An Advanced Implicit Solver for MHD

Bogdan Udrea

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

Doctor of Philosophy

University of Washington

1999

Program Authorized to Offer Degree: Aeronautics and Astronautics

University of Washington

Graduate School

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

Bogdan Udrea

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 Supervisory Committee:

_____

Uri Shumlak

Reading Committee:

_____

D. Scott Eberhardt

_____

Thomas R. Jarboe

_____

Uri Shumlak

Date: _____

In presenting this dissertation in partial fulfillment of the requirements for the Doctorial 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 thesis is allowable only for scholary purposes, consistant with "fair use" as prescribed in the U.S. Copyright Law. Requests for copying or reproduction of this dissertation may be referred to University Microfilms, 1490 Eisenhower Place, P.O. Box 975, Ann Arbor, MI 48106, 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

An Advanced Implicit Solver for MHD

by Bogdan Udrea

Chair of Supervisory Committee

Research Assistant Professor Uri Shumlak
Aeronautics and Astronautics

A new implicit algorithm has been developed for the solution of the time-dependent, viscous and resistive single fluid magnetohydrodynamic (MHD) equations. The algorithm is based on an approximate Riemann solver for the hyperbolic fluxes and central differencing applied on a staggered grid for the parabolic fluxes. The algorithm employs a locally aligned coordinate system that allows the solution to the Riemann problems to be solved in a natural direction, normal to cell interfaces. The result is an original scheme that is robust and reduces the complexity of the flux formulas. The evaluation of the parabolic fluxes is also implemented using a locally aligned coordinate system, this time on the staggered grid. The implicit formulation employed by WARP3 is a two level scheme that was applied for the first time to the single fluid MHD model. The flux Jacobians that appear in the implicit scheme are evaluated numerically. The linear system that results from the implicit discretization is solved using a robust symmetric Gauss-Seidel method. The code has an explicit mode capability so that implementation and test of new algorithms or new physics can be performed in this simpler mode. Last but not least the code was designed and written to run on parallel computers so that complex, high resolution runs can be per-

formed in hours rather than days. The code has been benchmarked against analytical and experimental gas dynamics and MHD results. The benchmarks consisted of one-dimensional Riemann problems and diffusion dominated problems, two-dimensional supersonic flow over a wedge, axisymmetric magnetoplasmadynamic (MPD) thruster simulation and three-dimensional supersonic flow over intersecting wedges and spheromak stability simulation. The code has been proven to be robust and the results of the simulations showed excellent agreement with analytical and experimental results. Parallel performance studies showed that the code performs as expected when run on parallel computers and the speedups obtained were excellent.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

The author wishes to express sincere appreciations to the members of his Ph.D. committee for the help given all the way towards the completion of this work. I am especially indebted to my advisor, Professor Uri Shumlak, for his advice and many useful ideas that resulted from discussions on and off the subject of computational MHD and plasma physics. Through class work and many discussions Professors Scott Eberhardt and Randall LeVeque provided me with the knowledge and tools that were essential to the successful implementation of the algorithms presented in this dissertation. While I was feeling my way around numerical methods Professors Thomas Jarboe and Reiner Decher took care that I did not stray too far away from the real world and kept focused on the physical aspect of what I was putting in code.

I also wish to thank my colleague Ogden Jones, with whom I worked together on this project for about two years and a half, for showing me around and introducing me to computational MHD. Both David Taflin and Stephen Pluntze, with whom I shared an office in the CFD lab, contributed a lot to creating a great study and work environment and also provided the occasional stress relief when the bugs in the code were too hard to find. Brian Leverson, the Department's system administrator, not only maintained the computers in ship-shape condition but his original take on the world made the lunches we had together the highlights of the day. Thanks guys for the good times we had together.

There is a long list of educators and teachers that contributed to my progress through school and life and I want to thank all of them for their help, especially to my parents who instilled in me the importance of a good education and did a fine job in raising a strong and independently minded individual even under the harsh years of communism in native Romania.

Last but not least I want to thank my wife Roxana who was at my side all the five years that took to complete this work, providing constant moral support and encouragement, and to my daughter Anna Matilda who provided most of the entertainment and absorbed the last drop of spare time for the last two years. Thank you girls.

DEDICATION

To my wife Roxana and our daughter Anna Matilda.

Chapter 1

INTRODUCTION

Plasma is also known as the fourth state of matter and it can be encountered in many aspects of the day-to-day life. The most conspicuous plasmas around us are present in the Sun that produces energy by fusion of hydrogen isotopes and in lighting elements that produce light by means of an electrical discharge through a slightly ionized a gas.

A few of the applications of plasmas of immediate and near term interest are magnetic confinement fusion plasmas, plasmas that are used as work fluids in electric thrusters and the plasmas produced by heating the air that passes through a shock wave in front of a hypersonic vehicle. There is a need for three-dimensional plasma physics simulations that can be used to design, optimize and study the above described devices and phenomena.

The goal of the work described here was to develop a code that uses methods similar to those developed by the computational fluid dynamics (CFD) community for the solution of the set of partial differential equations that describe the single fluid magnetohydrodynamics (MHD) model.

The single fluid MHD model is a mixed set of hyperbolic-parabolic PDEs. The hyperbolic part of the system of PDEs is also known as the ideal MHD model and describes wave-like behavior of the variables that describe the model (density, momentum, magnetic field and total energy.) The parabolic part describes the diffusion of the variables due to resistivity and viscosity. The hyperbolic PDEs are solved us-

ing an approximate Riemann solver that updates the variables inside the cells that discretize the domain of interest by calculating the fluxes at the cell faces based on the solution of an approximate Riemann problem. The traditional approach of generalized coordinate transformation proved extremely complicated and prone to errors so that an original method of calculating the fluxes at cell interfaces has been developed and implemented. The method locally aligns the coordinate system to the faces of each cell and allows the solution of the Riemann problem, in a more natural direction, normal to the cell face. Thus the complexity of the approximate Riemann solver algorithm is reduced and the errors present in a previous version of the code are eliminated. The parabolic terms PDEs are solved using a similar approach that involves a locally aligned coordinate system this time aligned with the faces of cell of a staggered grid. The staggered grid is a fictional grid that is generated on the fly, as needed.

Another advance of the state of the art is the implementation of a fully implicit dual level scheme that proved beneficial for both steady state and time dependent simulations. An implicit scheme compared to an explicit scheme extends the stability region of a certain discretization of the model equations. The dual level scheme was developed for and it is used in regular gas dynamics codes and this is the first time, to the author's knowledge, that it is applied to an MHD algorithm. The scheme is based on an implicit discretization of the system of PDEs and introduction of a fourth dimension in which the linear equations obtained from the discretization are iterated in a time-marching fashion. Due to this time-marching similarity the fourth dimension is called the pseudo time, however it has no connection with the physical time. The flux Jacobians that result from the discretization are calculated numerically using either a traditional approach that gives a first order approximation of the derivatives or using a complex number formulation that gives a robust second order approximation.

The advantage of the implicit scheme over the explicit scheme for steady state

computations is that the implicit scheme adds the right amount of numerical diffusion to the scheme so that the residual drops more that twelve orders of magnitude indifferent of the size of the time step or number of iterations of the implicit solver. In contrast the explicit scheme reaches a true steady solution only when the time step is very small that has the effect of adding numerical diffusion, indiscriminately, everywhere in the domain. Similar results where observed in gas dynamics simulations and it seems that this is the first time these observation are made for MHD benchmarks (see Section 5 for details.)

Another goal of the work described here and an original contribution was to develop a robust and portable parallel code. The advantage of a parallel code is rapid turnaround time for the solution of large, complex computational problems. Parallel codes are designed to run on multiple processor computers. Ideally, the speed-up is equal to the number of processors assigned to the task. (For example, for the solution of the same problem a parallel code running on four processors would be four times faster than a serial version running on one of these processors.) The parallel code takes advantage of the fact that data dependency of the algorithm used to solve the MHD model is loose. Loose data dependency is illustrated by the stencils needed to calculate the hyperbolic and parabolic fluxes. Only two cells placed symmetrically about the faces of the cell to be updated are needed for flux evaluations. In consequence, if the domain on which the solution is sought is divided into subdomains and the algorithm is applied concurrently to the data on each subdomain, then a limited amount of data is exchanged in order to calculate the fluxes at the boundaries between subdomains and thus to maintain the "physical connection" between them. This programming paradigm is called domain decomposition or coarse grain parallelization, and the type of code that implements it is called single program multiple data (SPMD). Tests presented in Section 5 showed that a parallel code implemented using a domain decomposition approach can be developed making abstraction of the architecture of the parallel computer and confirmed the speed-up expectations. The

portability of the code between three parallel computers has been tested and the performance of the parallel is excellent, with the speed-up results confirming the validity of the approach.

Last but not least WARP3 has been thoroughly tested with one, two and three-dimensional benchmarks and proven to be robust and reliable. The gas dynamics and MHD one-dimensional Riemann problem benchmarks were used to validate the approximate Riemann solver. Diffusion dominated one-dimensional benchmarks were used to validate the parabolic flux formulations. Two-dimensional gas dynamics and MHD supersonic flows past wedges were validated against Rankine-Hugoniot analytical solutions. An axisymmetric benchmark compares WARP3 simulation of a continuous flow MPD thruster versus experimental and other simulation results. A three-dimensional gas dynamics supersonic flow past intersecting wedges simulation was validated against experimental and numerical results. Last but not least stability studies of spheromaks enclosed in cylindrical flux conservers proved that WARP3 is a robust code that can perform three-dimensional time dependent simulations of fusion interest. This is the first time spheromak stability studies were performed with a non-linear code in three-dimensional configurations.

1.1 Organization of the Dissertation

The dissertations is divided in four main sections, each corresponding to a major chapter. The first section (Chapter 2) presents the derivation of the single fluid MHD model and the assumptions used in this derivation as well as the limitations of the model. It is hoped that the derivation of the model includes sufficient information so that new researchers are able to acquire the needed physical background at a high rate without agonizing over the algebraic details of the derivations. The following section (Chapter 3) gives a detailed description of the algorithm developed for the solution of the single fluid model and includes a survey of the methods used for the

solution of the single fluid MHD model. The third section (Chapter 4) describes the details of the parallel implementation and for the student that will only run the code the first two subsections and the User's Manual (presented in Appendices) provide the necessary information. For workers actively involved in parallel code development Chapter 4 might be a useful guide in the implementation of coarse grain parallelization of codes that use structured grids. Last (but not least) section (Chapter 5) presents the benchmarks of the code and discusses the performance of the parallel code. Chapter 6 contains a summary, conclusions and suggestions for future work. Information that other researchers might find useful is given in the appendices. Appendices A, B and C contain information on and the formulas of the eigensystem of the approximate Riemann solver that were too bulky to include in the main text of the dissertation. Appendix D gives the details of attempts to find "Roe" averages for MHD as reference for future workers that might find the methods or the approaches useful. Appendix E contains a description of the derivation of the axisymmetric source terms and their formulas. Appendix F describes a fast and accurate method for calculating cell metrics such as face areas and centroid position that might be useful in the development of similar codes. In Appendix G the method that applies a special treatment to non-simply connected grids is briefly described. Last but not least Appendix H contains a User's Manual for the code.

Chapter 2

SINGLE FLUID MAGNETOHYDRODYNAMIC MODEL

This section describes the derivation of the single fluid magnetohydrodynamic model from principles of statistical gas dynamics. An analysis of the physical basis for this model is presented along with the assumptions and approximations used in the derivation of the model.

2.1 Plasma Models

A plasma is a collection of positively and negatively charged ions, electrons, and neutral atoms. Computer simulation of plasmas requires a description of how the particles in a plasma interact with each other and with externally applied fields.

There are two general categories of plasma models, the kinetic model and the fluid model. Kinetic modeling of the plasma is accomplished by numerically solving the plasma kinetic equations (Vlasov or Fokker-Plank) or by using particle in cell (PIC) descriptions of the plasma. Generally PIC codes solve Maxwell's equations on a spatial grid (a discretization of the physical domain where a solution is sought.) Macro-particles or clouds (aggregates of many fundamental particles) are pushed by the electromagnetic fields through the relativistic Lorenz equation and the currents created by the motion of these charged particles are used, in turn, to update the electric and magnetic fields. The updated solution is used as the starting point for the next time step [8]. Three dimensional applications of the kinetic model are limited to relatively low density plasmas, such as contamination of spacecraft by ion thrusters [60] or gaseous electronics such as klystron oscillator tubes [61].

Fluid models assume that the particles in a plasma have collective behavior so that equations of conservation for mass, momentum and energy can be derived. Approximate transport coefficient such as viscosity, resistivity and heat transfer coefficients are also obtained in the derivations. Hybrid plasma models in which fluid and particle treatments are applied to different components of the plasma are also investigated. For example Swift [73] describes a code that calculates the interaction of fully kinetic ions and a massless electron fluid with the magnetic field.

The thrust of this research is the development of a single fluid model computer code for plasmas that incorporates advanced methods developed for electrically non-conducting fluids. In the following sections the equations that describe this single fluid model are derived.

2.2 Statistical Plasma Dynamics

The model described in this section is derived assuming that externally applied fields, such as electric and magnetic fields, and pressure gradients introduce only small perturbations in the motion of individual particles. When the perturbing effect of external fields is relatively small and there exists a large number of collisions between the particles, it is convenient to average the perturbing effect over the particles making up the plasma. A statistical approach is used to derive macroscopic properties such as density, average velocity and pressure. These properties are related to the particles' position and velocity by a distribution function $f_\alpha(\mathbf{r}, \mathbf{c}, t)$. The distribution function f_α gives the statistical probability per unit volume of phase space $(\mathbf{r}, \mathbf{c})$ of finding particles of species α at position $\mathbf{r}$ and velocity $\mathbf{c}$ at time t.

2.2.1 Densities

Number density of species α at position $\mathbf{r}$ and time t is the integral of the distribution function over all velocity space

$$n_\alpha = \int_{-\infty}^{+\infty} f_\alpha(\mathbf{r}, \mathbf{c}, t) d\mathbf{c}, \tag{2.1}$$

where it has been assumed that $f_\alpha \to 0$ as $\mathbf{c} \to \pm\infty$.

Mass and charge densities are

$$\rho_\alpha^m = m_\alpha \int f_\alpha(\mathbf{r}, \mathbf{c}, t) d\mathbf{c} = m_\alpha n_\alpha, \tag{2.2}$$

and

$$\rho_\alpha^q = q_\alpha \int f_\alpha(\mathbf{r}, \mathbf{c}, t) d\mathbf{c} = q_\alpha n_\alpha, \tag{2.3}$$

where m_α and q_α are the mass and respectively the charge of a particle of species α. (From here on the superscript m for mass density and the integration limits are dropped for simplicity.) Total mass density of the plasma is defined as

$$\rho = \sum_\alpha \rho_\alpha = \sum_\alpha n_\alpha m_\alpha. \tag{2.4}$$

In general an average of any property is obtained by integration of that property over the distribution function

$$n_\alpha \langle A \rangle_\alpha = \int A(\mathbf{c}) f_\alpha d\mathbf{c} \tag{2.5}$$

where property $A(\mathbf{c})$ can be a scalar, vector or tensor.

2.2.2 Fluxes

Similarly to the definition of averages, an average flux of a property Θ_α of species α can also be defined. The flux of Θ_α of a single particle through an infinitezimaly small surface $d\mathbf{s}$ which is stationary with respect to the laboratory is $\Theta_\alpha \mathbf{w} \cdot d\mathbf{s}$, where $\mathbf{w}$ is

the particle velocity with respect to laboratory coordinates. The total flux, obtained by integrating over the distribution function, is

$$Flux = \left(\int \Theta_\alpha \mathbf{w} f_\alpha d\mathbf{w} \right) \cdot d\mathbf{s},$$

and the flux vector is

$$\mathbf{F} = \int \Theta_\alpha \mathbf{w} f_\alpha d\mathbf{w} = n_\alpha \langle \Theta \mathbf{w} \rangle_\alpha. \tag{2.6}$$

2.2.3 Velocity

The particle number flux, obtained by making $\Theta_\alpha = 1$ in Eqn (2.6), is

$$particle\ flux\ vector = \int \mathbf{w} f_\alpha d\mathbf{w} = n_\alpha \langle \mathbf{w} \rangle_\alpha.$$

The average velocity of species α at position $\mathbf{r}$ and time t is defined as the ratio of the particle-flux vector to the number density of species α.

$$\mathbf{v}_\alpha = \frac{1}{n_\alpha} \int \mathbf{w} f_\alpha d\mathbf{w} = \langle \mathbf{w} \rangle_\alpha. \tag{2.7}$$

2.2.4 Mass Flux

The mass flux of species α is obtained by replacing Θ_α with m_α in Eqn (2.6) so that

$$mass\ flux\ vector = \dot{\mathbf{m}}_\alpha = \int m_\alpha \mathbf{w} f_\alpha d\mathbf{w} = n_\alpha m_\alpha \langle \mathbf{w} \rangle_\alpha = \rho_\alpha \mathbf{v}_\alpha.$$

The mass flux vector is used to define a total mass-flux vector,

$$\dot{\mathbf{m}} = \sum_\alpha \dot{\mathbf{m}}_\alpha = \sum_\alpha \rho_\alpha \mathbf{v}_\alpha. \tag{2.8}$$

From Eqns (2.4) and (2.8) the mass-averaged velocity is defined as

$$\mathbf{v} = \frac{\dot{\mathbf{m}}}{\rho} = \frac{\sum_\alpha \rho_\alpha \mathbf{v}_\alpha}{\sum_\alpha \rho_\alpha}. \tag{2.9}$$

The above definition also implies that $\mathbf{v}$ is the velocity of the center of mass of the plasma. The random thermal velocity of species α is defined as the random velocity with respect to the mass-averaged velocity $\mathbf{v}$,

$$\mathbf{c}_\alpha = \mathbf{w}_\alpha - \mathbf{v}. \tag{2.10}$$

The average of $\mathbf{c}_\alpha$ is the diffusion velocity $\mathbf{V}_\alpha$ of species α.

$$\mathbf{V}_\alpha = \langle \mathbf{c} \rangle_\alpha = \frac{1}{n_\alpha} \int \mathbf{c} f_\alpha d\mathbf{c} = \langle \mathbf{w}_\alpha - \mathbf{v} \rangle = \mathbf{v}_\alpha - \mathbf{v}. \tag{2.11}$$

Velocities defined above ($\mathbf{v}, \mathbf{V}_\alpha$, and $\mathbf{c}_\alpha$) are used in the following sections in the definition of the electric current, momentum flux and partial pressure of species α.

2.2.5 Electric Current

The charge flux or electric current $\mathbf{J}_\alpha$ is found by letting $\Theta_\alpha = q_\alpha$.

$$\mathbf{J}_\alpha = n_\alpha q_\alpha \mathbf{v}_\alpha = n_\alpha q_\alpha \mathbf{V}_\alpha + n_\alpha q_\alpha \mathbf{v} \tag{2.12}$$

The first term on the right hand side of the equation is called the conduction current and the second term is called the convection current of species α.

2.2.6 Momentum Flux and Pressure

If property Θ_α is a vector then its flux is defined as a dyad,

$$\overleftrightarrow{Flux} = \int \Theta_\alpha \mathbf{w} f_\alpha d\mathbf{w}$$

The momentum flux with respect to laboratory coordinates is found by making $\Theta_\alpha = m_\alpha \mathbf{w}$. Thus the momentum flux of species α becomes

$$\overleftrightarrow{\Pi}_\alpha = m_\alpha \int \mathbf{w}\mathbf{w} f_\alpha d\mathbf{w}. \tag{2.13}$$

The expression for the random thermal velocity from Eqn (2.10) is used to obtain $\mathbf{w} = \mathbf{v} + \mathbf{c}_\alpha$. Replacing the expression for $\mathbf{w}$ in Eqn (2.13) yields

$$\overleftrightarrow{\Pi}_\alpha = \rho_\alpha \left(\mathbf{v}\mathbf{v} + \mathbf{V}_\alpha \mathbf{v} + \mathbf{v}\mathbf{V}_\alpha \right) + \rho_\alpha \langle \mathbf{c}\mathbf{c} \rangle_\alpha, \tag{2.14}$$

where Eqns (2.11) and (2.2) were also used.

Momentum flux represents the force exerted by the particles of species α per unit area. The first term represents the contribution of species α to the total mass-averaged momentum, the second term is the diffusion of momentum flux, the third one is the momentum of diffusion. The fourth term $\overleftrightarrow{p}_\alpha = \rho_\alpha \langle \mathbf{cc} \rangle_\alpha$ is the momentum flux associated with random thermal motion and therefore is the partial pressure tensor of the species.

The total pressure tensor is

$$\overleftrightarrow{p} = \sum_\alpha \overleftrightarrow{p}_\alpha, \tag{2.15}$$

and the total shear stress tensor is

$$\overleftrightarrow{\tau} = \overleftrightarrow{p} - \overleftrightarrow{I} \cdot \overleftrightarrow{p}, \tag{2.16}$$

where $\overleftrightarrow{I}$ is the unit tensor.

The trace of a tensor, the sum of the diagonal terms, is an invariant. Thus the trace of the partial pressure tensor is associated with the scalar partial pressure of species α,

$$p_\alpha = \sum_{i,j} \delta_{ij} \overleftrightarrow{p}_{ij} = \rho_\alpha \sum_{i=1}^{3} \langle c_i c_i \rangle_\alpha, \tag{2.17}$$

where δ_{ij} is Dirac's delta.

Assuming that the distribution function is isotropic about $\mathbf{c}$, the partial pressure of species α becomes

$$p_\alpha = \frac{\rho_\alpha}{3} \langle c^2 \rangle_\alpha, \tag{2.18}$$

since in an isotropic system the square of the velocity in one direction is equal to a third of total velocity squared. If the distribution function is isotropic then $\int \mathbf{c} f_\alpha d\mathbf{c} = 0$. This implies that there is no preferred direction for the random thermal velocity of

particles of species α. Mathematically this is equivalent to saying that an isotropic distribution function is an invariant to rotation.

If the distribution function is Maxwellian then the pressure given by Eqn (2.18) is

$$p_\alpha = n_\alpha k T_\alpha, \tag{2.19}$$

where k is Boltzmann's constant and T_α is the temperature of species α.

2.3 Species Conservation Equations

Conservation equations describe how averaged quantities in a moving plasma change in time. The quantities conserved during particle collisions are mass, momentum and energy. In a control volume the time variation of these averaged quantities is due only to the fluxes of the quantities entering and leaving the volume. It is the purpose of this section to describe how the conservation equations are derived for each species.

The conservation equations are derived by taking moments of the Boltzmann equation. This equation describes how the distribution function introduced in the previous section changes when the state of the gas changes. Boltzmann equation is given by

$$\frac{\partial f_\alpha}{\partial t} + \mathbf{w} \cdot \nabla f_\alpha + \frac{\mathbf{F}_\alpha}{m_\alpha} \cdot \nabla_w f_\alpha = \left(\frac{\partial f_\alpha}{\partial t} \right)_{coll}, \tag{2.20}$$

where $\mathbf{F}_\alpha = q_\alpha(\mathbf{E} + \mathbf{w} \times \mathbf{B})$ and ∇_w means that the differentials are taken with respect to the velocity components of the phase space. The term $\left(\frac{\partial f_\alpha}{\partial t}\right)_{coll}$ describes how the distribution function changes in the presence of collisions. This term is null if there are no collisions present or if equilibrium has been established, in which case the collisions do not affect the distribution function. If the collisional term in Eqn (2.20) is null, then the equation is referred to as Vlasov equation.

Taking moments of Boltzmann equation is equivalent to multiplying Eqn (2.20) by a particle property Θ_α that is conserved in collisions, and then integrating over

velocity space gives

$$\int \Theta_\alpha \frac{\partial f_\alpha}{\partial t} d\mathbf{w} + \int \Theta_\alpha \mathbf{w} \cdot \nabla f_\alpha d\mathbf{w} + \int \Theta_\alpha \frac{\mathbf{F}_\alpha}{m_\alpha} \cdot \nabla_w f_\alpha d\mathbf{w} = \int \Theta_\alpha \left(\frac{\partial f_\alpha}{\partial t} \right)_{coll} d\mathbf{w} \tag{2.21}$$

For the zeroth moment $\Theta_\alpha = m_\alpha$, and the integration yields the conservation of mass equation. The first moment, with $\Theta_\alpha = m_\alpha \mathbf{w}_\alpha$, yields momentum conservation, and the second moment, with $\Theta_\alpha = m_\alpha w_\alpha^2/2$, yields energy conservation.

It is useful to introduce the definition of the collision term before working out the moments of Eqn (2.20). The collision term is defined as

$$\left(\frac{\partial f_\alpha}{\partial t} \right)_{coll} = \sum_\beta C_{\alpha\beta}, \tag{2.22}$$

where $C_{\alpha\beta}$ represents collisions of particles of species α with particles of species β.

In plasmas of interest to this work, such as fusion and electric thruster plasmas, the collisions are predominantly elastic Coulomb collisions, between both like and unlike particles. It is also assumed from here on that the plasma is sufficiently hot so that only positively charged ions and electrons are present in the plasma.

2.3.1 Conservation of Mass ($\Theta_\alpha = m_\alpha$)

After some straightforward algebra Eqn (2.21) becomes

$$\frac{\partial}{\partial t}(n_\alpha m_\alpha) + \nabla \cdot (n_\alpha m_\alpha \langle \mathbf{w} \rangle_\alpha) = \int m_\alpha \left(\frac{\partial f_\alpha}{\partial t} \right)_{coll} d\mathbf{w}. \tag{2.23}$$

The mass density of species α is $\rho_\alpha = n_\alpha m_\alpha$ and the average velocity is $\mathbf{v}_\alpha = \langle \mathbf{w} \rangle_\alpha$. For the conservation of mass equation the elastic Coulomb collisions produce no variation of plasma density so that

$$\int C_{ee} d\mathbf{w} = \int C_{ii} d\mathbf{w} = \int C_{ei} d\mathbf{w} = \int C_{ie} d\mathbf{w} = 0.$$

Substituting the above relations in Eqn (2.23) the conservation of mass equation for species α becomes

$$\frac{\partial \rho_\alpha}{\partial t} + \nabla \cdot (\rho_\alpha \mathbf{v}_\alpha) = 0 \tag{2.24}$$

2.3.2 Conservation of Momentum ($\Theta_\alpha = m_\alpha \mathbf{w}_\alpha$)

Evaluation of the integrals in Eqn (2.21) for the first moment gives

$$\frac{\partial}{\partial t}\left(\rho_\alpha \langle \mathbf{w} \rangle_\alpha\right) + \nabla \cdot \left(\rho_\alpha \langle \mathbf{ww} \rangle_\alpha\right) - n_\alpha \langle \left(\mathbf{F}_\alpha \cdot \nabla_w\right) \mathbf{w} \rangle_\alpha = \int m_\alpha \mathbf{w} \left(\frac{\partial f_\alpha}{\partial t}\right)_{coll} d\mathbf{w} \quad (2.25)$$

To simplify Eqn (2.25) the following substitutions are performed:

$$\begin{aligned}
\langle \mathbf{w} \rangle_\alpha &= \mathbf{v}_\alpha \\
\langle \mathbf{ww} \rangle_\alpha &= \mathbf{vv} + \mathbf{vV}_\alpha + \mathbf{V}_\alpha \mathbf{v} + \langle \mathbf{cc} \rangle_\alpha \\
\rho_\alpha \langle \mathbf{cc} \rangle_\alpha &= \overleftrightarrow{p}_\alpha \\
\mathbf{F}_\alpha \cdot \nabla_w \mathbf{w} &= \mathbf{F}_\alpha \\
\langle \mathbf{F} \rangle_\alpha &= q_\alpha \langle \mathbf{E} + \mathbf{w} \times \mathbf{B} \rangle_\alpha = q_\alpha (\mathbf{E} + \mathbf{v} \times \mathbf{B}) + q_\alpha \mathbf{V}_\alpha \times \mathbf{B} = \\
&= q_\alpha (\mathbf{E}^* + \mathbf{V}_\alpha \times \mathbf{B}),
\end{aligned}$$

where $\mathbf{E}^* = \mathbf{E} + \mathbf{v} \times \mathbf{B}$.

