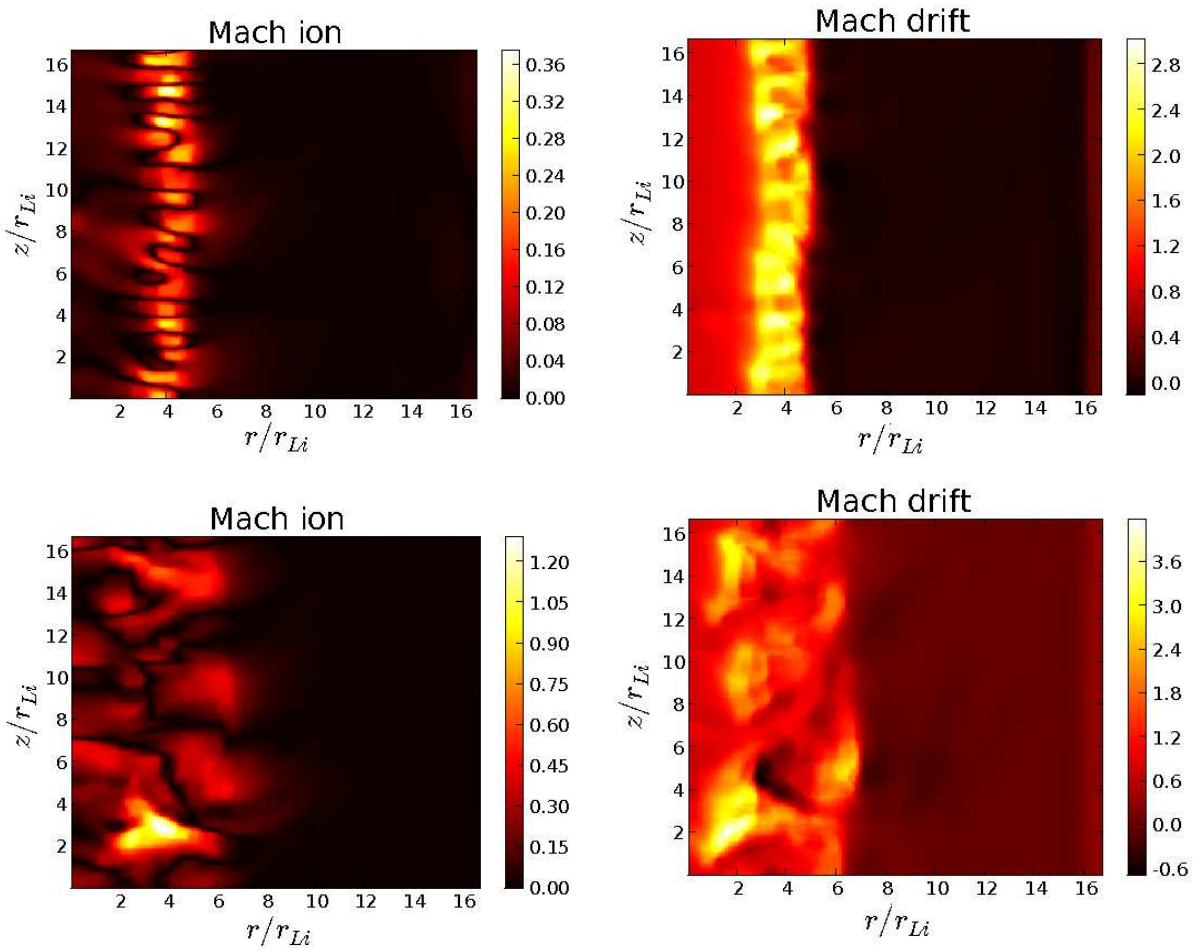

Figure 8.8: The ion Mach number (left) and the drift parameter (right) are computed at $2\tau_a$ (top plots) and $4\tau_a$ (bottom plots). The initial conditions have 0 ion velocity.

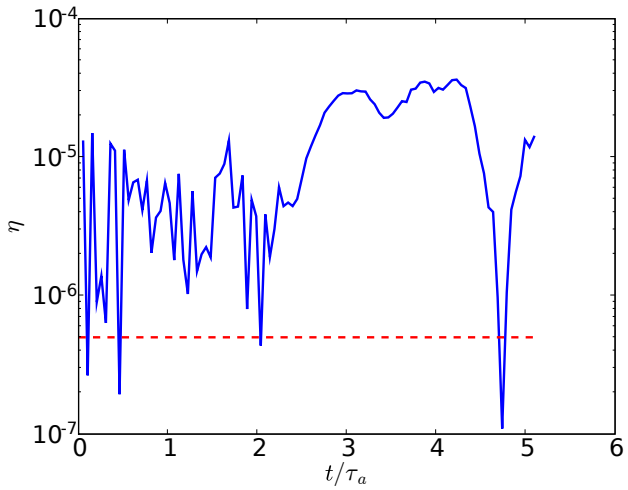

Figure 8.9: Two-fluid small-wavelength instability results in an anomalous resistivity even though a collisionless model is used. The blue line shows the anomalous resistivity as a function of the Alfvén transit time and the dashed red line shows the classical Spitzer resistivity for this regime. Note the anomalous resistivity is substantially higher than the classical resistivity.

presented here spans a large range of drift velocities from subsonic to supersonic over $5\tau_a$. Figure 8.9 shows the effective resistivity that is computed using

$$\eta = \iint \frac{\mathbf{E}\cdot\mathbf{J}}{\mathbf{J}\cdot\mathbf{J}} dr dz. \tag{8.1}$$

This anomalous resistivity is observed even though a collisionless two-fluid model is used without any explicit resistivity or viscosity. The presence of two-fluid effects by including electron mass and displacement currents brings about additional physics. The electron mass allows regimes where the electrons can be unmagnetized. The displacement currents allow local non-neutral effects to be captured. Figure 8.9 shows that the anomalous resistivity calculated is at least an order of magnitude larger than the classical Spitzer resistivity for this regime. This anomalous resistivity results from the two-fluid small-wavelength instability that evolves from the single wavelength perturbation. Similar anomalous resistivity has been observed in the Translation, Confinement, Sustainment-Upgrade (TCSU) device at

the Redmond Plasma Physics Laboratory (RPPL)[89, 4].

8.2 Asymptotic Studies of Mass Ratio on Small-Wavelength Instability

Asymptotic studies of mass ratio are conducted to determine whether the artificially large electron mass affects the two-fluid small-wavelength instability. The initial conditions are the same as described in Sec. 5.5 except for the speed of light which is chosen to be an order of magnitude higher so that decreasing the electron mass does not make the electron sound speed relativistic. For an ion-to-electron mass ratio of $M = 25$, $c/csi \approx 145$, $c/cse \approx 30$, $c/v_A \approx 195$. All other parameters remain the same. For $M = 50$, $c/cse \approx 20$, and the electron plasma and cyclotron frequencies change accordingly, but all other parameters remain the same. Likewise, for $M = 100$, $c/cse \approx 14$. The Z-pinch is initialized using a single wavelength perturbation and the evolution of the ion density is observed for each of the mass ratios after $2\tau_a$ and $4\tau_a$ in Fig. 8.10. It is noted that changing the mass ratio does not substantially affect the evolution of the instability.

Furthermore, to better understand the behavior of the instability, the growth rates for each of the mass ratios are plotted in Fig. 8.11. It is seen that the instability growth rates are in agreement for the various mass ratios where the fastest growing mode dominates the growth rate. The following sections will present a more detailed analysis on the fastest growing modes of the instability.

8.3 Asymptotic Studies of Drift Parameter on Small-Wavelength Instability

The drift parameter, v_d/v_{thi}, plays a critical role in the evolution of the small-wavelength drift-turbulence instability. The axisymmetric Z-pinch is investigated for several drift parameters to study the modes that result. The same initializations are used in Sec. 8.1 but a higher drift parameter regime is chosen, $v_d/v_{thi} = 2.7$ which results in $c/c_{si} \approx 13$, $c/c_{se} \approx 65$, $c/v_A \approx 60$, and $R_p/r_{Li} \approx 1.3$. Figure 8.12 shows ion density and azimuthal magnetic field after $1\tau_a$ and $2\tau_a$. Notice the formation of the instability in the ion fluid that is on the order of the ion Larmor radius.

Performing an FFT analysis similar to the $v_d/v_{thi} = 0.9$ case presented in Sec. 8.1, Fig. 8.13 shows the FFT versus frequency of each mode ranging from 1-14. Mode 1, which

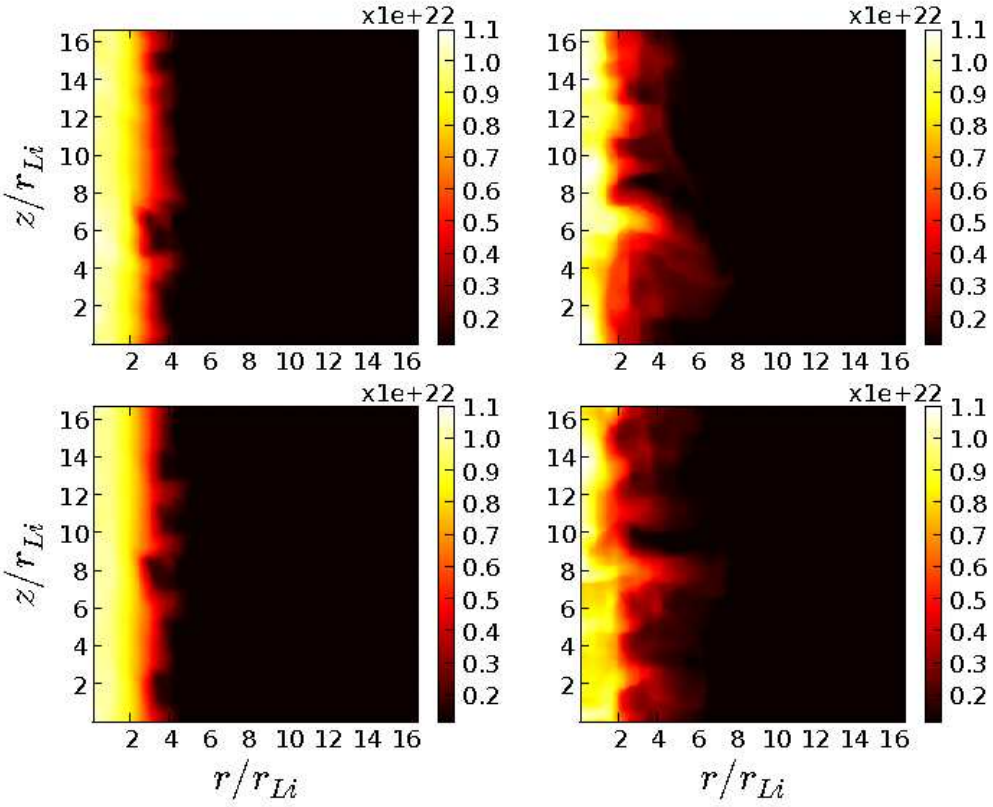

Figure 8.10: Solutions of ion density for an axisymmetric Z-pinch using ion-to-electron mass ratios of $M = 50$ (top plots), and $M = 100$ (bottom plots) at $t = 2\tau_a$ and $t = 4\tau_a$ from left to right. All plots are on the same scale and match the scale of the $M = 25$ plot in Fig. 8.1. Changing the mass ratio does not substantially affect the evolution of the two-fluid instability.

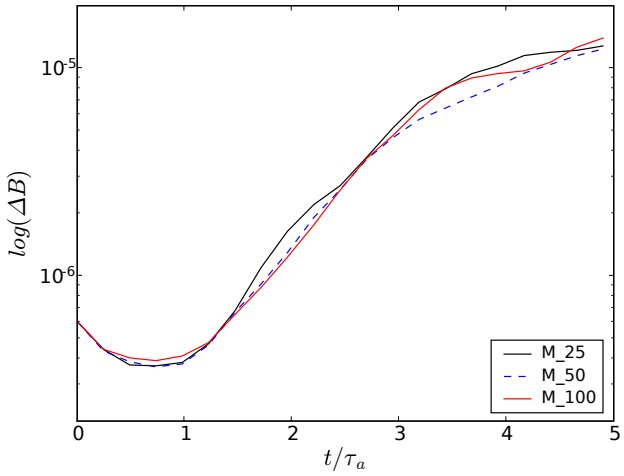

Figure 8.11: Two-fluid instability growth rates for an axisymmetric Z-pinch using ion-to-electron mass ratios of $M = 25$, 50, and 100.

is the single-wavelength mode, is the initialized perturbation. Mode 2 refers to two wavelengths, etc. For this problem, mode 4 is the fastest growing mode in the system whose wavelength corresponds to r_{Li}.

Just as with the $v_d/v_{thi} = 0.9$ solution, the drift parameter gets very large as the solution evolves and approaches values close to 10 near the pinch radius as is seen from Fig. 8.14 after $1\tau_a$. The ion velocity was initialized to 0 and the ion Mach number approaches sonic speeds after $1\tau_a$. After $2\tau_a$, the ion velocity is supersonic and the drift parameter approaches 15 near the axis. At $2\tau_a$, the instability is fully developed and has destroyed pinch. The ion fluid experiences shocks in the system and becomes supersonic as the solution evolves. Heating occurs at locations of the shock just as with the $v_d/v_{thi} = 0.9$ solution. Also, for $v_d/v_{thi} = 2.7$ the instability clearly evolves to be of order $kr_{Li} \sim 1$.

Ref.[90] presents an analysis to show that LHDI arises from drifts across fields in the presence of radial inhomogeneities in the plasma density. This instability is shown to arise in the presence of strongly magnetized electrons and unmagnetized ions and occurs at $\omega \sim \omega_{LH}$ when $\mathbf{k} \cdot \mathbf{B} = 0$. While all these criteria match the regime being explored for the two-fluid instability, the two-fluid instability observed here is not LHDI. The primary reason for this

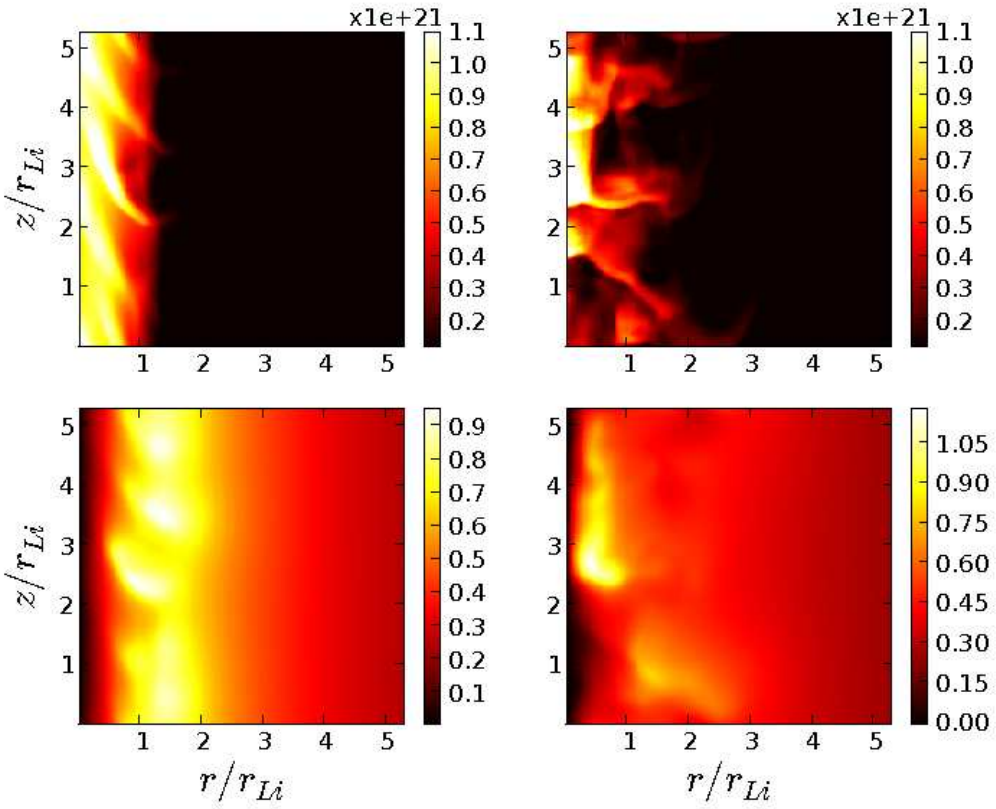

Figure 8.12: Two-fluid drift-turbulence instability ion density for the axisymmetric Z-pinch after $1\tau_a$ (left plot) and $2\tau_a$ (right plot). The top plots are of ion density and the bottom plots are of the azimuthal magnetic field. The wavelengths of the instability are on the order of the ion Larmor radius.

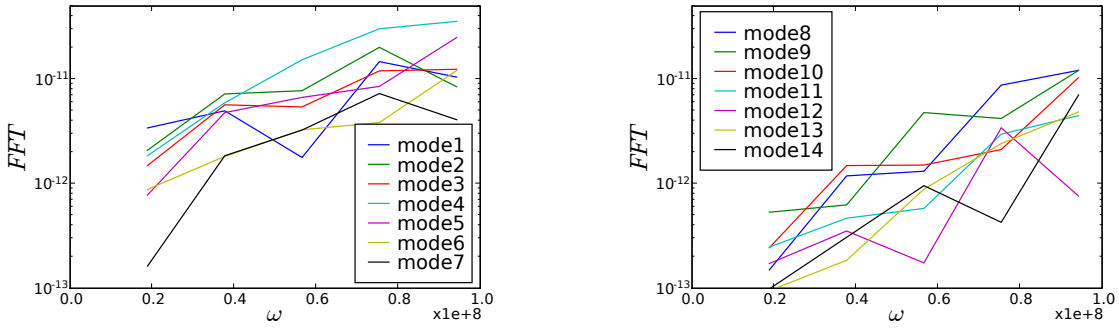

Figure 8.13: FFT plots for the axisymmetric Z-pinch instability showing growth rates of individual modes ranging from 1 to 14 for $v_d/v_{thi} = 2.7$. A resolution of 128×128 cells is used. Mode 1 corresponds to the initial sausage mode perturbation. Mode 4 is the fastest growing mode in the system.

discrepancy is because the wavelength of the instability is not on the order of the electron Larmor radius and is consistently and clearly on the order of the ion Larmor radius.

Exploring a low drift parameter regime such that $v_d/v_{thi} = 0.27$ which results in $c/c_{si} \approx 650$, $c/c_{se} \approx 130$, $c/v_A \approx 610$, and $R_p/r_{Li} \approx 13$, Fig. 8.15 shows that only a single wavelength perturbed sausage mode develops and grows faster than any other mode in the system. This solution is similar to the single-fluid MHD case where the ion Larmor radius is small enough such that the ion fluid is also magnetized. Ions need to be unmagnetized for the two-fluid instability to exist.