The momentum conservation for species α becomes

$$\frac{\partial}{\partial t}\left(\rho_\alpha \mathbf{v}_\alpha\right) + \nabla \cdot \left[\rho_\alpha (\mathbf{vV}_\alpha + \mathbf{V}_\alpha \mathbf{v})\right] + \nabla \cdot \left(\rho_\alpha \mathbf{vv}\right) + \nabla \cdot \overleftrightarrow{p}_\alpha = \int m_\alpha \mathbf{w} \sum_\beta C_{\alpha\beta} d\mathbf{w} + n_\alpha q_\alpha (\mathbf{E}^* + \mathbf{V}_\alpha \times \mathbf{B}).$$

Usually, the contributions to the momentum from diffusion are neglected, so that $\nabla \cdot \left[\rho_\alpha (\mathbf{vV}_\alpha + \mathbf{V}_\alpha \mathbf{v})\right] = 0$. These contributions are orders of magnitude smaller than the contribution to diffusion from collisions. However, they might be important in low density, collisionless plasmas in the presence of high temperature gradients. With this simplification the momentum conservation equation for species α is

$$\frac{\partial}{\partial t}\left(\rho_\alpha \mathbf{v}_\alpha\right) + \nabla \cdot \left(\rho_\alpha \mathbf{vv}\right) + \nabla \cdot \overleftrightarrow{p}_\alpha = \int m_\alpha \mathbf{w} \sum_\beta C_{\alpha\beta} d\mathbf{w} + n_\alpha q_\alpha (\mathbf{E}^* + \mathbf{V}_\alpha \times \mathbf{B}). \quad (2.26)$$

The terms on the right hand side of Eqn (2.26) will be further discussed, in Section (2.4), where the single fluid conservation equations are obtained.

2.3.3 Conservation of Energy ($\Theta = \frac{1}{2}m_\alpha w_\alpha^2$)

The conservation equation for the energy of a species is derived similarly to the mass and momentum conservation equations. The algebra involved is slightly more complicated than that for the previous conservation equations and is described in plasma physics textbooks such as those by Freidberg [25] and by Sutton and Sherman [71].

Replacing Θ_α with $m_\alpha w_\alpha^2/2)$ in Eqn (2.21), performing the integrations and taking the averages yields

$$\frac{\partial}{\partial t}(\frac{1}{2}\rho_\alpha v^2 + \rho_\alpha \mathbf{v}\cdot\mathbf{V}_\alpha + \epsilon_\alpha) + \nabla\cdot[(\frac{1}{2}\rho_\alpha v^2 + \epsilon_\alpha)\mathbf{v}] + \nabla\cdot\mathbf{q}_\alpha + \nabla\cdot\{\rho_\alpha[\frac{1}{2}v^2\mathbf{V}_\alpha + \mathbf{v}(\mathbf{v}\mathbf{V}_\alpha)]\} + \nabla\cdot(\mathbf{v}\cdot\overleftrightarrow{p}) = \mathbf{E}\cdot\mathbf{J}_\alpha + \int \frac{1}{2}m_\alpha w_\alpha^2 \sum_\beta C_{\alpha\beta} d\mathbf{w},$$

where ϵ_α is the internal energy of species α, $\mathbf{q}_\alpha$ is the heat flux vector of species α and $\mathbf{J}_\alpha$ is the electric current carried by species α. Component i of the heat flux vector is $q_\alpha^{(i)} = \rho_\alpha \langle c_i c^2\rangle_\alpha/2$.

Similarly to the momentum equation the contributions to energy from diffusion are neglected so that the conservation of energy for species α becomes

$$\frac{\partial}{\partial t}(\frac{1}{2}\rho_\alpha v^2 + \epsilon_\alpha) + \nabla\cdot[(\frac{1}{2}\rho_\alpha v^2 + \epsilon_\alpha)\mathbf{v}] + \nabla\cdot\mathbf{q}_\alpha + \nabla\cdot(\mathbf{v}\cdot\overleftrightarrow{p}) = \mathbf{E}\cdot\mathbf{J}_\alpha + \int \frac{1}{2}m_\alpha w_\alpha^2 \sum_\beta C_{\alpha\beta} d\mathbf{w}. \quad (2.27)$$

Some plasma models use the species conservation equations derived above to describe the plasma. They are generally known as two (electron and ion) fluid models, or as three fluid models if neutral particles are also accounted for.

2.4 Single Fluid Magnetohydrodynamics Equations

In this section the magnetohydrodynamics (MHD) single fluid conservation equations are derived by summing the species conservation equations over all species. Approximations introduced here will lead to the simplification of the equations without loss

of information. These approximations also limit the range of application of the single fluid MHD model and the limitations will be discussed at the end of the section.

The single fluid MHD equations are formulated in terms of conserved variables and their fluxes. In general, a conservation equation is formulated as

$$\frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot \mathbf{f}_h(\mathbf{q}) = \nabla \cdot \mathbf{f}_p(\mathbf{q}). \tag{2.28}$$

where $\mathbf{q}$ is the conserved variables vector, $\mathbf{f}_h(\mathbf{q})$ is the hyperbolic or advective flux tensor and $\mathbf{f}_p(\mathbf{q})$ is the parabolic or diffusive flux tensor. The diffusive flux vector term contains the non-ideal transport effects such as viscous and resistive diffusion and heat transfer.

2.4.1 Maxwell's Equations

Since Maxwell's equations will be used in the derivations they are introduced first and the effect the approximations have on these equations will be discussed. Maxwell's equations are the fundamental equations of electrodynamics. They describe how the electric and magnetic fields are generated, interact and propagate and are given by a set of partial differential equations

$$\nabla \cdot \mathbf{E} = \frac{1}{\epsilon_0}\rho^q \tag{2.29}$$

$$\nabla \cdot \mathbf{B} = 0 \tag{2.30}$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \tag{2.31}$$

$$\nabla \times \mathbf{B} = \mu_0 \mathbf{j} + \mu_0 \epsilon_0 \frac{\partial \mathbf{E}}{\partial t} \tag{2.32}$$

Plasma is assumed to be made of electrons an ions only. An assumption that will be used throughout the derivations of the single fluid conservation equations is that of charge neutrality which states that $n_e = n_i$. This means that the overall plasma charge density is null $\rho^q = \sum_\alpha \rho^q_\alpha = \sum_\alpha n_\alpha q_\alpha = 0$. The assumption is valid if the characteristic frequency of the plasma is much less than the electron plasma frequency

$\omega \ll \omega_{pe}$, $\omega_{pe} = \sqrt{ne^2/m_e\epsilon_0}$ and if the characteristic length of the plasma is much larger than the Debye shielding length, $a \gg \lambda_d$, $\lambda_d = c_e/\omega_{pe}$, $c_\alpha = \sqrt{2T_\alpha/m_\alpha}$. From Eqn (2.29) it can be seen that this approximation leads to $\epsilon_0 \nabla \cdot \mathbf{E} = 0$. Another approximation is that the electromagnetic waves of interest have phase velocities less than the speed of light $\omega/k \ll c$ and that the characteristic thermal velocities are non-relativistic $c_e, c_i \ll c$. This approximation implies that the displacement current $(\partial E/\partial t)$ in Eqn (2.32) can be neglected. With these approximation Maxwell's equations become

$$\nabla \cdot \mathbf{E} = 0 \tag{2.33}$$

$$\nabla \cdot \mathbf{B} = 0 \tag{2.34}$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \tag{2.35}$$

$$\nabla \times \mathbf{B} = \mu_0 \mathbf{j} \tag{2.36}$$

2.4.2 Single Fluid Conservation of Mass

Summing Eqn (2.24) over all species yields

$$\frac{\partial}{\partial t}(\sum_\alpha \rho_\alpha) + \nabla \cdot (\sum_\alpha \rho_\alpha \mathbf{v}_\alpha) = 0.$$

The single fluid mass conservation equation is obtained by substituting Eqns (2.4) and (2.9) in the above equation.

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0, \tag{2.37}$$

where ρ is plasma density and $\mathbf{v}$ is the velocity of the center of mass of the plasma.

The first term of Eqn (2.37) represents the time variation of the mass density in a control volume, and the second term is the divergence of the flux of density (equal to the momentum).

2.4.3 Single Fluid Conservation of Momentum

Before proceeding with the summation of the species momentum conservation equation over all species it is useful to analyze the collision terms. Let $\mathbf{R}_\alpha$ be the collision term from Eqn (2.26).

$$\mathbf{R}_\alpha = \int m_\alpha \mathbf{w}_\alpha \sum_\beta C_{\alpha\beta} d\mathbf{w}$$

Summing $\mathbf{R}_\alpha$ over all species gives

$$\begin{aligned}\sum_\alpha \mathbf{R}_\alpha &= \int \sum_\alpha (m_\alpha \mathbf{w}_\alpha \sum_\beta C_{\alpha\beta}) d\mathbf{w} \\ &= \int m_e \mathbf{w}_e C_{ee} d\mathbf{w} + \int m_i \mathbf{w}_i C_{ii} d\mathbf{w} + \int (m_e \mathbf{w}_e C_{ei} + m_i \mathbf{w}_i C_{ie}) d\mathbf{w} \\ &= 0,\end{aligned}$$

since momentum is conserved between like particle collisions and total momentum is conserved between unlike particles collisions.

The single fluid momentum conservation equation is obtained by summing Eqn (2.26) over all species. The resulting equation is

$$\frac{\partial}{\partial t}(\rho \mathbf{v}) + \nabla \cdot (\rho \mathbf{v}\mathbf{v}) + \nabla \cdot \overleftrightarrow{p} = \sum_\alpha n_\alpha q_\alpha \mathbf{E}^* + \sum_\alpha n_\alpha q_\alpha \mathbf{V}_\alpha \times \mathbf{B}. \tag{2.38}$$

Further simplification of this equation is possible since the plasma has been assumed neutral. Thus the first term on the right hand side of equation Eqn (2.38) is null. Plasma neutrality also leads to the fact that the total current consists only of conduction current since the total convection current is null, ($\sum_\alpha n_\alpha q_\alpha \mathbf{v} = \rho^q \mathbf{v} = 0$). Thus, the total current is

$$\mathbf{J} = \sum_\alpha \mathbf{J}_\alpha = \sum_\alpha n_\alpha q_\alpha \mathbf{V}_\alpha = \mathbf{j}. \tag{2.39}$$

With these simplifications the single fluid momentum conservation equation becomes

$$\frac{\partial}{\partial t}(\rho \mathbf{v}) + \nabla \cdot (\rho \mathbf{v}\mathbf{v}) + \nabla \cdot \overleftrightarrow{p} = \mathbf{j} \times \mathbf{B}.$$

The pressure tensor can be written as its diagonal plus the stress tensor as shown in Eqn (2.16). With the assumption that the distribution function is isentropic the pressure tensor is

$$\overleftrightarrow{p} = \overleftrightarrow{\tau} + \overleftrightarrow{I}p, \tag{2.40}$$

where p is the scalar pressure of the plasma, $p = \sum_\alpha p_\alpha$. At this point it is useful to define a plasma temperature based on the plasma pressure p. Since $p = nkT = p_e + p_i = n_e kT_e + n_i kT_i = nk(T_e + T_i)$ the temperature of the plasma is defined as $T = T_e + T_i$.

Replacing the pressure tensor expression in the single fluid momentum equation yields

$$\frac{\partial}{\partial t}(\rho \mathbf{v}) + \nabla \cdot (\rho \mathbf{v}\mathbf{v}) + \nabla \cdot (\overleftrightarrow{I}p) + \nabla \cdot \overleftrightarrow{\tau} = \mathbf{j} \times \mathbf{B}. \tag{2.41}$$

It is instructive to note that the momentum equation for a neutral plasma is the same with that found for a non-conducting fluid in regular fluid dynamics with an additional term ($\mathbf{j} \times \mathbf{B}$) which represents a body force. The expression of this force is formulated in terms of magnetic field only by replacing the current with the expression obtained from Maxwell's equation Eqn (2.36),

$$\mathbf{j} = \frac{1}{\mu_0}\nabla \times \mathbf{B}. \tag{2.42}$$

After some straightforward vector algebra the expression for the electromagnetic force becomes

$$\mathbf{j} \times \mathbf{B} = \frac{1}{\mu_0}(\nabla \times \mathbf{B}) \times \mathbf{B} = \frac{1}{\mu_0}[\nabla \cdot (\mathbf{B}\mathbf{B}) - \frac{1}{2}\nabla B^2 - (\nabla \cdot \mathbf{B})\mathbf{B}]. \tag{2.43}$$

Eqn (2.43) is further simplified before substitution in Eqn (2.41). Replacing Maxwell's equation $\nabla \cdot \mathbf{B} = 0$ in the last term of the right hand side of the expression gives

$$\mathbf{j} \times \mathbf{B} = \frac{1}{\mu_0}[\nabla \cdot (\mathbf{B}\mathbf{B}) - \frac{1}{2}\nabla B^2].$$

Finally the single fluid momentum equation is written in conservative form so that fluxes can be identified

$$\frac{\partial}{\partial t}(\rho \mathbf{v}) + \nabla \cdot [\rho \mathbf{v}\mathbf{v} - \frac{1}{\mu_0}\mathbf{B}\mathbf{B} + \overleftrightarrow{I}(p + \frac{1}{2}\frac{B^2}{\mu_0})] = -\nabla \cdot \overleftrightarrow{\tau}. \tag{2.44}$$

Comparing the MHD single fluid momentum equation with the non-conducting fluid momentum equation reveals that the additional flux term proportional to $\mathbf{BB}$ acts as a force and the additional term proportional to B^2 acts as a scalar pressure on the fluid.

To complete the definition of the single fluid momentum Eqn (2.44) an expression for the stress tensor is needed. The expression for the shear stress tensor is

$$\tau_{ij} = \mu \left[- \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} \right) + \frac{2}{3}\delta_{ij} \nabla \cdot \mathbf{v} \right], \tag{2.45}$$

where μ is the viscosity which is a function of composition and temperature and i and j indices correspond to the spatial coordinates. For example $v_1 \equiv v_x$ and $x_2 \equiv y$.

2.4.4 Ohm's Law

The objective of this section is to find a relationship between the total conduction current $\mathbf{j} = n_e e(\mathbf{V}_i - \mathbf{V}_e)$ in terms of only the electric and magnetic field and electron pressure. The relationship obtained is known as generalized Ohm's law. To obtain Ohm's law the electron momentum conservation equation is multiplied through by the ion electric charge and is subtracted from the ion momentum equation which is multiplied through by the electron charge. The first approximation made in the derivation of Ohm's law, is that the electron inertia is neglected. It is a valid approximation since the mass of the electron is about $1,800$ times smaller than the mass of the lightest ion, a proton. The resulting equation is

$$\frac{m_e}{ne^2}\frac{\partial \mathbf{j}}{\partial t} = \mathbf{E} + \mathbf{v} \times \mathbf{B} - \frac{1}{ne}\mathbf{j} \times \mathbf{B} + \frac{1}{ne}\nabla \cdot \overleftrightarrow{p}_e - \overleftrightarrow{\eta} \cdot \mathbf{j}, \tag{2.46}$$

which describes how the electric current varies in time under the influence of the terms on the right hand side.

The approximation that the characteristic plasma frequencies are much lower than the electron plasma frequency allows us to neglect the left hand side of Eqn (2.46). The electron pressure tensor is expressed in terms of the electron stress tensor ($\overleftrightarrow{\tau}_e$) and the electron scalar pressure (p_e) as $\overleftrightarrow{p}_e = \overleftrightarrow{\tau}_e + \overleftrightarrow{I} p_e$. Using these simplifications Eqn (2.46) becomes

$$\mathbf{E} + \mathbf{v} \times \mathbf{B} = \frac{1}{ne}\mathbf{j} \times \mathbf{B} - \frac{1}{ne}\nabla p_e - \frac{1}{ne}\nabla \cdot \overleftrightarrow{\tau}_e + \overleftrightarrow{\eta} \cdot \mathbf{j} \tag{2.47}$$

Eqn (2.47) is known as generalized Ohm's law. The first term on the right hand side of the equation is the contribution of the Hall current to Ohm's law. The curvature of electron trajectory in a magnetic field is responsible for the current transverse to the magnetic field, generally referred to as the Hall current. The second term on the right hand side of the equation is the electron diamagnetic drift effect. In general, a diamagnetic current is generated by the differential gyro-motion of charge carriers in the presence of a magnetic field and pressure gradient. This type of current is observed at the plasma edge in magnetic confinement fusion devices and at the interface between the solar wind and the magnetopause. The third term on the right hand side of Eqn (2.47) is the divergence of the electron stress tensor and is normally neglected due to its low magnitude relative to the Hall and diamagnetic drif terms. The fourth term on the right hand side of Eqn (2.47) represents the contribution of momentum transfer in electron-ion collisions and $\overleftrightarrow{\eta}$ is the plasma electric resistivity tensor.

With the assumption that the divergence of the electron stress tensot is negligible compared to the other terms Eqn (2.47) becomes

$$\mathbf{E} + \mathbf{v} \times \mathbf{B} = \frac{1}{ne}\mathbf{j} \times \mathbf{B} - \frac{1}{ne}\nabla p_e + \overleftrightarrow{\eta} \cdot \mathbf{j} \tag{2.48}$$

The Hall term needs a special treatment due to stability constraints it imposes on the solver. Another graduate student, Julian Becerra-Sagredo [7], implemented a scheme that treats the Hall terms. The benchmarks of the method showed mixed results and

more research is necessary in order to come up with a robust scheme. Implementation of the diamagnetic drift terms is currently researched by yet another graduate student, Ward Vuillemot, who is working on implementing a multiple temperature model that will allow an exact treatment of the ∇p_e term.

As a consequence, Ohm's law that it is currently implemented in the code includes only resistive effects and is given by

$$\mathbf{E} + \mathbf{v} \times \mathbf{B} = \overleftrightarrow{\eta} \cdot \mathbf{j}. \tag{2.49}$$

2.4.5 Induction Equation

In this section the variation of the expression of the magnetic field variation with time is derived starting from Eqn (2.31). Curl of the electric field obtained from Eqn (2.49) is taken and the resulting expression is replaced in Eqn (2.31). Thus the induction equation becomes

$$\frac{\partial \mathbf{B}}{\partial t} + \frac{1}{\mu_0} \nabla \cdot (\mathbf{vB} - \mathbf{Bv}) = \nabla \cdot \overleftrightarrow{E}_{res}, \tag{2.50}$$

where the notation $\nabla \cdot \overleftrightarrow{E}_{res} = -\nabla \times [\overleftrightarrow{\eta} \cdot (\nabla \times \mathbf{B})]$ was used for simplicity.

2.4.6 Single Fluid Energy Conservation

An approach similar to that used for the momentum equation is used to analyze the collision terms present in the species energy equation. Let r_α be the collision term from Eqn (2.27).

$$r_\alpha = \int \frac{1}{2} m_\alpha w_\alpha^2 \sum_\beta C_{\alpha\beta} d\mathbf{w}$$

Summing r_α over all species gives

$$\begin{aligned}\sum_\alpha r_\alpha &= \int \sum_\alpha (\frac{1}{2} m_\alpha w_\alpha^2 \sum_\beta C_{\alpha\beta}) d\mathbf{w} \\ &= \int \frac{1}{2} m_e w_e^2 C_{ee} d\mathbf{w} + \int \frac{1}{2} m_i w_i^2 C_{ii} d\mathbf{w} + \int \frac{1}{2}(m_e w_e^2 C_{ei} + m_i w_i^2 C_{ie}) d\mathbf{w} \\ &= 0,\end{aligned}$$

since energy is conserved between like particle collisions and total energy is conserved between unlike particle collisions.

The single fluid energy conservation equation is obtained by summing Eqn (2.27) over all species.

$$\frac{\partial}{\partial t}(\epsilon + \frac{1}{2}\rho v^2) + \nabla \cdot [(\epsilon + \frac{1}{2}\rho v^2)\mathbf{v}] + \nabla \cdot \mathbf{q} + \nabla \cdot (\mathbf{v} \cdot \overleftrightarrow{\tau}) + \nabla \cdot (p\mathbf{v}) = \mathbf{E} \cdot \mathbf{j}, \qquad (2.51)$$

where the total internal energy is $\epsilon = \sum_\alpha \epsilon_\alpha$, and the total heat flux vector is $\mathbf{q} = \sum_\alpha \mathbf{q}_\alpha$. The expressions for the total current given by Eqn (2.39) and for the pressure tensor given by Eqn (2.40) have also been used.

Comparison with the energy conservation equation for an electrically non-conducting fluid shows that the additional term in Eqn (2.51) is $\mathbf{E} \cdot \mathbf{j}$. This term is named Joule or ohmic heating and represents the amount of electromagnetic energy transferred into thermal energy by means of collisions between charged particles.

Replacement of the expressions for the electric current in Eqn (2.42) and for the electric field, obtained from Ohm's law given by Eqn (2.49), in the energy equation yields an equation that contains a total energy term under the spatial derivative. Total energy is defined as internal energy plus kinetic energy plus magnetic energy

$$e = \epsilon + \frac{1}{2}\rho v^2 + \frac{1}{2}\frac{B^2}{\mu_0}.$$

This total energy term is not present under the time derivative. Taking the dot product of the the induction Eqn (2.50) with the magnetic field yields an expression that is added to both sides of the energy equation. Thus the total energy is introduced

under the time derivative and the conservation equation for total energy is

$$\frac{\partial e}{\partial t} + \nabla \cdot [(e + p + \frac{1}{2}\frac{B^2}{\mu_0})\mathbf{v} - (\mathbf{B} \cdot \mathbf{v})\frac{\mathbf{B}}{\mu_0}] = \nabla \cdot \{\mathbf{v} \cdot \overleftrightarrow{\tau} + \overleftrightarrow{k} \cdot \nabla T - \frac{1}{\mu_0^2}[\overleftrightarrow{\eta} \cdot (\nabla \times \mathbf{B})] \times \mathbf{B}\}, \quad (2.52)$$

where the heat flux vector has been replaced by $\mathbf{q} = -\overleftrightarrow{k} \cdot \nabla T$, $\overleftrightarrow{k}$ is the thermal conductivity tensor, T is the temperature, and B is the magnitude of the magnetic field ($B = \sqrt{\mathbf{B} \cdot \mathbf{B}}$).

2.4.7 Transport Coefficients - Viscosity, Electrical and Thermal Conductivities

Transport coefficients for momentum (viscosity), magnetic field (electrical resistivity) and thermal energy (thermal conductivity) are obtained by linearizing the species distribution function (f_α) solving Boltzmann eqaution given by Eqn (2.20) for each term of the linearization.

$$\left(\frac{\partial f}{\partial t}\right)_{coll} = f_{\alpha_0} + \beta f_{\alpha_1} + \beta^2 f_{\alpha_2} + \dots , \quad (2.53)$$

where β is a small parameter, linear in pressure and density gradients, forces, and f_{α_0} is Maxwellian. Like and unlike particle collision geometries are studied in order to determine the the coefficients and the expression of the terms on the right hand side of Eqn (2.53).

For example, for a fully ionized non-magnetized plasma Spitzer [67] finds the following expression for the dynamic viscosity μ, electric conductivity σ (inverse of resistivity) and thermal conductivity k.

$$\mu = 2.204 \times 10^{-14} \frac{M_i T^{5/2}}{Z^2 ln\Lambda} \quad (2.54)$$

$$\sigma = 2.634 \times 10^{-2} \gamma_E(Z) \frac{T^{3/2}}{Z ln\Lambda} \quad (2.55)$$

$$k = 1.96 \times 10^{-9} \epsilon_T(Z) \delta_T(Z) \frac{T^{5/2}}{Z ln\Lambda} \quad (2.56)$$

where M_i is the atomic number of the ions in the plasma, T is the temperature of the plasma in degrees Kelvin, Z is the mean ion charge and $ln\Lambda$ is the Coulomb logarithm. The Coulomb logarithm is the natural logarithm of the number of particles in a Debye sphere. Coefficients $\gamma_E(Z)$, $\epsilon_T(Z)$ and $\delta_T(Z)$ are correction factors. For example, for $Z = 1$ $\gamma_E = 0.582$, $\epsilon_T = 0.419$ and $\delta_T = 0.225$. Eqns (2.54) to (2.56) use the MKS system of units.

In the case of a magnetized plasma the electric and thermal conductivity are no longer scalars but are tensors. This non-homogeneous behavior is due the charged particles describing gyro motion about magnetic field lines. Charged particles are relatively free to move along magnetic field lines. Motion transversal to the magnetic field lines is possible only after a sufficiently strong collision with another particle alters the trajectory and the particle "jumps" on another field line. The electric conductivity tensor is expressed as

$$\overleftrightarrow{\sigma} = \sigma_{||}\mathbf{bb} + \sigma_H \mathbf{b} \times \overleftrightarrow{I} + \sigma_\perp(\overleftrightarrow{I} - \mathbf{bb}) \tag{2.57}$$

where $\mathbf{b}$ is a unit vector oriented along the magnetic field, $\sigma_{||}$ is the electric conductivity parallel to the magnetic field, σ_H is the Hall conductivity and $\sigma_\perp$ is the transverse conductivity. The electric conductivities $\sigma_{||}$, σ_H and $\sigma_\perp$ are scalars and their expression can be found in plasma formularies such as [1]. (The thermal conductivity tensor is expressed similarly.) To illustrate the tensor form of electrical (or thermal) conductivity it is useful to assume that the magnetic field is assumed to be parallel to the Oz axis. The electrical conductivity tensor written in matrix form is

$$\overleftrightarrow{\sigma} = \begin{bmatrix} \sigma_\perp & \sigma_H & 0 \\ -\sigma_H & \sigma_\perp & 0 \\ 0 & 0 & \sigma_{||} \end{bmatrix}. \tag{2.58}$$

To illustrate the relative freedom of motion along the field lines it is useful to compare the ratio between the electric conductivities parallel and normal to the magnetic

field

$$\frac{\sigma_{||}}{\sigma_{\perp}} = 1 + \frac{\tau_{ee}}{\tau_{pe}}, \tag{2.59}$$

where $\tau_{pe} = 2\pi/\omega_{pe}$ is the electron plasma period. For a typical fusion reactor plasma $\tau_{ee} \approx 10^{-5}$ and $\tau_{pe} \approx 10^{-12}$ so that the ratio of resistivities is $\sigma_{||}/\sigma_{\perp} = 10^{7}$.

The inverse of the electric conductivity tensor ($\overleftrightarrow{\sigma}$) is the electric resistivity tensor

$$\overleftrightarrow{\eta} = \eta_{||}\mathbf{bb} + \eta_H \mathbf{b} \times \overleftrightarrow{I} + \eta_{\perp}(\overleftrightarrow{I} - \mathbf{bb}),$$

with components given by

$$\eta_{||} = \frac{1}{\sigma_{||}},\ \eta_H = -\frac{\sigma_H}{\sigma_H^2 + \sigma_{\perp}^2},\ \eta_{\perp} = \frac{\sigma_{\perp}}{\sigma_H^2 + \sigma_{\perp}^2}.$$

2.5 Tensor Form of the Single Fluid MHD Equations

The mass conservation equation Eqn (2.37), momentum conservation equation Eqn (2.44) and energy conservation equation Eqn (2.52) together with the induction equation given by Eqn (2.50) describe the single fluid model used in this work. They can be grouped in a system of partial differential equations that is expressed as

$$\frac{\partial}{\partial t}\begin{bmatrix} \rho \\ \rho\mathbf{v} \\ \mathbf{B} \\ e \end{bmatrix} + \nabla \cdot \begin{bmatrix} \rho\mathbf{v} \\ \rho\mathbf{v}\mathbf{v} - \frac{1}{\mu_0}\mathbf{B}\mathbf{B} + \overleftrightarrow{I}(p + \frac{1}{2}\frac{B^2}{\mu_0}) \\ \frac{1}{\mu_0}(\mathbf{v}\mathbf{B} - \mathbf{B}\mathbf{v}) \\ (e + p + \frac{1}{2}\frac{B^2}{\mu_0})\mathbf{v} - (\mathbf{B}\cdot\mathbf{v})\frac{\mathbf{B}}{\mu_0} \end{bmatrix} =$$

$$\nabla \cdot \begin{bmatrix} 0 \\ \mu\overleftrightarrow{\tau} \\ \overleftrightarrow{E}_{res} \\ \mu\mathbf{v}\cdot\overleftrightarrow{\tau} - \frac{1}{\mu_0^2}[\overleftrightarrow{\eta}\cdot(\nabla \times B)] \times \mathbf{B} + \overleftrightarrow{k}\cdot\nabla T \end{bmatrix}, \tag{2.60}$$

where the viscosity has been pulled out of the expression for the stress tensor in Eqn (2.45) and $\overleftrightarrow{\tau}$ represents the spatial derivatives of the velocity components only.

Eqn (2.60) has eight partial differential equations and nine unknowns. The unknowns are density (ρ), three components of velocity vector ($\mathbf{v}$), three components of the magnetic field ($\mathbf{B}$), total energy (e) and pressure (p). The ninth equation that closes the system is the equation of state. An equation of state defines the relationship between the pressure, density and temperature of the plasma. The simplest equation of state is the ideal gas equation of state $p/\rho = RT$, where R is the ideal gas constant. For a plasma that obeys the ideal equation of state the specific (per mass) internal energy (ϵ) depends on temperature only. Substitution of the ideal gas equation of state in the expression for total energy gives the closing equation for Eqn (2.60)

$$e = \frac{p}{\gamma - 1} + \frac{1}{2}\rho v^2 + \frac{1}{2}\frac{B^2}{\mu_0}, \tag{2.61}$$

where $\gamma = c_p/c_v$ is the ratio of the specific heats. It has to be noted that other equations of state can be used to close the system of Eqns (2.60) either in analytical or tabular form.

The system of Eqns (2.60) is a mixed set of hyperbolic and parabolic partial differential equations (PDEs). The left hand side of Eqn (2.60) contains the ideal MHD terms. If the right hand side were zero, then the system would be purely hyperbolic. In the context of conservation laws hyperbolic PDEs describe wave-like propagation of perturbations in the conserved variables. The right hand side of Eqn (2.60) contains dissipation or diffusive terms. These terms are generally referred to as the parabolic terms and they are associated with diffusion of momentum (viscous effects), diffusion of magnetic field (resistive effects) and diffusion of energy (heat conduction).

2.5.1 Normalization of the Single Fluid MHD Equations

It is useful to write the system of Eqns (2.60) in non-dimensional or normalized form. In non-dimensional form the problems solved using this model are self-similar and thus can be scaled. The scaling or reference variables are the characteristic plasma size (a), magnetic field (B_{ref}), density (ρ_{ref}), resistivity (η_{ref}), kinematic

viscosity (ν_{ref}), and heat conduction coefficient (k_{ref}). The Alfvèn speed, defined at the reference magnetic field and density, $c_a = B_{ref}/\sqrt{\mu_0 \rho_{ref}}$, is used as the reference speed. The non-dimensional variables, denoted with a bar, are obtained using the following expressions

$$\begin{aligned}
\bar{x} &= \frac{x}{a},\ \bar{y} = \frac{y}{a},\ \bar{z} = \frac{z}{a} \\
\bar{t} &= \frac{t}{a/c_a} \\
\bar{\rho} &= \frac{\rho}{\rho_{ref}} \\
\bar{\mathbf{v}} &= \frac{\mathbf{v}}{c_a}, \\
\bar{\mathbf{B}} &= \frac{\mathbf{B}}{B_{ref}}, \\
\bar{e} &= \frac{e}{B_{ref}^2/\mu_0} = \frac{e}{\rho_{ref} c_a^2} \\
\bar{p} &= \frac{p}{B_{ref}^2/\mu_0}, \\
\bar{\nu} &= \frac{\nu}{\nu_{ref}},\ \bar{\eta} = \frac{\eta}{\eta_{ref}},\ \bar{k} = \frac{k}{k_{ref}}.
\end{aligned}$$

For reference purposes the normalization of the mass conservation equations Eqn (2.37) and momentum conservation equation Eqn (2.44) are presented. Normalization of the induction equation Eqn (2.50) and energy conservation equation Eqn (2.52) is similar.

Each of the variables in the mass and momentum conservation equations is multiplied and divided by its corresponding reference variable (which is a constant). For the mass conservation equation this leads to

$$\frac{\rho_{ref} c_a}{a} \frac{\partial \bar{\rho}}{\partial \bar{t}} + \frac{\rho_{ref} c_a}{a} \bar{\nabla}(\bar{\rho}\bar{\mathbf{v}}) = 0,$$

where $\bar{\nabla} = \partial/\partial \bar{x}_i$ is the normalized differential operator. Cancellation of the $\rho_{ref} c_a/a$ term leads to an equation similar to the dimensional mass conservation equation but in terms of normalized conserved variables.

A similar procedure applied to the momentum conservation equation yields

$$\frac{\rho_{ref}c_a^2}{a}\frac{\partial}{\partial \bar{t}}(\bar{\rho}\bar{\mathbf{v}}) + \frac{\rho_{ref}c_a^2}{a}\bar{\nabla}\cdot[\bar{\rho}\bar{\mathbf{v}}\bar{\mathbf{v}} - \bar{\mathbf{B}}\bar{\mathbf{B}} + \overleftrightarrow{I}(\bar{p} + \frac{1}{2}\bar{B}^2)] = -\frac{\mu_{ref}}{c_a a^2}\bar{\mu}\bar{\nabla}\cdot\overleftrightarrow{\tau},$$

where $\mu_{ref} = \rho_{ref}\nu_{ref}$ is the reference dynamic viscosity. Division of each side of the equation by $\rho_{ref}c_a^2/a$ gives the non-dimensional form of momentum conservation equation

$$\frac{\partial}{\partial \bar{t}}(\bar{\rho}\bar{\mathbf{v}}) + \bar{\nabla}\cdot[\bar{\rho}\bar{\mathbf{v}}\bar{\mathbf{v}} - \bar{\mathbf{B}}\bar{\mathbf{B}} + \overleftrightarrow{I}(\bar{p} + \frac{1}{2}\bar{B}^2)] = -\frac{1}{ReAl}\bar{\mu}\bar{\nabla}\cdot\overleftrightarrow{\tau},$$

where $Re = Va/\nu_{ref}$ and $Al = c_a/V$ are two similarity parameters.

The induction and energy conservation equations are normalized similarly and the non-dimensional form of the single fluid MHD equations is

$$\frac{\partial}{\partial t}\begin{bmatrix}\rho\\ \rho\mathbf{v}\\ \mathbf{B}\\ e\end{bmatrix} + \nabla\cdot\begin{bmatrix}\rho\mathbf{v}\\ \rho\mathbf{v}\mathbf{v} - \mathbf{B}\mathbf{B} + \overleftrightarrow{I}(p + \frac{1}{2}B^2)\\ \mathbf{v}\mathbf{B} - \mathbf{B}\mathbf{v}\\ (e + p + \frac{1}{2}B^2)\mathbf{v} - (\mathbf{B}\cdot\mathbf{v})\mathbf{B}\end{bmatrix} =$$

$$\nabla\cdot\begin{bmatrix}0\\ (ReAl)^{-1}\bar{\mu}\overleftrightarrow{\tau}\\ (LuAl)^{-1}\overleftrightarrow{E}_{res}\\ (ReAl)^{-1}\bar{\mu}\mathbf{v}\cdot\overleftrightarrow{\tau} - (LuAl)^{-1}[\overleftrightarrow{\eta}\cdot(\nabla\times\mathbf{B})]\times\mathbf{B} + (PeAl)^{-1}\overleftrightarrow{k}\cdot\nabla T\end{bmatrix}, \quad (2.62)$$

from where the bar notation has been dropped for the conserved variables and the tensors.

The expression for non-dimensional total energy is

$$e = \frac{p}{\gamma - 1} + \frac{1}{2}\rho v^2 + \frac{1}{2}B^2, \quad (2.63)$$

from where the bar notation has also been dropped.

The non-dimensional or similarity parameters are

$$\text{Reynolds number: } Re = \frac{aV}{\nu_{ref}} \tag{2.64}$$

$$\text{Lundquist number: } Lu = \frac{\mu_0 aV}{\eta_{ref}} \tag{2.65}$$

$$\text{Peclet number: } Pe = \frac{aV}{k_{ref}} \tag{2.66}$$

$$\text{Alfvèn number: } Al = \frac{c_a}{V} \tag{2.67}$$

where V is the magnitude of the velocity ($V = \sqrt{\mathbf{v} \cdot \mathbf{v}}$), and the reference viscosity, resistivity and heat conduction coefficients are used.

The similarity parameters have important meanings and they are discussed briefly. The Reynolds number (Re) is a measure of the ratio of the inertial force to the viscous force in the fluid. If Re is small then the effect of viscosity is important throughout the flow. If Re is large then the effect of viscosity is important only in regions where shear flows are present such as in a region near a solid boundary (called a boundary layer.) The Lundquist number (Lu) describes how the magnetic field is influenced by the fluid flow and is a measure of the ratio between the flow velocity and magnetic field diffusion velocity. For $Lu \gg 1$ the magnetic field will move with the flow (frozen-in field) and for $Lu \ll 1$ the magnetic field will not be noticeably influenced by plasma motion. The Peclet number is a measure of the ratio between heat advection and heat conduction. At large Pe the heat advection is much more important than heat conduction.

2.5.2 Summary of Approximations and Limitations

The approximations introduced throughout the derivation of the single fluid MHD model limit the applicability of the model to plasmas that have certain properties. To summarize, the approximations made and the limits imposed are

- Plasma is sufficiently hot that only singly positive ions and electrons are present and the plasma is neutral ($n_e = n_i$.)

- Ions and electrons are in equilibrium and have Maxwellian distributions and their temperatures are assumed equal.
- Collisions between like and unlike particles are elastic Coulomb collisions.
- Electromagnetic waves of interest have phase velocities less than the speed of light and the characteristic thermal speeds are non-relativistic.
- Electron inertia is neglected.
- The MHD time scales must be long compared to the characteristic electron times
- The scale length of the system must be large compared to the ion Larmor radius and the Debye length

Chapter 3

NUMERICAL ALGORITHM

This chapter presents the methods used to obtain the numerical solution of the system of PDEs described by Eqn (2.62). The first section gives a broad overview of the numerical methods used in the code and of the concepts used throughout the derivation of the numerical algorithm. It is followed by a section describing the explicit scheme that will facilitate the understanding of the methods used throughout the code. Formulation of the implicit scheme and a description of its solution algorithm are given in the last section.

3.1 Overview - Numerical Solution of the Single Fluid MHD Model

Eqns (2.62) describe a system of non-linear, coupled hyperbolic-parabolic PDEs. The hyperbolic part, obtained by neglecting the right-hand side of Eqn (2.62), is in general associated with wave-like propagation of perturbations in the conserved variables and is sometimes called the advective part of the equations. The hyperbolic system is also known as the ideal MHD model. To obtain the parabolic part of the system, the second term of the left-hand side of Eqn (2.62) is neglected. The parabolic PDEs describe diffusion-like behavior and in general they describe how some of the conserved variables diffuse or decay due to the viscous and resistive nature of the fluid. For a problem to be defined the system of equations has to be accompanied by a set of initial and boundary conditions. The initial conditions specify the values of the conserved variables at the beginning of the simulation. The boundary conditions specify the values of the conserved variables or of their gradients at the boundaries of the domain

and they effectively describe how the fluid under study interacts with the outside world. The problem thus defined is called an initial boundary value problem (IBVP).

The goal of this work has been to develop a robust code that solves numerically IBVPs defined by Eqn (2.62) and the appropriate initial and boundary conditions with high temporal and spatial accuracy on complex geometries. Implementation of the numerical methods described in the following sections resulted in a code called WARP3 for University of **W**ashington **A**pproximate **R**iemann solver for **P**lasmas in three (**3**) dimensions.

A finite volume scheme has been used for the implementation of the algorithm. Finite volume schemes are a class of schemes used to obtain a discretized representation of Eqn (2.62). Discretization of the domain on which the solution of the system of PDEs is sought is accomplished by dividing the domain into cells. The result of this discretization is called a grid or mesh. The points (or nodes) of the grid are the vertices of the cells making up the domain. WARP3 uses structured grids which have hexahedral cells. (A hexahedron is a six-face-polyhedron.) Cell ordering of a structured grid is implicit, meaning that information related to the cells, e.g. the position of their centroids, or values of the conserved variables in each cell are stored in and retrieved from three dimensional arrays. The convention used in WARP3 is that positive j direction is $North$, positive i direction is $East$, negative j direction is $South$, negative i direction is $West$. In the third dimension the positive k direction is Top and negative k direction is $Bottom$. For example the cell at position (i,j,k) has cell ($i-1$,j,k) as its western neighbor and cell ($i+1$,j,k) as its eastern neighbor and so-forth, as illustrated in Figure 3.1. The advantage of implicit ordering is the ease of access to cell related data, since variables are stored in three-dimensional arrays.

The alternative to structured grids and implicit ordering is unstructured grids and explicit ordering. Unstructured grids, more common to finite element methods, typically have tetrahedral (four-face) cells. The explicit ordering information of unstructured grids is implemented either through linked-lists or mapping arrays that add

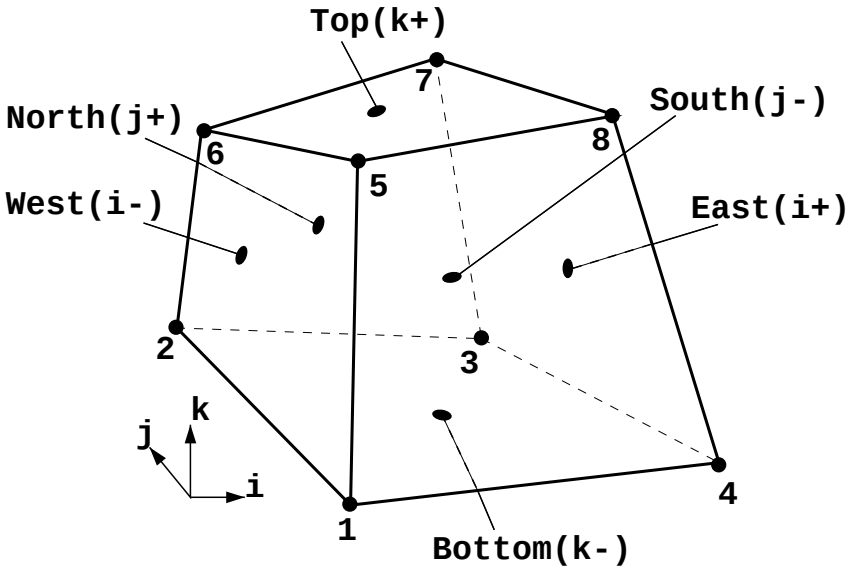

Figure 3.1: Cell of a three-dimensional structured grid showing the (i,j,k) directions, face notation convention, and vertex numbering.

to the storage requirements. The advantage of unstructured grids is the relative ease of generating grids about or inside complicated geometries. A finite volume method such as that described in this section is equally applicable on both types of grids.

Time variation of the conserved variables in each cell is calculated based on the fluxes of the conserved variables that leave and enter the cell through its faces. These fluxes are vector valued functions derived such that the overall scheme has certain properties such as high spatial accuracy and good shock capturing. They are called numerical fluxes and are not to be confused with the fluxes that appear in Eqn (2.62). A particular class of finite volume schemes called approximate Riemann solvers is used to compute the hyperbolic numerical fluxes at cell faces. Details of the implementation of these schemes is presented in Section 3.2.2. For the calculation of the parabolic numerical fluxes a staggered grid approach is used. The staggered grid is a fictional grid that is overlayed on the top of the real grid and its elements are calculated "on the fly" when they are needed for the computation of the parabolic numerical

fluxes. Section 3.2.6 presents details of the algorithm used to compute the parabolic fluxes. Both hyperbolic and parabolic numerical fluxes are calculated using the values of the conserved variables at the same time level. This is equivalent to having the solution advecting and diffusing at the same time. Some schemes use fractional steps and for example apply first the algorithm that solves for an intermediate advective solution. Then the values of this intermediate solution are used to calculate the diffusive solution.

The code can be run in both explicit and implicit modes. An explicit method computes the solution at time level $n+1$ using only the solutions from previous time levels, that have been already calculated, e.g. $\mathbf{Q}^{n+1} = f(\mathbf{Q}^n, \mathbf{Q}^{n-1}, \dots)$. An implicit scheme computes the solution at time level $n+1$ using the solutions at the current time level and previous time levels $\mathbf{Q}^{n+1} = f(\mathbf{Q}^{n+1}, \mathbf{Q}^n, \mathbf{Q}^{n-1}, \dots)$. Basic stability analysis of one-dimensional linear PDEs, such as the linear wave equation, shows that the time step, used in the discretization of the time variable, is limited to values proportional to the wave (or signal) speed and inversely proportional to the cell size. These explicit time step restrictions are dictated by the Courant-Friederichs-Levy condition which states that the time step ($\Delta t = t_{n+1} - t_n$) taken to advance the solution to the next time level has to be small enough so that the fastest wave does not cross the boundaries of the neighboring cells. Similar stability analysis shows that in the case of implicit schemes this stability constraint relaxes and larger time steps can be taken. Although not provable directly the results of this analysis carry on to non-linear three-dimensional systems of hyperbolic PDEs. Implicit schemes are beneficial when the ratio of the time scales in the problem of interest is large (the problem is said to be stiff) or when a steady state solution is sought. The disadvantage of implicit schemes is that they require the solution of large sparse linear systems of equations. However, the implicit scheme can be designed such that storage of large sparse matrices is avoided. All the components of the code used in the explicit mode are also used in the implicit mode so the work required to upgrade and maintain both

types of schemes in a single code is minimal. The formulation of the implicit scheme is presented in Section 3.3 along with implementation details.

3.2 Explicit Scheme and Finite Volume Formulation

In this section the concepts used in the discretization of Eqn (2.62) and in the derivation of the explicit algorithm are introduced. The concepts presented are general and can be applied to any system of hyperbolic-parabolic PDEs. However, some of the issues presented are peculiar to the MHD model and the original contributions will be emphasized.

3.2.1 Numerical Solution of the Hyperbolic System of PDEs

A hyperbolic system of PDEs is written in tensor form as

$$\frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot \overset{\leftrightarrow}{f}_h(\mathbf{q}) = 0, \tag{3.1}$$

where $\mathbf{q}$ is the conserved variables vector and $\overset{\leftrightarrow}{f}_h(\mathbf{q})$ is the hyperbolic flux tensor which is a function of the conserved variables that has the flux vectors as components. In orthogonal Cartesian coordinates $\overset{\leftrightarrow}{f}_h = \mathbf{f}_h\mathbf{i}+\mathbf{g}_h\mathbf{j}+\mathbf{h}_h\mathbf{k}$, where $\mathbf{f}_h$,$\mathbf{g}_h$, and $\mathbf{h}_h$ are the flux vectors in the x, y, and z directions and $(\mathbf{i},\mathbf{j},\mathbf{k})$ are the unit vectors along the axes. Hyperbolicity of the system implies that the eigenvalues of the flux Jacobians (e.g. $\partial\mathbf{f}_h/\partial\mathbf{q}$) are real and their eigenvectors are linearly independent. For the sake of simplicity the subscript h that denotes hyperbolic terms is dropped in the following sections.

A finite volume approach is used to find an approximate solution to Eqns (3.1). This is accomplished by taking the volume integral of Eqn (3.1) over a cell $\mathcal{C}$

$$\frac{\partial}{\partial t}\int_{V_c} \mathbf{q} d\tau + \int_{V_c} \nabla \cdot \overset{\leftrightarrow}{f}(\mathbf{q}) d\tau = 0. \tag{3.2}$$

The value of the conserved variables is assumed constant inside cell $\mathcal{C}$ and $\mathbf{Q}$ is used instead of $\mathbf{q}$ in Eqn (3.2) to emphasize the approximation. Application of the diver-

gence theorem lowers the order of the flux integral from a volume to a surface integral. Thus Eqn (3.2) becomes

$$V_{\mathcal{C}} \frac{\partial \mathbf{Q}}{\partial t} + \oint_{\Sigma_{\mathcal{C}}} d\sigma \cdot \overleftrightarrow{f}(\mathbf{Q}) = 0 \tag{3.3}$$

where $V_{\mathcal{C}}$ is the volume of cell $\mathcal{C}$, $\Sigma_{\mathcal{C}}$ is the surface bounding the cell, and $d\sigma$ is an infinitesimally small area element of $\Sigma_{\mathcal{C}}$. The area element $d\sigma$ is a vector normal to the $\Sigma_{\mathcal{C}}$ and oriented such that points away from the cell center.

The time derivative of the conserved variables is discretized using a first order finite difference approximation so that

$$\frac{\partial \mathbf{Q}}{\partial t} \approx \frac{\mathbf{Q}^{n+1} - \mathbf{Q}^{n}}{\Delta t} \tag{3.4}$$

where $\Delta t = t_{n+1} - t_n$ is the time step and Q^n is the approximate solution to Eqn (3.1) at time t_n. The surface integral is approximated by a sum over the faces of the cell

$$\oint_{\Sigma} d\sigma \cdot \overleftrightarrow{f}(\mathbf{Q}) \approx \sum_{i=1}^{faces} \overleftrightarrow{F}(\mathbf{Q}^n) \cdot \mathbf{n}_i A_i \tag{3.5}$$

where the flux of the conserved variables $\overleftrightarrow{f}(\mathbf{Q})$ has been replaced with the numerical flux $\overleftrightarrow{F}(\mathbf{Q}^n)$, $\mathbf{n}_i$ is the unit vector normal to face i and A_i is the area of face i. The numerical flux $\overleftrightarrow{F}$ is calculated at cell faces. Replacing Eqns (3.4) and (3.5) in Eqn (3.3) yields a formula for the time update of the solution to the hyperbolic system of PDEs

$$\mathbf{Q}^{n+1} = \mathbf{Q}^{n} - \frac{\Delta t}{V_{\mathcal{C}}} \sum_{i=1}^{faces} \overleftrightarrow{F}(\mathbf{Q}^n) \cdot \mathbf{n}_i A_i. \tag{3.6}$$

If the numerical flux tensor $\overleftrightarrow{F}$ were evaluated at $\mathbf{Q}^{n+1}$ instead of $\mathbf{Q}^n$ the scheme would have been implicit. At this point it is useful to note that the update formula given by Eqn (3.6) was derived without any reference to the particular type of cell used. Thus the scheme is applicable to any type of cell.

Two popular classes of methods for calculating the numerical fluxes are the flux vector split methods such as that of Steger and Warming [69] and the approximate

Riemann solver of Roe [57]. These schemes are examples of upwind schemes in which physical information propagation is introduced into the discretization of the governing equations.

The flux vector split scheme is not practical for the single fluid MHD model since an appropriate flux vector splitting cannot be found. Brio and Wu [11] developed a method that requires an adiabatic index γ=2 to obtain the analytic flux vector split.

Recent efforts to develop approximate Riemann solvers for ideal MHD have been successful. It seems that Zachary *et* al. [85, 86] where the first to develop a successful approximate Riemann solver method for the ideal MHD equations by introducing a normalization of the eigenvectors of the hyperbolic system that avoids degeneracies. Dai and Woodward [18] also developed an approximate Riemann solver, however their method is more complicated than Zachary's and involves shock tracking. Powell *et* al. [54] developed a successful multidimensional approximate Riemann solver that incorporates eigenvector normalizations similar to Zachary's and an additional source term that leads to the elimination of a null eigenvalue. In WARP3 the eigenvectors of the MHD system are calculated with a method similar to Powell's and a detailed description of the eigensystem is presented in 3.2.3. Cargo and Gallice [14, 13, 15, 12] also developed an approximate Riemann solver for the one-dimensional ideal MHD equations. Their approach has been studied in an attempt to extend it to three spatial dimensions. However, as shown in Appendix D the eigensystem obtained is too complicated to be useful in any multidimensional algorithm. More recently Aslan [4, 2, 3] has developed a method for deriving the numerical hyperbolic fluxes for the ideal MHD equations with an original sonic fix.