8.4 3-D Z-Pinch

Simulations of a fully 3-dimensional Z-pinch are done using the two-fluid plasma model in a Cartesian mesh as well as a cylindrical single-block general geometry grid described in Sec. 3.4.8. The initializations for this problem are similar to the normalized 2-dimensional Z-pinch initializations detailed in Sec. 4.4.4. The ion-to-electron mass ratio is 25 and the parameter regime is identical to the 2-dimensional case with the drift parameter $v_e/v_{thi} \approx 8$, where v_e is the electron drift velocity and v_{thi} is the ion thermal speed. The ratio of the pinch radius to the ion Larmor radius, $R_p/r_{Li} \approx 3$ such that $R_p = 0.5L$ where L is the domain length and $c \sim 16v_A$. 3-dimensional sausage and kink perturbations are initialized

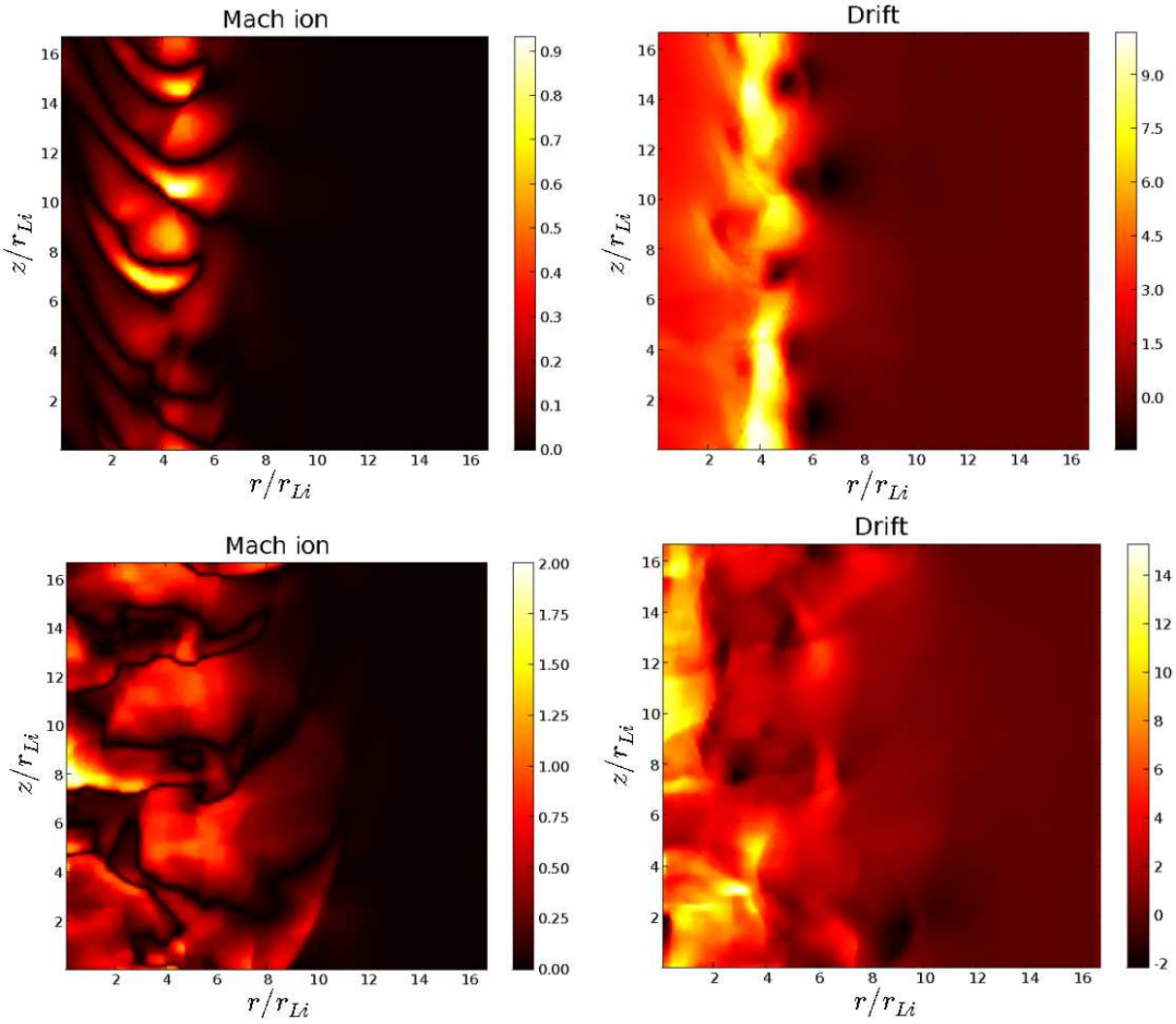

Figure 8.14: Top plots: The ion Mach number (left) and the drift parameter (right) are computed after $1\tau_a$. Bottom plots: The ion Mach number (left) and the drift parameter (right) are computed after $2\tau_a$. The initial condition has 0 ion velocity. Note the development of very large drift parameters of 10 near the pinch radius after only $1\tau_a$. After $2\tau_a$, the ions are supersonic and the drift parameter at the axis reaches values of 15.

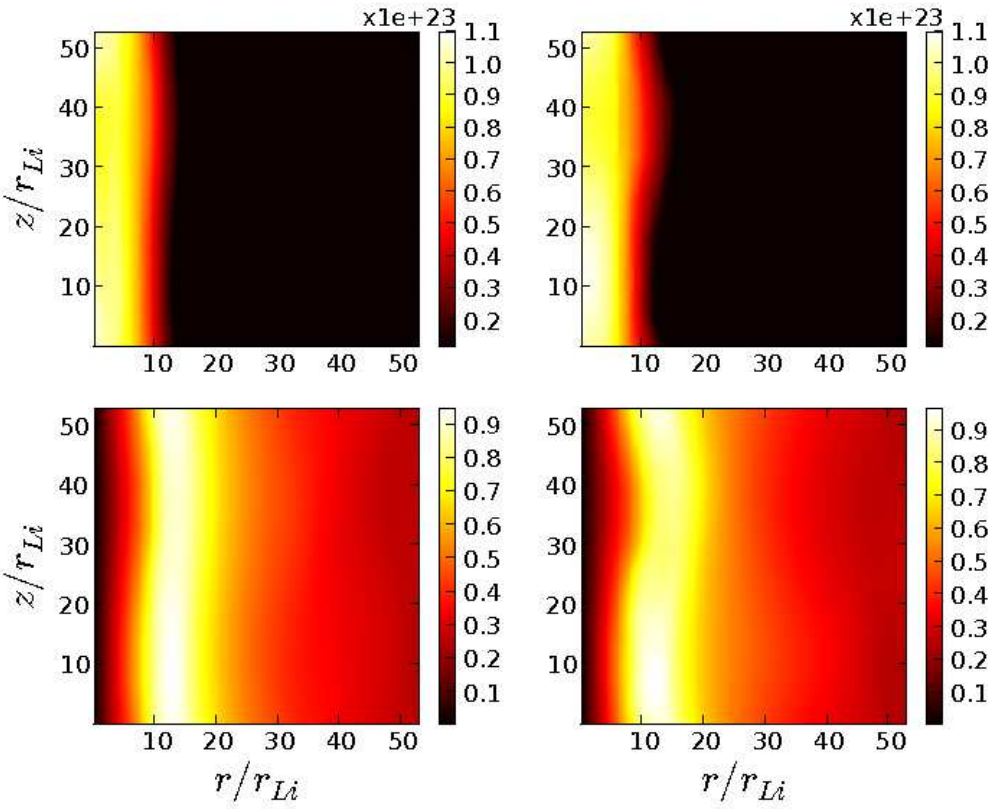

Figure 8.15: Two-fluid drift-turbulence instability ion density for the axisymmetric Z-pinch after $2.5\tau_a$ (left plot) and $5\tau_a$ (right plot). The top plots are of ion density and the bottom plots are of the azimuthal magnetic field. No short-wavelength instabilities are observed in this solution.

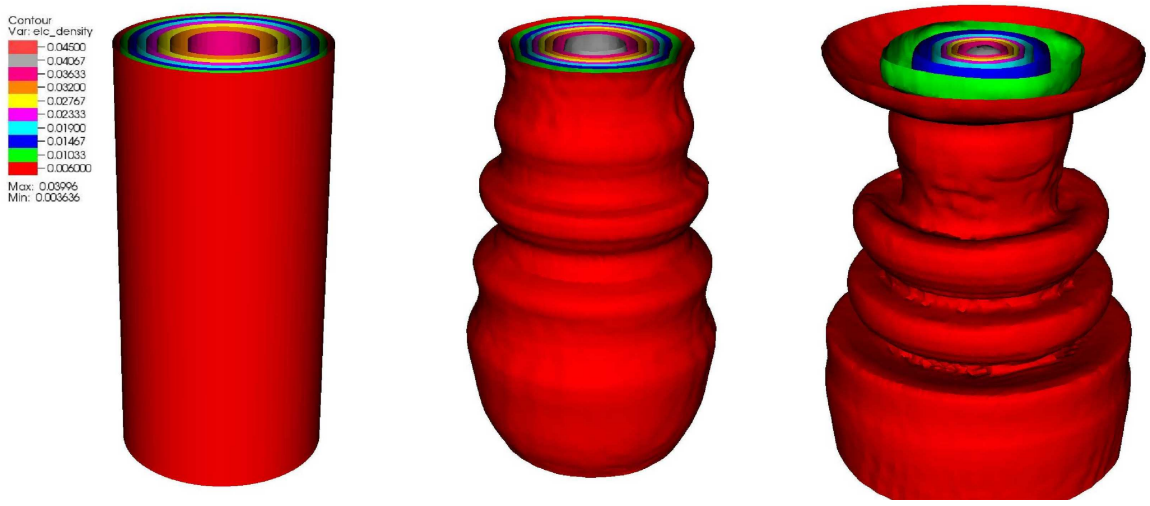

Figure 8.16: Electron density in a 3-D Z-pinch using the two-fluid plasma model. Left plot is the initial condition for electron density. Middle plot is the electron density after 4 Alfvén transit times. Right plot is the electron density after 5 Alfvén transit times. Notice the formation of the short-wavelength instabilities on top of the single wavelength sausage instability.

in an equilibrium Z-pinch using single wavelength perturbations.

The evolution of the macro- and micro-instabilities is observed. Figures 8.16 and 8.17 present a Cartesian geometry solution of the the short-wavelength lower-hybrid drift instability that forms on top of the single-wavelength sausage and kink instabilities. The solutions shown for the sausage and the kink instabilities are after 4 and 5 Alfvén transit times. The wavelength of the two-fluid instability observed is on the order of the ion Larmor radius as with the 2-D cases explored previously.

8.5 3-D Astrophysical Jets

The 3-D Z-pinch application can be extended to include the solution of astrophysical jets using WARPX. To simulate astrophysical phenomena, the regime is appropriately chosen such that $R_P/r_{Li} \approx 10$ with other initial conditions remaining the same as the Z-pinch. A kink mode perturbation of 3 wavelengths is initialized and the 3 wavelength mode remains the fastest growing mode in the system as seen in Fig. 8.18. This application shows that

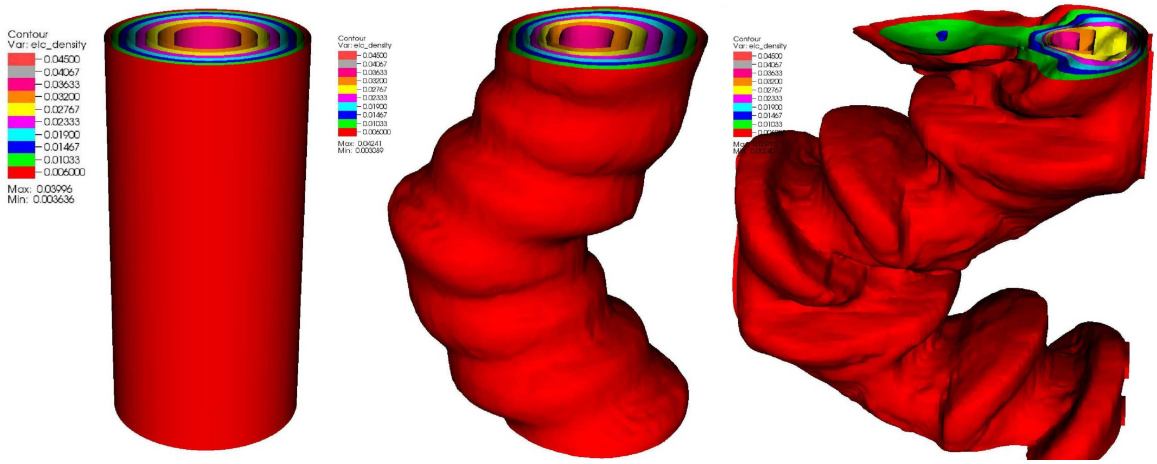

Figure 8.17: Electron density in a 3-D Z-pinch using the two-fluid plasma model. Left plot is the initial condition for electron density. Middle plot is the electron density after 4 Alfvén transit times. Right plot is the electron density after 5 Alfvén transit times. Notice the formation of the short-wavelength instabilities on top of the single wavelength kink instability.

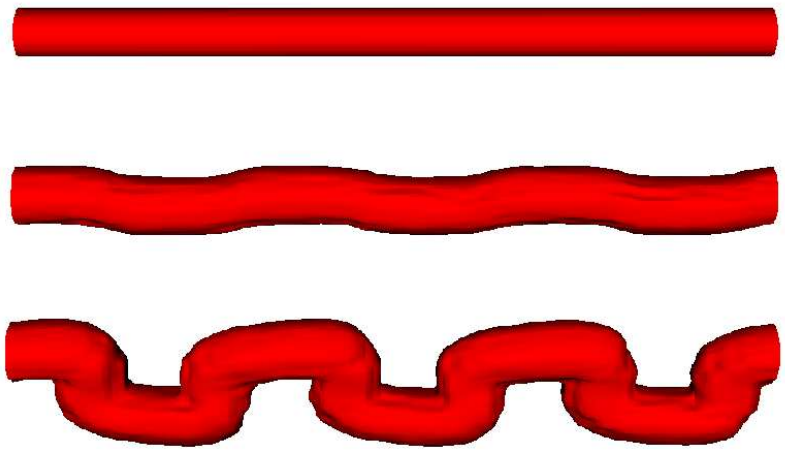

Figure 8.18: Electron density for a 3-D astrophysical jet using the two-fluid plasma model. Top plot is the initial condition for electron density. Middle plot is the electron density after 15 Alfvén transit times. Bottom plot is the electron density after 25 Alfvén transit times.

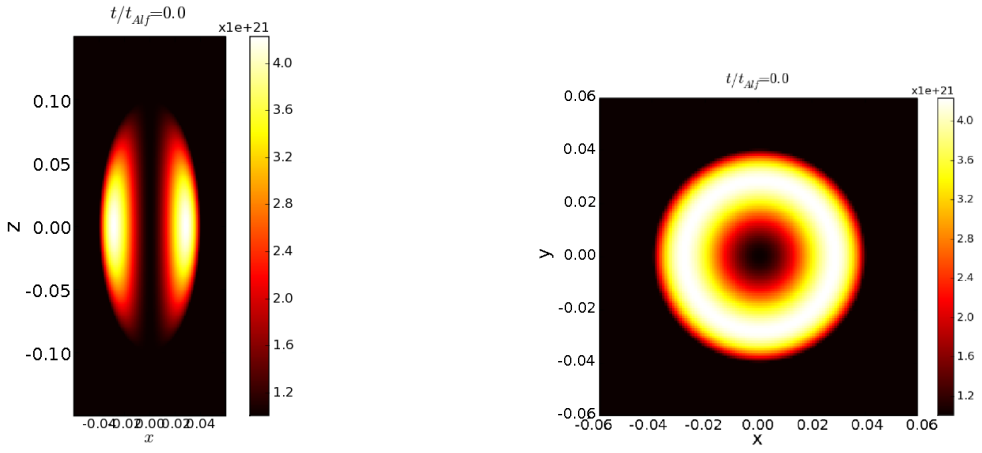

Figure 8.19: Initial condition of ion density for a 3-D Hill's vortex FRC using the two-fluid plasma model. Left plot is a cross-section in the x-z plane for the midplane in y. Right plot is a cross-section for the x-y plane for the midplane in z.

WARPX and the two-fluid plasma model can be applied to a number of plasma problems spanning a wide range of parameter regimes from the single-fluid MHD regime to a full two-fluid regime where electrons are also unmagnetized.

8.6 3-D Hill's Vortex FRC

A 3-D Hill's vortex FRC is simulated using the same initial conditions and parameters described in Sec. 7.4 with the domain radius, $r_c = 0.06$, the separatrix radius $r_s = 0.04$, $n_0 \approx 10^{22}/\text{m}^{-3}$, $T_i = T_e = 100\text{eV}$, $s \approx 4$, $M = 25$, $c/v_A \approx 22$. Figure 8.19 shows the initial condition of the ion density. The FRC is initialized with the parameters described in Sec. 7.4. Following this relaxation, it is allowed to run until instabilities set in. No perturbation is applied so numerical errors and grid effects are expected to eventually drive an instability.

Figure 8.20 shows the evolution of the ion density after $2.5\tau_a$ and $5\tau_a$. A two-fluid instability very similar to that observed in the Z-pinch forms in the FRC. This instability is on the order of the ion Larmor radius as with the Z-pinch results. Later in time, local grid effects set in and distort the instability as seen in the solution after $5\tau_a$. The drift parameter for this calculation is about 1.1 at the separatrix radius. Figure 8.20 shows that the solution

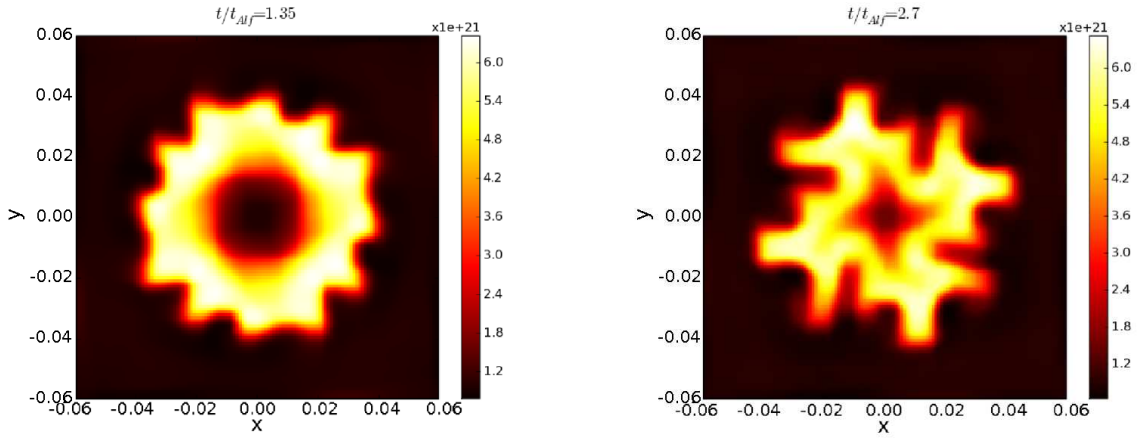

Figure 8.20: Initial condition of ion density for a 3-D Hill's vortex FRC using the two-fluid plasma model. Left plot is a cross-section of the ion density after $2.5\tau_a$ in the x-y plane for the midplane in z. Right plot is a cross-section of the ion density after $5\tau_a$ in the x-y plane for the midplane in z. Note formation of the small wavelength instability

appears to be affected by the rectangular boundaries as it becomes more rectangular with time.