An approximate Riemann solver was originally implemented in WARP3 by Jones [43, 44] that used a method derived by Yee *et* al. for the one and two-dimensional Euler equations. Jones' original contribution was to implement a modified version of Yee's scheme into the MHD algorithm and to be the first to show that it is possible to solve the single fluid MHD model on three-dimensional, curvilinear grids with an

approximate Riemann solver. However, the method he chose to calculate the numerical fluxes was highly complex and led to errors in the implementation. A simple test presented in the Section 5 reveals one of the errors of Jones' implementation. The new algorithm presented in this work follows Jones' approach but significantly reduces the the complexity of the numerical fluxes and improves the robustness and reliability of the code. The new method resolves the vector variables in directions normal to the face cells and computes a flux at the face based on the solution of a Riemann problem normal to the cell face. The flux vectors obtained in the locally aligned coordinate system are resolved to the Ox, Oy and Oz axes and the conserved variables are updated in the $Oxyz$ frame of reference.

Approximate Riemann Solvers

A Riemann problem is an initial value problem consisting of a system of conservation equations (system of strictly hyperbolic PDEs) and an initial discontinuity in one or more of the conserved variables. The concepts introduced in this section are readily understandable if a one-dimensional system of conservation laws is used to derive the algorithm. Eqn (3.1) is reduced from three-dimensions to one-dimension

$$\frac{\partial \mathbf{q}}{\partial t} + \frac{\partial \mathbf{f}}{\partial x} = 0. \tag{3.7}$$

Application of the chain rule to the derivative of the flux gives the quasi-linear form of Eqn (3.7)

$$\frac{\partial \mathbf{q}}{\partial t} + A\frac{\partial \mathbf{q}}{\partial x} = 0, \tag{3.8}$$

where $A = \partial \mathbf{f}/\partial \mathbf{q}$ is the flux Jacobian. Since boundaries are not treated in a purely Riemann solver the domain is the entire Ox axis. The Ox axis is discretized by dividing it into intervals, each of the intervals being a cell. Cell $\mathcal{C}_i$ represents the interval between $x_{i-\frac{1}{2}}$ and $x_{i+\frac{1}{2}}$ ($\mathcal{C}_i = \{x : x_{i-\frac{1}{2}} \leq x \leq x_{i-\frac{1}{2}}\}$). As illustrated in Figure 3.2 the fractional indices denote values at cell interfaces. For example $x_{i+\frac{1}{2}}$ is the coordinate of the face common to cells i and $i+1$.

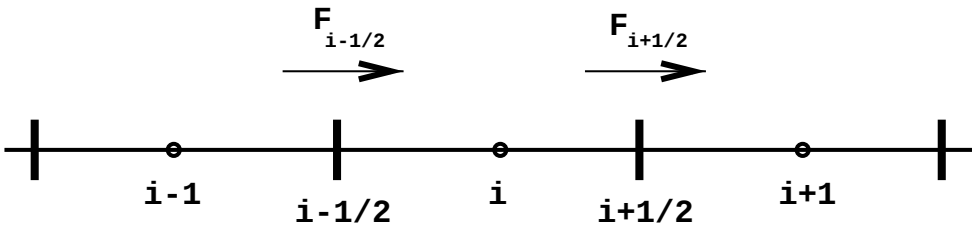

Figure 3.2: Three adjacent cells of the one-dimensional domain. Solid circles represent the cells centers and cell interfaces are denoted by fractional indices.

With these notations the one-dimensional update formula is

$$\mathbf{Q}_i^{n+1} = \mathbf{Q}_i^n - \frac{\Delta t}{x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}}(\mathbf{F}_{i+\frac{1}{2}}^n - \mathbf{F}_{i-\frac{1}{2}}^n), \tag{3.9}$$

where $\mathbf{F}_{i+\frac{1}{2}}^n$ is the value of the numerical flux calculated at interface $i+\frac{1}{2}$ based on values of the conserved variables at time step n.

Pairs of states $(\mathbf{Q}_L, \mathbf{Q}_R)$ at the left and right of cell interfaces define a sequence of Riemann problems. A Riemann solver algorithm was devised first by Godunov [27] for the one-dimensional Euler equations, and was reproduced by Richtmeyer and Morton [56] and Holt [32]. Godunov's assumption was that the conserved variables are constant in a cell and he used an analytic method to find the exact solution of the simplified Riemann problem thus obtained. The solution in the entire domain of interest is obtained by solving the sequence of Riemann problems at cell interfaces. It is possible however to increase the accuracy of the method by prescribing a certain variation of the conserved variables inside a cell. Some methods use piecewise linear variation (PLM) or piecewise parabolic variation (PPM). In WARP3 the conserved variables are assumed constant inside a cell, consistent with the discretizations presented in Eqns (3.6) and (3.9).

For multi-dimensional systems of conservative equations exact solutions of the Riemann problem are either difficult to obtain or would involve a prohibitive amount of computer time so that a further simplification of the method was devised by Roe.

Roe's contribution [57] was to generalize the application of an algorithm developed for the linear advection equation

$$\frac{\partial Q}{\partial t} + a\frac{\partial Q}{\partial x} = 0, \qquad a = \text{const.} \tag{3.10}$$

to the case of non-linear systems. Roe's approximate Riemann solver finds the exact solution to an approximate problem

$$\frac{\partial \mathbf{Q}}{\partial t} + \tilde{A}\frac{\partial \mathbf{Q}}{\partial x} = 0, \tag{3.11}$$

where $\tilde{A}$ is a constant flux Jacobian chosen so that it is representative of local conditions. Eqn (3.11) is sometimes referred to as the "locally frozen" system since $\tilde{A}$ is constant inside each cell.

The solution of the Riemann problem at the interface is a similarity solution. The value of the conserved variables at the cell interface ($\mathbf{Q}_{i+\frac{1}{2}}$) is a function only of the states at the left and right of the interface $\mathbf{Q}_{i+\frac{1}{2}} = f(\mathbf{Q}_i, \mathbf{Q}_{i+1})$, provided that the time step is small enough that the waves from the Riemann problem do not interfere. By carefully constructing $\tilde{A}$ to be "representative of local conditions", the approximate Riemann solver contains the necessary and sufficient physical information to yield the correct solution to the original non-linear system of hyperbolic PDEs. For $\tilde{A}$ to be a valid construct it has to satisfy certain properties. Roe has identified these properties and he named them collectively property U to emphasize that they ensure *uniform* validity across discontinuities.

The four properties are

1. $\tilde{A}$ constitutes a linear mapping from the vector space $\mathbf{Q}$ to the vector space $\mathbf{F}$;

2. As $\mathbf{Q}_L \to \mathbf{Q}_R, \tilde{A} \to A(\mathbf{Q})$;

3. For any pair $(\mathbf{Q}_L, \mathbf{Q}_R), \tilde{A}(\mathbf{Q}_L, \mathbf{Q}_R)(\mathbf{Q}_L - \mathbf{Q}_R) = \mathbf{F}_L - \mathbf{F}_R$;

4. The eigenvectors of $\tilde{A}$ are linearly independent.

Property **1** allows the hyperbolic system to be locally decoupled, meaning that in each cell, each of the hyperbolic PDEs can be treated as a linear advection equation. The non-linearity of the global system of PDEs is maintained, however, since the wave speeds and wave strengths are non-linear functions of all conserved variables. Property **2** is necessary to recover smoothly the non-linear system from the linear system. Property **3** is a sufficient condition for the algorithm produced to be conservative (proven by Roe.) Finally, properties **3** and **4** are necessary and sufficient conditions for the algorithm to recognize a shock wave. (If $(\mathbf{Q}_L, \mathbf{Q}_R)$ satisfy the Rankine-Hugoniot jump condition across a shock then $\mathbf{F}_L - \mathbf{F}_R = S(\mathbf{Q}_L - \mathbf{Q}_R)$, where S is the shock speed.)

The approximate flux Jacobian $\tilde{A}$ contains a wealth of information in its eigensystem. Its eigenvalues are associated with the wave speeds of the Riemann problem, the projection of the jump in conserved variables across the interface on the left eigenvectors of $\tilde{A}$ are the jumps which take place between intermediate states, and the right eigenvectors point in the direction of propagation of the waves (in phase space).

For illustration the eigensystems of the approximate flux Jacobian of two one-dimensional Riemann problems are analyzed. The first example is the one-dimensional gas dynamics Riemann problem modeled by the one-dimensional Euler equations and an initial discontinuity in the conserved variables. The approximate flux Jacobian of this problem has three eigenvalues $v_x + c$, v_x and $v_x - c$ where c is the speed of sound. The values for the intermediate state between the v_x and $v_x + c$ characteristics (Figure 3.3) is obtained by vector dot multiplication of the jump across the interface $(\mathbf{Q}_L - \mathbf{Q}_R)$ with the left eigenvector of the flux Jacobian corresponding to the eigenvalue $v_x + c$. The second example is the one-dimensional MHD Riemann problem. This problem is modeled by the one-dimensional version of Eqn (2.62) and an initial jump in the conserved variables. The approximate flux Jacobian has eight eigenvalues corresponding to each of the waves present in the solution. Two waves are collapsed and they are propagating with v_x (Figure 3.4). The other six waves propagate with

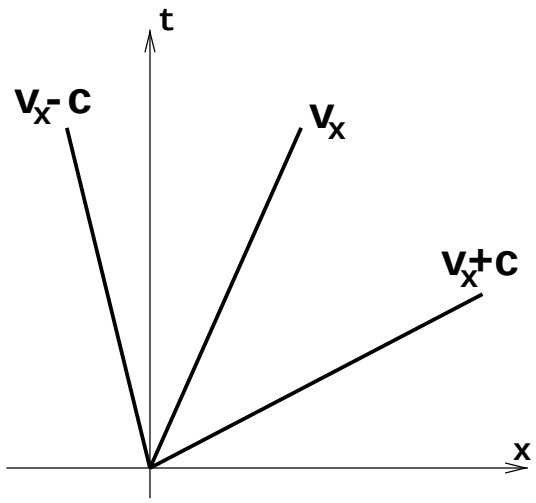

Figure 3.3: The three characteristics of the one-dimensional Euler (gas dynamics) problem

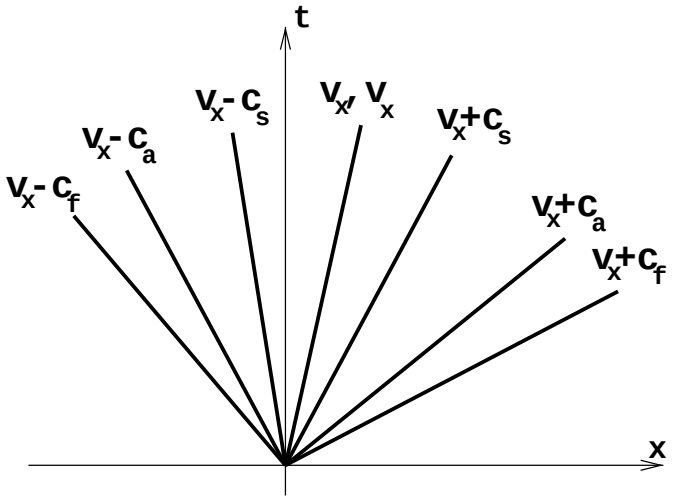

Figure 3.4: The eight characteristics of the one-dimensional MHD problem

$v_x \pm c_f$, $v_x \pm c_s$ and $v_x \pm c_a$, where c_f is the fast magneto-acoustic speed, c_s is the slow magneto-acoustic speed, and c_a is the Alfvèn speed. (A detailed description of the eigensystem of the single fluid MHD model is presented in Section 3.2.3.)

The only drawback of a Roe type approximate Riemann solver is that the solution admits only shocks. Thus the numerical fluxes have to incorporate an *entropy* or *sonic* fix that identifies the presence of rarefaction waves and incorporates them into the solution.

3.2.2 Hyperbolic Numerical Fluxes

This section describes the concepts behind the derivation of the numerical hyperbolic fluxes. The desired properties of the scheme are incorporated into the solution algorithm through these fluxes. These properties are high accuracy and capturing of discontinuities without spurious oscillations. In the context of this work "high accuracy" means that the spatial accuracy of the algorithm is second order in regions where the solution is smooth and first order in the regions where sharp gradients of the conserved variables are present. Temporal accuracy of the explicit scheme is first order. Shock capturing without oscillations is a property formally known as total variation diminishing (TVD) or total variation non-increasing (TVNI). Total variation of a scheme is defined in terms of the solution at time step n, $\mathbf{Q}^n$, as

$$TV(\mathbf{Q}^n) = \sum_{i=-\infty}^{\infty} |\mathbf{Q}_{i+1}^n - \mathbf{Q}_i^n|. \tag{3.12}$$

A scheme is TVD if the total variation of the solution at time step $n+1$ is less than or equal to the total variation at time step n, $TV(\mathbf{Q}^{n+1}) \leq TV(\mathbf{Q}^n)$.

To arrive at a scheme that has both properties an approach similar to that of Yee *et* al. [84] has been followed. Their recipe for deriving the numerical hyperbolic fluxes is not the only method of incorporating the desired properties into the algorithm. For example, Aslan and Kammash [4] developed an original method to find the numerical

fluxes. Yee's method has been preferred due to the experience gained during the development of WARP3.

The recipe for constructing a numerical flux that produces the desired properties of a scheme is to combine a high order flux ($\mathbf{F}^H$) with a low order flux ($\mathbf{F}^L$). (By high or low order flux it is meant a a numerical flux that gives high or respectively low accuracy to the scheme.) The high order flux can be viewed as consisting of the low order flux plus a correction

$$\mathbf{F}^H = \mathbf{F}^L + (\mathbf{F}^H - \mathbf{F}^L), \tag{3.13}$$

where the correction is the second term of the right hand side. In a flux limiter method, such as that used by Yee, the magnitude of the correction is limited depending on the smoothness of the data. Thus Eqn (3.13) becomes

$$\mathbf{F} = \mathbf{F}^L + \Phi(\mathbf{F}^H - \mathbf{F}^L), \tag{3.14}$$

where the flux limiter is Φ. The role of the flux limiter is to smoothly switch the scheme between high and low accuracy. In the regions of space where the solution is smooth the flux limiter should take null or close to null values so that the scheme has high accuracy in those regions. In the regions where the solution has sharp gradients the limiter should take values that make the scheme low order accurate and thus avoid spurious, non-physical oscillations. An illustration of the effects of non-physical oscillations produced by a non-TVD scheme are shown in Figure 3.5. LeVeque showed in [46] that for the scheme to be TVD the flux limiter has to be bound. Sweeby [72] showed that if the high order flux ($\mathbf{F}^H$) is second order accurate then for the scheme to be second order accurate Φ has to pass smoothly through $\Phi(1) = 1$.

The choice for the low order flux in Yee's scheme (and the majority of modern schemes) is a first order upwind scheme. The flux at the interface between cells i and $i+1$ is

$$\mathbf{F}^L_{i+\frac{1}{2}} = \frac{\mathbf{F}_i + \mathbf{F}_{i+1}}{2} - \frac{1}{2} sgn(\tilde{\Lambda}_{i+\frac{1}{2}})(\mathbf{F}_{i+1} - \mathbf{F}_i), \tag{3.15}$$

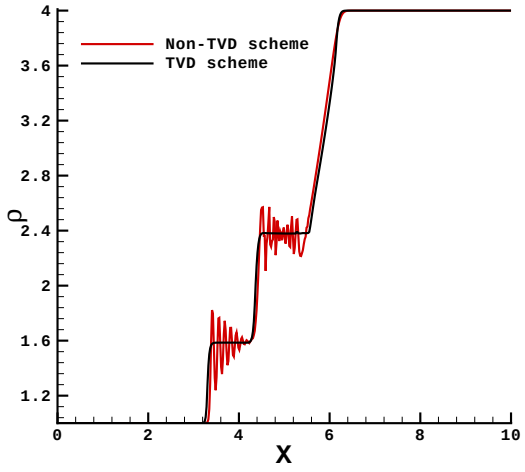

Figure 3.5: TVD and non-TVD solutions for a gas dynamics Riemann problem. The non-physical oscillations of the non-TVD schemes are present downstream of the shock and contact discontinuity.

where $\mathbf{F}_i = \mathbf{F}(\mathbf{Q}_i)$, $\tilde{\Lambda}_{i+\frac{1}{2}}$ is a diagonal matrix with elements equal to the eigenvalues of the approximate flux Jacobian $\tilde{A}_{i+\frac{1}{2}}$, and the *signum* function is applied to each of the elements of $\tilde{\Lambda}_{i+\frac{1}{2}}$. The *signum* function returns the sign of its argument

$$sgn(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

The upwinding property of the scheme is illustrated by assuming that the m-th eigenvalue of $\tilde{A}$ is positive. The first order flux corresponding to this eigenvalue, obtained by substitution in Eqn (3.15), is $\mathbf{F}^{L,m}_{i+\frac{1}{2}} = \mathbf{F}^m_i$. A positive eigenvalue corresponds to a right-going wave so that information present "upwind" at i is carried by the wave to the interface $i + \frac{1}{2}$. (The characteristics of the right-going waves are at the right of the time axis in Figures 3.3 and 3.4.)

From the Rankine-Hugoniot condition (properties **3** and **4** of Roe's property *U*)

the flux difference can be expressed as

$$\mathbf{F}_{i+1} - \mathbf{F}_i = \tilde{A}_{i+\frac{1}{2}}(\mathbf{Q}_{i+1} - \mathbf{Q}_i) \tag{3.16}$$

The eigenvector decomposition of $\tilde{A}$ ($\tilde{A} = \tilde{R}\tilde{\Lambda}\tilde{L}$), is used to obtain

$$\mathbf{F}_{i+1} - \mathbf{F}_i = (\tilde{R}\tilde{\Lambda}\tilde{L})_{i+\frac{1}{2}}(\mathbf{Q}_{i+1} - \mathbf{Q}_i), \tag{3.17}$$

where $\tilde{R}$ is the right eigenvectors matrix, and $\tilde{L} = \tilde{R}^{-1}$ is the left eigenvectors matrix.

Replacing Eqn (3.17) in Eqn (3.15) yields

$$\mathbf{F}^L_{i+\frac{1}{2}} = \frac{\mathbf{F}_i + \mathbf{F}_{i+1}}{2} - \frac{1}{2}(\tilde{R}|\tilde{\Lambda}|\tilde{L})_{i+\frac{1}{2}}(\mathbf{Q}_{i+1} - \mathbf{Q}_i), \tag{3.18}$$

where $|\tilde{\Lambda}| = sgn(\tilde{\Lambda})\tilde{\Lambda}$. The strength of the m-th wave is the dot-product of the m-th left eigenvector ($\mathbf{l}^m$) and the jump of the conserved variables across the interface

$$\alpha^m_{i+\frac{1}{2}} = \mathbf{l}^m_{i+\frac{1}{2}} \cdot (\mathbf{Q}_{i+1} - \mathbf{Q}_i). \tag{3.19}$$

With this definition the second term of the right hand side of Eqn (3.18) can be written as a sum over the conserved variables

$$\mathbf{F}^L_{i+\frac{1}{2}} = \frac{\mathbf{F}_i + \mathbf{F}_{i+1}}{2} - \frac{1}{2}\sum_m (|\tilde{\lambda}^m|\tilde{\alpha}^m\tilde{\mathbf{r}}^m)_{i+\frac{1}{2}}. \tag{3.20}$$

If the numerical flux given by Eqn (3.20) were used in the update formula in Eqn (3.9) the scheme would be a first order accurate in space Roe-type approximate Riemann solver.

At this point the sonic or entropy fix is introduced in the formula. The fix identifies rarefaction waves and introduces a small dissipation necessary for the algorithm to solve for the physically admissible solution that does not violate the entropy condition. The entropy condition states that the entropy of a gas that passes through a shock wave increases. An approximate Riemann solver of Roe-type produces only shock waves so that a rarefaction shock wave replaces a smooth rarefaction wave (also known as an expansion fan.) This rarefaction shock wave violates the entropy condition since

the entropy should decrease through a rarefaction. Yee's scheme corrects this non-physical solution by using a fix that smoothes out the eigenvalues in a vicinity around zero. The absolute value of the eigenvalue is replaced by a function that switches between the absolute value and a parabolic variation so that

$$\sigma(\lambda^m) = \begin{cases} |\lambda^m| & \text{if } |\lambda^m| \geq \epsilon \\ \frac{(\lambda^m)^2+\epsilon^2}{2\epsilon} & \text{otherwise,} \end{cases} \tag{3.21}$$

where ϵ is a small number. Another sonic fix was developed by van Leer *et* al. [78] that treats each wave differently. However their fix is computationally more intensive and does not seem to bring any major benefits. The final form of the first order numerical flux used in WARP3 is

$$\mathbf{F}^L_{i+\frac{1}{2}} = \frac{\mathbf{F}_i + \mathbf{F}_{i+1}}{2} - \frac{1}{2}\sum_m \sigma(\tilde{\lambda}^m_{i+\frac{1}{2}})\tilde{\alpha}^m_{i+\frac{1}{2}}\tilde{\mathbf{r}}^m_{i+\frac{1}{2}}, \tag{3.22}$$

where the fractional index $i+\frac{1}{2}$ denotes values calculated at the interface between cells i and $i+1$.

The derivation of the high order correction flux $\mathbf{F}^H$ used in Yee's scheme rests on the fact that the exact solution to Eqn (3.7) is TVD due to the propagation of information along characteristics, and is independent of the particular form of the flux. Thus to achieve second order accuracy the flux in Eqn (3.22) is modified to be

$$\mathbf{F}_{i+\frac{1}{2}} = \frac{\mathbf{F}_i + \mathbf{F}_{i+1}}{2} + \frac{1}{2}\sum_m \Phi^m_{1+\frac{1}{2}}\tilde{\mathbf{r}}^m_{i+\frac{1}{2}}, \tag{3.23}$$

where the flux limiter is

$$\Phi^m_{i+\frac{1}{2}} = g^m_i + g^m_{i+1} - \sigma(\tilde{\lambda}_{i+\frac{1}{2}} + \gamma^m_{i+\frac{1}{2}})\tilde{\alpha}^m_{i+\frac{1}{2}}, \tag{3.24}$$

and

$$g^m_i = S \;\; max[0, min(\frac{1}{2}\sigma^m_{i+\frac{1}{2}}|\tilde{\alpha}^m_{i+\frac{1}{2}}|, \frac{S}{2}\sigma^m_{i-\frac{1}{2}}\tilde{\alpha}^m_{i-\frac{1}{2}})], \tag{3.25}$$

$$S = sgn(\tilde{\lambda}^m_{i+\frac{1}{2}}), \;\; \sigma^m_{i+\frac{1}{2}} = \sigma(\tilde{\lambda}^m_{i+\frac{1}{2}}) \tag{3.26}$$

$$\gamma^m_{i+\frac{1}{2}} = \begin{cases} \frac{g^m_{i+1}-g^m_i}{\tilde{\alpha}^m_{i+\frac{1}{2}}}, & \tilde{\alpha}^m_{i+\frac{1}{2}} \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.27}$$

The derivation is presented by Harten in [28, 29] and Yee *et* al. in [84]. The limiter in the flux formula in Eqn (3.23) is the so-called *minmod* limiter. Sweeby [72] reviewed a number of flux limiters that have the properties necessary to make a general flux limiter scheme TVD and highly accurate in space. The most notable ones are

$$\begin{aligned}
\text{minmod} \quad & \Phi(\theta) = max(0, min(\theta, 1)) \\
\text{superbee} \quad & \Phi(\theta) = max(0, min(2\theta, 1), min(\theta, 2))) \\
\text{van Leer} \quad & \Phi(\theta) = \frac{|\theta| + \theta}{1 + \theta} \\
\text{van Albada} \quad & \Phi(\theta) = \frac{2\theta^2 + \theta}{2\theta^2 - \theta + 2}
\end{aligned}$$

All these limiters lay inside the second-order TVD region found by Sweeby and they also pass smoothly through point $\Phi(1) = 1$. This last property is a necessary condition for the overall scheme to be at least second order accurate [46]. In the context of Yee's scheme θ is

$$\theta = S \frac{\sigma^m_{i+\frac{1}{2}} |\tilde{\alpha}^m_{i+\frac{1}{2}}|}{\sigma^m_{i-\frac{1}{2}} \tilde{\alpha}^m_{i-\frac{1}{2}}},$$

and the limiters are actually used in the correction fluxes g_{i+1} and g_i in Eqn (3.24). A graph of the various limiters and of the second order TVD scheme is shown in Figure 3.6. It is interesting to note that that the minmod limiter lays on the lower boundary of the region and the superbee limiter lays on the top boundary of the region. The van Leer limiter has an asymptote at $\Phi = 2$ and the van Albada limiter has an asymptote at $\Phi = 1$. All four limiters listed above have been tested in the algorithm and it has been found that the van Leer, van Albada, and the minmod flux limiters give almost the same results, with the minmod limiter being slightly less diffusive. The superbee limiter has been found to be too compressional. A test case with smooth initial data such as a sine wave transformed into a square wave as the solution evolved in time if the superbee limiter were used. Also, for the case of a one-dimensional gas dynamics Riemann problem the superbee limiter produced very

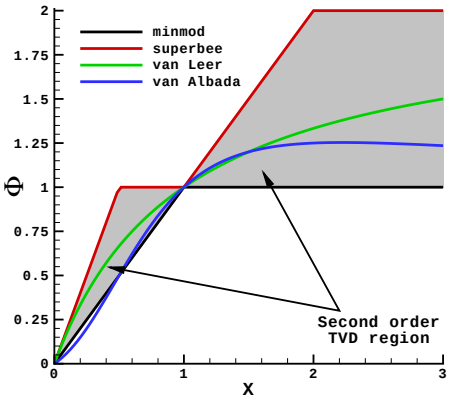

Figure 3.6: Plots of various flux limiters showing that they lay inside the the second order TVD region. (The second order TVD region has a solid background)

sharp discontinuities but also some post-shock oscillations in the solution. Since the minmod limiter required the least calculations it has been chosen as the default flux limiter in WARP3.

3.2.3 Eigensystem of the Single Fluid MHD Model

To this point the method described makes no reference to the hyperbolic system used so that theoretically it is applicable to any system of hyperbolic PDEs. In this section the eigensystem of the single fluid MHD model used in WARP3 is introduced. It is this eigensystem that differentiates the MHD model from the regular gas dynamics model. Not only the single fluid MHD model has three extra variables (the components of the magnetic field) but the eigenvectors have to be normalized to avoid degeneracies and the flux Jacobians are modified to eliminate null eigenvalues. The derivation of the eigensystem used in WARP3 is similar to that presented by Powell *et* al. [54] and by Roe and Balsara [58].

The hyperbolic part of Eqn (2.62) given below in tensor form

$$\frac{\partial}{\partial t}\begin{bmatrix} \rho \\ \rho\mathbf{v} \\ \mathbf{B} \\ e \end{bmatrix} + \nabla \cdot \begin{bmatrix} \rho\mathbf{v} \\ \rho\mathbf{v}\mathbf{v} - \mathbf{B}\mathbf{B} + \overleftrightarrow{I}(p + \frac{1}{2}B^2) \\ \mathbf{v}\mathbf{B} - \mathbf{B}\mathbf{v} \\ (e + p + \frac{1}{2}B^2)\mathbf{v} - (\mathbf{B}\cdot\mathbf{v})\mathbf{B} \end{bmatrix} = 0, \tag{3.28}$$

is written using flux vectors rather than the flux tensor. In orthogonal Cartesian coordinates the system is

$$\frac{\partial \mathbf{q}}{\partial t} + \frac{\partial \mathbf{f}}{\partial x} + \frac{\partial \mathbf{g}}{\partial y} + \frac{\partial \mathbf{h}}{\partial z} = 0, \tag{3.29}$$

where $\mathbf{f}$, $\mathbf{g}$, and $\mathbf{h}$ are the flux vectors in the x, y, and z directions. For example the flux vector in the x direction is

$$\mathbf{f} = \begin{bmatrix} \rho v_x \\ \rho v_x^2 - B_x^2 + p + B^2/2 \\ \rho v_x v_y - B_x B_y \\ \rho v_x v_z - B_x B_z \\ 0 \\ v_x B_y - B_x v_y \\ v_x B_z - B_x v_z \\ (e + p + B^2/2)v_x - (\mathbf{B}\cdot\mathbf{v})B_x \end{bmatrix}. \tag{3.30}$$

The flux Jacobians appear in the quasilinear form of Eqn (3.29)

$$\frac{\partial \mathbf{q}}{\partial t} + A_c\frac{\partial \mathbf{q}}{\partial x} + B_c\frac{\partial \mathbf{q}}{\partial y} + C_c\frac{\partial \mathbf{q}}{\partial z} = 0, \tag{3.31}$$

where $A_c = \partial\mathbf{f}/\partial\mathbf{q}$, $B_c = \partial\mathbf{g}/\partial\mathbf{q}$, and $C_c = \partial\mathbf{h}/\partial\mathbf{q}$ are the flux Jacobians, and the subscript c is used to emphasize that the flux Jacobians are calculated in terms of conservative variables. The flux Jacobians have simpler expressions if they were calculated in terms of primitive variables rather than in terms of conserved variables.

The primitive variables for the MHD model are density, velocity vector, magnetic field vector and pressure

$$\mathbf{w} = [\rho, v_x, v_y, v_z, B_x, B_y, B_z, p]^T. \tag{3.32}$$

In terms of primitive variables the quasilinear form is

$$\frac{\partial \mathbf{q}}{\partial t} + A_p \frac{\partial \mathbf{w}}{\partial x} + B_p \frac{\partial \mathbf{w}}{\partial y} + C_p \frac{\partial \mathbf{w}}{\partial z} = 0, \tag{3.33}$$

where $A_w = \partial \mathbf{f}/\partial \mathbf{w}$, $B_w = \partial \mathbf{g}/\partial \mathbf{w}$, and $C_w = \partial \mathbf{h}/\partial \mathbf{w}$ are the flux Jacobians in terms of primitive variables. A straightforward transformation is applied to obtain the flux Jacobian in terms of conserved variables. The transformation formulas and their derivation are presented in Appendix A.

The Jacobian of the flux in the x direction in terms of primitive variables is

$$A_p = \begin{bmatrix} v_x & \rho & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & v_x & 0 & 0 & -\frac{B_x}{\rho} & \frac{B_y}{\rho} & \frac{B_z}{\rho} & \frac{1}{\rho} \\ 0 & 0 & v_x & 0 & -\frac{B_y}{\rho} & -\frac{B_x}{\rho} & 0 & 0 \\ 0 & 0 & 0 & v_x & -\frac{B_z}{\rho} & 0 & -\frac{B_x}{\rho} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & B_y & -B_x & 0 & -v_y & v_x & 0 & 0 \\ 0 & B_z & 0 & -B_x & -v_z & 0 & v_x & 0 \\ 0 & \gamma p & 0 & 0 & (\gamma - 1)\mathbf{v} \cdot \mathbf{B} & 0 & 0 & v_x \end{bmatrix}. \tag{3.34}$$

A Riemann solver based on the eigensystem of A_p would not work since this flux Jacobian has a zero eigenvalue which is non-physical. (Since the primitive and conservative forms of the flux Jacobian are equivalent this null eigenvalue is present in both formulations and is not peculiar to the primitive formulation.) The approach taken by Powell [54] and followed in the implementation of WARP3 is to modify A_p so that it becomes non-singular. The properties of the new flux Jacobian A'_p should be

- The eigenvalues and eigenvectors corresponding to the seven waves in the one-dimensional ($B_x = \text{const.}$) Riemann problem remain unchanged;
- The eigenvalue corresponding to the new eighth wave is equal to v_x;
- The left and right eigenvectors corresponding to the new eighth wave "make sense";
- In the case $B_x = \text{const.}$, the eight-wave Riemann problem reduces to the seven-wave Riemann problem.

The resulting modified Jacobian that satisfies these requirements is

$$A'_p = \begin{bmatrix} v_x & \rho & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & v_x & 0 & 0 & 0 & \frac{B_y}{\rho} & \frac{B_z}{\rho} & \frac{1}{\rho} \\ 0 & 0 & v_x & 0 & 0 & -\frac{B_x}{\rho} & 0 & 0 \\ 0 & 0 & 0 & v_x & 0 & 0 & -\frac{B_x}{\rho} & 0 \\ 0 & 0 & 0 & 0 & v_x & 0 & 0 & 0 \\ 0 & B_y & -B_x & 0 & 0 & v_x & 0 & 0 \\ 0 & B_z & 0 & -B_x & 0 & 0 & v_x & 0 \\ 0 & \gamma p & 0 & 0 & 0 & 0 & 0 & 0 & v_x \end{bmatrix}. \tag{3.35}$$

The eigenvalues of A'_p are

(1) Entropy wave: $\lambda_e = v_x$

(2,3) Alfvén waves: $\lambda_a = v_x \pm c_a$

(4,5) Fast magneto-acoustic waves: $\lambda_f = v_x \pm c_f$

(6,7) Slow magneto-acoustic waves: $\lambda_s = v_x \pm c_s$

(8) Magnetic flux wave: $\lambda_d = v_x$,

where c_a is the Alfvén speed, and c_f and c_s are the fast and respectively slow magneto-acoustic speeds. The expressions for the Alfvén and magneto-acoustic wave speeds

are

$$c_a = \frac{B_x}{\sqrt{\rho}} \tag{3.36}$$

$$c_{f,s}^2 = \frac{1}{2}\left[\frac{\gamma p + B^2}{\rho} \pm \sqrt{\left(\frac{\gamma p + B^2}{\rho}\right)^2 - 4\frac{\gamma p B_x^2}{\rho^2}}\right], \tag{3.37}$$

where the fast magneto-acoustic speed corresponds to the plus in front of the square root, and $B^2 = B_x^2 + B_y^2 + B_z^2$.

The eighth wave, named the magnetic flux wave, was introduced by the modifications that rendered the flux Jacobian non-singular. The eigenvectors of A'_p are reproduced in Appendix B for reference.

Powell notes that the first seven waves yield the same eigenvectors and eigenvalues as the seven-wave Riemann problem, with the additional property that none of them carries a change in B_x (the fifth element of each right eigenvector is null), and none of the wave strenghts is proportional to a jump in B_x (the fifth element of each left eigenvector is null.) The new eighth wave travels with the x-component of the flow speed and it carries a jump in B_x (the only non-zero element in the left eigenvector corresponds to B_x), and it affects only the x-component of the magnetic field (the only non-zero element in the right eigenvector corresponds to B_x.)

Since the expressions for the eigenvectors are simpler in the primitive form the flux calculations in WARP3 are implemented using this form. The wave strength of the m-th wave is calculated as

$$\alpha^m = \mathbf{l}_p^m \cdot (\mathbf{w}_{i+1} - \mathbf{w}_i), \tag{3.38}$$

where $\mathbf{l}_p^m$ is the left eigenvector corresponding to variable m derived in primitive form. The wave strength obtained with Eqn (3.38) is equivalent to the one obtained in Eqn 3.19). The right eigenvectors (also calculated in primitive form) are transformed to conservative form in order to have the numerical flux formula in Eqn (3.20) in conservative form. The transformation from primitive to conservative form is explained in Appendix A and the formula used is on the right of Eqn (A.10).

The other two flux Jacobians B_p and C_p are modified similarly so that they become non-singular. The effect of these transformations on the original system of Eqns (3.28) is shown by collecting the terms that modify the original flux Jacobians and by reverting the system of PDEs to conserved variables. The resulting system of PDEs is

$$\frac{\partial}{\partial t}\begin{bmatrix}\rho\\ \rho\mathbf{v}\\ \mathbf{B}\\ e\end{bmatrix}+\nabla\cdot\begin{bmatrix}\rho\mathbf{v}\\ \rho\mathbf{v}\mathbf{v}-\mathbf{B}\mathbf{B}+\overleftrightarrow{I}(p+\frac{1}{2}B^2)\\ \mathbf{v}\mathbf{B}-\mathbf{B}\mathbf{v}\\ (e+p+\frac{1}{2}B^2)\mathbf{v}-(\mathbf{B}\cdot\mathbf{v})\mathbf{B}\end{bmatrix}=-\begin{bmatrix}0\\ \mathbf{B}\\ \mathbf{v}\\ \mathbf{v}\cdot\mathbf{B}\end{bmatrix}\nabla\cdot\mathbf{B}. \qquad (3.39)$$

The algebra involved in this transformation is straightforward if lengthy and has been checked both by hand and with the symbolic mathematics package Mathematica.

The left hand side of Eqn (3.39) is exactly the same with the left hand side of Eqn (3.28) and the right hand side is a source term proportional to $\nabla\cdot\mathbf{B}$. At the PDE level, only terms that are equal to zero have been added to the conservative form of the governing equations. Indeed Eqns (3.39) are the mathematically correct equations if during their derivation (presented in Chapter 2) $\nabla\cdot\mathbf{B}=0$ were not used. If $\nabla\cdot\mathbf{B}=0$ at the beginning of the simulation than Eqns (3.28) and (3.39) are equivalent. From a numerical point of view the formulation in Eqn (3.39) is preferable since the eigensystem of the flux Jacobians is complete. More than that, in the case $\nabla\cdot\mathbf{B}$ is created locally due to truncation errors these non-physical deviations of the divergence will be advected away in the case of a domain with open boundaries. If the domain has closed boundaries then non-zero $\nabla\cdot\mathbf{B}$ can accumulate and give non-physical results. The effects that non-zero $\nabla\cdot\mathbf{B}$ has on the solution of the MHD equations are analyzed by Brackbill and Barnes [10]. For the simulations that model a closed domain a "divergence cleaning" procedure can be turned on in WARP3. This "divergence cleaning" procedure is based on the Hodge projection method. A Hodge

projection method defines a modified field, $\mathbf{B}'$

$$\mathbf{B}' = \mathbf{B} + \nabla b \tag{3.40}$$

where b is a scalar calculated such that $\mathbf{B}'$ is divergence free, and $\mathbf{B}$ is the original non-zero divergence field. Taking the divergence of Eqn (3.40) gives

$$\nabla \cdot \mathbf{B}' = \nabla \cdot \mathbf{B} + \nabla^2 b = 0 \tag{3.41}$$

The divergence of the original field $\nabla \cdot \mathbf{B}$ is known so that the equation for b is an elliptic PDE of Poisson type

$$\nabla^2 b = -\nabla \cdot \mathbf{B}. \tag{3.42}$$

The PDE in Eqn (3.42) is solved by an iterative method for elliptic equations. Th divergence cleaning that uses a Hodge projection method is used in other MHD codes such as that of Zachary *et* al. [86].

The eigenvectors of the modified flux Jacobian in Eqn (3.35) have not been normalized and are prone to scaling problems in the following cases:

1. $B_x = 0$. In this case, both the Alfvén speed (c_a) and the slow magneto-acoustic speed (c_s) are null, leading to degenerate left and right eigenvectors for the slow waves.

2. $B_y^2 + B_z^2 = 0$, $B_x^2 \neq \gamma p$. In this case the fast magneto-acoustic speed becomes equal to the sound speed ($c_f = c(= \sqrt{\gamma p/\rho})$), and the slow magneto-acoustic speed becomes equal to the Alfvén speed ($c_s = c_a$). The left and right eigenvectors for the Alfvén waves become degenerate.

3. $B_y^2 + B_z^2 = 0$, $B_x^2 = \gamma p$. In this case all speeds become equal ($c_f = c_s = c_a = c$), and the left and right eigenvectors of the slow and fast magneto-acoustic waves become degenerate.

The design of a robust approximate Riemann solver requires that these degeneracies be treated at the analytical level rather than having a series of procedures for each case and calling the appropriate one if the corresponding condition is satisfied. A robust normalization was proposed by Zachary *et* al. [85] and refined by Roe and Balsara [58]. This normalization makes the set of eigenvectors ortho-normal (the dot-product of an eigenvector with itself is equal to one and the dot-product with the other eigenvectors is null.) Again, the algebra involved is straightforward but lengthy and has been checked analytically. The set of normalized eigenvectors that are used in WARP 3 are presented in Appendix C.

Last but not least the values of the conserved variables at cell interfaces have to be calculated (e.g. $\mathbf{Q}_{i+\frac{1}{2}}$). These values are needed for computing the numerical fluxes at cell interfaces ($\mathbf{F}_{i+\frac{1}{2}}$). Ideally the values of the conserved variables at cell interfaces are calculated so that they satisfy the Rankine-Hugoniot conditions for the jump across the interface. Roe [57] noticed that the difficulty in constructing $\tilde{A}$ is in satisfying property **3** and he proposes an elegant method for accomplishing it using a parameter vector $\mathbf{u}$ such that $\mathbf{Q} = \mathbf{Q}(\mathbf{u})$ and $\mathbf{F} = \mathbf{F}(\mathbf{u})$. Roe defined the jump in the conserved variables as $\mathbf{Q}_L - \mathbf{Q}_R = \tilde{B}(\mathbf{u}_L - \mathbf{u}_R)$, and the jump in the fluxes as $\mathbf{F}_L - \mathbf{F}_R = \tilde{C}(\mathbf{u}_L - \mathbf{u}_R)$, From the combination of these two equations it can be seen that $\tilde{A} = \tilde{C}\tilde{B}^{-1}$. The so called *Roe averages* of the flow variables are easily identified in $\tilde{A}$. For the three-dimensional gas dynamics equations (three-dimensional Euler equations) the parameter vector used by Roe was

$$\mathbf{u} = \rho^{1/2}[1, v_x, v_y, v_z, h]^T, \tag{3.43}$$

where $h = (e + p)/\rho$ is the total enthalpy per unit volume. The Roe averages for the

flow variables for three-dimensional Euler equations are

$$\begin{aligned}
\rho^{\text{face}} &= \rho_L^{1/2} + \rho_R^{1/2} \\
v_x^{\text{face}} &= \frac{\rho_L^{1/2} v_{x,L} + \rho_R^{1/2} v_{x,R}}{\rho_L^{1/2} + \rho_R^{1/2}} \\
v_y^{\text{face}} &= \frac{\rho_L^{1/2} v_{y,L} + \rho_R^{1/2} v_{y,R}}{\rho_L^{1/2} + \rho_R^{1/2}} \\
v_z^{\text{face}} &= \frac{\rho_L^{1/2} v_{z,L} + \rho_R^{1/2} v_{z,R}}{\rho_L^{1/2} + \rho_R^{1/2}} \\
h^{\text{face}} &= \frac{\rho_L^{1/2} h_L + \rho_R^{1/2} h_R}{\rho_L^{1/2} + \rho_R^{1/2}},
\end{aligned}$$

where the superscript **face** and subscripts L for *left* and R for *right* rather than $i+\frac{1}{2}$, i, and $i+1$ have been used to simplify the notation. The value of the pressure or of the total energy of the fluid at a cell face is calculated from the enthalpy at the cell face. Unfortunately these expressions cannot be applied directly to the ideal MHD equations.

A Roe average for the three-dimensional ideal MHD equations has not been found yet. Powell *et* al. [54] propose a Roe average but their method can lead to complex values for the Alfvén speed. Cargo and Gallice [14, 13, 15, 12] found Roe averages for the one-dimensional MHD equations and attempts to extend their method to three dimensions failed due to the complicated expressions for the eigenvalues and eigenvectors. Last but not least the method proposed by Aslan [2, 3] to find Roe averages works only for the cases when the magnetic field is constant in the direction parallel to that in which the Riemann problem is solved. Since this is not a realistic assumption Aslan's Roe averages are not very useful. (The expressions for Aslan's averages are similar to that of Roe's only that the average density is $\rho^{\text{face}} = \sqrt{\rho_L \rho_R}$.) Descriptions of two attempts to find a Roe averaged state and the corresponding eigensystem for the single fluid MHD equations using Powell's and then Cargo and Gallice's approaches are given in Appendix D.

In the computational MHD community it is accepted that schemes that use a simple arithmetic average to find the values of the conserved variables at the cell faces are acceptable and give the correct physical results. This is illustrated by the solutions of the one dimensional gas dynamics and MHD Riemann problems presented in the benchmark section.

3.2.4 Solution on Three-Dimensional Curvilinear Grids

This section describes how the hyperbolic PDEs in Eqn (3.39) are solved on three-dimensional curvilinear grids. A curvilinear grid is a grid that has the edges of the cells oriented at arbitrary angles with respect to each other. In contrast, an orthogonal grid has the edges of the cells normal to each other. The most common examples are orthogonal Cartesian or cylindrical polar grids. One approach used to solve Eqn (3.39) is the traditional approach of coordinate transformation used by Jones [43] and implemented by him in a previous version of the code. This approach is introduced only as a reference and is not detailed. The new and original approach that uses a locally aligned coordinate system is presented in detail and its advantages are discussed.

Traditional Approach - Coordinate Transformation

The approach traditionally used to obtain an update formula for the conserved variables on curvilinear, three dimensional grids is to transform the physical space defined by coordinates (x, y, z) in the so called computational space defined by the generalized coordinates (ξ, η, ζ). The transformation is defined such that the cells of a structured computational grid are cubes with the edge length equal to one. Details of the transformation procedure are presented in computational fluid dynamics texts such as those by Hoffmann [31] and Tannehill [74].

The hyperbolic conservation equation Eqn (3.1) is transformed to computational

space so that

$$\frac{\partial \tilde{\mathbf{q}}}{\partial t} + \nabla \cdot \overleftrightarrow{\mathcal{F}}(\tilde{\mathbf{q}}) = 0, \tag{3.44}$$

where $\tilde{\mathbf{q}} = J^{-1}\mathbf{q}$ is the transformed conserved variables vector, J^{-1} is the inverse of the Jacobian of the transformation, the transformed hyperbolic flux tensor is $\overleftrightarrow{\mathcal{F}}(\tilde{\mathbf{q}}) = \tilde{\mathbf{f}}\mathbf{n}_\xi + \tilde{\mathbf{g}}\mathbf{n}_\eta + \tilde{\mathbf{h}}\mathbf{n}_\zeta$, $\mathbf{n}_\xi$, $\mathbf{n}_\eta$, and $\mathbf{n}_\zeta$ are the unit vectors of the computational space axes, the flux in the ξ direction is $\tilde{\mathbf{f}} = \mathbf{q}\xi_t + \mathbf{f}\xi_x + \mathbf{g}\xi_y + \mathbf{h}\xi_z$, and ξ_t, ξ_x ,ξ_y, and ξ_z, are the derivatives of the coordinate ξ with respect to time and the x, y, z coordinates respectively.

If the update formula Eqn (3.6) is applied to the transformed system in Eqn (3.44), then the second term on the right hand side, which is the approximation of the surface integral of the fluxes, is particularly simple. Since the cells in the computational space are cubes with the edge equal to one their volume is equal to one ($V_\mathcal{C} = 1$), the face areas are equal to one ($A_i = 1$, $i = 1, 2, \ldots, 6$), and the unit vectors normal to the faces are equal to the unit vectors of the computational space axes. The update formula for cell (i, j, k) becomes

$$\tilde{\mathbf{Q}}^{n+1}_{i,j,k} = \tilde{\mathbf{Q}}^{n}_{i,j,k} - \Delta t[\delta_\xi \tilde{\mathbf{F}}^n + \delta_\eta \tilde{\mathbf{G}}^n + \delta_\zeta \tilde{\mathbf{H}}^n], \tag{3.45}$$

where $\tilde{\mathbf{F}}^n \equiv \tilde{\mathbf{F}}(\tilde{\mathbf{Q}}^n)$ is the numerical flux in the computational space evaluated at time level n and

$$\begin{aligned}
\delta_\xi \tilde{\mathbf{F}}^n &= \tilde{\mathbf{F}}^n_{i+\frac{1}{2},j,k} - \tilde{\mathbf{F}}^n_{i-\frac{1}{2},j,k} \\
\delta_\eta \tilde{\mathbf{G}}^n &= \tilde{\mathbf{G}}^n_{i,j+\frac{1}{2},k} - \tilde{\mathbf{G}}^n_{i,j-\frac{1}{2},k} \\
\delta_\zeta \tilde{\mathbf{H}}^n &= \tilde{\mathbf{H}}^n_{i,j,k+\frac{1}{2}} - \tilde{\mathbf{H}}^n_{i,j,k-\frac{1}{2}}.
\end{aligned}$$

Fractional indices such as $i + \frac{1}{2}$ denote a cell interface. For example $\tilde{\mathbf{F}}^n_{i+\frac{1}{2},j,k}$ is the flux calculated at the interface between cells (i, j, k) and $(i+1, j, k)$. This notation is illustrated for a two-dimensional grid in Fig. 3.7.

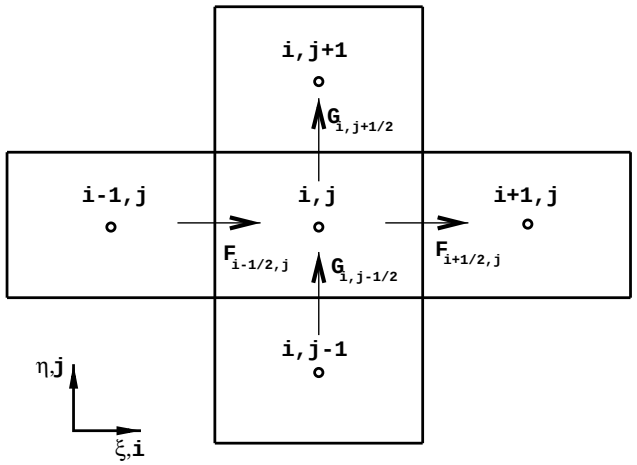

Figure 3.7: Cell (i, j) of a two-dimensional grid that illustrates cell center and flux at face notation.

Each of the fluxes is evaluated using a one-dimensional numerical flux formula such as that in Eqn (3.23) only that now the eigensystem is that of the fluxes expressed in generalized coordinates.

Unfortunately this approach, which is the norm for CFD algorithms for regular gas dynamics, is not directly applicable to the single fluid MHD model due to the complications that arise in the calculation of the numerical fluxes. These complications are encountered at the modification of the Jacobians of fluxes $\tilde{\mathbf{F}}$, $\tilde{\mathbf{G}}$, and $\tilde{\mathbf{H}}$ to eliminate the null eigenvalues and at the normalization of their eigenvectors. A generalized coordinates formulation for the single fluid MHD model has been attempted by Jones [43] but due to the extremely complicated algebra some errors are present in his numerical fluxes.

New Approach - Locally Aligned Coordinate System

Rather than using a transformation from Cartesian coordinates to generalized coordinates the original approach developed and implemented in WARP3 uses a locally aligned coordinate system. The vector conserved variables are the momentum per

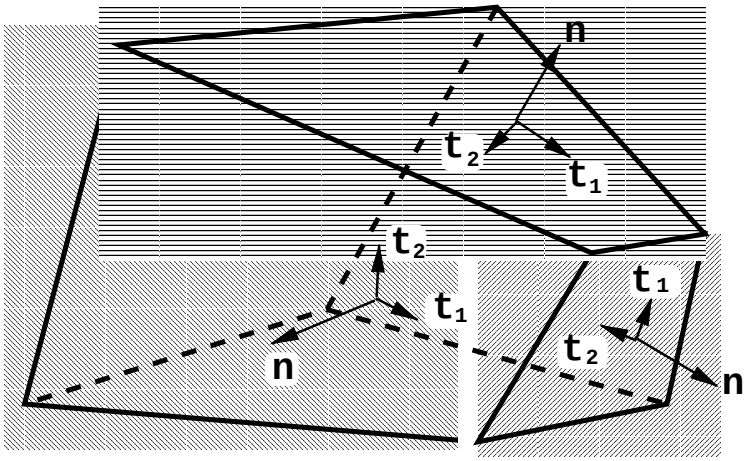

Figure 3.8: Illustration of the locally aligned coordinate system for faces of a hexahedral cell.

unit volume (ρv_x, ρv_y, ρv_z) and the magnetic field (B_x, B_y, B_z). They are initialized, stored and updated in the fixed orthogonal Cartesian reference frame defined by $Oxyz$. The idea behind the locally aligned coordinate system is to consistently rotate and align the momentum and magnetic field vectors from two cells sharing a face such that one component is normal to the face and the other two components are parallel to the face. The new coordinate system is still orthogonal Cartesian, with axis On normal to the face, and Ot_1 and Ot_2, parallel to the face as illustrated in Figure 3.8. (This notation is also used to identify the components of the vector conserved variables. For the magnetic field, for example, B_n is the component normal to the face and B_{t_1} and B_{t_2} are the components parallel to the face.) The advantage of this approach is that only one numerical flux vector and its Jacobian need to be calculated. The solution of the hyperbolic PDE system is now obtained by solving sequences of Riemann problems in directions normal to the faces. The Riemann problems are solved at the interfaces in a natural direction to generate fluxes that are automatically orthogonal to the interface. The fluxes obtained in the rotated reference frame (Ont_1t_2) are rotated back to the fixed reference frame ($Oxyz$) and

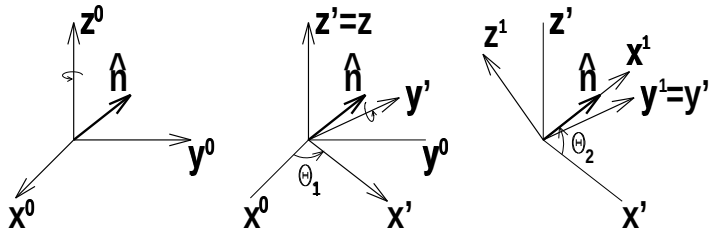

Figure 3.9: Rotation of a cell coordinate system that aligns x^o with the normal to the cell face **n**. The new Ox^1 axis corresponds to On, Oy^1 corresponds to Ot_1 and Oz^1 corresponds to Ot_2.

the values of the conserved variables are updated in that reference frame.