A general geometry solution is used to investigate the effect of using a cylindrical grid to capture the cylindrical geometry. This is done to study whether the rectangular boundary has an impact on the evolution of the solution. Figure 8.21 shows results from a simulation with this grid after $2.5\tau_a$ based on the geometry described in Fig. 3.13. It is seen that the heavily distorted elements along the diagonals skew the solution. To investigate this further, the grid from Fig. 3.14 is explored with the same initial conditions and the solution after $2.5\tau_a$ is shown in Fig. 8.22.

Figure 8.22 shows that after $2.5\tau_a$ the solution appears to be stable and no two-fluid instability is observed. Being cylindrical, this grid eliminates effects of having rectangular boundaries on the solution. Also, the most distorted elements are near the conducting wall boundaries and do not affect the solution like the cylindrical grid in Fig. 8.21. In fact, even after $4\tau_a$, the solution looks stable without the development of the two-fluid drift-turbulence instability. For this problem, the two-fluid drift turbulence instability needs to be triggered by altering the parameter regime such that the kinetic parameter is smaller. In the

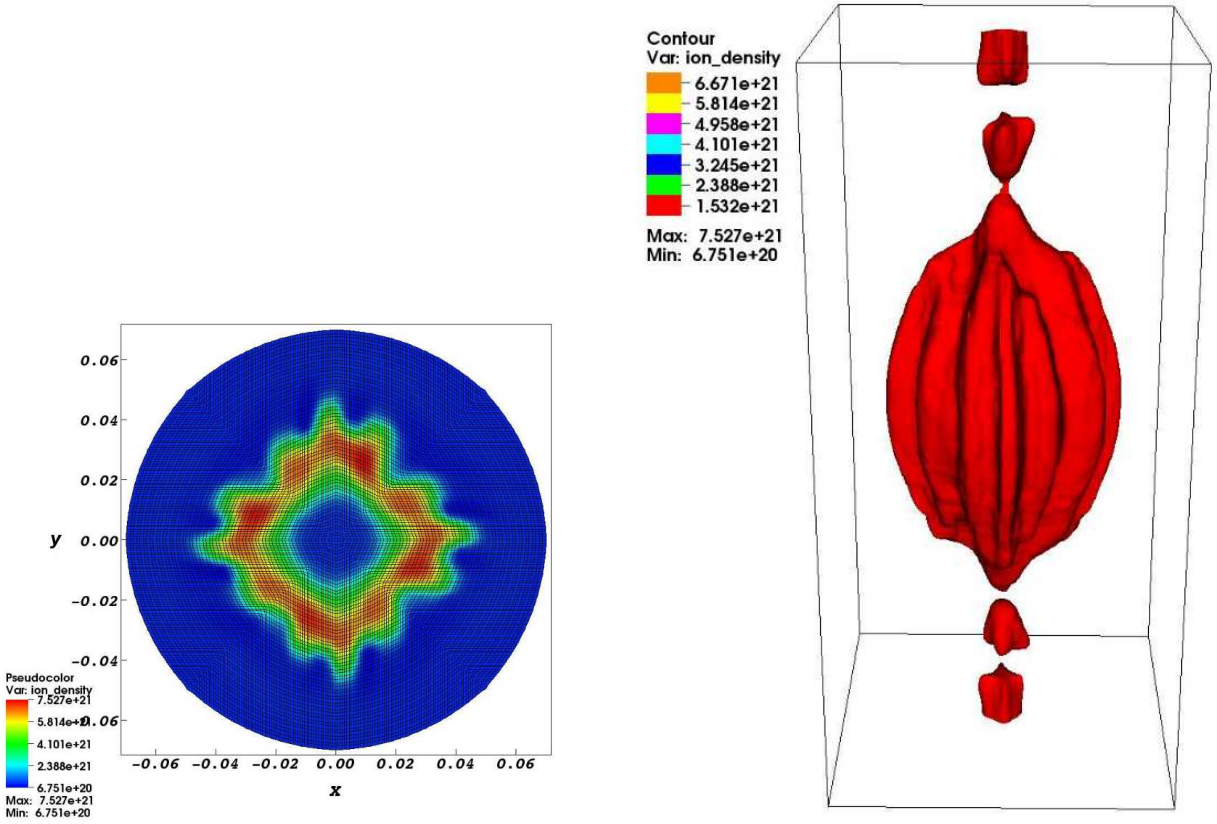

Figure 8.21: ρ_i for a 3-D Hill's vortex FRC using the two-fluid plasma model with cylindrical grid. Left plot is a cross-section of the ion density after $2.5\tau_a$ in the x-y plane for the midplane in z. Right plot is a 3-D contour plot of ρ_i at $2.5\tau_a$.

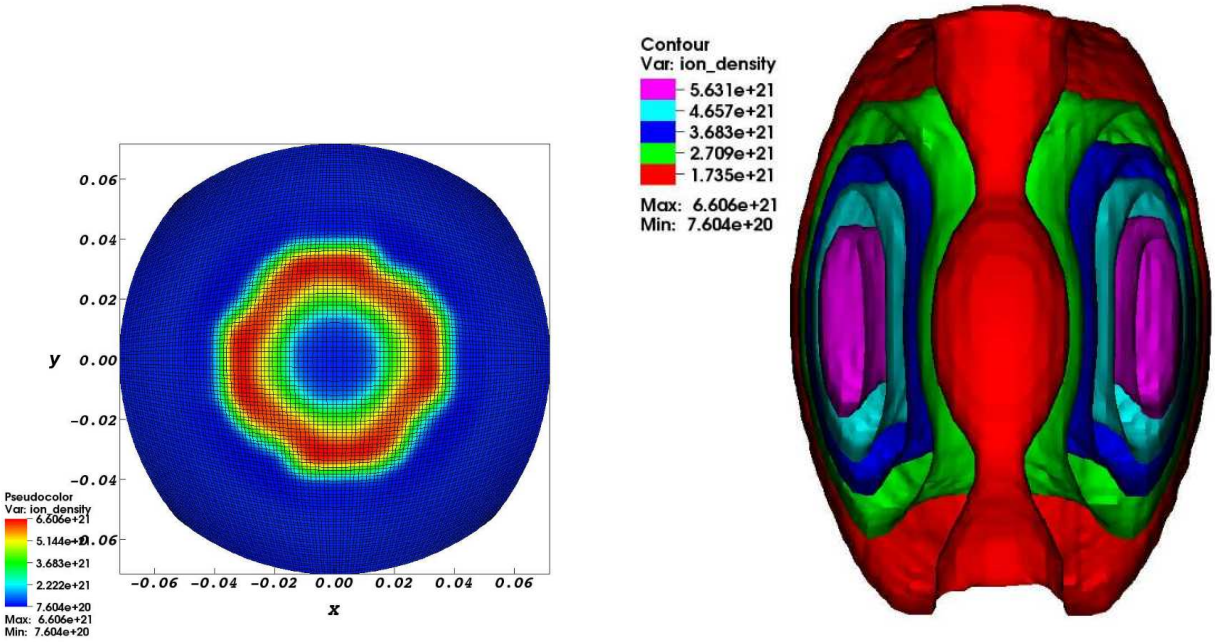

Figure 8.22: ρ_i for a 3-D Hill's vortex FRC using the two-fluid plasma model with cylindrical grid. Left plot is a cross-section of the ion density after $2.5\tau_a$ in the x-y plane for the midplane in z. Right plot is a 3-D contour plot of ρ_i at $2.5\tau_a$. This grid does not show the development of the two-fluid instability. It also does not display the grid effect seen in the Cartesian geometry.

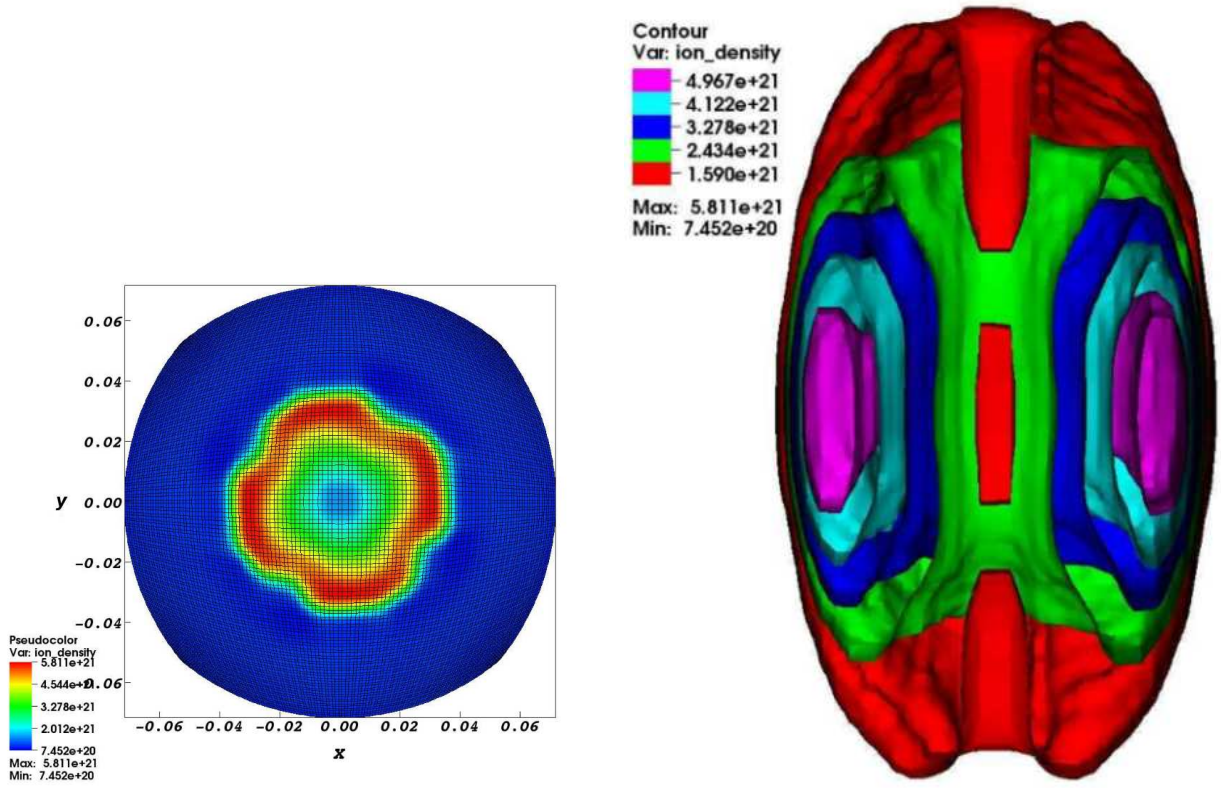

Figure 8.23: ρ_i for a 3-D Hill's vortex FRC using the two-fluid plasma model with cylindrical grid. Left plot is a cross-section of the ion density after $4\tau_a$ in the x-y plane for the midplane in z. Right plot is a 3-D contour plot of ρ_i at $4\tau_a$.

Cartesian geometry, the grid provided sufficient perturbation to trigger the instability. The grid effect in the Cartesian geometry coupled with a rectangular conducting wall boundary for the cylindrical geometry of the FRC could be responsible for the instability. However, in the cylindrical grid general geometry solution, a larger perturbation may be needed. It is possible that the regime explored is close to a stability boundary and a higher drift parameter or kinetic parameter could push the FRC to go unstable for the cylindrical geometry as well. Another way to approach this problem is to run it in a regime such that there is a large density gradient across a region containing a small kinetic parameter. The two-fluid drift-turbulence instability is only observed in the presence of sufficiently large Larmor radii and sufficiently large density gradients as investigated with the axisymmetric Z-pinch instability for different drift parameters and Larmor radii. Driving the 3-D FRC problem to a regime such that there is a large density gradient across a small kinetic parameter will most likely trigger the two-fluid drift turbulence instability in the FRC.

Chapter 9

CONCLUSIONS

9.1 Contributions

9.1.1 Analytical Study of Two-Fluid Plasma Model and Asymptotic Approximations

The two-fluid plasma model is studied and compared to reduced fluid models. An analytical study of dispersion diagrams in Chapter 2 shows that while the two-fluid plasma model has disparate speeds and frequencies, namely the speed of light, the fluid speeds of sound, the ion and electron plasma frequencies, and the ion and electron cyclotron frequencies, all the waves in the system reach an asymptote and can be resolved with explicit time-steps. For Hall-MHD, however, the Whistler wave grows without bound and requires a cut-off frequency or an artificial hyper-resistivity to truncate the high frequency modes. The time-step restriction of the two-fluid model can be overcome by implementing an implicit time-integration scheme that utilizes advanced features such as physics-based preconditioning, spectral multi-grid, and Newton Krylov iterative methods. An important feature of the ideal two-fluid plasma model is that it contains purely dispersive source terms and no explicit dissipation exists in the system. These source terms account for the wide variety of plasma waves in the system.

9.1.2 Implementation of Non-ideal Two-Fluid Plasma Model

A non-ideal two-fluid plasma model is implemented to include transport through terms such as momentum transfer, viscosity, resistivity, and heat flux as derived by Braginskii. Gyro-viscosity is neglected in this dissertation for simplicity, therefore, an absence of magnetic field is assumed for computing the viscous stress tensor. The inclusion of transport allows a physically relevant two-fluid model to be applied to self-consistently include dissipative effects. Chapter 2 describes the non-ideal terms, and their applications are presented in Chapter 5.

9.1.3 Implementation of 3-D Generalized Geometry DG Method

The numerical methods explored in this dissertation include a high resolution wave propagation method, a finite volume method, and a discontinuous Galerkin (DG) method, a finite element method. These are described in Chapter 3. The implementation details of each of the numerical algorithms in WARPX are provided in detail here. The DG method implemented in this dissertation is generalized for any spatial order and is implemented in 3 dimensions in WARPX. Also, to minimize grid effects in non-rectangular geometries, a general geometry implementation using the DG algorithm is included in WARPX in 3-dimensions as a part of this dissertation and all the implementation details are provided in Chapter 3. For now, a single block general geometry implementation is included. Multi-block and unstructured grids constitute future work. Implementation details for general geometry, Hall-MHD, and the non-ideal two-fluid plasma model are provided in Chapter 3.

9.1.4 Implementation of Explicit and Implicit-DG

The DG algorithm is implemented with explicit 2^{nd} and 3^{rd} order Runge-Kutta time-integration schemes, and with 1^{st} and 2^{nd} order implicit backward differencing and 2^{nd} order Crank-Nicholson schemes. Chapter 6 compares the implicit- and explicit-DG implementations for a 1-D electromagnetic plasma shock. The implicit-DG algorithm is actually a semi-implicit scheme for better computational cost and accuracy. Since the ion time-scales must be resolved as a minimum to capture two-fluid physics, the ions are evolved explicitly in time using a Runge-Kutta time integration with the DG method. The electron Euler equations and Maxwell's equations are solved implicitly using a Crank-Nicholson scheme. Solving a strong shock problem using an implicit algorithm is non-trivial due to highly non-linear evolution that can make convergence difficult. However, it is demonstrated that the implicit-DG algorithm is best suited for solving the MHD-shock problem when realistic parameters are used such that a real ion-to-electron mass ratio and a real light speed-to-sound speed ratio is used. This makes the explicit algorithm very restrictive since the speed of light and the electron plasma frequency become very large driving the time step to very small values. Additionally, if high divergence correction speeds are chosen for the

perfectly hyperbolic Maxwell's equations, an implicit algorithm may be suitable. For a 2-D implementation, further work is required to make the implicit algorithm more robust.

9.1.5 Comparing Finite Volume and Finite Element Methods for the Two-Fluid Plasma Model

Prior to selecting the DG algorithm for simulations performed in this dissertation, comparisons are performed between the high-resolution wave propagation algorithm and the DG algorithm specifically for applications to equation systems with purely dispersive source terms. Chapter 4 presents these results. It is noted that the wave propagation method provides very accurate and computationally efficient solutions when the source terms are small compared to the hyperbolic components. For small source terms, the wave propagation method provides an exact solution when run with CFL= 1 as long as the speeds in the system are not disparate. This is explored using a model equation system, a dispersive Euler equation system, to simulate quasi-neutral ion cyclotron waves. This system has purely dispersive source terms and the characteristic speeds of the system are not disparate. When the source terms become large, phase errors are noted in the solutions using the wave-propagation method which can be overcome by going to higher grid resolution. When the equation system has disparate speeds like the two-fluid equation system, the wave propagation method experiences diffusive errors since the slower speeds in the fluids are damped out by the time-step that is set based on the fastest speeds and frequencies in the system. Comparisons in Chapter 4 show that the RKDG method is more computationally efficient and provides more accurate solutions for the two-fluid plasma model even at lower grid resolutions particularly when it is run with higher spatial orders.

9.1.6 Benchmark Problems using Two-Fluid Plasma Model

The two-fluid plasma model is benchmarked to previously published results in Chapter 5. A 1-D electromagnetic shock problem is studied for divergence of electric field errors to determine if the solution changes. $\nabla \cdot \mathbf{E}$ errors can be problematic in regions near shocks. A 2-D GEM (Geospace Environment Modeling) challenge magnetic reconnection problem is sim-

ulated and benchmarked to previously published results using the two-fluid plasma model. Following this, a realistic parameter regime is used to simulate magnetic reconnection for a space plasma to compare the ideal two-fluid plasma model to the non-ideal two-fluid model that includes transport coefficients that were derived by Braginskii. This is done to ensure that the regime chosen does not have artificial unphysical resistivity, viscosity, and heat flux. The transport terms are self-consistently computed from the conserved variables at any given time-step. The two-fluid model differs from the Hall-MHD model in allowing electron de-magnetization by including electron inertia and in allowing local non-neutral effects to be captured by including displacement currents. In order to study the regime the fully two-fluid regime, magnetic reconnection is simulated for a current sheet that is initially thinner than the electron skin depth. It is seen that large local electric fields develop in the ideal two-fluid solution in this regime that eventually drive the solution unstable. It is also observed that early in time, the fluids become turbulent and this could result in the large gradients that develop in the electric fields that drive the solution unstable. The inclusion of transport dissipates such terms and allows the solution to remain in a steady-state.

9.1.7 Implementation of Hall-MHD in WARPX and Comparing to Two-Fluid Plasma Model

The two-fluid plasma model is compared to Hall-MHD for an explicit scheme to study the differences in the physics captured and computational effort in Chapter 7. This comparison is performed for a 1-D electromagnetic plasma shock, a 2-D GEM challenge magnetic reconnection, a 2-D axisymmetric Z-pinch two-fluid instability, and for a 2-D axisymmetric Hill's vortex FRC equilibrium. The plasma shock and magnetic reconnection applications benchmark the Hall-MHD implementation in WARPX to previously published results. The Z-pinch small-wavelength instability is captured by both fluid models, however some differences are seen. The Hall-MHD model resolves smaller wavelengths as the grid resolution is increased because it has an unbounded Whistler wave that needs to have a cut-off wave number and often collapses to the grid scale. Inclusion of hyper-resistivity in the ideal Hall-MHD model produces a result similar to the ideal two-fluid model solution. However,

the explicit Hall-MHD model requires at least an order of magnitude larger computational cost and it is more efficient and accurate to use the full two-fluid plasma model. It is unclear whether even an efficient implicit solver for Hall-MHD can out-perform an efficient, implicit or explicit solver for the full two-fluid plasma model. Using artificial ion-to-electron mass ratios and artificial light speed-to-fluid sound speed ratios for the full two-fluid plasma model provides a comparable two-fluid solution that is both accurate and computationally efficient.