It has been found that to consistently align the local coordinates with a face only two rotations are needed. The angles between the normal to the face and the fixed $Oxyz$ reference are calculated and the axis corresponding to the minimum angle (in magnitude) is aligned with the normal. The direction of rotation is always positive trigonometric (counter-clockwise). The rotating coordinate frame is initially aligned with the $Oxyz$ axis and the axes are named x^o, y^o and z^o. For example, if x^o is to be aligned with the normal then the first rotation is about the z^o axis with an angle θ_1. The second rotation is about y', with an angle θ_2, which aligns the coordinate system with the interface normal. The new reference frame coordinate axes are (On, Ot_1, Ot_2). This particular rotation is illustrated in Figure 3.9 where the new Ox^1 axis corresponds to On, Oy^1 corresponds to Ot_1 and Oz^1 corresponds to Ot_2.

The three pairs of rotations used in WARP3 are given below for reference. In the following formulas n_x, n_y, and n_z are the components of the unit vector normal to the face.

1. If the x^o axis is aligned with the normal, then the rotation angles are

$$\theta_1 = \tan^{-1}\frac{n_y}{n_x}, \qquad \theta_2 = -\tan^{-1}\frac{n_z}{\sqrt{n_x^2+n_y^2}}, \tag{3.46}$$

and the rotation matrices are

$$R_1 = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{rotation about } z^o \tag{3.47}$$

$$R_2 = \begin{bmatrix} \cos\theta_2 & 0 & \sin\theta_2 \\ 0 & 1 & 0 \\ -\sin\theta_2 & 0 & \cos\theta_2 \end{bmatrix} \quad \text{rotation about } y'. \tag{3.48}$$

2. If the y^o axis is aligned with the normal, then the rotation angles are

$$\theta_1 = -\tan^{-1}\frac{n_x}{n_y}, \qquad \theta_2 = \tan^{-1}\frac{n_z}{\sqrt{n_x^2 + n_y^2}}, \tag{3.49}$$

and the rotation matrices are

$$R_1 = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{rotation about } z^o \tag{3.50}$$

$$R_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_2 & -\sin\theta_2 \\ 0 & \sin\theta_2 & \cos\theta_2 \end{bmatrix} \quad \text{rotation about } x'. \tag{3.51}$$

3. If the z^o axis is to be aligned with the normal, then the rotation angles are

$$\theta_1 = -\tan^{-1}\frac{n_y}{n_z}, \qquad \theta_2 = \tan^{-1}\frac{n_x}{\sqrt{n_y^2 + n_z^2}}, \tag{3.52}$$

and the rotation matrices are

$$R_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_2 & -\sin\theta_2 \\ 0 & \sin\theta_2 & \cos\theta_2 \end{bmatrix} \quad \text{rotation about } x^o \tag{3.53}$$

$$R_2 = \begin{bmatrix} \cos\theta_2 & 0 & \sin\theta_2 \\ 0 & 1 & 0 \\ -\sin\theta_2 & 0 & \cos\theta_2 \end{bmatrix} \quad \text{rotation about } y'. \tag{3.54}$$

To rotate the fluxes, obtained by solving the Riemann problem in a direction normal to the face, back to the $Oxyz$ coordinate system the inverses of the matrices given above are needed. Since the rotation matrices are orthogonal matrices their inverses are equal to their transposes so that there is no computation required to calculate them. The cost of applying a rotation (or an inverse rotation) is small and involves nine floating point operations (FLOPs) per vector variable per face. (A floating point operation (FLOP) is defined as a multiplication and addition operation performed by the arithmetic-logic unit of the processor of a digital computer.) Thus the total cost of rotations and inverse rotations is 36 FLOPs/face. The cost of these additional matrix multiplication operations is offset by the simplicity of the flux formula that requires less operations than those necessary to calculate the numerical fluxes in a generalized coordinates approach.

The update formula in Eqn (3.6) for the locally aligned coordinate system becomes

$$\mathbf{Q}^{n+1} = \mathbf{Q}^n - \frac{\Delta t}{V_C} \sum_{i=1}^{faces} \mathbf{F}_n^n A_i, \tag{3.55}$$

where $\mathbf{F}_n^n$ is the numerical flux calculated at time level n in the direction normal to the face. The formula for $\mathbf{F}_n^n$ is given by Eqn (3.23) with the eigensystem calculated in a locally aligned coordinate frame, as shown in Appendix C.

The new method has been found to be robust and was able to successfully match the analytical results obtained from linear theory for the tilt instability of spheromaks as shown in Section 5. The generalized coordinates version of the code failed during the first few iterations.

3.2.5 Numerical Solution of the Parabolic System of PDEs

A parabolic system of PDEs is written in tensor form as

$$\frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot \overleftrightarrow{f}_p(\mathbf{q}, \mathbf{q}', \mathcal{K}_d) = 0, \tag{3.56}$$

where $\mathbf{q}$ is the conserved variables vector, $\mathbf{q}'$ generically denotes the spatial derivatives of the conserved variables with respect to space, $\mathcal{K}_d$ generically denotes the diffusion coefficients, and $\overleftrightarrow{f}_p(\mathbf{q}, \mathbf{q}', \mathcal{K}_d)$ is the parabolic fluxes tensor which is a function of the conserved variables and their derivatives and of the diffusion coefficients. For the single fluid MHD model in Eqn (2.62) the non-dimensional (or normalized) the parabolic PDE part of the system is

$$\frac{\partial}{\partial t}\begin{bmatrix} \rho \\ \rho\mathbf{v} \\ \mathbf{B} \\ e \end{bmatrix} = \nabla\cdot\begin{bmatrix} 0 \\ (ReAl)^{-1}\bar{\mu}\overleftrightarrow{\tau} \\ (LuAl)^{-1}\overleftrightarrow{E}_{res} \\ (ReAl)^{-1}\bar{\mu}\mathbf{v}\cdot\overleftrightarrow{\tau} - (LuAl)^{-1}[\overleftrightarrow{\eta}\cdot(\nabla\times\mathbf{B})]\times\mathbf{B} + (PeAl)^{-1}\overleftrightarrow{k}\cdot\nabla T \end{bmatrix}, \tag{3.57}$$

where the non-dimensional parameters Re, Al, Lu, and Pe have been discussed in Section 2.5.1, $\overleftrightarrow{\tau}$ is the fluid shear stress tensor, $\overleftrightarrow{E}_{res}$ was used to simplify the notation and it is defined such that $\nabla\cdot\overleftrightarrow{E}_{res} = -\nabla\times[\overleftrightarrow{\eta}\cdot(\nabla\times\mathbf{B})]$, $\overleftrightarrow{\eta}$ is the electrical resistivity tensor, $\overleftrightarrow{k}$ is the heat conductivity tensor, and T is the temperature of the plasma.

The shear stress tensor is

$$\overleftrightarrow{\tau} = \begin{bmatrix} 2\frac{\partial v_x}{\partial x} - \frac{2}{3}\nabla\cdot\mathbf{v} & \frac{\partial v_x}{\partial y} + \frac{\partial v_y}{\partial x} & \frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x} \\ \frac{\partial v_y}{\partial x} + \frac{\partial v_x}{\partial y} & 2\frac{\partial v_y}{\partial y} - \frac{2}{3}\nabla\cdot\mathbf{v} & \frac{\partial v_y}{\partial z} + \frac{\partial v_z}{\partial y} \\ \frac{\partial v_z}{\partial x} + \frac{\partial v_x}{\partial z} & \frac{\partial v_z}{\partial y} + \frac{\partial v_y}{\partial z} & 2\frac{\partial v_z}{\partial z} - \frac{2}{3}\nabla\cdot\mathbf{v} \end{bmatrix}. \tag{3.58}$$

The expression for $\overleftrightarrow{E}_{res}$, obtained after some straightforward vector algebra, is

$$\overleftrightarrow{E}_{res} = \begin{bmatrix} 0 & \eta_z\left(\frac{\partial B_y}{\partial x} - \frac{\partial B_x}{\partial y}\right) & -\eta_y\left(\frac{\partial B_x}{\partial z} - \frac{\partial B_z}{\partial x}\right) \\ \eta_x\left(\frac{\partial B_z}{\partial y} - \frac{\partial B_y}{\partial z}\right) & 0 & -\eta_z\left(\frac{\partial B_y}{\partial x} - \frac{\partial B_x}{\partial y}\right) \\ \eta_y\left(\frac{\partial B_x}{\partial z} - \frac{\partial B_z}{\partial x}\right) & -\eta_x\left(\frac{\partial B_z}{\partial y} - \frac{\partial B_y}{\partial z}\right) & 0 \end{bmatrix}, \tag{3.59}$$

where the resistivity tensor has been assumed to be diagonally dominant

$$\overleftrightarrow{\eta} = \begin{bmatrix} \eta_x & 0 & 0 \\ 0 & \eta_y & 0 \\ 0 & 0 & \eta_z \end{bmatrix}. \tag{3.60}$$

The effect of this assumption is to eliminate the Hall resistivity from the model, which is consistent with the assumptions made in Section 2 that allowed the elimination of the Hall terms from the generalized Ohm's law.

At the time of this writing a procedure that treats the parabolic term that models the heat transfer in the plasma

$$\frac{\partial e}{\partial t} = \frac{1}{PeAl} \nabla \cdot (\overleftrightarrow{k} \cdot \nabla T)$$

is being implemented in WARP3 by another graduate student, Ward Vuillemot.

The treatment of the parabolic terms is similar to that of the hyperbolic terms in that a finite volume approach is used in the derivation of the numerical fluxes. However, each of the parabolic terms act upon a different set of conserved variables and it is a function of different variables. For example the viscous terms influence the momentum and the total energy of the plasma while the resistive terms affect the magnetic field and the total energy. In consequence numerical fluxes are derived for each of the parabolic terms. (The implementation of the heat transfer term which influences the total energy only is being implemented using a similar approach.)

3.2.6 Parabolic Numerical Fluxes

It is desired that the discretization of the parabolic terms give a spatial accuracy at least equal to that obtained from the discretization of the hyperbolic terms. Thus, the numerical parabolic fluxes are obtained using a second order finite volume approach. The formulas presented in the following subsections are rather complicated, since they are used to calculate the parabolic fluxes on curvilinear grids. However, it can be easily shown that if applied to an orthogonal Cartesian grid the formulas reduce to a second order spatial central differencing which is a rather standard method of discretizing parabolic PDEs.

Viscous Fluxes

The viscous terms from the parabolic system of PDEs influence the momentum and the total energy of the fluid. The equations that describe these influences are

$$\left.\frac{\partial(\rho \mathbf{v})}{\partial t}\right|_{\text{visc}} = \frac{1}{ReAl}\nabla \cdot \overleftrightarrow{\tau} \tag{3.61}$$

$$\left.\frac{\partial e}{\partial t}\right|_{\text{visc}} = \frac{1}{ReAl}\nabla \cdot (\overleftrightarrow{\tau} \cdot \mathbf{v}), \tag{3.62}$$

where the **vise** subscript has been used to emphasize that only the viscous terms are taken in account in the equations.

Each of the above equations are integrated over the volume of a cell. For example the integration of the momentum diffusion Eqn (3.61) gives

$$\int_{V_c} \left.\frac{\partial(\rho \mathbf{v})}{\partial t}\right|_{\text{visc}} d\tau = \frac{1}{ReAl}\int_{V_c} \nabla \cdot \overleftrightarrow{\tau} d\tau. \tag{3.63}$$

Similarly to the method used for the derivation of the hyperbolic terms the momentum (which is a conserved variable) is assumed constant inside the cell so that it can be taken out of the integral on the left hand side of Eqn (3.63). The integral term on the right hand side is also treated similarly to the flux integral of the hyperbolic part of the PDE system and its order is lowered by applying the divergence theorem. Thus the volume integral becomes a surface integral and the new equation is

$$\left.\frac{\partial(\rho \mathbf{v})}{\partial t}\right|_{\text{visc}} = \frac{1}{ReAlV_C}\int_{\Sigma_c} \overleftrightarrow{\tau} \cdot d\sigma, \tag{3.64}$$

where the equation has been divided through by the cell volume. A similar integration is used for the energy equation to yield

$$\left.\frac{\partial e}{\partial t}\right|_{\text{visc}} = \frac{1}{ReAlV_C}\int_{\Sigma_c} (\overleftrightarrow{\tau} \cdot \mathbf{v}) \cdot d\sigma. \tag{3.65}$$

The time derivative on the left hand side of each equations is substituted by a first order finite difference approximation, e.g. $\partial e/\partial t \approx (e^{n+1} - e^n)/\Delta t$. The integrals are approximated as sums over the faces of each cell and all the terms under the integral

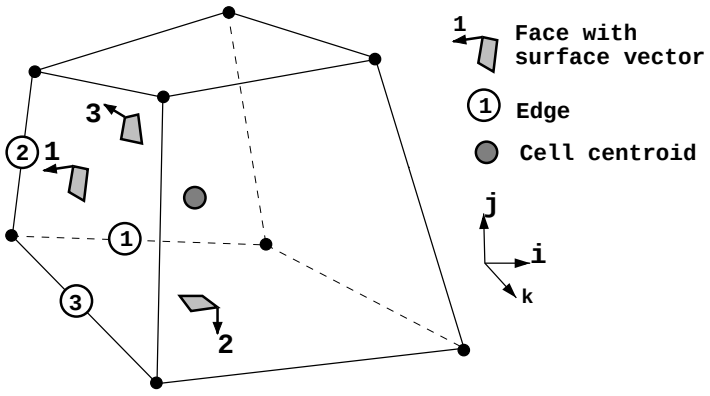

Figure 3.10: Face and vertex notation convention of a real cell used for the calculation of the parabolic fluxes. Face 1 area vector points in the negative i direction, face 2 area vector points in the negative j direction and face 3 area vector points in the negative k direction.

are computed explicitly using the values of the conserved variables at time level n. The update formulas for the momentum and energy parabolic PDEs are

$$(\rho\mathbf{v})^{n+1} = (\rho\mathbf{v})^n + \frac{\Delta t}{ReAlV_{\mathcal{C}}} \sum_{i=1}^{6} \overleftrightarrow{\tau} \cdot \mathbf{n}_i A_i \tag{3.66}$$

$$e^{n+1} = e^n + \frac{\Delta t}{ReAlV_{\mathcal{C}}} \sum_{i=1}^{6} (\overleftrightarrow{\tau} \cdot \mathbf{v}) \cdot \mathbf{n}_i A_i. \tag{3.67}$$

The terms containing the sums are called the viscous fluxes. In order to evaluate them a staggered grid approach is used. This approach is necessary since the spatial derivatives of the velocity in the stress tensor $\overleftrightarrow{\tau}$ have to be evaluated at the faces of the real cell $\mathcal{C}$. The elements of the cells of the staggered grid (volumes and face area vectors) are calculated on the fly as they are needed, using simple averages of the elements of the real cells. The face and edge notation used to identify the cells of the staggered grid is presented in Figures 3.10 and 3.11.

The contributions of all six faces of the auxiliary cell are added to obtain the full approximation of the stress tensor for face 1 of the real cell. Term xx of the stress

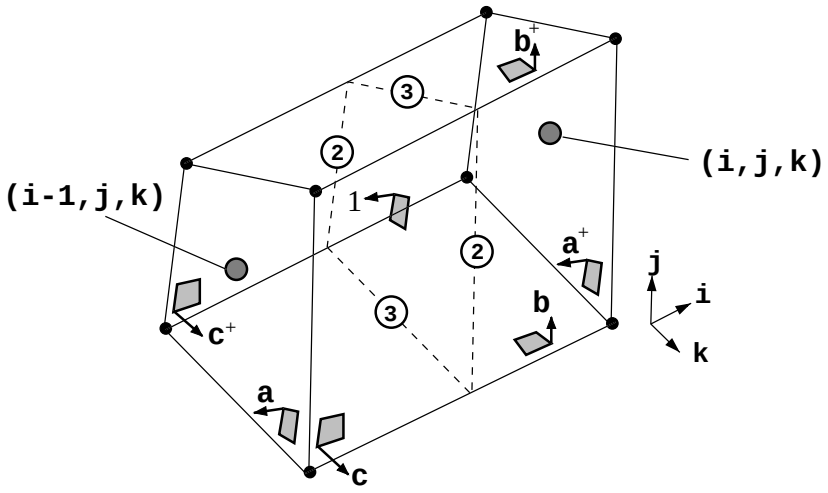

Figure 3.11: Auxiliary cell for face 1. Faces a and a^+ correspond to the i direction and pass through the centroids of cells $(i-1,j,k)$ and (i,j,k) respectively. Faces b and b^+ correspond to the j direction and faces c and c^+ correspond to the k direction.

tensor is calculated as

$$\tau_{xx} \approx \tau_{xx}^{a} + \tau_{xx}^{a^+} + \tau_{xx}^{b} + \tau_{xx}^{b^+} + \tau_{xx}^{c} + \tau_{xx}^{c^+}, \tag{3.68}$$

where a, a^+, b, b^+, c, and c^+ are the face of the auxiliary cell shown in Figure 3.11.

The spatial derivatives are computed using the divergence theorem and approximating the integral as a sum over the faces of the auxiliary cell. For example the

contribution of face a to the elements of the stress tensor are

$$\begin{aligned}
\tau_{xx}^a &\approx \frac{1}{V^a}\left[-2A_x^a v_x^a + \frac{2}{3}(A_x^a v_x^a + A_y^a v_y^a + A_z^a v_z^a)\right], \\
\tau_{xy}^a = \tau_{yx}^a &\approx \frac{1}{V^a}(-A_x^a v_y^a - A_y^a v_x^a), \\
\tau_{yy}^a &\approx \frac{1}{V^a}\left[-2A_y^a v_y^a + \frac{2}{3}(A_x^a v_x^a + A_y^a v_y^a + A_z^a v_z^a)\right], \\
\tau_{yz}^a = \tau_{zy}^a &\approx \frac{1}{V^a}(-A_y^a v_z^a - A_z^a v_y^a), \\
\tau_{xz}^a = \tau_{zx}^a &\approx \frac{1}{V^a}(-A_x^a v_z^a - A_z^a v_x^a), \\
\tau_{zz}^a &\approx \frac{1}{V^a}\left[-2A_z^a v_z^a + \frac{2}{3}(A_x^a v_x^a + A_y^a v_y^a + A_z^a v_z^a)\right],
\end{aligned}$$

where V_a is the volume of the auxiliary cell corresponding to face a, (A_x^a, A_y^a, A_z^a) are the components of the face area vector of face a and (v_x^a, v_y^a, v_z^a) are the components of the velocity vector corresponding at face a. Similar formulas are used to calculate the contributions from the other five faces.

The values of the geometric terms are calculated using simple averages of the values of the volume and face area vectors of the physical cells. For example

$$V^a = \frac{1}{2}(V_{ijk} + V_{i-1jk}), \tag{3.69}$$

$$\mathbf{A}^a = \frac{1}{2}(\mathbf{A}_{ijk}^1 + \mathbf{A}_{i-1jk}^1). \tag{3.70}$$

The volumes corresponding to the other five faces are

$$V^{a+} = \frac{1}{2}(V_{i+1jk} + V_{ijk}), \tag{3.71}$$

$$V^b = \frac{1}{2}(V_{ijk} + V_{ij-1k}), \tag{3.72}$$

$$V^{b+} = \frac{1}{2}(V_{ij+1k} + V_{ijk}), \tag{3.73}$$

$$V^c = \frac{1}{2}(V_{ijk} + V_{ijk-1}), \tag{3.74}$$

$$V^{c+} = \frac{1}{2}(V_{ijk+1} + V_{ijk}) \tag{3.75}$$

Similar formulas are employed for calculation of the face area vectors for each of the faces of the auxiliary cell. The value of the the velocity vector at face a is the same

with that in cell $(i-1,j,k)$ since face a has been so chosen that it passes through the centroid of that cell as shown in Figure 3.11. The same is true for the velocity at face a^+. However, faces b, b^+ and c, $c+$ do not pass through cell centers so that the velocity at these faces is computed using a simple average between the values at cell centers in the corresponding directions. The velocity vectors are calculated as follows

$$\mathbf{v}^a = \mathbf{v}_{i-1jk} \tag{3.76}$$

$$\mathbf{v}^{a^+} = \mathbf{v}_{ijk} \tag{3.77}$$

$$\mathbf{v}^b = \frac{1}{2}(\mathbf{v}_{ij-1k} + \mathbf{v}_{ijk}) \tag{3.78}$$

$$\mathbf{v}^{b^+} = \frac{1}{2}(\mathbf{v}_{ijk} + \mathbf{v}_{ij+1k}) \tag{3.79}$$

$$\mathbf{v}^c = \frac{1}{2}(\mathbf{v}_{ijk-1} + \mathbf{v}_{ijk}) \tag{3.80}$$

$$\mathbf{v}^{c^+} = \frac{1}{2}(\mathbf{v}_{ijk} + \mathbf{v}_{ijk+1}). \tag{3.81}$$

A better approximation for the geometric terms and velocities is obtained if instead of the simple averages the distances from cell centroids to face centers were used as weights. It is expected that for highly deformed grids the cell centroid to face center approximation performs better. The calculation of the distances between cell centroids and face centers is performed in the code and the diffusive flux calculations can be easily modified to use these distances as weights instead of simple averaging.

All the terms presented up to this point are used to calculate the stress tensor at face 1. The values of the stress tensor for face 2 and 3 of the real cell are calculated similarly, only that the auxiliary cell will extend in the j and k directions respectively. Only these three stress tensors need be calculated for each cell since the stress tensors corresponding to the the other three faces are calculated for the neighboring cells.

Resistive Flux

The parabolic PDEs that involve the resistive terms are

$$\left.\frac{\partial \mathbf{B}}{\partial t}\right|_{res} = -(LuAl)^{-1}\nabla \times [\overleftrightarrow{\eta} \cdot (\nabla \times \mathbf{B})]$$
$$\left.\frac{\partial e}{\partial t}\right|_{res} = -(LuAl)^{-1}\nabla \cdot \{[\overleftrightarrow{\eta} \cdot (\nabla \times \mathbf{B})] \times \mathbf{B}\}$$

Before integrating over cell $\mathcal{C}$ it is useful to define a vector that has the expression of an electric field. Remembering the $\mathbf{j} = \nabla \times \mathbf{B}$ and defining the resistive electric field as $\mathbf{E}_\eta = \overleftrightarrow{\eta} \cdot \mathbf{j}$ the resistive diffusion equations become

$$\left.\frac{\partial \mathbf{B}}{\partial t}\right|_{res} = -(LuAl)^{-1}\nabla \times \mathbf{E}_\eta \tag{3.82}$$
$$\left.\frac{\partial e}{\partial t}\right|_{res} = (LuAl)^{-1}(\mathbf{B} \times \mathbf{E}_\eta) \tag{3.83}$$

Using an approach similar to that used for the viscous fluxes the resistive diffusive terms are integrated over cell $\mathcal{C}$. For the induction Eqn (3.82) the integration gives

$$\mathbf{B}^{n+1} = \mathbf{B}^n - \frac{\Delta t}{LuAlV_\mathcal{C}} \oint_{\Sigma_\mathcal{C}} d\sigma \times \mathbf{E}_\eta, \tag{3.84}$$

where Stokes' theorem has been used rather than divergence to lower the order of the volume integral. The integral is approximated by a sum over the faces of a cell so that the update formula that gives the time variation of the magnetic field due to the resistive effects becomes

$$\mathbf{B}^{n+1} = \mathbf{B}^n - \frac{\Delta t}{LuAlV_\mathcal{C}} \sum_{i=1}^{6} \mathbf{n} \times \mathbf{E}_\eta A_i. \tag{3.85}$$

Since the integration is performed in a locally aligned coordinate system $\mathbf{n} = (1, 0, 0)$ and $\mathbf{E}_\eta = (E_n, E_{t_1}, E_{t_2})$, where E_{t_1} and E_{t_2}. Thus the cross-product under the sum is $\mathbf{n} \times \mathbf{E}_\eta = (0, -E_{t_2}, E_{t_1})$. It is interesting to note that only the tangential components of the electric field are needed. To simplifiy the expressions the result of the cross-product for face i is denoted $\mathcal{B}_i$. Substituting the formula for the cross-product in

Eqn (3.85) gives

$$\mathbf{B}^{n+1} = \mathbf{B}^n - \frac{\Delta t}{LuAlV_{\mathcal{C}}} \sum_{i=1}^{6} \mathcal{B}_i A_i, \tag{3.86}$$

where $\mathcal{B}_i = (0, -E_{t_2}, E_{t_1})_i$ is a vector.

For the energy equation the integration over a cell volume and application of the divergence theorem gives

$$\mathbf{e}^{n+1} = \mathbf{e}^n - \frac{\Delta t}{LuAlV_{\mathcal{C}}} \oint_{\Sigma_{\mathcal{C}}} (\mathbf{B} \times \mathbf{E}_\eta) \cdot d\sigma. \tag{3.87}$$

Again the integral is approximated by a sum over the faces of the cell and the update formula for the variation of the total energy due to resistive effects becomes

$$\mathbf{e}^{n+1} = \mathbf{e}^n - \frac{\Delta t}{LuAlV_{\mathcal{C}}} \sum_{i=1}^{6} (\mathbf{B} \times \mathbf{E}_\eta) \cdot \mathbf{n} A_i. \tag{3.88}$$

The dot-product between the unit face area vector $\mathbf{n}$ and $\mathbf{B} \times \mathbf{E}_\eta$ is computed in a locally aligned coordinate system also to give $(\mathbf{B} \times \mathbf{E}_\eta) \cdot \mathbf{n} = B_{t_1} E_{t_2} - B_{t_2} E_{t_1}$. To simplify the expressions the result of the dot-product is denoted $\mathcal{E} = B_{t_1} E_{t_2} - B_{t_2} E_{t_1}$. Substituting the expression for the dot-product in Eqn (3.88) gives the update formula for the energy equation due to resistive terms

$$\mathbf{e}^{n+1} = \mathbf{e}^n - \frac{\Delta t}{LuAlV_{\mathcal{C}}} \sum_{i=1}^{6} \mathcal{E}_i A_i, \tag{3.89}$$

where $\mathcal{E}_i$ is evaluated at face i.

$\mathbf{E}_\eta$ $(= \overleftrightarrow{\eta} \cdot \nabla \times \mathbf{B})$ has to be evaluated at each cell face in order to close the update formulas. To accomplish this the same auxiliary cells used for the viscous terms are employed.

Using Stokes' theorem the curl of the magnetic field can be calculated approximately using the following approach

$$V_a \nabla \times \mathbf{B} \approx \int_{V_a} \nabla \times \mathbf{B} d\tau = \oint_{\Sigma_a} d\sigma \times \mathbf{B} \Rightarrow \nabla \times \mathbf{B} \approx \oint_{\Sigma_a} d\sigma \times \mathbf{B}, \tag{3.90}$$

where the integration has been performed over the auxiliary cell and V_a is the volume of this cell. At this point it is important to note that only at the faces of the real cells the coordinate system is locally aligned. For the integration in Eqn (3.90), involving the faces of the auxiliary cell, the coordinate system has a general orientation and the cross-product under the integral is a full vector.

Only the tangential components of $\mathbf{E}_\eta$ are needed so that only the tangential components of the curl of the magnetic field are calculated. The expressions for these components are

$$\begin{aligned}(\nabla \times \mathbf{B})_{t_1} = \frac{1}{V_a}[&(A_{t_2}^{a^+} B_n^{a+} - A_n^{a^+} B_{t_2}^{a+}) - (A_{t_2}^a B_n^a - A_n^a B_{t_2}^a) + \\ &(A_{t_2}^{b^+} B_n^{b+} - A_n^{b^+} B_{t_2}^{b+}) - (A_{t_2}^b B_n^b - A_n^b B_{t_2}^b) + \\ &(A_{t_2}^{c^+} B_n^{c+} - A_n^{c^+} B_{t_2}^{c+}) - (A_{t_2}^c B_n^c - A_n^c B_{t_2}^c)],\end{aligned} \tag{3.91}$$

and

$$\begin{aligned}(\nabla \times \mathbf{B})_{t_2} = \frac{1}{V_a}[&(A_{t_1}^{a^+} B_n^{a+} - A_n^{a^+} B_{t_1}^{a+}) - (A_{t_1}^a B_n^a - A_n^a B_{t_1}^a) + \\ &(A_{t_1}^{b^+} B_n^{b+} - A_n^{b^+} B_{t_1}^{b+}) - (A_{t_1}^b B_n^b - A_n^b B_{t_1}^b) + \\ &(A_{t_1}^{c^+} B_n^{c+} - A_n^{c^+} B_{t_1}^{c+}) - (A_{t_1}^c B_n^c - A_n^c B_{t_1}^c)],\end{aligned} \tag{3.92}$$

where (A_n, A_{t_1}, A_{t_2}) are the components of the face are vector in the locally aligned coordinate system and the components of the magnetic field at the auxiliary cell faces are computed with formulas similar to those for the velocities (that were used to obtain the viscous fluxes.)

Components of the magnetic field and of the face are vectors need not be rotated. It is easily shown using simple vector algebra that, for example, $A_{t_2} B_n \equiv A_z B_x$. Components of $\overleftrightarrow{\eta}$ at real cell faces are calculated using simple averages.

3.3 Implicit Scheme

The problems described by the single fluid MHD model are often "stiff" due to the large differences (orders of magnitude) in characteristic speeds and length scales. Both types of stiffness are due to discretizations. The characteristic speeds are peculiar to the approximate Riemann solver that is used to discretize the hyperbolic part of the system of PDEs. Different length scales are used in cases where viscous and resistive effects are localized such as in a boundary layer or a current sheet. To be able to capture the phenomena that take place in the narrow regions where the diffusive effects are important the regions are discretized using cells that are much finer than in the rest of the domain where the diffusive effects are less important. Stability constraints of explicit schemes force a time step orders of magnitude smaller than that required for accuracy, leading to extremely long computational times. This difficulty is addressed by implicit schemes which eliminate or reduce the stability constraints. More than that the implicit scheme implemented in WARP3 adds an appropiate amount a numerical diffusion to the scheme so that it drives the residual steady state problems to zero machine faster than the explicit scheme, although it is computationally more intensive. It is shown in Section 5 that only for time steps almost one order of magnitude smaller than the time step required for stability the explicit scheme drives the residual to zero machine. For time dependent simulations the implicit scheme in WARP3 is marginally faster that the explicit scheme.

This section describes how the solution of the single fluid MHD system of PDEs is obtained with an implicit dual-time stepping scheme and how the implicit scheme is implemented in WARP3.

3.3.1 Implicit Dual Level Scheme

The method employed by WARP3 to advance the solution of the hyperbolic-parabolic system of PDEs uses a dual level scheme. The outer level consists of an implicit

discretization of the governing equations. At each time step the implicit discretization is solved iteratively with a method similar to a time-marching scheme that drives a residual to zero machine. The iterative method represents the inner level of the scheme and the iterations are called pseudo time iterations. The dual level scheme allows (physical) time steps larger than those of an explicit scheme at the expense of CPU time since pseudo time steady state is obtained by solving a large sparse linear system. The dual-time stepping formulation is used not only for finding steady state solutions but also for tracking time-dependent solutions of initial-boundary value problems.

A similar dual level scheme was first introduced by Jameson [39] with the difference that the pseudo time iterations were performed with an explicit scheme. The WARP3 implementation of the dual-time stepping scheme is similar to that described by Dubuc *et* al. [19]. Dubuc's scheme was developed for two-dimensional Euler equations (unsteady inviscid flow) on deformable (time-varying) grids. The three-dimensional implementation of this scheme for the viscous and resistive MHD equations is a first. The main difference between Dubuc's implementation and WARP3 is that the linear system resulting from the implicit discretization is solved (in WARP3) using a block symmetric Gauss-Seidel scheme rather than a generalized conjugate gradient scheme. Also at the present time WARP3 is able to handle only static grids (that not vary in time.)

The implicit scheme is derived starting with a generalized set of governing equations written in tensor form

$$\frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot \overleftrightarrow{F}_h(\mathbf{q}) = \nabla \cdot \overleftrightarrow{F}_p(\mathbf{q}), \tag{3.93}$$

where $\overleftrightarrow{F}_h$ is the hyperbolic (advective) flux tensor and $\overleftrightarrow{F}_p$ is the parabolic (diffusive) flux tensor. Similar to the explicit scheme, the implicit scheme is derived by making abstraction of the nature of the governing equations and can be applied in general to any system of hyperbolic-parabolic PDEs.

To obtain the implicit scheme Eqn (3.93) is integrated over a cell volume. As explained in Section 3.2, the values of the conserved variables ($\mathbf{Q}$) are assumed constant inside cell $\mathcal{C}$. The result of the integration is

$$V_{\mathcal{C}} \frac{d\mathbf{Q}}{dt} + \mathbf{R}_h(\mathbf{Q}) = \mathbf{R}_p(\mathbf{Q}), \tag{3.94}$$

where the partial differential (∂) has become ordinary (d) since the conserved variables are a function of time only, and $\mathbf{R}_h$ and $\mathbf{R}_p$ are the volume integrals over cell $\mathcal{C}$ of the hyperbolic and parabolic fluxes respectively, also called the flux residuals. The residuals represent the net fluxes for cell $\mathcal{C}$. Methods used to calculate the volume integrals using a TVD scheme (for the hyperbolic fluxes) and a central differencing scheme on staggered cells (for the parabolic fluxes) were presented in Section 3.2.

The hyperbolic and parabolic flux residuals are evaluated at physical timestep $n+1$ and the time derivative is approximated using a second order accurate discretization. The resulting equation is

$$V_{\mathcal{C}} \frac{3\mathbf{Q}^{n+1} - 4\mathbf{Q}^n + \mathbf{Q}^{n-1}}{2\Delta t} + \mathbf{R}(\mathbf{Q}^{n+1}) = 0, \tag{3.95}$$

where the hyperbolic and parabolic residuals are combined into a single term $\mathbf{R}(\mathbf{Q}^{n+1}) = \mathbf{R}_h(\mathbf{Q}^{n+1}) - \mathbf{R}_p(\mathbf{Q}^{n+1})$ for a simplified notation. Treatment of each of the flux residuals is presented in the following sections. To be noted here is that in the code the second order time accuracy is obtained at the expense of memory since three time levels are stored (instead of two levels such as for the explicit scheme.) Eqn (3.95) is nonlinear in $\mathbf{Q}^{n+1}$ and has to be solved numerically. A method to find a solution for Eqn (3.95) is derived by defining a new residual $\mathbf{R}^*$ such that

$$\mathbf{R}^*(\mathbf{Q}^{n+1}) = V_{\mathcal{C}} \frac{3\mathbf{Q}^{n+1} - 4\mathbf{Q}^n + \mathbf{Q}^{n-1}}{2\Delta t} + \mathbf{R}(\mathbf{Q}^{n+1}) = 0. \tag{3.96}$$

The new residual called pseudo time residual from here on is used to define a time-marching equation in pseudo time (t^*)

$$\frac{d\mathbf{Q}^{n+1}}{dt^*} + \frac{1}{V_{\mathcal{C}}} \mathbf{R}^*(\mathbf{Q}^{n+1}) = 0, \tag{3.97}$$

which has been already integrated over cell $\mathcal{C}$ and divided through by the volume of the cell. A steady state solution to Eqn (3.97), $d\mathbf{Q}^{n+1}/dt = 0$, also satisfies

$$\mathbf{R}^*(\mathbf{Q}^{n+1}) = 0 \tag{3.98}$$

and thus it is a solution to the unsteady Eqn (3.95).

The steady state problem in pseudo time given by Eqn (3.97) is discretized using a first order approximation of the pseudo time derivative and an implicit formulation. The result of this discretization is an expression for the variation of $\mathbf{Q}$ with respect to pseudo time, also called a Δ formulation. The discretized pseudo time implicit formulation is

$$\frac{\Delta\mathbf{Q}}{\Delta t^*} = -\frac{1}{V_\mathcal{C}}\mathbf{R}^*(\mathbf{Q}^{m+1}), \tag{3.99}$$

where m is the pseudo time level, $\Delta\mathbf{Q} = \mathbf{Q}^{m+1} - \mathbf{Q}^m$, Δt^* is the pseudo time step, and $\mathbf{Q}^m = \mathbf{Q}^{m+1} \equiv \mathbf{Q}^{n+1}$ at steady state. To find a solution to Eqn (3.99), which is nonlinear in $\mathbf{Q}^{m+1}$, the pseudo time residual ($\mathbf{R}^*$) has to be expressed in terms of known quantities. In this case the known quantities are conserved variables at pseudo time level m. This is accomplished by linearizing $\mathbf{R}^*(\mathbf{Q}^{m+1})$ with respect to pseudo time. The linearization is obtained using a Taylor series expansion about pseudo time so that

$$\mathbf{R}^*(\mathbf{Q}^{m+1}) \approx \mathbf{R}^*(\mathbf{Q}^m) + \left.\frac{\partial\mathbf{R}^*}{\partial t^*}\right|_m \Delta t^*, \tag{3.100}$$

where $\left.\frac{\partial\mathbf{R}^*}{\partial t^*}\right|_m$ means that the residual is calculated based on values of the conserved variables at time step m, and only the first two terms of the Taylor series have been kept. The derivative of the residual with respect to pseudo time is replaced by a derivative with respect to the conserved variables using the chain rule

$$\frac{\partial\mathbf{R}^*}{\partial t^*} = \frac{\partial\mathbf{R}^*}{\partial\mathbf{Q}}\frac{\partial\mathbf{Q}}{\partial t^*}, \tag{3.101}$$

where $\partial \mathbf{R}^*/\partial \mathbf{Q}$ is the Jacobian of the pseudo time residual. The partial derivative of the conserved variables with respect to pseudo time is discretized using the same finite difference as that used for Eqn (3.99). With this substitution Eqn (3.101) becomes

$$\frac{\partial \mathbf{R}^*}{\partial t^*} \approx \frac{\partial \mathbf{R}^*}{\partial \mathbf{Q}} \frac{\Delta \mathbf{Q}}{\Delta t^*}. \tag{3.102}$$

Replacement of Eqn (3.102) in Eqn (3.100) yields the expression for the linearized pseudo time residual

$$\mathbf{R}^*(\mathbf{Q}^{m+1}) \approx \mathbf{R}^*(\mathbf{Q}^m) + \left.\frac{\partial \mathbf{R}^*}{\partial \mathbf{Q}}\right|_m \Delta \mathbf{Q}. \tag{3.103}$$

For the scheme to be complete the Jacobian of the pseudo time residual has to be found. This Jacobian is obtained from Eqn (3.96) by taking the partial derivative with respect to the conserved variables at time level m. The resulting expression is

$$\left.\frac{\partial \mathbf{R}^*}{\partial \mathbf{Q}}\right|_m = \left.\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\right|_m + \frac{3}{2}\frac{V_C}{\Delta t} I, \tag{3.104}$$

where use have been made of the fact that $\mathbf{Q}^m \equiv \mathbf{Q}^{n+1}$, and I is the unit matrix with rank equal to the number of conserved variables. The expression of the linearized pseudo time residual, obtained by replacing Eqn (3.104) in Eqn (3.103), is

$$\mathbf{R}^*(\mathbf{Q}^{m+1}) \approx \mathbf{R}^*(\mathbf{Q}^m) + \left(\left.\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\right|_m + \frac{3}{2}\frac{V_C}{\Delta t} I \right) \Delta \mathbf{Q}. \tag{3.105}$$

Substituting Eqn (3.105) in Eqn (3.99) gives the expression for the solution advance ($\Delta \mathbf{Q}$) in one pseudo time step. The resulting expression is

$$\left[V_{\mathbf{C}} \left(\frac{1}{\Delta t^*} + \frac{3}{2\Delta t} \right) I + \left.\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\right|_m \right] \Delta \mathbf{Q} = -\mathbf{R}^*(\mathbf{Q}^m), \tag{3.106}$$

where the Jacobian of the combined flux residual is

$$\left.\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\right|_m = \left.\frac{\partial \mathbf{R}_h}{\partial \mathbf{Q}}\right|_m - \left.\frac{\partial \mathbf{R}_p}{\partial \mathbf{Q}}\right|_m. \tag{3.107}$$

The difference between a steady state implicit formulation such as that presented by Vanden and Whitfield [79], for example, and the time-dependent formulation presented here is that each of the left and right hand sides contain an

additional term. The residual on the right hand side of Eqn (3.106) contains an additional term due to the time variation of the solution that has the expression $V_{\mathcal{C}}(3\mathbf{Q}^{n+1} - 4\mathbf{Q}^{n} + \mathbf{Q}^{n-1})/(2\Delta t)$. The quantity proportional to the volume on the left hand side contains the extra term $3/2\Delta t V_{\mathcal{C}} I$.

Written for the entire domain Eqn (3.106) has the form of a linear system

$$A\mathbf{x} = \mathbf{b}, \tag{3.108}$$

where matrix A, also called the implicit operator is given by

$$A = \left[V_{\mathbf{C}} \left(\frac{1}{\Delta t^*} + \frac{3}{2\Delta t} \right) I + \left. \frac{\partial \mathbf{R}}{\partial \mathbf{Q}} \right|_m \right], \tag{3.109}$$

and the right hand side, or the inhomogeneity term, is

$$\mathbf{b} = -\mathbf{R}^*(\mathbf{Q}^m). \tag{3.110}$$

The unknowns vector $\mathbf{x}$ is the time advance of solution $\Delta\mathbf{Q}$ in each cell.

Matrix A is a block multi-diagonal matrix and for the single fluid MHD model which has eight conserved variables the elements of A are 8×8 matrices. (For the three-dimensional Euler equations, for example, the elements would be 5×5 matrices since the model has five conserved variables.) The structure of matrix A is determined by the method used to calculate the Jacobians of the residuals and by the ordering of the time advance of the solution in the vector $\mathbf{x}$. In general the structure of matrix A is sparse and in CFD practice iterative methods are used to find the solution to Eqn (3.108). For the linear system resulting from the discretization of a three-dimensional system of PDEs direct methods (such as Gaussian elimination) are not desired due to large computer memory requirements. Vanden and Whitfield [79] investigated the use of a direct method for the solution of a linear system obtained from the implicit discretization of Euler equations and compared it to factored and non-factored iterative methods. They found out that their implementation of the direct method required approximately 3.5 more memory and 14.5 more CPU time

than a Gauss-Seidel iterative method. WARP3 uses a Gauss-Seidel iterative method to obtain the solution of Eqn (3.108) that results from the implicit discretization of the single fluid MHD model. Implementation of the Gauss-Seidel iterative method is also presented later on. Before describing the solution procedure however, the methods used to calculate the Jacobians of the residuals are introduced.

3.3.2 Flux Jacobians

The residuals are defined at the beginning of Section 3.3 as volume integrals of the flux terms. Following an approach similar to that used in the derivation of the explicit scheme (Section 3.2) the volume integral of the flux terms is transformed into a closed surface integral that is approximated with a sum of the flux at each cell face times the corresponding face area.

Thus, computation of the Jacobians of the residuals reduces to finding out the Jacobians of the fluxes for the hyperbolic and parabolic terms. Since the methods used to calculate each of the fluxes are different due to the nature of the PDEs the computation of the flux Jacobians is presented for each type of flux.

The analytic derivation of the flux Jacobians is highly complex and if obtained the CPU time needed to evaluate the resulting lengthy analytical expressions would slow down the code considerably. As a consequence the flux Jacobians for both hyperbolic and parabolic fluxes are computed numerically.

Hyperbolic Flux Jacobians

The purpose of this section is to describe the method used to calculate the Jacobians of the hyperbolic fluxes. These Jacobians are not to be confused with the approximate flux Jacobian $\tilde{A}$ used in the Roe-type approximate Riemann solver. They are the Jacobians of the numerical fluxes derived in Section 3.2.2.

After calculation of the hyperbolic fluxes in a locally aligned coordinate frame and rotation back to a coordinate system parallel to $Oxyz$ (as described in Section 3.2.4)

the residual of the hyperbolic fluxes becomes

$$\mathbf{R}_h = \mathbf{F}^m_{i+\frac{1}{2},j,k} - \mathbf{F}^m_{i-\frac{1}{2},j,k} + \mathbf{G}^m_{i,j+\frac{1}{2},k} - \mathbf{G}^m_{i,j-\frac{1}{2},k} + \mathbf{H}^m_{i,j,k+\frac{1}{2}} - \mathbf{H}^m_{i,j,k-\frac{1}{2}}, \tag{3.111}$$

where $\mathbf{F}$, $\mathbf{G}$ and $\mathbf{H}$ are the numerical hyperbolic fluxes in the x, y and z direction respectively, the fractional indices denote cell faces, and superscript $n+1$ means that the fluxes are calculated based on values of the conserved variables at time step $n+1$. It is important to note that each of the fluxes in Eqn (3.111) should be multiplied with the corresponding face area. These multiplications have been omitted on purpose to simplify notation and from here on the multiplication of the flux vector with the area is assumed to take place when the flux is computed. (This assumption is consistent with the implementation of flux calculations in WARP3 where the multiplication with the face area takes place in the procedure that computes the fluxes.)

The Jacobian of the hyperbolic residual in Eqn (3.107) is obtained by taking the derivative of Eqn (3.111) with respect to the conserved variables vector $\mathbf{Q}$. Normally the numerical flux formulas are based on values of the conserved variables in four cells, two on each side of a cell interface. For example the flux in the x direction is calculated based on the value of the conserved variables in two cells to right and two cells to left of the interface of interest (see Figure 3.12). The functional expression for the numerical hyperbolic flux in the x direction at face $i+\frac{1}{2}, j, k$ is

$$\mathbf{F}_{i+\frac{1}{2},j,k} = f(\mathbf{Q}_{i-1,j,k}, \mathbf{Q}_{i,j,k}, \mathbf{Q}_{i+1,j,k}, \mathbf{Q}_{i+2,j,k}). \tag{3.112}$$

It has been reported by Whitfield [82] that computation of the flux Jacobians with an expression for the fluxes that uses conserved variables from only one cell on each side of the interface is acceptable. This effectively produces a Jacobian based on a first order flux. The functional expression for the numerical hyperbolic flux in the x direction at face $i+\frac{1}{2}, j, k$ becomes then

$$\mathbf{F}_{i+\frac{1}{2},j,k} = f(\mathbf{Q}_{i,j,k}, \mathbf{Q}_{i+1,j,k}). \tag{3.113}$$

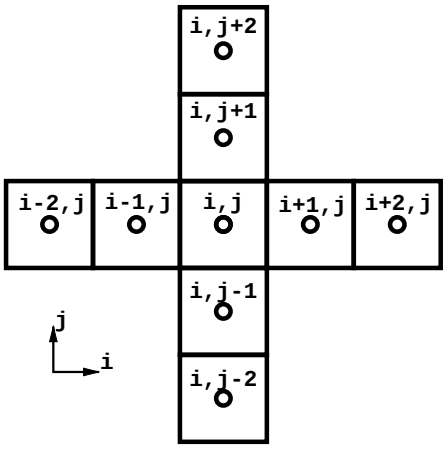

Figure 3.12: Two-dimensional stencil of the hyperbolic fluxes. Four cells placed symmetrically with respect to the face of interest are needed to calculate the flux at that face. Extension to three-dimensions is straightforward. The stencil in the third dimension would require the addition of two cells pointing up ($k+$ direction) and two cells pointing down ($k-$ direction) with respect to the plane of the paper.

There are two reasons for using a first order stencil for the calculation of the flux Jacobian. The first reason is that a reduction of the band width of the implicit operator (matrix A in Eqn (3.108)) leads to lower memory and CPU time requirements than for a second order stencil. The second reason also pointed out by Whitfield is that the convergence of the scheme does not improve significantly if a second order stencil is used. It is important to note at this point that on the right hand side of Eqn (3.108) the fluxes in the pseudo time residual ($\mathbf{R}^*$) are calculated using the full second order stencil.

The expression for the residual in Eqn (3.112) is written in terms of first order fluxes as

$$\begin{aligned}\mathbf{R}_h =& \hat{\mathbf{F}}_{i+\frac{1}{2},j,k}(\mathbf{Q}^m_{i,j,k}, \mathbf{Q}^m_{i+1,j,k}) - \hat{\mathbf{F}}_{i-\frac{1}{2},j,k}(\mathbf{Q}^m_{i-1,j,k}, \mathbf{Q}^m_{i,j,k}) + \\ & \hat{\mathbf{G}}_{i,j+\frac{1}{2},k}(\mathbf{Q}^m_{i,j,k}, \mathbf{Q}^m_{i,j+1,k}) - \hat{\mathbf{G}}_{i,j-\frac{1}{2},k}(\mathbf{Q}^m_{i,j-1,k}, \mathbf{Q}^m_{i,j,k}) + \\ & \hat{\mathbf{H}}_{i,j,k+\frac{1}{2}}(\mathbf{Q}^m_{i,j,k}, \mathbf{Q}^m_{i,j,k+1}) - \hat{\mathbf{H}}_{i,j,k-\frac{1}{2}}(\mathbf{Q}^m_{i,j,k-1}, \mathbf{Q}^m_{i,j,k}),\end{aligned} \tag{3.114}$$

where the hat notation has been used to emphasize that the fluxes are calculated using a first order stencil. Taking the derivative of $\mathbf{R}_h$ given by Eqn (3.114) with respect to $\mathbf{Q}$ yields

$$\left.\frac{\partial \mathbf{R}_h}{\partial \mathbf{Q}}\right|_m = \frac{\partial \hat{\mathbf{F}}_{i+\frac{1}{2},j,k}}{\partial \mathbf{Q}_{i,j,k}} + \frac{\partial \hat{\mathbf{F}}_{i+\frac{1}{2},j,k}}{\partial \mathbf{Q}_{i+1,j,k}} - \frac{\partial \hat{\mathbf{F}}_{i-\frac{1}{2},j,k}}{\partial \mathbf{Q}_{i-1,j,k}} - \frac{\partial \hat{\mathbf{F}}_{i-\frac{1}{2},j,k}}{\partial \mathbf{Q}_{i,j,k}} + \frac{\partial \hat{\mathbf{G}}_{i,j+\frac{1}{2},k}}{\partial \mathbf{Q}_{i,j,k}} + \frac{\partial \hat{\mathbf{G}}_{i,j+\frac{1}{2},k}}{\partial \mathbf{Q}_{i,j+1,k}} - \frac{\partial \hat{\mathbf{G}}_{i,j-\frac{1}{2},k}}{\partial \mathbf{Q}_{i,j-1,k}} - \frac{\partial \hat{\mathbf{G}}_{i,j-\frac{1}{2},k}}{\partial \mathbf{Q}_{i,j,k}} + \frac{\partial \hat{\mathbf{H}}_{i,j,k+\frac{1}{2}}}{\partial \mathbf{Q}_{i,j,k}} + \frac{\partial \hat{\mathbf{H}}_{i,j,k+\frac{1}{2}}}{\partial \mathbf{Q}_{i,j,k+1}} - \frac{\partial \hat{\mathbf{H}}_{i,j,k-\frac{1}{2}}}{\partial \mathbf{Q}_{i,j,k-1}} - \frac{\partial \hat{\mathbf{H}}_{i,j,k-\frac{1}{2}}}{\partial \mathbf{Q}_{i,j,k}}, \tag{3.115}$$

where the subscript m of the conserved variables has been dropped for simplicity. The Jacobian of the hyperbolic flux residual used in the formulation of the implicit operator in Eqn (3.106) is given by Eqn (3.115).

Due to the complicated expressions of the hyperbolic fluxes analytical expressions for the flux Jacobians are extremely difficult to obtain. Thus in WARP3 the flux Jacobians are evaluated numerically. A study by Orkwis and Vanden [52] compared the accuracy of numerical and analytical Jacobians for a Navier-Stokes solver for supersonic to hypersonic flows (Mach 2.0 to 14.1). They showed that for these high speed flows solved on grids with large aspect ratio cells numerical Jacobians retain their accuracy.

Two methods have been implemented in WARP3 for the numerical evaluation of the hyperbolic flux Jacobians. The first method called the limit formulation is a first order approximation of the derivative, obtained with a simple Taylor expansion about $\mathbf{Q}$. This approach has been used to evaluate the Jacobians by Vanden and Whitfield [79] for example. The second method called the complex numbers formulation is also obtained by a Taylor series expansion only that the function that is expanded is evaluated using a complex variable. This method of calculating derivatives of real functions using complex variables was recently introduced by Squire and Trapp [68]. It has been reported by Whitfield and Taylor and their co-workers [83]

that the evaluation of Jacobians with complex numbers has been succesfully implemented in three-dimensional compressible and incompressible Navier-Stokes codes.

The limit formulation method is derived starting with a Taylor series expansion

$$\mathbf{F}(\mathbf{Q}+\epsilon) = \mathbf{F}(\mathbf{Q}) + \epsilon\frac{\partial \mathbf{F}}{\partial \mathbf{Q}} + \frac{\epsilon^2}{2}\frac{\partial \mathbf{F}^2}{\partial^2 \mathbf{Q}} + \text{H.O.T.}, \tag{3.116}$$

where ϵ is a small number. The Jacobian is obtained from the second term of the right hand side as

$$\frac{\partial \mathbf{F}}{\partial \mathbf{Q}} \approx \frac{\mathbf{F}(\mathbf{Q}+\epsilon) - \mathbf{F}(\mathbf{Q})}{\epsilon} + \mathrm{O}(\epsilon) \tag{3.117}$$

which is a first order approximation of the continuous derivative. (Note that it is possible to obtain a second order accurate Jacobian if a central differencing were used instead of a one sided differencing.) Numerical experiments showed that an appropriate value for ϵ is 1×10^{-12} on the DEC Alphastation 433au. Eqn (3.117) is called a limit formulation because it is a discrete representation of the definition of a derivative

$$\frac{\partial f}{\partial q} = \lim_{\epsilon \to 0} \frac{f(q+\epsilon) - f(q)}{\epsilon}.$$

Element (p, r) of the flux Jacobian is calculated with the formula

$$\frac{\partial \mathbf{F}}{\partial \mathbf{Q}}(p, r) = \frac{\mathbf{F}_p(\mathbf{Q}^r) - \mathbf{F}_p(\mathbf{Q})}{\epsilon}, \tag{3.118}$$

where $\mathbf{Q}^r = \mathbf{Q} + \epsilon\mathbf{e}^r$ is the conserved vector perturbed in the r-th variable and $\mathbf{e}^r$ is the r-th canonical vector. For example for $r = 2$ (corresponding to the momentum in x direction) $\mathbf{e}^r = [0, 1, 0, 0, 0, 0, 0, 0]^T$. Fortran 90 allows aggregate array assignments and operations so that in WARP3 an entire row of the Jacobian is evaluated in one step ($p = 1, 2, \ldots, 8$).