9.1.8 Study of Two-Fluid Instabilities in 3-D Z-pinch and 3-D FRC

Two-fluid instabilities are studied in an axisymmetric Z-pinch, a fully 3-D Z-pinch and a 3-D FRC with an initial Hill's vortex profile. The development of the small-wavelength two-fluid instability is studied when only a single wavelength perturbation is applied to the Z-pinch and no perturbation is applied to the FRC. It is noted that the small-wavelength instability grows on top of the single wavelength perturbation. This instability is in the appropriate regime for the lower-hybrid drift instability (LHDI), i.e. $\mathbf{k} \cdot \mathbf{B} = 0$, strong density gradients in the solution, unmagnetized ions with strongly magnetized electrons, and $T_e \sim T_i$. The fastest growing modes for LHDI are $\sim kr_{Le}$ and the longer wavelength modes are $\sim k\sqrt{r_{Le}r_{Li}}$. However, for the instability observed here, the instability modes are $\sim kr_{Li}$. This raises the question as to whether these instabilities are actually LHDI or a low frequency drift instability that manifests itself in a similar parameter regime. Analysis presented in this dissertation suggests that this instability is not LHDI although LHDI could be responsible for initially triggering the instability. The two-fluid drift-turbulence instability observed corresponds to previously published analysis of a similar instability that occurs as a result of reactive coupling between backward propagating fast whistler waves and forward propagating slow ion sound waves[86].

9.1.9 Development of WARPX

The code, WARPX, has been developed to include explicit and implicit methods linking to PETSc for implicit solvers and preconditioners using the DG method. Additionally, it

is now a general geometry code for both the finite volume method and the finite element method in 3-dimensions. The DG algorithm implemented in WARPX as a part of this dissertation uses a generalized implementation for any desired spatial order and produces formally higher order spatial accuracy by accounting for additional coefficients as compared to Ref.[2]. Auxiliary variable implementation using the DG method is included in WARPX to allow arbitrary spatial orders for an arbitrary number of auxiliary variables included with any given equation system. This allows both auxiliary and conserved variables to achieve the same accuracy. Higher order limiters are implemented for DG that are advantageous for problems that do not have strong shocks. WARPX has been generalized and significantly expanded in the course of this dissertation. While it certainly requires further development to become and remain competitive with new computing architectures, WARPX has been developed to a point where it can simulate simple experimental configurations.

9.2 Suggested Future Work

The research presented in this dissertation is certainly not complete and lays the path for future work with the two-fluid plasma model. Future work remains in improving the numerical methods, in including additional physics in the two-fluid plasma model, and in the code development of WARPX.

The axisymmetric and 3-D FRC can be investigated using a numerical Grad-Shafranov MHD solver to provide a more accurate equilibrium initially. Ideally, a true two-fluid FRC equilibrium computed numerically would provide the best results. Once this is done, the two-fluid evolution can be investigated and the development of a toroidal magnetic field can be studied and compared to Hall-MHD. With a better initial equilibrium profile, the development and evolution of the two-fluid drift-turbulence instability can be investigated using a sharper density gradient with a smaller kinetic parameter.

A study of advanced implicit time-integration schemes for strongly non-linear systems is a first step towards improving the present numerical methods. Physics-based preconditioning and p-multigrid methods are some examples of preconditioners that can be used to make the implicit solvers more efficient. Some knowledge of the problem and parameters is necessary in order to fully utilize the advantages of a physics-based preconditioner. The

preconditioner as described in Ref.[43] assumes that the fluid velocities are small. Depending on the problem, the velocities may not be small, but the magnetic fields or electric fields may be small. Since the number of such approximations is limited, several approximations can be hard-coded into WARPX and the choice of the approximation and preconditioner can be specified directly in the input file depending on the problem parameters. This would allow a general implicit implementation with sufficient flexibility to choose the appropriate approximation for the preconditioner depending on the problem being solved.

A number of problems have shocks in the ion fluid but the gradients are relatively smooth in the electron fluid and electric and magnetic fields. Hence, solving a continuous finite element method for the electron fluid and the fields with an implicit implementation might make the solution more linear and this is expected to be easier on the implicit Newton solvers. The ions can be still be solved with an explicit-DG scheme since they contain most of the shocks in the system. A robust implicit implementation will make the two-fluid plasma model very competitive with other fluid models due to its ability to capture additional physics without the time-step restriction brought about by the displacement currents and electron inertia.

Additional improvements in the numerical methods include the development of more robust limiters for DG. Limiters have been a topic of research for DG methods for many years and the development of improved limiters is critical to obtaining accurate solutions especially when using general geometry. Additionally, the implementation of multiple-block grids and unstructured grids constitute major future work and can significantly improve the application of WARPX to real experiments.

In terms of physics, the inclusion of gyro-viscosity with the two-fluid plasma model will make the non-ideal implementation more complete. Gyro-viscosity couples the magnetic field to the fluid sheared flow, and it is expected to include the effects of a magnetic drag in some sense. Gyro-viscosity is expected to make the transport of longitudinal and transverse momentum more accurate.

A 10-moment and better yet, a 13-moment plasma model constitutes future work with the inclusion of anisotropic pressure tensor. This will provide a more complete description of the plasma. This is applicable for problems such as an FRC where the ions are kinetic,

so the ions could be modeled using a 13-moment model while the electrons could still use a 5-moment model. This would be reversed in a plasma sheath where the electrons are highly kinetic. This is presently being investigated by the computational plasma dynamics group at the University of Washington.

WARPX, while a useful fluid plasma code, can use significant improvements to make it competitive with other fluid codes. In addition to improved numerical methods and inclusion of additional physics, the code infrastructure can be modified to use current computing architectures. This is presently being investigated by the computational plasma dynamics group to include OpenCL and expand the platform that WARPX runs on to include, CPUs, GPUs, and other processors. Additionally, a useful feature would be to allow WARPX to internally choose the fluid model appropriate for the regime of interest based on the mean-free-path and the parameters of a given problem.

BIBLIOGRAPHY

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

[2] J. Loverich and U. Shumlak. A discontinuous Galerkin method for the full two-fluid plasma model. *Computer Physics Communications*, 169:251–255, 2005.

[3] J. Loverich and U. Shumlak. Nonlinear full two-fluid study of m=0 sausage instabilities in an axisymmetric Z pinch. *Physics of Plasmas*, 13(082310), 2006.

[4] A. Hakim and U. Shumlak. Two-fluid physics and field-reversed configurations. *Physics of Plasmas*, 14(055911), 2007.

[5] A.G. Kulikoviskii, N.V. Pogorelov, and A.Y. Semenov. *Mathematical Aspects of Numerical Solutions of Hyperbolic Systems*. Chapman and Hall/CRC, 2001.

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

[7] W. Gropp, E. Lusk, D. Ashton, P. Balaji, D. Buntinas, R. Butler, A. Chan, D. Goodell, J. Krishna, G. Mercier, R. Ross, R. Thakur, and B. Toonen. MPICH2 user's guide. Technical Report 1.2.1, Argonne National Laboratory, 2009.

[8] E. Gabriel, G.E. Fagg, G. Bosilca, T. Angskun, J.J. Dongarra, J.M. Squyres, V. Sahay, P. Kambadur, B.Barrett, A. Lumsdaine, R.H. Castain, D.J. Daniel, R.L. Graham, and T.S. Woodall. Open MPI: Goals, concept, and design of a next generation MPI implementation. In *Proceedings, 11th European PVM/MPI Users' Group Meeting*, pages 97–104, September 2004.

[9] The HDF Group. HDF5 user's guide. Technical Report 1.8.4, 2009.

[10] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J.D. Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. LAPACK user guide. Technical Report 3^{rd} Edition, Society for Industrial and Applied Mathematics, 1999.

[11] S. Balay, K. Buschelman, V. Eijkhout, W.D. Gropp, D. Kaushik, M.G. Knepley, L.C. McInnes, B.F. Smith, and H. Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.0.0, Argonne National Laboratory, 2008.

[12] S. Knight. SCons user guide. Technical Report 1.3.0, 2008.

[13] VisIt user's manual. Technical Report 1.5, Lawrence Livermore National Laboratory, 2005.

[14] C.-D. Munz, P. Omnes, R. Schneider, E. Sonnendrücker, and U. Voß. Divergence correction techniques for Maxwell solvers based on a hyperbolic model. *Journal of Computational Physics*, 161(2):484–511, 2000.

[15] A. Dedner, F. Kemm, D. Kröner, C.-D. Munz, T. Schnitzer, and M. Wesenberg. Hyperbolic divergence cleaning for the MHD equations. *Journal of Computational Physics*, 175:645–673, 2002.

[16] R. Fitzpatrick. Scaling of forced magnetic reconnection in the Hall-magnetohydrodynamic Taylor problem. *Physics of Plasmas*, 11(3):937–946, 2004.

[17] S.I. Braginskii. Transport processes in a plasma. *Reviews of Plasma Physics*, 1:205, 1965. Authorized translation from Russian by Herbert Lashinsky, University of Maryland, USA. Edited by M.A. Leontovich.

[18] D. Bale, R. J. LeVeque, S. Mitran, and J. A. Rossmanith. A wave-propagation method for conservation laws and balance laws with spatially varying flux functions. *SIAM Journal of Scientific Computing*, 24:955–978, 2002.

[19] J.O. Langseth and R.J. LeVeque. A wave propagation method for three-dimensional hyperbolic conservation laws. *Journal of Computational Physics*, 165:126–166, 2000.

[20] P. Roe. Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, 43:357–372, 1981.

[21] G. Strang. On the construction and comparison of difference schemes. *SIAM Journal of Numerical Analysis*, 5(3):506–517, 1968.

[22] B. Cockburn, G.E. Karniadakis, and C.-W. Shu. *The Development of Discontinuous Galerkin Methods. In: Discontinuous Galerkin Methods: Theory, Computation and Applications. Lecture notes in Computational Science and Engineering.*, volume 11. Springer, 2000.

[23] W.H. Reed and T.R. Hill. Triangular mesh methods for the neutron transport equation. *Technical Report LA-UR-73-479*, Los Alamos Scientific Laboratory, 1973.

[24] B. Cockburn and C.-W. Shu. TVB runge-Kutta local projection discontinous Galerkin finite element for conservation laws ii. general framework. *Mathematics of Computation*, 52(186):411–435, 1989.

[25] C.-W. Shu. Total-Variation-Diminishing time discretizations. *SIAM Journal of Scientific and Statistical Computing*, 9(6):1073–1084, 1988.

[26] B. Cockburn and C.-W. Shu. Runge-Kutta discontinous Galerkin methods for convection-dominated problems. *Journal of Scientific Computing*, 16:173–261, 2001.

[27] F. Bassi and S. Rebay. A high order accurate discontinuous finite element method for the numerical solution of the compressible Navier-Stokes equations. *Journal of Computational Physics*, 131:267–279, 1997.

[28] J.T. Oden, I. Babuŝka, and C.E. Baumann. A discontinuous *hp* finite element method for diffusion problems. *Journal of Computational Physics*, 146:491–519, 1998.

[29] P. Percell and M.F. Wheeler. A local residual finite element procedure for elliptic equations. *SIAM Journal on Numerical Analysis*, 15(4):705–714, 1978.

[30] L.M. Delves and C.A. Hall. An implicit matching principle for global element calculations. *Journal of the Institute of Mathematical Applications*, 23:223–234, 1979.

[31] J.A. Hendry and L.M. Delves. The global element method applied to a harmonic mixed boundary value problem. *Journal of Computational Physics*, 33:33–44, 1979.

[32] B. Cockburn, F. Li, and C.-W. Shu. Locally divergence-free discontinuous Galerkin methods for the Maxwell equations. *Journal of Computational Physics*, 194(2):588–610, 2004.

[33] J.S. Hesthaven and T. Warburton. High-order nodal discontinuous Galerkin methods for the Maxwell eigenvalue problem. *Royal Society London*, 362:493–524, 2004.

[34] F. Li and C.-W. Shu. Locally divergence-free discontinuous Galerkin methods for MHD equations. *Journal of Scientific Computing*, 22-23:413–442, 2003.

[35] D. Levy, C.-W. Shu, and J. Yan. Local discontinuous Galerkin methods for nonlinear dispersive equations. *Journal of Computational Physics*, 196:751–772, 2004.

[36] M. Zhang and C.-W. Shu. An analysis of and a comparison between the discontinuous galerkin and the spectral finite volume methods. *Computers & Fluids*, 34:581–592, 2005.

[37] R. J. Spiteri and S. J. Ruuth. A new class of optimal high-order strong-stability-preserving time discretization methods. *SIAM Journal of Numerical Analysis*, 40(2):469–491, 2002.

[38] S. Gottlieb, D.I. Ketcheson, and C-W. Shu. High order strong stability preserving time discretizations. *Journal of Scientific Computing*, 38(3):251–289, 2008.

[39] B. Cockburn, S. Hou, and C.-W. Shu. The Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws iv: the multidimensional case. *Mathematics of Computation*, 54:545–581, 1990.

[40] L. Krivodonova. Limiters for high-order discontinuous Galerkin methods. *Journal of Computational Physics*, 226:879–896, 2007.

[41] L. Wang and D. J. Mavriplis. Implicit solution of the unsteady Euler equations for high-order accurate discontinuous Gakerkin discretizations. *AIAA Aerospace Sciences Meeting and Exhibit Proceedings*, AIAA 2006(109), 2006.

[42] L. Wang and D. J. Mavriplis. Implicit solution of the unsteady Euler equations for high-order accurate discontinuous Gakerkin discretizations. *Journal of Computational Physics*, 225:1994–2015, 2007.

[43] L. Chacón. An optimal, parallel, fully implicit Newton-Krylov solver for three-dimensional viscoresistive magnetohydrodynamics. *Physics of Plasmas*, 15(056103), 2008.

[44] K.J. Fidowski, T.A. Oliver, J. Lu, and D.L. Darmofal. p-Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier-Stokes equations. *Journal of Computational Physics*, 207:92–113, 2005.

[45] J. Grandy. Efficient computation of volume of hexahedral cells. Technical Report UCRL-ID-128886, Lawrence Livermore National Laboratory, 2007.

[46] D.A. Calhoun, C. Helzel, and R.J. LeVeque. Logically rectangular grids and finite volume methods for PDEs in circular and spherical domains. *SIAM Review*, 50(4):723–752, 2008.

[47] B. Srinivasan, A. Hakim, and U. Shumlak. Numerical methods for two-fluid dispersive fast MHD phenomena. *Communications in Computational Physics*, Submitted for review 2010.

[48] S. Baboolal. Finite-difference modeling of solitons induced by a density hump in a plasma multi-fluid. *Mathematics and Computers in Simulation*, 55:309–316, 2001.

[49] G. Toth. The $\nabla \cdot \mathbf{B} = 0$ constraint in shock-capturing magnetohydrodynamics codes. *Journal of Computational Physics*, 161:605–652, 2000.

[50] D.S. Balsara. Divergence-free adaptive mesh refinement for magnetohydrodynamics. *Journal of Computational Physics*, 174:614–648, 2001.

[51] K.G. Powell. An approximate Riemann solver for magnetohydrodynamics. *ICASE Report*, (94-24), 1994.

[52] C.R. Evans and J.F. Hawley. Simulation of magnetohydrodynamic flows: a constrained transport method. *Astrophysical Journal*, 332:659–677, 1988.

[53] J.U. Brackbill and D.C. Barnes. The effect of nonzero $\nabla \cdot \mathbf{B}$ on the numerical solution of the magnetohydrodynamic equations. *Journal of Computational Physics*, 35:426–430, 1980.

[54] U. Shumlak and J. Loverich. Approximate Riemann Solver for the Two Fluid Plasma Model. *Journal of Computational Physics*, 187:620–638, 2003.

[55] J. Birn et al. Geospace environmental modeling (GEM) magnetic reconnection challenge. *Journal of Geophysical Research*, 106:3715, 2001.

[56] J. Birn and M. Hesse. Geospace environment modeling (gem) magnetic reconnection challenge: Resistive tearing, anisotropic pressure and hall effects. *Journal of Geophysical Research*, 106(A3):3737, 2001.

[57] A. Otto. Geospace environment modeling (gem) magnetic reconnection challenge: MHD and hall MHD - constant and current dependent resistivity models. *Journal of Geophysical Research*, 106:3751–3757, 2001.

[58] M.A. Shay, J.F. Drake, B.N. Rogers, and R.E. Denton. Alfvénic collisionless magnetic reconnection and the hall term. *Journal of Geophysical Research*, 106(A3):3759–3772, 2001.

[59] Z.W. Ma and A. Bhattacharjee. Hall magnetohydrodynamic reconnection: The geospace environment modeling challenge. *Journal of Geophysical Research*, 106(A3):3773, 2001.

[60] M. Hesse, M. Kuznetsova, and J. Birn. Particle-in-cell simulations of three-dimensional collisionless magnetic reconnection. *Journal of Geophysical Research*, 106:29831–29841, 2001.

[61] P. Pritchett. Geospace environment modeling magnetic reconnection challenge: Simulations with a full particle electromagnetic code. *Journal of Geophysical Research*, 106:3783–3798, 2001.

[62] M. Kuznetsova, M. Hesse, and J. Birn. Collisionless reconnection supported by nongyrotropic pressure effects in hybrid and particle simulations. *Journal of Geophysical Research*, 106:3799–3810, 2001.

[63] E. Priest and T. Forbes. *Magnetic reconnection: MHD theory and applications*. Cambridge University Press, 2000.

[64] A. Zocco, L. Chacon, and A.N. Simakov. Current sheet bifurcation in electron magnetohydrodynamics. *Physics of Plasmas*, 16(110703), 2009.

[65] S. Pope. *Turbulent Flows*. Cambridge University Press, 2000.

[66] D.S. Harned and Z. Mikić. Accurate semi-implicit treatment of the Hall effect in magnetohydrodynamic computations. *Journal of Computational Physics*, 83:1–15, 1989.

[67] J.D. Huba. *"Hall-Magnetohydrodynamics - A Tutorial" in Space Plasma Simulations*. Springer, 2003.

[68] L. Arnold, J. Dreher, and R. Grauer. A semi-implicit Hall-MHD solver using whistler wave preconditioning. *Computer Physics Communications*, 178:553–557, 2008.

[69] L. Chacón and D.A. Knoll. A 2d high-β Hall MHD implicit nonlinear solver. *Journal of Computational Physics*, 188:573–592, 2003.

[70] R. Falgout, A. Baker, V.E. Henson, U.M. Yang, T. Kolev, B. Lee, J. Painter, C. Tong, and P. Vassilevski. Hypre user's manual. Technical Report 2.0.0, Lawrence Livermore National Laboratory, 2006.

[71] D. Hysom and A. Pothen. Euclid user manual. Technical Report 01.a, Lawrence Livermore National Laboratory, 2001.

[72] B. Srinivasan, U. Shumlak, and A. Hakim. Comparisons and applications of two-fluid plasma algorithms. *AIAA Plasmadynamics Conference Proceedings*, 2008.

[73] Y.A. Omelchenko. *Physics of Plasmas*, 7(5):1443–1451, 2000.

[74] H.Y. Guo, A.L. Hoffman, K.E. Miller, and L.C. Steinhauer. Flux conversion and evidence of relaxation in a high-β plasma formed by high-speed injection into a mirror confinement structure. *Physical Review Letters*, 92(24), 2004.

[75] R.C. Davidson and N.T. Gladd. Anomalous transport properties associated with the lower-hybrid-drift instability. *The Physics of Fluids*, 18(10):1327–1335, 1975.

[76] R.C. Davidson, N.T. Gladd, C.S. Wu, and J.D. Huba. Effects of finite plasma beta on the lower-hybrid-drift instability. *The Physics of Fluids*, 20(2):301–310, 1977.

[77] P. H. Yoon, A. T. Y. Lui, and M. I. Sitnov. Generalized lower-hybrid drift instabilities in current-sheet equilibrium. *Physics of Plasmas*, 9(5):1526–1538, 2002.

[78] G. Lapenta and J.U. Brackbill. Nonlinear evolution of lower hybrid drift instability: current sheet thinning and kinking. *Physics of Plasmas*, 9(5):1544–1554, 2002.

[79] J.D. Huba, J.F. Drake, and N.T. Gladd. Lower-hybrid-drift instability in field reversed plasmas. *The Physics of Fluids*, 23(3):552–561, 1980.

[80] A. A. Galeev. *"Magnetospheric tail dynamics" in Magnetospheric Plasma Physics*. Center for Academic Publications, Japan and Reidel Publishing Company, London, 1982. Edited by A. Nishida.

[81] W. Daughton. Electromagnetic properties of the lower-hybrid drift instability in a thin current sheet. *Physics of Plasmas*, 10(8):3103–3119, 2003.

[82] D. Biskamp. *Magnetic Reconnection in Plasmas*. Cambridge University Press, 2000.

[83] J.D. Huba. Hall dynamics of the Kelvin-Helmholtz instability. *Physical Review letters*, 72(13):2033–2036, 1994.

[84] A.B. Mikhailovskii and L.I. Rudakov. The stability of a spatially inhomogeneous plasma in a magnetic field. *Soviet Physics JETP*, 44:912–918, 1963.

[85] W. Daughton. *Journal of Geophysical Research*, 104(A12):701–707, 1999.

[86] H. Ji, R. Kulsrud, W. Fox, and M. Yamada. *Journal of Geophysical Research*, 110(A08212), 2005.

[87] R. Kulsrud, H. Ji, W. Fox, and M. Yamada. *Physics of Plasmas*, 12(082301), 2005.

[88] J.P. Friedberg and R.A. Gerwin. Lower hybrid drift instability at low drift velocities. *The Physics of Fluids*, 20(8):1311–1315, 1977.

[89] H.Y. Guo, A.L. Hoffman, and R.D. Milroy. Rotating magnetic field current drive of high-temperature field reversed configurations with high ζ scaling. *Physics of Plasmas*, 14(112502), 2007.

[90] N.A. Krall and P.C. Liewer. Low-frequency instabilities in magnetic pulses. *Physical Review A*, 4:2094–2103, 1971.

[91] R.C. Davidson. *Methods in Nonlinear Plasma Theory.* Academic Press, 1972.

[92] Jeffrey P. Friedberg. *Ideal Magnetohydrodynamics.* Plenum Press, 1987.

[93] P. Jamet. Galerkin-type approximations which are discontinuous in time for parabolic equations in a variable domain. *SIAM Journal on Numerical Analysis*, 15:912–928, 1978.

[94] L.C. Steinhauer, H. Yamada, and A. Ishida. Two-fluid flowing equilibria of compact plasmas. *Physics of Plasmas*, 8(9):4053–4061, 2001.

[95] J. Qiu and C.-W. Shu. Runge-Kutta discontinuous Galerkin method using WENO limiters. *SIAM Journal on Scientific Computing*, 26:907–929, 2005.

[96] R. W. Motley and N.D'Angelo. Excitation of electrostatic plasma oscillations near the ion cyclotron frequency. *Physics of Fluids*, 6(2):296–299, 1963.

[97] T.R. Fogarty and R.J. LeVeque. High-resolution finite-volume methods for acoustic waves in periodic or random media. *Journal of Acoustical Society of America*, 106(1):17–28, 1999.

[98] Y. Suzuki and B. van Leer. A discontinuous Galerkin method with Hancock-type time integration for hyperbolic systems with stiff relaxation source terms.

[99] H. T. Huynh. An upwind moment scheme for conservation laws. *ICCFD 3, Toronto, Canada*, 2004.

[100] D. Biskamp, E. Schwarz, and J.F. Drake. Ion-controlled collisionless magnetic reconnection. *Physical Review Letters*, 75:3850–3853, 1995.

[101] L. Rastätter, M. Hesse, and K. Schindler. Hall-mhd modeling of near-Earth magnetotail current sheet thinning and evolution. *Journal of Geophysical Research*, 104(A6):12301–12311, 1999.

[102] E. Ahedo, P. Martínez-Cerezo, and M. Martínez-Sánchez. One-dimensional model of the plasma flow in a Hall thruster. *Physics of Plasmas*, 8(6):3058–3068, 2001.

[103] A. Cohen-Zur, A. Fruchtman, J. Ashkenazy, and A. Gany. Analysis of the steady-state axial flow in a Hall thruster. *Phyics of Plasmas*, 9(10):4363–4374, 2002.

[104] R. Winglee, T. Ziemba, L. Giersch, J. Prager, J. Carscadden, and B.R. Roberson. Simulation and laboratory validation of magnetic nozzle effects for the high power helicon thruster. *Phyics of Plasmas*, 14(063501), 2007.

[105] J.D. Huba, J.G. Lyon, and A.B. Hassam. Theory and simulation of the Rayleigh-Taylor instability in the limit of large Larmor radius. *Physical Review letters*, 59(26):2971–2974, 1987.

[106] E.V. Belova, R.C. Davidson, H. Ji, and M. Yamada. Kinetic effects on the stability properties of field-reversed configurations. I. Linear stability. *Physics of Plasmas*, 10(6):2361–2371, 2003.

[107] A. Hakim. Extended MHD modelling with the ten-moment equations. *Journal of Fusion Energy*, 27:36–43, 2008.

[108] B. Srinivasan and U. Shumlak. A semi-implicit, ideal, full two-fluid plasma model. To be submitted. 2010.

[109] S. Baboolal and R. Bharuthram. Two-scale numerical solution of the electromagnetic two-fluid plasma-maxwell equations: shock and soliton simulation. *Mathematics and Computers in Simulation*, 76:3–7, 2007.

[110] N.A. Krall. Low-frequency stability for field reversed configuration parameters. *Physics of Fluids*, 3:878–883, 1987.

Appendix A

BRAGINSKII'S TRANSPORT COEFFICIENTS IMPLEMENTED IN WARPX

The non-ideal two-fluid equations[17] are

$$\frac{\partial \rho_e}{\partial t} + \nabla \cdot (\rho_e \mathbf{u}_e) = 0 \tag{A.1}$$

$$\frac{\partial \rho_e \mathbf{u}_e}{\partial t} + \nabla \cdot (\rho_e \mathbf{u}_e \mathbf{u}_e + p_e \mathbf{I}) = \frac{\rho_e q_e}{m_e}(\mathbf{E} + \mathbf{u}_e \times \mathbf{B}) - \nabla \cdot \overleftrightarrow{\mathbf{\Pi}} + \mathbf{R}_{ei} \tag{A.2}$$

$$\frac{\partial \epsilon_e}{\partial t} + \nabla \cdot ((\epsilon_e + p_e)\mathbf{u}_e) = \frac{\rho_e q_e}{m_e}\mathbf{u}_e \cdot \mathbf{E} - \nabla \cdot \mathbf{q}_e - \overleftrightarrow{\mathbf{\Pi}}_e : \nabla \mathbf{u}_e + Q_e \tag{A.3}$$

with subscript e for electrons and

$$\frac{\partial \rho_i}{\partial t} + \nabla \cdot (\rho_i \mathbf{u}_i) = 0 \tag{A.4}$$

$$\frac{\partial \rho_i \mathbf{u}_i}{\partial t} + \nabla \cdot (\rho_i \mathbf{u}_i \mathbf{u}_i + p_i \mathbf{I}) = \frac{\rho_i q_i}{m_i}(\mathbf{E} + \mathbf{u}_i \times \mathbf{B}) - \nabla \cdot \overleftrightarrow{\mathbf{\Pi}}_i - \mathbf{R}_{ei} \tag{A.5}$$

$$\frac{\partial \epsilon_i}{\partial t} + \nabla \cdot ((\epsilon_i + p_i)\mathbf{u}_i) = \frac{\rho_i q_i}{m_i}\mathbf{u}_i \cdot \mathbf{E} - \nabla \cdot \mathbf{q}_i - \overleftrightarrow{\mathbf{\Pi}}_i : \nabla \mathbf{u}_i + Q_i \tag{A.6}$$

$$\tag{A.7}$$

with subscript i for ions. Here, $\mathbf{R}_{ei}$ is the momentum transfer term, $\overleftrightarrow{\mathbf{\Pi}}$ is the viscous stress tensor, $\mathbf{q}$ is the heat flux, and Q is the thermal equilibration between the species. The $\overleftrightarrow{\mathbf{\Pi}} : \nabla \mathbf{u}$ term represents the heat generated due to viscosity. In this dissertation, gyroviscosity is neglected for simplicity therefore, an absence of magnetic field is assumed for computing the viscous stress tensor. Each of the transport terms are elaborated in the following sections and use the same formulas and notations are detailed in Ref.[17]. All coefficients used are for $Z = 1$.

A.1 Momentum Transfer

The momentum transfer term accounts for the inter-species transfer of friction and thermal forces. It is given by

$$\mathbf{R}_{ei} = \mathbf{R}_U + \mathbf{R}_T \tag{A.8}$$

where R_U is the friction force and R_T is the thermal force. The friction force is defined as

$$\mathbf{R}_U = -\alpha_{\parallel}\mathbf{u}_{\parallel} - \alpha_{\perp}\mathbf{u}_{\perp} - \alpha_{\wedge}\mathbf{u}_{\wedge} \tag{A.9}$$

where subscript $\parallel$ refers to parallel to the $\mathbf{B}$ field, subscript $\perp$ refers to perpendicular to the $\mathbf{B}$ field, and subscript $\wedge$ refers to the direction that is perpendicular to both the $\mathbf{B}$ field and the vector quantity, in this case $\mathbf{u}$. $\mathbf{u}$ is the relative velocity given by $\mathbf{u} = \mathbf{u}_e - \mathbf{u}_i$ and its components are defined as

$$\mathbf{u}_{\parallel} = \frac{\mathbf{B}}{|B|}\left(\mathbf{u}\cdot\frac{\mathbf{B}}{|B|}\right) \tag{A.10}$$

$$\mathbf{u}_{\perp} = \mathbf{u} - \mathbf{u}_{\parallel} \tag{A.11}$$

$$\mathbf{u}_{\wedge} = \frac{\mathbf{B}}{|B|}\times\mathbf{u}. \tag{A.12}$$

The friction force coefficients, basically electrical resistivity, are described as

$$\alpha_{\parallel} = \rho_e\nu_e\alpha_0 \tag{A.13}$$

$$\alpha_{\perp} = \rho_e\nu_e\frac{1-\alpha_1' x^2+\alpha_0'}{\Delta} \tag{A.14}$$

$$\alpha_{\wedge} = \rho_e\nu_e x\frac{\alpha_1'' x^2+\alpha_0''}{\Delta} \tag{A.15}$$

where ρ_e is the electron mass density, ν_e is the electron collision frequency, $x = \omega_{ce}/\nu_e$, $\Delta = x^4 + \delta_1 x^2 + \delta_0$. The coefficients chosen for $Z = 1$ are $\alpha_0 = 0.5129$, $\alpha_1' = 6.416$, $\alpha_0' = 1.837$, $\alpha_1'' = 1.704$, $\alpha_0'' = 0.7796$, $\delta_0 = 3.7703$, and $\delta_1 = 14.69$. The collision frequency

is defined as

$$\nu_e = \frac{n_e q_e^4 \ln \Lambda}{12\epsilon_0^2 m_e^{1/2} (\pi k T_e)^{3/2}} \tag{A.16}$$

where n_e is the electron number density, q_e is the electron charge, T_e is the electron temperature in K, $\ln \Lambda = \ln 12\pi n_e \lambda_D^3$, and λ_D is the species Debye length. Often, a $\ln \Lambda = 10$ is chosen which accounts for most plasma configurations.

The thermal force term is defined as

$$\mathbf{R}_T = -\beta_{\parallel}^{UT} \nabla_{\parallel} T_e - \beta_{\perp}^{UT} \nabla_{\perp} T_e - \beta_{\wedge}^{UT} \nabla_{\wedge} T_e \tag{A.17}$$

The thermal force coefficients are described as

$$\beta_{\parallel}^{UT} = n_e \beta_0 \tag{A.18}$$

$$\beta_{\perp}^{UT} = n_e \frac{1 - \beta_1' x^2 + \beta_0'}{\Delta} \tag{A.19}$$

$$\beta_{\wedge}^{UT} = n_e x \frac{\beta_1'' x^2 + \beta_0''}{\Delta} \tag{A.20}$$

where the coefficients chosen for $Z = 1$ are $\beta_0 = 0.7110$, $\beta_1' = 5.101$, $\beta_0' = 2.681$, $\beta_1'' = 3/2$, and $\beta_0'' = 3.053$ with all others as previously defined. The $\parallel$, $\perp$, and $\wedge$ components of ∇T_e are obtained in the same manner as Eqs.(A.10-A.12). The equation of state, $P_e = n_e k T_e$ is used to obtain the electron temperature from the continuity and energy equations.

A.2 Viscous Stress Tensor

The viscous stress tensor accounts for the viscosity within each species. Ignoring gyroviscosity, the viscous stress tensor used assumes an absence of magnetic field. This viscous stress tensor is defined for the electrons as

$$\overleftrightarrow{\mathbf{\Pi}}_e = -\eta_0 \overleftrightarrow{\mathbf{W}}_e \tag{A.21}$$

where the viscosity coefficient is $\eta_0 = 0.73P_e/\nu_e$, and

$$\overleftrightarrow{\mathbf{W}}_e = \nabla_\alpha \mathbf{u}_{e\beta} + \nabla_\beta \mathbf{u}_{e\alpha} - \frac{2}{3}\delta_{\alpha\beta}\nabla \cdot \mathbf{u}_e. \tag{A.22}$$

Here the second term on the right-hand-side is basically the transpose of the first right-hand-side term, the velocity gradient. The $\delta_{\alpha\beta}$ ensures that the divergence of the velocity is applied only for the diagonal components of the velocity gradient tensor. The resulting viscous stress tensor matrix is a symmetric 3×3 matrix. The ion viscous stress tensor is developed in a similar manner with

$$\overleftrightarrow{\mathbf{\Pi}}_i = -\eta_0 \overleftrightarrow{\mathbf{W}}_i \tag{A.23}$$

where the viscosity coefficient is $\eta_0 = 0.96P_i/\nu_i$, and

$$\overleftrightarrow{\mathbf{W}}_i = \nabla_\alpha \mathbf{u}_{i\beta} + \nabla_\beta \mathbf{u}_{i\alpha} - \frac{2}{3}\delta_{\alpha\beta}\nabla \cdot \mathbf{u}_i. \tag{A.24}$$

The collision frequency for the ions, ν_i, is defined in the same manner as the electrons.

A.3 Viscous Heating

The viscous heating term in the energy equation accounts for the heat generated to due viscosity within each species. It is given by

$$\overleftrightarrow{\mathbf{\Pi}}_e : \nabla \mathbf{u}_e = \sum_m \sum_n \Pi_e[m][n] \nabla u_e[n][m] \tag{A.25}$$

$$\overleftrightarrow{\mathbf{\Pi}}_i : \nabla \mathbf{u}_i = \sum_m \sum_n \Pi_i[m][n] \nabla u_i[n][m] \tag{A.26}$$

where both tensors are symmetric due to neglecting gyroviscosity.

A.4 Heat Flux

The electron and ion heat flux terms are included to account for thermal conduction within each species. The electron heat flux given by

$$\mathbf{q}_e = \mathbf{q}_U^e + \mathbf{q}_T^e \tag{A.27}$$

where $\mathbf{q}_U^e$ results from the relative velocity due to the presence of the thermal forces. $\mathbf{q}_T^e$ refers to the electron heat flux due to thermal conduction. $\mathbf{q}_U^e$ is given by

$$\mathbf{q}_U^e = \beta_\parallel^{TU}\mathbf{u}_\parallel + \beta_\perp^{TU}\mathbf{u}_\perp + \beta_\wedge^{TU}\mathbf{u}_\wedge \tag{A.28}$$

where the coefficients are defined as

$$\beta_\parallel^{TU} = \beta_\parallel^{UT}T_e \tag{A.29}$$

$$\beta_\perp^{TU} = \beta_\perp^{UT}T_e \tag{A.30}$$

$$\beta_\wedge^{TU} = \beta_\wedge^{UT}T_e \tag{A.31}$$

with $\mathbf{u}_\parallel$, $\mathbf{u}_\perp$, and $\mathbf{u}_\wedge$ defined in Eqs.(A.10-A.12).

The thermal conduction heat flux term is given by

$$\mathbf{q}_T^e = -\chi_\parallel\nabla_\parallel T_e - \chi_\perp\nabla_\perp T_e - \chi_\wedge\nabla_\wedge T_e \tag{A.32}$$

where the thermal conductivities are defined as

$$\chi_\parallel = \frac{P_e}{m_e\nu_e}\gamma_0 \tag{A.33}$$

$$\chi_\perp = \frac{P_e}{m_e\nu_e}\frac{\gamma_1' x^2 + \gamma_0'}{\Delta} \tag{A.34}$$

$$\chi_\wedge = \frac{P_e}{m_e\nu_e}x\frac{\gamma_1'' x^2 + \gamma_0''}{\Delta} \tag{A.35}$$

where $\gamma_0 = 3.1616$, $\gamma_1' = 4.664$, $\gamma_0' = 11.92$, $\gamma_1'' = 5/2$, and $\gamma_0'' = 21.67$.

The ion heat flux contains only the thermal conduction term because the heat flux due

to the relative velocity is assumed negligible for the massive ions. The ion heat flux is given by

$$\mathbf{q}_i = -\chi_{\parallel}\nabla_{\parallel}T_i - \chi_{\perp}\nabla_{\perp}T_i + \chi_{\wedge}\nabla_{\wedge}T_i \tag{A.36}$$

where the thermal conductivities are defined as

$$\chi_{\parallel} = 3.906\frac{P_i}{m_i\nu_i} \tag{A.37}$$

$$\chi_{\perp} = \frac{P_i}{m_i\nu_i}\frac{2x^2 + 2.645}{\Delta} \tag{A.38}$$

$$\chi_{\wedge} = \frac{P_i}{m_i\nu_i}x\frac{\frac{5}{2}x^2 + 4.65}{\Delta} \tag{A.39}$$

where $x = \omega_{ci}/\nu_i$, $\Delta = x^4 + 2.70x^2 + 0.677$.

A.5 Thermal Equilibration

The heat generated due to collisions of electrons with ions results in a thermal equilibration between the two species. The thermal equilibration is defined as

$$Q_{\Delta} = 3\frac{m_e}{m_i}n_e\nu_e k(T_e - T_i). \tag{A.40}$$

For ions, $Q_i = Q_{\Delta}$ and the heat due to the friction and thermal forces can be neglected since ions are assumed to be massive. For electrons however,

$$Q_e = -\mathbf{R}\cdot\mathbf{u} - Q_{\Delta} \tag{A.41}$$

$$= -\mathbf{R}_U\cdot\mathbf{u} - \mathbf{R}_T\cdot\mathbf{u} - Q_{\Delta} \tag{A.42}$$

where $\mathbf{R}_U$ is the friction force, $\mathbf{R}_T$ is the thermal force, and $\mathbf{u}$ is the relative velocity.

Appendix B

CODE STRUCTURE OF WARPX

WARPX is a 3-dimensional finite volume and finite element general geometry code. It was developed at the University of Washington and is hosted with the Plasma Science and Innovation (PSI) Center. It can be obtained by going to `http://psicenter.org/warpx/main.cgi` and requesting a copy of the code. Directions for generating an ssh key to submit with the download request are provided in this website. Once approved, instructions for downloading the code using subversion will be sent.

This appendix describes the general code and algorithm structure of WARPX. This hierarchy is well suited for any numerical algorithm and any equation system that a user desires to simulate with special emphasis for hyperbolic systems. The `trunk` directory in WARPX has a folder named `src` which contains all the source code for WARPX. Under `src`, the code is distributed as described by the flow chart in Fig. B.1. Since the SCons tool is used to generate the object files and the executable, all `.cc` files must be included in the `SConscript` file for the folder that they are in. The `src` directory contains an `SConstruct` file that serves to link all the `SConscript` files.

The `lib` directory contains all the software engineering aspects of WARPX. The array structure and parallel array structures (`pararrays`) are defined here. The `gridrange` and `gridbox` are defined with `splitbox` splitting up the grid among the specified number of processors. The algorithm used to divide the grid among processors tries to maintain an equal (or close to equal) volume-to-area ratio of the grid among all processors. The calls to MPI and HDF5 routines are performed in this directory. An indexer is defined that keeps track of the indices of the variables within each grid cell. A sequencer is defined to loop over the local (for each processor) or global (over the entire) domain independent of the number of dimensions in the system. The sequencer also allows a general implementation of boundary conditions by looping over desired layers of the domain for each local or global

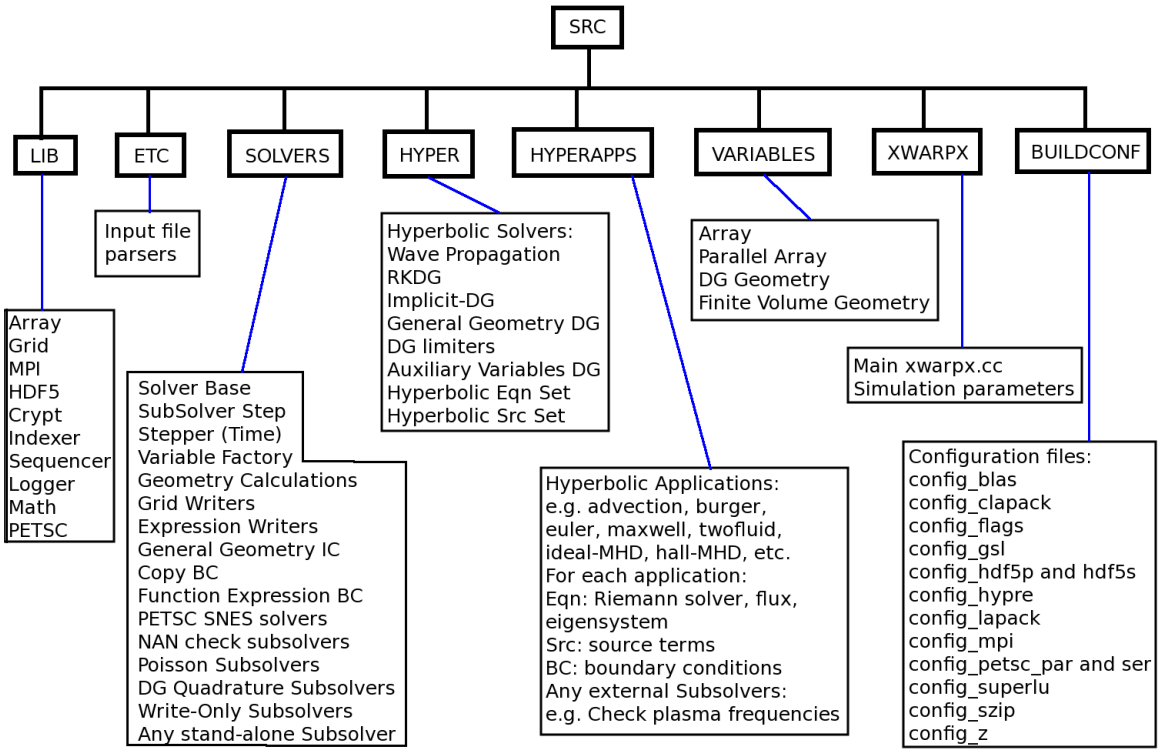

Figure B.1: Flowchart indicating the general code structure and organization of WARPX.

domain. A logger is defined to display a log stream depending on user specifications in the input file. Warnings, errors, critical messages, and non-critical messages are all passed through the logger. The expression parser reads the input files and uses input file parsers defined in the `etc` directory.

The `solvers` directory contains the infrastructure for the various solvers and subsolvers implemented in WARPX. The `wxsolver` base class deals with initializing the object, getting the grid, getting the variables, setting up the subsolvers, reading and writing data, getting the time-step, and advancing the solution. In setting up the subsolvers, the solver base class ensures that it implements start-only, per time-step, end write-only subsolvers according to user specifications in the input file. The `wxsubsolver` base class deals with defining the subsolvers that derive from solvers and contain the numerical algorithms. Examples of subsolvers include any finite difference algorithm, initial conditions, geometry calculations, boundary conditions, expression writers, grid writers, DG quadrature calculations, PETSc SNES solvers, NAN detection subsolvers, etc. All hyperbolic equation solvers are also of type subsolver but are in the `hyper` directory. Any subsolver that is not a hyperbolic equation solver such as Poisson subsolvers are housed in the `solvers` directory. All WARPX subsolvers need to be registered so that the user-specified options in the input file are appropriately linked to during compilation and run-time. In the `solvers` directory this registration is done under `wxsubsolver_register`.

The `hyper` directory specifically contains hyperbolic equation subsolvers. These subsolvers include the wave propagation algorithm, the RKDG algorithm, the Implicit-DG algorithm, general geometry algorithm using DG and wave propagation, limiters for DG, and auxiliary variable subsolvers for DG to name some major ones. The `wxhyperboliceqnset` and `wxhyperbolicsrcset` contain the information that ties each of the numerical algorithms to each of the hyperbolic applications such as Riemann problem computations, fluxes, sources, eigensystems, etc. The hyperbolic subsolver algorithms are written in a general manner such that they are applicable to any hyperbolic equation system. All hyperbolic subsolvers that are created are registered so that the input file specifications appropriately link to the code.

The `hyperapps` directory contains hyperbolic equation systems. Examples of these are

advection equation, Burger's equation, Euler equations, Maxwell's equations, ideal-MHD, Hall-MHD, Two-fluid equations, Ten-moment equations, etc. For each application, an equation file is defined that contains the number of equations and waves in the system, the Riemann problem computations, flux calculations, and eigensystem computations. The source terms are defined separately to allow for flexibility to use the same equation system with and without sources. All equation systems and sources that are created contain a registration file similar to the subsolvers to appropriately link to specifications in the input file. Equation specific boundary conditions (or other subsolvers) such as wall boundaries or axis boundaries are defined within the specific application folder in the `hyperapps` directory and are appropriately registered. Auxiliary variables such as electron currents and electric fields for Hall-MHD are also defined in the specific application folder using the same framework as the hyperbolic equations and sources.

The `variables` directory contains array structures for serial and parallel arrays. The geometry data for finite volume methods and finite element methods are also defined as variables. If particles were to the added to WARPX, they would be included as variables in this directory. The `xwarpx` directory contains the main `xwarpx.cc` file and the simulation parameters. The `buildconf` directory contains all the configuration files to link the external dependencies to WARPX. For configurations that are specific to a computer and are not defined in `buildconf`, a `config.py` file is created and the external dependencies can be specified there (e.g. `mpi='/usr/local/mpich2'`). WARPX is then compiled using desired flags like `scons debug=no parallel=yes`. A `build` directory is created where all the object files and executable are stored.

The main `trunk` directory has a `regtests` directory that contains regression tests. These tests allow any major changes in WARPX to be benchmarked to previous results. The test input files can be included under the `tests` folder and they are specified under `xregtests.inp`. From the `regtests` directory, `./scripts/xregtests.py --help` provides the options for the regression tests. A set of accepted results can be created and stored as benchmark cases to compare results from future changes in WARPX.

Also in the `trunk` directory is a `scripts` directory that contains python scripts for preprocessing input files and post-processing data. For input files described in Appendix C, a

pre-processor is required to convert the `*.pin` file to a WARPX compatible `*.inp` file. This is done using `wxinpparse.py filename.pin`. For python post-processing, `wxdata.py` is used to read the WARPX parallel arrays from the HDF5 data files. `import wxdata` is used from within Python followed by `d = wxdata.WxData(filename, frame)` where filename is specified without the extension and frame is the HDF5 data file number. For the wave propagation method, `q = d.read('variable')` is used to read in the desired variable. For the DG method, this reads in only the average values within each cell. In order to get a reconstruction of the high order coefficients, `q = d.readDG('variable',meqn)` is used where `meqn` is the number of variables in the equation system. A high resolution projection is specified using `q = d.readDG('variable',meqn,nq)` where `nq` is the number of additional quadrature points per cell that the user desires for higher plotting resolution. Quick 1- and 2-dimensional plots in python are generated with the `wxplot.py` script where `wxplot.py -h` shows the options.

For 3-dimensional visualization, 2-D cross-sectional plots can be generated using the python scripts. Additionally, VisIt has been extensively used for 3-D post-processing of WARPX data especially using the DG algorithm as the python scripts are time-consuming with high resolution 3-D data. To use VisIt, a write-only subsolver needs to be defined in the input file that projects each of the variables in the equation system onto quadrature locations from within C++ at the end of each time-step. Examples of this are provided in Appendix C. Once this is done for the variables and the grid, the `wxxdmf.py` script is used to generate XMF files for each of the HDF5 data files. `wxxdmf.py -h` shows the options available for generating XMF files. For a general geometry grid, `wxxdmfgg.py` is used to generate the XMF files that are then opened in VisIt.

Appendix C

WARPX INPUT FILES

Some sample WARPX input files are provided in this chapter. Section C.1 shows modules of a sample input file using the RKDG method with the two-fluid plasma model. Macros are employed in writing the XML-based format input files such that modules that are repeated can be included using a cleaner implementation. The Hall-MHD input file using the RKDG method with auxiliary variables implemented is very similar to the auxiliary variable implementation of the Braginskii coefficients using the full two-fluid plasma model. Section C.2 shows modules of a sample implicit DG input file that uses the PETSc routines for a Newton-based iterative scheme. Section C.3 shows modules of a sample general geometry DG input file using a functional definition for the vertices of the desired geometry.

The input files described in this appendix apply macros and require pre-processing using the `wxinpparse.py` script to generate a WARPX compatible `*.inp` input file.

C.1 RKDG Input File for Collisional Two-Fluid Plasma Model

A sample input file is provided for a 2-D two-fluid simulation using Braginskii transport coefficients. The `"..."` in the input file is used to highlight lengthy descriptions that are truncated here. The sample input file is well-commented but a brief summary is provided here. The simulations parameters are defined in the beginning of the file and these are read in by the macro which is included in the input file using `#include filename.pin`. The `PAR_ARRAY` call to the macro makes the memory allocation for all array variables that live in each cell of the grid. The initial condition is specified using an expression subsolver. A check frequency subsolver is used to compute the plasma and cyclotron frequencies, which appear in the source terms, to determine the most restrictive time-step. A NAN check subsolver is included because the simulations often run without detecting such errors if they are not caught early in time.

The auxiliary variables are then computed using auxiliary equation solvers. Each of the auxiliary variables is limited in the same manner as the conserved variables. Once the auxiliary variables are computed, they are passed in to the `DGRHS` subsolver to advance the solution in time. The order in which the auxiliary variables are specified here is important as they are used in the auxiliary and conserved equation systems assuming a specific order. Once the right-hand-side is computed for the conserved variables, a Runge-Kutta stage 1 time advance is performed on the conserved variables. Limiters are then applied to the conserved intermediate variable `q1`. The above steps are then repeated for a Runge-Kutta stage 2 time advance and likewise if stage 3 is used. Prior to each Runge-Kutta stage, boundary conditions are applied to the conserved and auxiliary variables. At the end of each time-step, the variable `qnew` is copied to `qold`. The `SUBSOLVERSTEP` macro determines the list of subsolvers that are computed for a given step. The `SolverSequence` specifies the start-only subsolver-steps, the per time-step list of subsolver-steps in the required order, and the write-only or end-only list of subsolver-steps. All subsolvers, hyperbolic equations, and hyperbolic sources specify the registered name in the `Kind` field. The registration is briefly described in Appendix B.

An input file for Hall-MHD is implemented in much the same way as the two-fluid plasma model. In the two-fluid model, Braginskii terms are included as auxiliary variables and require auxiliary equations and solvers to be specified. For Hall-MHD, the electron currents computed from a reduced Ampere's law and the electric fields computed from generalized Ohm's law are the auxiliary variables. These auxiliary variables call the appropriate auxiliary equation and auxiliary sources by passing in the conserved variables. Limiters and boundary conditions are applied in the same manner described previously. The conserved variables are then updated using the auxiliary variables and are advanced in time using the desired Runge-Kutta order. WARPX contains a Hall-MHD implementation where the auxiliary variables are computed using a standard finite differencing scheme. While this implementation is easy and uses less computational effort it does not deliver the same accuracy as the auxiliary equation solver due to the absence of higher order auxiliary variable coefficients. A DG implementation of the auxiliary equations provides a better solution.

```
# -*- python -*-
# specify simulation parameters e.g.
import math
PI = math.pi

MI = 2.0*1.67e-27
ME = MI/25.0
...

# for macros used in this file
#include "tfmacro_braginskii_recon_separatesrc.pin"

<warpx>

 ##
 # Basic parameters
 ##
 Simulation = recon
 # one of debug, info, warning, error,
 #        critical, disabled
 Verbosity = info
 GlobalVerbosity = disabled

 <recon>
   Type = WxSolver
   Kind = comboSolver

   Time = [0.0, TEND] # start, end times
   Out = NOUT # no of output files to write
   Dt = 0.1 # initial dt to try

##
# Grid defintion
##
   <grid>
     Type = WxGridBox

     Lower = [XL, YL] # lower coordinates
     Upper = [XU, YU] # upper coordinates
     Cells = [XCELLS, YCELLS] # no of cells

     PeriodicDirs = [0] # X is periodic

   </grid>

##
# Array definitions
##
   <[PAR_ARRAY("qold", "grid", MEQN*SP_ORDER*SP_ORDER)]>
   <[PAR_ARRAY("qnew", "grid", MEQN*SP_ORDER*SP_ORDER)]>
   <[PAR_ARRAY("q1", "grid", MEQN*SP_ORDER*SP_ORDER)]>
   <[PAR_ARRAY("dq", "grid", MEQN*SP_ORDER*SP_ORDER)]>

   # aux var array definitions (first elc then ion)
   # grad Te, grad Ti, grad ue
   # grad ui, div ue, div ui
   # heat flux x,y,z, for electrons and ions
   # viscous stress tensor for electrons and ions
   <[PAR_ARRAY("gtAux", "grid", 6*SP_ORDER*SP_ORDER)]>
   <[PAR_ARRAY("guAux", "grid", 18*SP_ORDER*SP_ORDER)]>
   <[PAR_ARRAY("duAux", "grid", 2*SP_ORDER*SP_ORDER)]>
   <[PAR_ARRAY("qhfAux", "grid", 6*SP_ORDER*SP_ORDER)]>
   <[PAR_ARRAY("piAux", "grid", 18*SP_ORDER*SP_ORDER)]>

##
# Subsolver definitions
##
   # Initialize qold,qnew
   <initialcond>
     Type = WxSubSolver
     Kind = funcDGArraySetter
     WriteVars = [qold, qnew]
     OnGrid = grid
     spatialOrder = SP_ORDER
     meqn = MEQN

     <func>
       Type = WxFunction
       Kind = exprFunc

       # specify parameters to be used in progn

       progn = [ \
           "bx0 = b0*tanh(y/dlambda)", \
           ...
           "Ei = 5.0*b0*b0*rhoi/(mi*mu0*12.*gas_gamma1*n0)"]

       exprList = ["rhoe", \
                   ...
                   "bx", \
                   "by", \
                   "0.0", \
                   "0.0", \
                   "0.0" ]

     </func>

   </initialcond>

   # use q to see if wpe, wpi, wce, wci are resolved
   <chk_freq>
     Type = WxSubSolver
     Kind = checkFreqTimeStep

     charge = Q
     ionmass = MI
     elcmass = ME
     epsilon0 = EPS0

     isTransport = true
     boltz = BOLTZ
     gas_gamma = GAMMA
     dx = (YU-YL)/YCELLS

     OnGrid = grid

     ReadVars = [qold]

   </chk_freq>

    # Check to see if NANs exist in qold
   <nancheck>
     Type = WxSubSolver
     Kind = isNANCheck

     OnGrid = grid
```

```
  ReadVars = [qold]
</nancheck>

# use qold to get gt aux vars for elc and ion
# (temp gradient vectors)
<[GTAUX("gtAux_qold", "gtAux", "qold", "gtAux")]>

# apply limiter to gtAux
<[GTLIMIT("gtAux_limit", "component", "gtAux")]>

# use qold to get gu aux vars for elc and ion
# (velocity gradient tensors)
<[GUAUX("guAux_qold", "guAux", "qold", "guAux")]>

# apply limiter to guAux
<[GULIMIT("guAux_limit", "component", "guAux")]>

# use qold to get du aux vars for elc and ion
# (divergence velocity scalars)
<[DUAUX("duAux_qold", "duAux", "qold", "duAux")]>

# apply limiter to duAux
<[DULIMIT("duAux_limit", "component", "duAux")]>

# use qold to get qhf aux vars for elc and ion
# (heat flux vectors)
<[QHFAUX("qhfAux_qold", "qhfAux", "qold", "gtAux",
           "qhfAux")]>

# apply limiter to qhfAux
<[QHFLIMIT("qhfAux_limit", "component", "qhfAux")]>

# use qold to get pi aux vars for elc and ion
# (viscosity tensors)
<[PIAUX("piAux_qold", "piAux", "qold", "guAux",
          "duAux", "piAux")]>

# apply limiter to piAux
<[PILIMIT("piAux_limit", "component", "piAux")]>

# use qold to compute dq
<[DGRHS("dgrhs_qold", "qold", "gtAux", "qhfAux",
          "piAux", "guAux", "dq")]>

# use dq and qold to compute q1
<rk_stage1>
  Type = WxSubSolver
  Kind = linearCombiner

  OnGrid = grid

  ReadVars = [qold, dq]
  coeffs   = [1.0, 1.0]
  WriteVars = [q1]

</rk_stage1>

# subsolver to apply limiter to q1
<[DGLIMIT("q1_limit", "characteristic", "q1")]>

# use q1 to get gt aux vars for elc and ion
# (temp gradient vectors)
<[GTAUX("gtAux_q1", "gtAux", "q1", "gtAux")]>

# use q1 to get gu aux vars for elc and ion
# (velocity gradient tensors)
<[GUAUX("guAux_q1", "guAux", "q1", "guAux")]>

# use q1 to get du aux vars for elc and ion
# (divergence velocity scalars)
<[DUAUX("duAux_q1", "duAux", "q1", "duAux")]>

# use q1 to get qhf aux vars for elc and ion
# (heat flux vectors)
<[QHFAUX("qhfAux_q1", "qhfAux", "q1", "gtAux",
           "qhfAux")]>

# use q1 to get pi aux vars for elc and ion
# (viscosity tensors)
<[PIAUX("piAux_q1", "piAux", "q1", "guAux",
          "duAux", "piAux")]>

# use q1 to compute dq
<[DGRHS("dgrhs_q1", "q1", "gtAux", "qhfAux",
          "piAux", "guAux", "dq")]>

# use dq q1, and qold to compute qnew
<rk_stage2>
  Type = WxSubSolver
  Kind = linearCombiner

  OnGrid = grid

  ReadVars = [qold, q1, dq]
  coeffs   = [0.5, 0.5, 0.5]
  WriteVars = [qnew]

</rk_stage2>

# subsolver to apply limiter to qnew
<[DGLIMIT("qnew_limit", "characteristic", "qnew")]>

# define boundary condition applicator for gtAux
<[BC("leftBC_gtAux", "bcConductingGtAuxDG2d",
       "gtAux", 1, "lower")]>
<[BC("rightBC_gtAux", "bcConductingGtAuxDG2d",
       "gtAux", 1, "upper")]>

# define boundary condition applicator for guAux
<[BC("leftBC_guAux", "bcConductingGuAuxDG2d",
       "guAux", 1, "lower")]>
<[BC("rightBC_guAux", "bcConductingGuAuxDG2d",
       "guAux", 1, "upper")]>

# define boundary condition applicator for duAux
<[BC("leftBC_duAux", "bcConductingDuAuxDG2d",
       "duAux", 1, "lower")]>
<[BC("rightBC_duAux", "bcConductingDuAuxDG2d",
       "duAux", 1, "upper")]>

# define boundary condition applicator for qhfAux
<[BC("leftBC_qhfAux", "bcConductingQhfAuxDG2d",
       "qhfAux", 1, "lower")]>
<[BC("rightBC_qhfAux", "bcConductingQhfAuxDG2d",
       "qhfAux", 1, "upper")]>
```

```
# define boundary condition applicator for piAux
<[BC("leftBC_piAux", "bcConductingPiAuxDG2d",
       "piAux", 1, "lower")]>
<[BC("rightBC_piAux", "bcConductingPiAuxDG2d",
       "piAux", 1, "upper")]>

# define left boundary condition applicator for
# qold
<[BC("leftBC", "bcConductingNoSlipTwoFluidDG2d",
       "qold", 1, "lower")]>
<[BC("rightBC", "bcConductingNoSlipTwoFluidDG2d",
       "qold", 1, "upper")]>

# define left boundary condition applicator for q1
<[BC("leftBCq1", "bcConductingNoSlipTwoFluidDG2d",
       "q1", 1, "lower")]>
<[BC("rightBCq1", "bcConductingNoSlipTwoFluidDG2d",
       "q1", 1, "upper")]>

# define left boundary condition applicator for qnew
<[BC("leftBCqnew", "bcConductingNoSlipTwoFluidDG2d",
       "qnew", 1, "lower")]>
<[BC("rightBCqnew", "bcConductingNoSlipTwoFluidDG2d",
       "qnew", 1, "upper")]>

# Set qnew back to qold so it can be used in
# hyperbolicSolve
<copier>
  Type = WxSubSolver
  Kind = linearCombiner

  OnGrid = grid
  ReadVars = [qnew]
  coeffs = [1.0]
  WriteVars = [qold]
</copier>

# initialize the array
<[SUBSOLVERSTEP("initArrays", "initialcond",
                 "qnew, qold")]>

# chk_freq step
<[SUBSOLVERSTEP("checkFreqDt", "chk_freq")]>

# NAN check step
<[SUBSOLVERSTEP("isNAN", "nancheck")]>

# compute Grad T stage 1
<[SUBSOLVERSTEP("gradT_stage1", "gtAux_qold,
                  gtAux_limit", "gtAux")]>

# compute Grad U stage 1
<[SUBSOLVERSTEP("gradU_stage1", "guAux_qold,
                  guAux_limit", "guAux")]>

# compute Div U stage 1
<[SUBSOLVERSTEP("divU_stage1", "duAux_qold,
                  duAux_limit", "duAux")]>

# compute heat flux stage 1
<[SUBSOLVERSTEP("qhf_stage1", "qhfAux_qold,
                  qhfAux_limit", "qhfAux")]>

# compute viscosity stage 1
<[SUBSOLVERSTEP("pi_stage1", "piAux_qold,
                    piAux_limit", "piAux")]>

# compute RK stage 1
<[SUBSOLVERSTEP("rk_stage1_step", "dgrhs_qold,
                    rk_stage1")]>

# limit q1
<[SUBSOLVERSTEP("limit_q1_step", "q1_limit"
                   , "q1")]>

# compute Grad T stage 2
<[SUBSOLVERSTEP("gradT_stage2", "gtAux_q1,
                    gtAux_limit", "gtAux")]>

# compute Grad U stage 2
<[SUBSOLVERSTEP("gradU_stage2", "guAux_q1,
                    guAux_limit", "guAux")]>

# compute Div U stage 2
<[SUBSOLVERSTEP("divU_stage2", "duAux_q1,
                    duAux_limit", "duAux")]>

# compute heat flux stage 2
<[SUBSOLVERSTEP("qhf_stage2", "qhfAux_q1,
                    qhfAux_limit", "qhfAux")]>

# compute viscosity stage 2
<[SUBSOLVERSTEP("pi_stage2", "piAux_q1,
                    piAux_limit", "piAux")]>

# compute RK stage 2
<[SUBSOLVERSTEP("rk_stage2_step", "dgrhs_q1,
                    rk_stage2")]>

# limit qnew
<[SUBSOLVERSTEP("limit_qnew_step", "qnew_limit"
                   , "qnew")]>

# apply boundary conditions for gtAux
<[SUBSOLVERSTEP("applyBCgt", "leftBC_gtAux,
                     rightBC_gtAux")]>

# apply boundary conditions for guAux
<[SUBSOLVERSTEP("applyBCgu", "leftBC_guAux,
                     rightBC_guAux")]>

# apply boundary conditions for duAux
<[SUBSOLVERSTEP("applyBCdu", "leftBC_duAux,
                     rightBC_duAux")]>

# apply boundary conditions for qhfAux
<[SUBSOLVERSTEP("applyBCqhf", "leftBC_qhfAux,
                      rightBC_qhfAux")]>

# apply boundary conditions for piAux
<[SUBSOLVERSTEP("applyBCpi", "leftBC_piAux,
                     rightBC_piAux")]>

# apply boundary conditions for qold
<[SUBSOLVERSTEP("applyBC", "leftBC, rightBC")]>
```

```
   # apply boundary conditions for qnew
   <[SUBSOLVERSTEP("applyBCqnew", "leftBCqnew,
                                  rightBCqnew")]>

   # apply boundary conditions for q1
   <[SUBSOLVERSTEP("applyBCq1", "leftBCq1, rightBCq1")]>

   # copy step
   <[SUBSOLVERSTEP("copy", "copier")]>

##
# Solver sequence
##
   <SolverSequence>
     Type = WxSolverSequence

     StartOnly = [initArrays, applyBC]
    PerStep = [applyBC, isNAN, checkFreqDt, \
               gradT_stage1, applyBCgt, \
               gradU_stage1, applyBCgu, \
               divU_stage1, applyBCdu, \
               qhf_stage1, applyBCqhf, \
               pi_stage1, applyBCpi, \
               rk_stage1_step, limit_q1_step, applyBCq1, \
               gradT_stage2, applyBCgt, \
               gradU_stage2, applyBCgu, \
               divU_stage2, applyBCdu, \
               qhf_stage2, applyBCqhf, \
               pi_stage2, applyBCpi, \
               rk_stage2_step, limit_qnew_step, \
               copy] # apply at each step

   </SolverSequence>

 </recon>

</warpx>
```

The macro used in the input file by specifying `#include` is described here. `PAR-ARRAY` allocates the memory for all parallel arrays that live in the entire grid. The auxiliary subsolvers are specified for electron and ion temperature gradient vectors (`GTAUX`), velocity gradient vectors (`GUAUX`), velocity divergence scalars (`DUAUX`), heat flux vectors (`QHFAUX`), and viscous stress tensors (`PIAUX`). Auxiliary equation solvers act like conserved equation solvers and accept an arbitraty number of conserved and other auxiliary variables. An arbitrary number of auxiliary variables and each of their sizes is specified in the input file, so the `dgAuxSolver2d` subsolver packs all the auxiliary variables under one variable `qaux` that is used in all the flux and source calculations. After each of the auxiliary variables are computed, limiters (`GTLIMIT`, `GULIMIT`, `DULIMIT`, `QHFLIMIT`, `PILIMIT`) are applied to smooth out any large gradients that may arise in the auxiliary variables.

The right-hand-side of the hyperbolic equation, `DGRHS`, is then computed using a similar implementation with the `dgRhsCalc2d` subsolver by specifying all the Braginskii auxiliary variables and packing them under `qaux`. The source terms specified for the auxiliary and conserved equation calculations take in input, auxiliary input, and output indices. This allows flexibility within the input file and also restricts the source functions to only accept the variable indices that they need. `DGLIMIT` defines the limiter macro for the conserved variables.

`BC` allows any boundary condition to be chosen from the input file by specifying the subsolver boundary condition (`Kind`), the parallel array to apply the boundary condition to (`WriteVars`), the x, y, or z direction (`direction` of 0, 1, or 2), and the lower or upper edge (`edge`).

`SUBSOLVERSTEP` groups a set of subsolvers that are defined as a single step at the beginning of the simulation, at the end of the simulation, at any given time-step, or at the end of each time-step. The `DtFrac` field allows the choice of a fraction of a time-step over which the subsolver step is implemented; the default is 1. If a variable is specified for `SyncVars`, the subsolver step syncs the variable across all processors in the domain.

```
# -*- python -*-
#begin python
import string

def PAR_ARRAY(name, grid, numComponents):
    txt = '''
   <%s>
     Type = WxVariable
     Kind = parArray

     OnGrid = %s
     NumComponents = %d
     GhostCells = [2, 2]
   </%s>
   '''
    return txt % (name, grid, numComponents, name)

def GTAUX(name, gta, q, gt):
    txt = '''
   <%s>
     Type = WxSubSolver
     Kind = dgAuxSolver2d

     ReadVars = [%s, %s]
     WriteVars = [%s]
     OnGrid = grid

     # CFL number to use
     Cfl = FUDGE*0.5*0.9/(2*SP_ORDER-1.0)
     # maximum CFL allowed
     Cflm = FUDGE*0.5/(2*SP_ORDER-1.0)

     spatialOrder = SP_ORDER

     auxEqnNums = [18]

     Equations = [gtAuxEqn]

     # twofluid auxiliary equations for gradient
     # of Te and Ti
     <gtAuxEqn>
       Type = WxHyperbolicEqn
       Kind = gradTempAuxEqn # kind of equations

       gas_gamma = GAMMA
       boltz = BOLTZ
       mi = MI
       me = ME

       elcMinDensity = ELC_MIN_DENS
       ionMinDensity = ION_MIN_DENS
       elcMinPressure = ELC_MIN_PRES
       ionMinPressure = ION_MIN_PRES

     </gtAuxEqn>

   </%s>'''

    return txt % (name, gta, q, gt, name)

def GUAUX(name, gua, q, gu):
    txt = '''
   <%s>
     Type = WxSubSolver
     Kind = dgAuxSolver2d

     ReadVars = [%s, %s]
     WriteVars = [%s]
     OnGrid = grid

     # CFL number to use
     Cfl = FUDGE*0.5*0.9/(2*SP_ORDER-1.0)
     # maximum CFL allowed
     Cflm = FUDGE*0.5/(2*SP_ORDER-1.0)

     spatialOrder = SP_ORDER

     auxEqnNums = [18]

     Equations = [guAuxEqn]

     # twofluid auxiliary equations for gradient
     # of ue and ui
     <guAuxEqn>
       Type = WxHyperbolicEqn
       Kind = gradVelAuxEqn # kind of equations

     </guAuxEqn>

   </%s>'''

    return txt % (name, gua, q, gu, name)

def DUAUX(name, dua, q, du):
    txt = '''
   <%s>
     Type = WxSubSolver
     Kind = dgAuxSolver2d

     ReadVars = [%s, %s]
     WriteVars = [%s]
     OnGrid = grid

     # CFL number to use
     Cfl = FUDGE*0.5*0.9/(2*SP_ORDER-1.0)
     # maximum CFL allowed
     Cflm = FUDGE*0.5/(2*SP_ORDER-1.0)

     spatialOrder = SP_ORDER

     auxEqnNums = [18]

     Equations = [duAuxEqn]

     # twofluid auxiliary equations for divergence
     # of ue and ui
     <duAuxEqn>
       Type = WxHyperbolicEqn
       Kind = divVelAuxEqn # kind of equations

     </duAuxEqn>
   </%s>'''

    return txt % (name, dua, q, du, name)

def QHFAUX(name, qhfa, q, gt, qhf):
```

```
    txt = '''
   <%s>
     Type = WxSubSolver
     Kind = dgAuxSolver2d

     ReadVars = [%s, %s, %s]
     WriteVars = [%s]
     OnGrid = grid

     # CFL number to use
     Cfl = FUDGE*0.5*0.9/(2*SP_ORDER-1.0)
     # maximum CFL allowed
     Cflm = FUDGE*0.5/(2*SP_ORDER-1.0)

     spatialOrder = SP_ORDER

     auxEqnNums = [18,6]

     Equations = [qhfAuxEqn]
     Sources = [qhfAuxSrc]

     # twofluid auxiliary equations for elc and
     # ion heat flux
     <qhfAuxEqn>
       Type = WxHyperbolicEqn
       Kind = bragHeatFluxAuxEqn # kind of equations

     </qhfAuxEqn>

     # twofluid auxiliary sources for elc and
     # ion heat flux
     <qhfAuxSrc>
       Type = WxHyperbolicSrc
       Kind = bragHeatFluxAuxSrc # kind of equations

       charge = Q
       me = ME
       mi = MI
       epsilon0 = EPS0
       boltz = BOLTZ
       gas_gamma = GAMMA
       dx = (YU-YL)/YCELLS

       InpRange = [0,1,2,3,4,5]
       AuxInpRange = [0,1,2,3,4,5,6,7,8,9,10,11,12,
                      13,14,15,18,19,20,21,22,23]
       OutRange = [0,1,2,3,4,5]

     </qhfAuxSrc>

   </%s>'''

    return txt % (name, qhfa, q, gt, qhf, name)

def PIAUX(name, pia, q, gu, du, pi):
    txt = '''
   <%s>
     Type = WxSubSolver
     Kind = dgAuxSolver2d

     ReadVars = [%s, %s, %s, %s]
     WriteVars = [%s]
     OnGrid = grid

     # CFL number to use
     Cfl = FUDGE*0.5*0.9/(2*SP_ORDER-1.0)
     # maximum CFL allowed
     Cflm = FUDGE*0.5/(2*SP_ORDER-1.0)

     spatialOrder = SP_ORDER

     auxEqnNums = [18,18,2]

     Equations = [piAuxEqn]
     Sources = [piAuxSrc]

     # twofluid auxiliary equations for elc and
     # ion viscosity
     <piAuxEqn>
       Type = WxHyperbolicEqn
       Kind = bragViscosityAuxEqn # kind of equations

     </piAuxEqn>

     # source for viscosity auxiliary equations
     <piAuxSrc>
       Type = WxHyperbolicSrc
       Kind = bragViscosityAuxSrc # kind of sources

       charge = Q
       me = ME
       mi = MI
       epsilon0 = EPS0
       boltz = BOLTZ
       gas_gamma = GAMMA
       dx = (YU-YL)/YCELLS

       InpRange = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,
                   14,15,16,17]
       AuxInpRange = [0,1,2,3,4,5,6,7,8,9, 18,19,20,
                      21,22,23,24,25,26, 27,28,29,30,
                      31,32,33,34,35, 36, 37, 13,14,15]
       OutRange = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,
                   14,15,16,17]

     </piAuxSrc>

   </%s>'''

    return txt % (name, pia, q, gu, du, pi, name)

def DGRHS(name, q, gt, qhf, pi, gu, dq):
    txt = '''
   <%s>
     Type = WxSubSolver
     Kind = dgRhsCalc2d

     ReadVars = [%s, %s, %s, %s, %s]
     WriteVars = [%s]
     OnGrid = grid

     auxEqnNums = [6,6,18,18]

     # CFL number to use
     Cfl = FUDGE*0.5*0.9/(2*SP_ORDER-1.0)
     # maximum CFL allowed
```

```
Cflm = FUDGE*0.5/(2*SP_ORDER-1.0)

spatialOrder = SP_ORDER

Equations = [tfEqn] # equation to solve

Sources = [lorentzElc, lorentzIon, ionCurrents,
           elcCurrents, ionCharge, elcCharge,
           momtransport, heattransport,
           qvisctransport] # source terms

# two-fluid equations
<tfEqn>
  Type = WxHyperbolicEqn
  Kind = twoFluidBragLaxEqn

  gas_gamma = GAMMA
  c0 = LIGHT
  gamma = DIVB_SPEED
  chi = DIVE_SPEED
  elcMinDensity = ELC_MIN_DENS
  ionMinDensity = ION_MIN_DENS
  elcMinPressure = ELC_MIN_PRES
  ionMinPressure = ION_MIN_PRES
</tfEqn>

# Lorentz source term for electron
<lorentzElc>
  Type = WxHyperbolicSrc
  Kind = lorentzForces

  InpRange = [0,1,2,3, 10,11,12,13,14,15]
  OutRange = [1,2,3,4]

  mass = ME
  charge = -Q
</lorentzElc>

# Lorentz source term for ions
<lorentzIon>
  Type = WxHyperbolicSrc
  Kind = lorentzForces

  InpRange = [5,6,7,8, 10,11,12,13,14,15]
  OutRange = [6,7,8,9]

  mass = MI
  charge = Q
</lorentzIon>

# Current sources for electric field from electrons
<elcCurrents>
  Type = WxHyperbolicSrc
  Kind = currents

  InpRange = [1,2,3]
  OutRange = [10,11,12]

  mass = ME
  charge = -Q
  epsilon0 = EPS0
</elcCurrents>

# Current sources for electric field from ions
<ionCurrents>
  Type = WxHyperbolicSrc
  Kind = currents

  InpRange = [6,7,8]
  OutRange = [10,11,12]

  charge = Q
  mass = MI
  epsilon0 = EPS0
</ionCurrents>

# Charge sources for electric field divergence
# from electrons
<elcCharge>
  Type = WxHyperbolicSrc
  Kind = charges

  InpRange = [0]
  OutRange = [16]

  mass = ME
  charge = -Q
  epsilon0 = EPS0
  chi = DIVE_SPEED
</elcCharge>

# Charge sources for electric field divergence
# from ions
<ionCharge>
  Type = WxHyperbolicSrc
  Kind = charges

  InpRange = [5]
  OutRange = [16]

  mass = MI
  charge = Q
  epsilon0 = EPS0
  chi = DIVE_SPEED
</ionCharge>

# Momentum transport = frictional force + thermal force,
# based on Braginskii transport equations
<momtransport>
  Type = WxHyperbolicSrc
  Kind = bragFrictionThermalForce

  InpRange = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
  AuxInpRange = [0,1,2,3,4,5]
  OutRange = [0,1,2,3,4,5,6,7,8,9]

  gas_gamma = GAMMA
  me = ME
  mi = MI
  charge = Q
  epsilon0 = EPS0
  boltz = BOLTZ
</momtransport>

# heat acquired in collisions,
# based on Braginskii transport equations
```

```
    <heattransport>
      Type = WxHyperbolicSrc
      Kind = bragHeatCollision

      InpRange = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
      AuxInpRange = [0,1,2,3,4,5]
      OutRange = [4,9]

      gas_gamma = GAMMA
      me = ME
      mi = MI
      charge = Q
      epsilon0 = EPS0
      boltz = BOLTZ
    </heattransport>

    # heat due to viscosity, visc:grad(u)
    # based on Braginskii transport equations
    <qvisctransport>
      Type = WxHyperbolicSrc
      Kind = bragHeatViscosity

      InpRange = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
      AuxInpRange = [12,13,14,15,16,17,18,19,20, 21,22,
                     23,24,25,26,27,28,29, 30,31,32,33,
                     34,35,36,37,38, 39,40,41,42,43,44,
                     45,46,47]
      OutRange = [4,9]

      gas_gamma = GAMMA
      me = ME
      mi = MI
      charge = Q
      epsilon0 = EPS0
      boltz = BOLTZ
    </qvisctransport>

  </%s>'''

   return txt % (name, q, gt, qhf, pi, gu, dq, name)

def GTLIMIT(name, lim, q):
    txt = '''
   <%s>
     Type = WxSubSolver
     Kind = dgLimiter2d
     OnGrid = grid

     WriteVars = [%s]

     spatialOrder = SP_ORDER
     # one of characteristic, component, no-limiter
     limiter = %s

     Equations = [gtAuxEqn]

     # twofluid auxiliary equations for gradient
     # of Te and Ti
     <gtAuxEqn>
       Type = WxHyperbolicEqn
       Kind = gradTempAuxEqn # kind of equations

       gas_gamma = GAMMA
       boltz = BOLTZ
       mi = MI
       me = ME
     </gtAuxEqn>
   </%s>
   '''
    return txt % (name, q, lim, name)

def GULIMIT(name, lim, q):
    txt = '''
   <%s>
     Type = WxSubSolver
     Kind = dgLimiter2d
     OnGrid = grid

     WriteVars = [%s]

     spatialOrder = SP_ORDER
     # one of characteristic, component, no-limiter
     limiter = %s

     Equations = [guAuxEqn]

     # twofluid auxiliary equations for gradient
     # of ue and ui
     <guAuxEqn>
       Type = WxHyperbolicEqn
       Kind = gradVelAuxEqn # kind of equations

     </guAuxEqn>

   </%s>
   '''
    return txt % (name, q, lim, name)

def DULIMIT(name, lim, q):
    txt = '''
   <%s>
     Type = WxSubSolver
     Kind = dgLimiter2d
     OnGrid = grid

     WriteVars = [%s]

     spatialOrder = SP_ORDER
     # one of characteristic, component, no-limiter
     limiter = %s

     Equations = [duAuxEqn]

     # twofluid auxiliary equations for divergence
     # of ue and ui
     <duAuxEqn>
       Type = WxHyperbolicEqn
       Kind = divVelAuxEqn # kind of equations

     </duAuxEqn>

   </%s>
   '''
    return txt % (name, q, lim, name)
```

```
def QHFLIMIT(name, lim, q):
    txt = '''
   <%s>
     Type = WxSubSolver
     Kind = dgLimiter2d
     OnGrid = grid

     WriteVars = [%s]

     spatialOrder = SP_ORDER
     # one of characteristic, component, no-limiter
     limiter = %s

     Equations = [qhfAuxEqn]

     # twofluid auxiliary equations for elc and
     # ion heat flux
     <qhfAuxEqn>
       Type = WxHyperbolicEqn
       Kind = bragHeatFluxAuxEqn # kind of equations

     </qhfAuxEqn>

   </%s>
   '''
    return txt % (name, q, lim, name)

def PILIMIT(name, lim, q):
    txt = '''
   <%s>
     Type = WxSubSolver
     Kind = dgLimiter2d
     OnGrid = grid

     WriteVars = [%s]

     spatialOrder = SP_ORDER
     # one of characteristic, component, no-limiter
     limiter = %s

     Equations = [piAuxEqn]

     # twofluid auxiliary equations for elc
     # and ion viscosity
     <piAuxEqn>
       Type = WxHyperbolicEqn
       Kind = bragViscosityAuxEqn # kind of equations

     </piAuxEqn>

   </%s>
   '''
    return txt % (name, q, lim, name)

def DGLIMIT(name, lim, q):
    txt = '''
   <%s>
     Type = WxSubSolver
     Kind = dgLimiter2d
     OnGrid = grid

     WriteVars = [%s]

     spatialOrder = SP_ORDER
     # one of characteristic, component, no-limiter
     limiter = %s

     Equations = [tfEqn] # equation to limit

     # two-fluid equations
     <tfEqn>
       Type = WxHyperbolicEqn
       Kind = twoFluidBragLaxEqn

       gas_gamma = GAMMA
       c0 = LIGHT
       chi = DIVE_SPEED
       gamma = DIVB_SPEED
       elcMinDensity = ELC_MIN_DENS
       ionMinDensity = ION_MIN_DENS
       elcMinPressure = ELC_MIN_PRES
       ionMinPressure = ION_MIN_PRES

       # for euler eigensystem for characteristic
       # limiting
       minPressure = ELC_MIN_PRES
   </tfEqn>

   </%s>'''
    return txt % (name, q, lim, name)

def BC(name, kind, arr, dr, edge):
    txt = string.Template('''
   <$name>
     Type = WxSubSolver
     Kind = $kind

     OnGrid = grid
     WriteVars = [$arr]

     direction = $dr
     edge = $edge
   </$name>
   ''')
    return txt.substitute(name=name, kind=kind,
                          arr=arr, dr=dr, edge=edge)

def SUBSOLVERSTEP(name, ss, sv=None, dtFrac=1.0):
    if sv:
        s = "SyncVars = [%s]" % sv
    else:
        s = ""
    txt = string.Template('''
   <$name>
     Type = WxSubSolverStep

     SubSolvers = [$ss]
     $svar
     DtFrac = $dtFrac
   </$name>
    ''')
    return txt.substitute(name=name, ss=ss, svar=s,
                          dtFrac=dtFrac)

#end python
```

C.2 Implicit-DG Input File for Two-Fluid Plasma Model

A semi-implicit DG implementation for the two-fluid plasma model is performed in a similar manner to the explicit implementation. Sections of a 1-D sample input file are presented here. The difference being that the ions are treated in exactly the same manner as the explicit case but the electrons and Maxwell's equations are treated separately. Additional parallel arrays are introduced to separate the ion Euler equations from the electron Euler equations and Maxwell's equations.

```
##
# Array definitions
##
   <[PAR_ARRAY("qelcmaxold", "grid", MEQNELCMAX*SP_ORDER)]>
   <[PAR_ARRAY("qelcmaxnew", "grid", MEQNELCMAX*SP_ORDER)]>

   <[PAR_ARRAY("qionold", "grid", MEQNION*SP_ORDER)]>
   <[PAR_ARRAY("dqion", "grid", MEQNION*SP_ORDER)]>
   <[PAR_ARRAY("qion1", "grid", MEQNION*SP_ORDER)]>
   <[PAR_ARRAY("qionnew", "grid", MEQNION*SP_ORDER)]>
```

The initial conditions are applied separately for each of the `qelcmaxold` and `qionold` variables similar to the description in Sec. C.1. The implicit advance is performed first using the macro

```
   # Electron & Maxwell Implicit update
   # use implicit algo to compute qnew from qold for Electron & Maxwell
   <[DGRHSIM("dgrhs_elc_max", "qelcmaxold", "qionold", "qelcmaxnew")]>
```

where `qionold` is passed as an auxiliary variable to advance the fields through source terms. For the explicit solver only the right-hand-side is computed for `DGRHS` following which the desired Runge-Kutta is implemented from the input file. However, for the implicit solver, even though the same right-hand-side subsolvers are used within WARPX, the output of `DGRHSIM` is the solution of the conserved variables after one time-step. The solution is limited and the boundary conditions are applied for the `qelcmaxnew` variable. The explicit advance for the ions is then performed in the same manner described previously where `qelcmaxnew` is passed in as an auxiliary variable so the electric and magnetic fields update the ions.

```
# Euler Explicit update for ions
# Right-Hand-Side uses qold to compute dq for euler ion
<[DGRHSION("dgrhs_qold_ion", "qionold", "qelcmaxnew", "dqion")]>
# Runge-Kutta stage 1 update follows
```

In this manner any desired number of Runge-Kutta stages are implemented through the input file. The frequency check for the time-step only computes the ion plasma and cyclotron frequencies since the electrons are updated implicitly. Limiters and boundary conditions are applied for the ion fluid. The subsolver sequence is specified using

```
<SolverSequence>
  Type = WxSolverSequence

  StartOnly = [initArrays, applyBC_elc_max, applyBC_ion]
  PerStep = [chkFreqDt, impl_step, applyBCqnew_elc_max, \
             rk_stage1_step_ion, applyBCq1_ion, \
             rk_stage2_step_ion, applyBCqnew_ion, \
             copy] # apply at each step

</SolverSequence>
```

The macro for the semi-implicit implementation of the two-fluid model defines `DGRHSIM` to use `Kind = dgRhsCalc1dImplicit` and the remaining equations and sources are defined just for the electron Euler equations and Maxwell's equations. `qelcmaxold` is the conserved variable, `q` and `qionold` is the auxiliary variable, `qion`. `THETA` controls whether it is a BDF1 or CN2 scheme. `CFLM` is the maximum user desired CFL number for the implicit time-advance which can be more restrictive depending on the speeds and frequencies of the ion fluid. A portion of the macro is shown here. Similar implementation is done for the explicit ions. `bcSubSolvers` is specified to apply the boundary conditions between every iteration of the Newton-based implicit scheme.

```
def DGRHSIM(name, q, qion, qnew):
    txt = '''
   <%s>
     Type = WxSubSolver
     Kind = dgRhsCalc1dImplicit

     ReadVars = [%s, %s]
     WriteVars = [%s]
     OnGrid = grid
```

```
    auxEqnNums = [5]

    theta = THETA

    Cfl = CFLM
    Cflm = CFLM

    spatialOrder = SP_ORDER

    bcSubSolvers = [leftBC_elc_max, rightBC_elc_max]

    Equations = [eulerElc, maxEqn] # equation to solve
    Sources = [elcCurrents, ionCurrents, elcCharge, ionCharge, lorentzElc]
    ...
```

If a BDF2 scheme is desired, then 2 auxiliary variables are passed to the implicit advance and they are packed as one `qaux` in the subsolver within WARPX. The variable `qolder` represents the `qelcmax` variable 2 time-steps prior to the current time-step which is required for the BDF2 algorithm.

```
def DGRHSIM(name, q, qolder, qion, qnew):
    txt = '''
   <%s>
     Type = WxSubSolver
     Kind = dgRhsCalc1dImplicit

     ReadVars = [%s, %s, %s]
     WriteVars = [%s]
     OnGrid = grid

     auxEqnNums = [13,5]

     theta = THETA

     bdf = BDF

     Cfl = CFLM
     Cflm = CFLM

     spatialOrder = SP_ORDER

     bcSubSolvers = [leftBC_elc_max, rightBC_elc_max]

     Equations = [eulerElc, maxEqn] # equation to solve
     Sources = [elcCurrents, ionCurrents, elcCharge, ionCharge, lorentzElc]
     ...
```

C.3 RKDG Input File for Two-Fluid Plasma Model using General Geometry

The general geometry implementation presently requires the vertices locations to be specified using a functional representation. The vertices are defined as standard parallel arrays. A geometry variable is defined that contains all the volumes, surface areas, Jacobians, mass matrices, and all other grid metrics that are static throughout the simulation. If using periodic boundary conditions, care needs to be taken to define 2 separate grids; one for the vertices and geometry and the other for the variables being advanced. This is because a periodic boundary condition can be applied to the components of the vertices parallel array incorrectly severely distorting the grid. Note that the geometry variable is of type `dgGeometry`. A 3-D HDF5 data file using general geometry can become rather large hence, setting `writeOut = false` for the geometry variable significantly reduces the file size.

```
<vertices>
  Type = WxVariable
  Kind = parArray
  OnGrid = grid
  NumComponents = 3 # number of components
  GhostCells = [2, 3, 3] # no of ghost cells use
  WriteGhostCells = [0, 1, 1] # how many ghost cells to write

  # these are vertices, hence offsets should be 0.0
  xoffsets = [0.0, 0.0, 0.0]
  yoffsets = [0.0, 0.0, 0.0]
  zoffsets = [0.0, 0.0, 0.0]
</vertices>

<geometry>
  Type = WxVariable
  Kind = dgGeometry
  OnGrid = grid
  spatialOrder = SP_ORDER
  GhostCells = [2, 2, 2] # no of ghost cells to use
  writeOut = false # do not write geometry array to file
</geometry>
```

The vertices are defined using a functional representation in same manner as initial conditions are applied with `x`, `y`, and `z` representing the logical grid which is used to define the physical grid. The vertices are then used to compute volumes, surface areas, Jacobians, and mass matrix in the DG geometry subsolver.

```
    <verticesCalc>
      Type = WxSubSolver
      Kind = funcArraySetter

      WriteVars = [vertices]
      OnGrid = grid

      <func>
        Type = WxFunction
        Kind = exprFunc # name of initial-condition function
        pi = PI

        # list of preiliminary expressions to execute
       progn = ["xg = 2*x", \
                "yg = 0.2*y"]

       exprList = ["xg", "yg", "z"]
     </func>
    </verticesCalc>

    <geometryCalculator>
      Type = WxSubSolver
      Kind = dgGeometryCalc3d
      OnGrid = grid
      spatialOrder = SP_ORDER

      ReadVars = [vertices]
      WriteVars = [geometry]
    </geometryCalculator>

   # initialize the array
   <[SUBSOLVERSTEP("calcVertices", "verticesCalc", "vertices")]>

   # initialize the array
   <[SUBSOLVERSTEP("calcGeometry", "geometryCalculator", "geometry")]>
```

The grid metrics only need to be computed once at the start of the simulation so they are called in the `StartOnly` step.

```
##
# Solver sequence
##
   <SolverSequence>
     Type = WxSolverSequence
     StartOnly = [calcVertices, calcGeometry, initArrays]
     PerStep = [applyBC, checkFreqDt, rk_stage1_step, \
                applyBC_q1, rk_stage2_step, copy] # apply at each step
     WriteOnly = [writer]

   </SolverSequence>
```

The boundary conditions need to account for general geometry so the appropriate BC

subsolvers are included with the geometry variable sent in as a `ReadVar`. When general geometry is used, the macro files only change in terms of the subsolvers that they call and the additional geometry read variable. For the `DGRHS`, `Kind = dgRhsCalc3dGenGeom` and for `DGLIMIT`, `Kind = dgLimiter3dGenGeom` where the geometry variable is sent to the subsolver as an additional `ReadVar`. The macros for the general geometry case are very similar to the macro included in Sec. C.1.

C.4 DG Write-Only SubSolvers for VisIt

Write-only subsolvers rewrite the data by accounting for the spatial order and project the solution onto the grid with the quadrature points. In doing this, the data can be easily loaded onto VisIt after running simple python scripts that create XMF files based on the HDF5 data. The `gridWriter` module uses the `Kind = gridWriterDGGenGeom` subsolver to project the general geometry grid onto the physical quadrature locations. A similar implementation exists for a regular Cartesian mesh where `Kind = gridWriterDG` projects the grid onto quadrature locations. Similarly, the variables can also be projected onto the quadrature locations using the `exprWriterDG` subsolver described here. Any equation system can use this projection by specifying the `writeNames` (variables names that are written to the HDF5 data file at each time-step) and `indVars` (user-specified `ReadVars` read in the order of their indices). The `indVars` are used to define the output variables, `writeNames` as shown here.

```
<gridWriter>
  Type = WxSubSolver
  Kind = gridWriterDGGenGeom

  ReadVars = [geometry]

  OnGrid = grid
  # spatial order of scheme
  spatialOrder = SP_ORDER

  # set to false if interpolation is not desired at Gaussian points
  useGaussian = false

  # list of variables to write out
  writeNames = ["X", "Y", "Z"]
```

```
 </gridWriter>

# define subsolver to convert twofluid data
<twofluidWriter>
  Type = WxSubSolver
  Kind = exprWriterDG

  OnGrid = grid
  ReadVars = [qnew]

  # spatial order of scheme
  spatialOrder = SP_ORDER
  # number of equations
  meqn = MEQN
  # set to false if interpolation is not desired at Gaussian points
  useGaussian = false

  # list of variables to write out
  writeNames = ["elc_density", "elc_vx", "elc_vy", "elc_vz", "elc_pressure", \
                "ion_density", "ion_vx", "ion_vy", "ion_vz", "ion_pressure", \
                "ex", "ey", "ez",\
                "bx", "by", "bz"]

  # list of independent variables
  indVars = ["re", "rue", "rve", "rwe", "ere",\ # electron cons. variables
             "ri", "rui", "rvi", "rwi", "eri",\ # ion cons. variables
             "ex", "ey", "ez",\ # electric field
             "bx", "by", "bz",\ # magnetic field
             "err1", "err2"]

  # electron density
  <elc_density>
    me = ME
    exprList = ["re/me"]
  </elc_density>

  # electron velocity
  <elc_vx>
    exprList = ["rue/re"]
  </elc_vx>

  ...

</twofluidWriter>
```

This is then included in the subsolver sequence using an additional statement `WriteOnly = [writer]` where `writer` uses the macro `SUBSOLVERSTEP` to call the grid and expression writers.

VITA

Bhuvana Srinivasan grew up in 4 cities in India and in Thailand. She earned her B.S. in Aerospace & Mechanical Engineering from the Illinois Institute of Technology and her M.S. and Ph.D. from the Department of Aeronautics & Astronautics at University of Washington. She engaged in traveling, snowboarding, biking, mountaineering, and partying while in graduate school.