The complex numbers method is derived starting from a Taylor series expansion similar to Eqn (3.116) only that ϵ is replaced by the product between the fundamental complex number ($i = \sqrt{-1}$) and a small number h. The expansion about $\mathbf{Q}$ gives

$$\mathbf{F}(\mathbf{Q}+ih) = \mathbf{F}(\mathbf{Q}) + ih\frac{\partial \mathbf{F}}{\partial \mathbf{Q}} - \frac{h^2}{2}\frac{\partial \mathbf{F}^2}{\partial^2 \mathbf{Q}} - i\frac{h^3}{6}\frac{\partial \mathbf{F}^3}{\partial^3 \mathbf{Q}} + \text{H.O.T.}. \tag{3.119}$$

The Jacobian is obtained by taking the imaginary part of $\mathbf{F}(\mathbf{Q}+ih)$ and dividing through by h. Thus the expression for the Jacobian is

$$\frac{\partial \mathbf{F}}{\partial \mathbf{Q}} \approx \mathrm{Im}\left[\frac{\mathbf{F}(\mathbf{Q}+ih)}{h}\right] + \mathrm{O}(h^2), \tag{3.120}$$

which is a second order accurate approximation of the continuous differential. Complex numbers storage require two times more computer memory than real numbers and it is reported by Newman *et* al. [37] that arithmetic operations on complex numbers require in general three more times CPU time than on real numbers. In consequence care must be exercised in the design of the procedure that calculates the Jacobians using complex numbers. This method has been implemented such that the minimum number of complex numbers are used (and stored) for the computation of the Jacobians. They are stored in vectors with length equal to the number of conserved variables (eight for the single fluid MHD model). One of the variables is the complex "conserved variables vector" $\mathbf{Q}^r_{\mathrm{cmplx}} = \mathbf{Q} + ih\mathbf{e}^r$, where the superscript r of the complex conserved variables vector means that the r-th conserved variable has been perturbed. The other complex numbers used are the eigenvalues (vector of eight) and the left and right eigenvectors matrices (square matrices of rank eight) that are necessary to calculate the complex flux vector (also stored in a vector of eight). With these notations the formula for element (p, r) of the complex flux Jacobian is

$$\frac{\partial \mathbf{F}}{\partial \mathbf{Q}}(p,r) = \mathrm{Im}\left[\frac{\mathbf{F}_p(\mathbf{Q}^r_{\mathrm{cmplx}})}{h}\right] \tag{3.121}$$

Similarly to the limit formulation the evaluation of an entire row of the Jacobian is performed in one step. Through numerical experimentation it has been found that the method is insensitive to the values of h and values around 1×10^{-5} were appropriate for all cases studied.

For both methods the flux at the cell face is computed with the first order formula given by Eqn (3.15) which for the complex formulation has been adapted to work with complex numbers.

Although arithmetic operations in complex numbers are three times slower than the same operations in real numbers this drawback is offset by the fact that the complex flux has to be evaluated only once for the complex formulation but twice for the limit formulation. More than that, the complex number formulation is advantageous because it it not prone to catastrophic cancellation errors that lead to dramatic drops in precision in certain cases and the results are insensitive to the value of h. The largest potential advantage, however, is that since the complex numbers formulation allows evaluation of Jacobians with relatively large values of h, it is possible that most computations currently performed in double-precision arithmetic be performed in single-precision arithmetic. This would result in halving the memory requirements and also halving the CPU time required to perform the computations. (The reduction of the computational time by a factor of two is ideal and in practice it most likely that lower reductions factors would be achieved, depending on the processor architecture and the compiler used.)

Parabolic Flux Jacobians

The methods used to calculate each of the parabolic fluxes currently implemented in WARP3 were presented in Section 3.2.6. The different nature of the viscous fluxes and resistive fluxes calls for separate treatment of each of them. However the viscous and resistive flux Jacobians are calculated using the same approach. In this section only the description of the viscous flux is presented.

The three-dimensional stencil of the parabolic fluxes is made of twenty-seven cells. If the grid were orthogonal Cartesian the parabolic flux stencil would be a cube with three cells on each side. Figure 3.13) shows the parabolic fluxes stencil in two dimensions. If the full (twenty-seven-cell) stencil were used to calculate the parabolic flux Jacobians then the band structure of matrix A would have had twenty-seven diagonals. This would lead to a band width of twenty-seven, meaning that besides a main diagonal matrix A would have thirteen upper and thirteen lower diagonals.

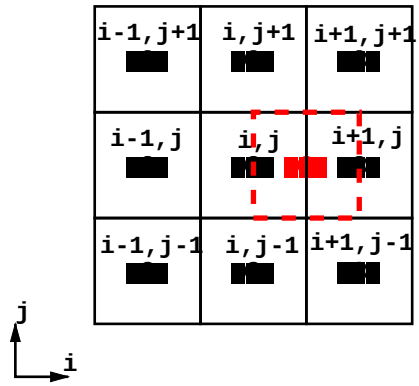

Figure 3.13: Two-dimensional stencil of the parabolic fluxes. The stencil is made of nine cells. Extension to three-dimensions requires the addition of two extra layers of nine cells each. One layer would be placed above ($k+$ direction) and the other layer would be placed below ($k-$ direction) the existing layer. The auxiliary cell between real cells (i, j) and $(i + 1, j)$ is shown.

Since the storage requirements in this case would be rather large the CFD practice is that the stencil can be reduced without loss of accuracy. The disadvantage is a reduced convergence rate of the implicit solver. In WARP3 a stencil of one cell is used for the calculation of the parabolic flux Jacobians. This leads to a formulation of the parabolic fluxes that uses only the value of the conserved variables in the cell at the center of the stencil (indices (i, j, k)). The effect on the structure of the linear operator A is that the parabolic flux Jacobians appear only in the main diagonal. On the right hand side of the linear system however the parabolic fluxes are evaluated using the full twenty-seven-cell stencil.

The evaluation of the parabolic flux Jacobian has been implemented in the limit formulation only. A complex numbers formulation would have required a complete rewrite of the procedures due to the rather complicated expression of the fluxes. The parabolic fluxes required in Eqn (3.117) are calculated as a function of the conserved variables at (i, j, k) only by dropping the other terms from the expressions. For

example the effect on the velocities at auxiliary cell faces given by Eqns (3.76-3.81) is

$$\mathbf{v}^{a} = 0 \tag{3.122}$$

$$\mathbf{v}^{a^+} = \mathbf{v}^{b} = \mathbf{v}^{b^+} = \mathbf{v}^{c} = \mathbf{v}^{c^+} = \mathbf{v}_{ijk}. \tag{3.123}$$

Similar expression are used for evaluation of the magnetic field at auxiliary cell faces. Similar to the limit formulation for the hyperbolic fluxes numerical experiments showed that an acceptable value for ϵ is 1×10^{-12}.

Structure of the Linear System

The purpose of this subsection is to describe the structure of the implicit operator A and the method used to find the solution to the linear system obtained by discretization of the implicit formulation.

The structure of the implicit operator is found by replacing the expressions of the Jacobians of the hyperbolic and parabolic residuals in Eqn (3.106). The shorthand notation used in Eqn (3.106) for simplicity obscures the fact that the term (enclosed in square brackets) that multiplies $\Delta\mathbf{Q}$ is actually an operator that is applied to $\Delta\mathbf{Q}$. To determine the full formula for the linear system that results from the implicit discretization the linearization formula in Eqn (3.103) is recalled. For example flux Jacobian $\partial\hat{\mathbf{F}}_{i+\frac{1}{2},j,k}/\partial\mathbf{Q}_{i+1,j,k}$ is multiplied in Eqn (3.106) by $\Delta\mathbf{Q}_{i+1,j,k}$. Following this procedure and after replacing all the terms in Eqn (3.106) the Δ equation for one step of the implicit method becomes

$$\begin{aligned} -\frac{\partial\hat{\mathbf{H}}_{i,j,k-\frac{1}{2}}}{\partial\mathbf{Q}_{i,j,k-1}}\Delta\mathbf{Q}_{i,j,k-1} - \frac{\partial\hat{\mathbf{G}}_{i,j-\frac{1}{2},k}}{\partial\mathbf{Q}_{i,j-1,k}}\Delta\mathbf{Q}_{i,j-1,k} - \frac{\partial\hat{\mathbf{F}}_{i-\frac{1}{2},j,k}}{\partial\mathbf{Q}_{i-1,j,k}}\Delta\mathbf{Q}_{i-1,j,k} + D_{i,j,k}\Delta\mathbf{Q}_{i,j,k} + \\ \frac{\partial\hat{\mathbf{F}}_{i+\frac{1}{2},j,k}}{\partial\mathbf{Q}_{i+1,j,k}}\Delta\mathbf{Q}_{i+1,j,k} + \frac{\partial\hat{\mathbf{G}}_{i,j+\frac{1}{2},k}}{\partial\mathbf{Q}_{i,j+1,k}}\Delta\mathbf{Q}_{i,j+1,k} + \frac{\partial\hat{\mathbf{H}}_{i,j,k+\frac{1}{2}}}{\partial\mathbf{Q}_{i,j,k+1}}\Delta\mathbf{Q}_{i,j,k+1} = \\ -\left[V_{i,j,k}\frac{3\mathbf{Q}^{m} - 4\mathbf{Q}^{n} + \mathbf{Q}^{n-1}}{2\Delta t} + \mathbf{R}(\mathbf{Q}^{m})\right], \end{aligned} \tag{3.124}$$

where the diagonal term is

$$D_{i,j,k} = \frac{\partial \hat{\mathbf{F}}_{i+\frac{1}{2},j,k}}{\partial \mathbf{Q}_{i,j,k}} + \frac{\partial \hat{\mathbf{G}}_{i,j+\frac{1}{2},k}}{\partial \mathbf{Q}_{i,j,k}} + \frac{\partial \hat{\mathbf{H}}_{i,j,k+\frac{1}{2}}}{\partial \mathbf{Q}_{i,j,k}} - \frac{\partial \hat{\mathbf{F}}_{i-\frac{1}{2},j,k}}{\partial \mathbf{Q}_{i,j,k}} - \frac{\partial \hat{\mathbf{G}}_{i,j-\frac{1}{2},k}}{\partial \mathbf{Q}_{i,j,k}} - \frac{\partial \hat{\mathbf{H}}_{i,j,k-\frac{1}{2}}}{\partial \mathbf{Q}_{i,j,k}} - \frac{\partial \hat{\mathbf{P}}_{i,j,k}}{\partial \mathbf{Q}_{i,j,k}} + V_{i,j,k}\left(\frac{1}{\Delta t^*} + \frac{3}{2\Delta t}\right) I \tag{3.125}$$

and $\partial \hat{\mathbf{P}}_{i,j,k}/\partial \mathbf{Q}_{i,j,k}$ represents the contribution from both viscous and resistive parabolic flux Jacobians, $V_{i,j,k}$ is the volume of cell (i,j,k), I is the unit matrix of rank equal to the number of conserved variables, and the residual $\mathbf{R}(\mathbf{Q}^m)$ was defined as

$$\mathbf{R}(\mathbf{Q}^m) = \mathbf{R}_h(\mathbf{Q}^m) - \mathbf{R}_p(\mathbf{Q}^m) = \mathbf{F}^m_{i+\frac{1}{2},j,k} - \mathbf{F}^m_{i-\frac{1}{2},j,k} + \mathbf{G}^m_{i,j+\frac{1}{2},k} - \mathbf{G}^m_{i,j-\frac{1}{2},k} + \mathbf{H}^m_{i,j,k+\frac{1}{2}} - \mathbf{H}^m_{i,j,k-\frac{1}{2}} - \mathbf{F}^{\mathrm{visc},m}_{i,j,k} - \mathbf{F}^{\mathrm{res},m}_{i,j,k}. \tag{3.126}$$

The hat notations for the flux Jacobians on the right hand side of Eqn (3.124) are used to emphasize that the stencils of the fluxes used to evaluate Jacobians are reduced in comparison to the stencils of the fluxes on the right hand side. It can be seen that the structure of the linear operator A is block heptadiagonal. The blocks are matrices of rank eight for the single fluid MHD model.

At this point a notation is introduced that reduces the complexity of Eqn (3.124). By defining the two Jacobians of $\mathbf{F}_{i+\frac{1}{2},j,k}$ as

$$\overline{A^m}_{i,j,k} = \frac{\partial \hat{\mathbf{F}}_{i+\frac{1}{2},j,k}}{\partial \mathbf{Q}_{i,j,k}} \qquad \overline{\overline{A^m}}_{i,j,k} = \frac{\partial \hat{\mathbf{F}}_{i+\frac{1}{2},j,k}}{\partial \mathbf{Q}_{i+1,j,k}}, \tag{3.127}$$

the other two flux Jacobians for the x direction are

$$\overline{A^m}_{i-1,j,k} = \frac{\partial \hat{\mathbf{F}}_{i-\frac{1}{2},j,k}}{\partial \mathbf{Q}_{i-1,j,k}} \qquad \overline{\overline{A^m}}_{i-1,j,k} = \frac{\partial \hat{\mathbf{F}}_{i-\frac{1}{2},j,k}}{\partial \mathbf{Q}_{i,j,k}}. \tag{3.128}$$

It is obvious now that although four flux Jacobians are needed for each spatial direction only two need to be stored. Similar expressions are defined for the flux Jacobians

in the y and z directions. With this new notation the linear system described by Eqn (3.124) becomes

$$-\overline{C^m}_{i,j,k-1}\Delta\mathbf{Q}_{i,j,k-1} - \overline{B^m}_{i,j-1,k}\Delta\mathbf{Q}_{i,j-1,k} - \overline{A^m}_{i-1,j,k}\Delta\mathbf{Q}_{i-1,j,k} + \tag{3.129}$$
$$D_{i,j,k}\Delta\mathbf{Q}_{i,j,k} + \overline{\overline{A^m}}_{i,j,k}\Delta\mathbf{Q}_{i+1,j,k} + \overline{\overline{B^m}}_{i,j,k}\Delta\mathbf{Q}_{i,j+1,k} + \overline{\overline{C^m}}_{i,j,k}\Delta\mathbf{Q}_{i,j,k+1} =$$
$$-\left[V_{i,j,k}\frac{3\mathbf{Q}^m - 4\mathbf{Q}^n + \mathbf{Q}^{n-1}}{2\Delta t} + \mathbf{R}(\mathbf{Q}^m)\right],$$

and

$$D_{i,j,k} = \overline{C^m}_{i,j,k} + \overline{B^m}_{i,j,k} + \overline{A^m}_{i,j,k} - \overline{\overline{A^m}}_{i-1,j,k} - \overline{\overline{B^m}}_{i,j-1,k} - \overline{\overline{C^m}}_{i,j,k-1} - \tag{3.130}$$
$$P_{i,j,k} + V_{i,j,k}\left(\frac{1}{\Delta t^*} + \frac{3}{2\Delta t}\right)I,$$

where $P_{i,j,k} = \partial\hat{\mathbf{P}}_{i,j,k}/\partial\mathbf{Q}_{i,j,k}$

Eqn (3.129) was written to reveal the elements of each of the diagonals. The terms multiplying the $\Delta\mathbf{Q}$s that contain a -1 are placed on the sub-diagonals and the terms multiplying the $\Delta\mathbf{Q}$s that contain a $+1$ are placed on the supra-diagonals.

Solution of the Linear System

Numerical solution of Eqn (3.130) is obtained with a symmetric Gauss-Seidel iterative method which is a well established method for the solution of linear systems [30].

The implicit operator A, which is the matrix of the linear system to be solved, is expressed as the sum of its diagonal and strict upper and lower triangular parts

$$A = L + D + U. \tag{3.131}$$

The linear equation is written as

$$(L + D + U)\mathbf{x} = \mathbf{b}. \tag{3.132}$$

The symmetric Gauss-Seidel method performs a forward sweep followed by a backward sweep through the computational domain. A step made of these two sweeps can be

written as

$$(L+D)\mathbf{x}^{(2p-1)} + U\mathbf{x}^{(2p-2)} = b \tag{3.133}$$

$$L\mathbf{x}^{(2p-1)} + (D+U)\mathbf{x}^{(2p)} = b, \tag{3.134}$$

where $p = 1, 2, \ldots$ is the iteration counter. Since the grid that discretizes the domain on which a solution is sought is made of multiple blocks the algorithm is not truly a Gauss-Seidel algorithm but what sometimes in the CFD community is called block Gauss-Seidel.

The reason why matrix A is split in this form becomes obvious if steps shown in Eqns (3.133) and (3.134) are expanded. Take for example the forward step from Eqn (3.133) which becomes

$$\begin{aligned} L_{i,j,k-1}x_{i,j,k-1}^{(2p-1)} + L_{i,j-1,k}x_{i,j-1,k}^{(2p-1)} + L_{i-1,j,k}x_{i-1,j,k}^{(2p-1)} + D_{i,j,k}x_{i,j,k}^{(2p-1)} + \\ U_{i+1,j,k}x_{i+1,j,k}^{(2p-2)} + U_{i,j+1,k}x_{i,j+1,k}^{(2p-2)} + U_{i,j,k+1}x_{i,j,k+1}^{(2p-2)} = b_{i,j,k}. \end{aligned} \tag{3.135}$$

During the forward sweep all the elements of the sub-vector $\mathbf{x}$ up to the (i, j, k) position (but not including it) have been calculated. (The values of the unknowns at iteration level $2p-2$ were calculated previously and those at level $2p-1$ were calculated in the current sweep.) Thus, the unknown element $x_{i,j,k}$ is computed as

$$\begin{aligned} D_{i,j,k}x_{i,j,k}^{(2p-1)} = b_{i,j,k} - L_{i,j,k-1}x_{i,j,k-1}^{(2p-1)} - L_{i,j-1,k}x_{i,j-1,k}^{(2p-1)} - L_{i-1,j,k}x_{i-1,j,k}^{(2p-1)} - \\ U_{i+1,j,k}x_{i+1,j,k}^{(2p-2)} - U_{i,j+1,k}x_{i,j+1,k}^{(2p-2)} - U_{i,j,k+1}x_{i,j,k+1}^{(2p-2)}, \end{aligned} \tag{3.136}$$

The greatest advantage of this method is that there is no need to store two levels of the solution since the old level can be safely overwritten with the new values. Similarly, the backward step is written as

$$\begin{aligned} D_{i,j,k}x_{i,j,k}^{(2p)} = b_{i,j,k} - L_{i,j,k-1}x_{i,j,k-1}^{(2p-1)} - L_{i,j-1,k}x_{i,j-1,k}^{(2p-1)} - L_{i-1,j,k}x_{i-1,j,k}^{(2p-1)} - \\ U_{i+1,j,k}x_{i+1,j,k}^{(2p)} - U_{i,j+1,k}x_{i,j+1,k}^{(2p)} - U_{i,j,k+1}x_{i,j,k+1}^{(2p)}. \end{aligned} \tag{3.137}$$

Elements of L and U are easily found by comparing Eqns (3.136) and (3.137) to Eqn (3.129). The elements of the strict lower matrix are

$$L_{i,j,k-1} = -\overline{C^m}_{i,j,k-1} \tag{3.138}$$

$$L_{i,j-1,k} = -\overline{B^m}_{i,j-1,k} \tag{3.139}$$

$$L_{i-1,j,k} = -\overline{A^m}_{i-1,j,k}, \tag{3.140}$$

the elements of the strict upper matrix are the

$$U_{i+1,j,k} = \overline{\overline{A^m}}_{i,j,k} \tag{3.141}$$

$$U_{i,j+1,k} = \overline{\overline{B^m}}_{i,j,k} \tag{3.142}$$

$$U_{i,j,k+1} = \overline{\overline{C^m}}_{i,j,k}, \tag{3.143}$$

and the elements of the diagonal are given by Eqn (3.130).

For the single fluid MHD model each of Eqns (3.136) and (3.137) represent actually linear systems with eight equations and eight unknowns. A Gauss elimination method with partial pivoting and scaling is used to obtain the solution for each system.

It is suggested by Whitfield [82] that the flux Jacobians need not be evaluated at each pseudo time level m since (in principle) the unknown x ($\equiv \Delta\mathbf{Q}$) goes to zero as the solution reaches steady state. This method has the potential to reduce CPU time but has not been implemented in WARP3. However, the method can be implemented with little modification to the code.

It has been found that the Gauss-Seidel method is extremely robust and requires only five to six iterations (p) to bring the error to less than 1×10^{-11}. The error is defined as the 2-norm of the difference between two consecutive solutions

$$\mathcal{E} = \sqrt{\sum_{i,j,k} \left[x_{i,j,k}^{(2p)}\right]^2 - \left[x_{i,j,k}^{(2p-2)}\right]^2}. \tag{3.144}$$

3.4 Boundary Conditions

A discussion of the solution algorithm for the single fluid MHD model would be incomplete without a description of the boundary conditions. In CFD simulations the boundary conditions are a description of how the fluid interacts with the world outside the domain. Boundary conditions specify the values or the gradients of the conserved/primitive variables at the boundaries of the domain.

In WARP3 boundary conditions are imposed in terms of the magnetic field (B_x, B_y, B_z) and fluid variables (ρ, v_x, v_y, v_z, p) and procedures that implement each type of boundary conditions are used.

The boundary conditions used for the magnetic field are

- perfectly conducting walls,
- walls with soaked-in field,
- axisymmetric boundaries,
- outflow boundaries,
- inflow boundaries,

and the boundary conditions for the fluid flow variables are

- walls with slip or no slip conditions,
- axisymmetric boundaries,
- outflow boundaries,
- inflow boundaries.

Each of these boundary conditions are discussed below. In the implementation the procedures that impose the magnetic BCs are called first and then the values of the magnetic field components are passed to the fluid (or hydro) BCs since the total energy per unit volume, which is a function of the magnetic field, has to be calculated at the boundaries.

To specify BCs in the context of a finite volume formulation the fluxes at the cell faces that lay on the boundary need to be evaluated. For example at a rigid wall the flux of momentum in the direction normal to the wall is null. (The flux of momentum in tangential direction at a rigid wall is null if the no-slip BC is imposed.) Layers of additional cells are added to each block in order to impose the BCs. Following a terminology used in finite volume texts such as that by LeVeque [46] the additional cells are called "ghost" cells. Hyperbolic fluxes have stencils extending two cells on each side of a cell that is updated so that two layers of ghost cells are added to the faces where BCs are imposed. The parabolic fluxes have a stencil extending only one cell all around (including corners) of the cell to be updated. For this reason two layers of ghost cells are sufficient, however special treatment is necessary in order to update the values of the conserved variables in the corner ghost cells (presented in Section 4.) A schematic of the two-dimensional stencil for the update of cell (i, j)=$(1, 1)$ is presented in Figure 3.14.

Following a somewhat standard practice in the CFD community cell indices vary from -1 to $icels + 2$, where $icels$ is the number of cells in the i direction. With this indexing the "real" cells extend from 1 to $icels$ for example. The position of the first ghost cell is -1, and of the second ghost cell is 0. Second to last ghost cell has index $icels + 1$ and last ghost cell has index $icels + 2$.

3.4.1 Magnetic Boundary Conditions

The physics of plasma-wall interactions is very rich and cannot be captured with the single fluid MHD model. Simple treatment of wall boundaries is possible by assuming

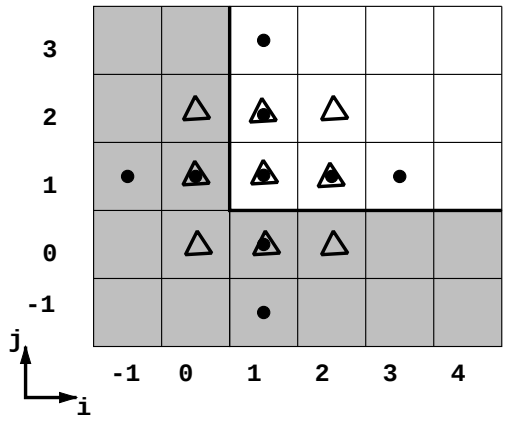

Figure 3.14: Two-dimensional stencil for the hyperbolic (•) and parabolic (△) fluxes that are used to update cell (i, j)=$(1, 1)$. The ghost cells are shaded.

that the walls are either perfect conductors or that the field that soaked in the wall is maintained at a constant value.

Perfectly Conducting Wall Magnetic BC

The perfectly conducting wall assumption is valid if the duration of the experiment is short enough such that the depth of the penetration of the magnetic field (skin depth) in the wall is negligible. The perfectly conducting wall assumption leads to conditions for the electric and magnetic fields at the wall. From Maxwell's equations applied over a control volume that straddles the wall the conditions obtained are that the tangential electric field and the time variation of the normal magnetic field vanish at the wall

$$\mathbf{n} \times \mathbf{E}|_{\text{wall}} = 0, \tag{3.145}$$

and

$$\mathbf{n} \cdot \left.\frac{\partial \mathbf{B}}{\partial t}\right|_{\text{wall}} = 0, \tag{3.146}$$

where $\mathbf{n}$ is the unit vector normal to the wall. Since WARP3 does not keep track of the electric field Ohm's law given by Eqn (2.49) and Ampère's law given by Eqn (2.36) are used to eliminate it from Eqn (3.145). The BCs resulting from substitution of the electric field in Eqn (3.145) are that the tangential components of the curl of the magnetic field are null at the wall

$$\mathbf{n} \times (\nabla \times \mathbf{B})|_{\text{wall}} = 0. \tag{3.147}$$

Two cases of perfectly conducting wall magnetic BCs are implemented in WARP3. The first case is that where initially there is no initial normal magnetic field in the wall. In this case for both conditions represented by Eqns (3.146) and (3.147) to be satisfied the perfectly conducting wall BC imposes a reflection of the normal component of the magnetic field and constant tangential magnetic field across the wall. For historical reasons this BC is called a perfectly conducting wall magnetic BC in WARP3. The second case of perfectly conducting wall BC is described by an initial normal magnetic field of finite value and this BC is called soaked-in magnetic BC and is described below.

Soaked-In Magnetic BC

The soaked-in BC for the magnetic field is the more general perfectly conducting wall BC and makes the assumption that the magnetic field has completely penetrated a conducting wall and for the duration of the experiment no other normal magnetic flux penetrates that wall. In consequence the values of the normal magnetic field components in the wall do not change. This boundary conditions is implemented by specifying the values of the magnetic field in analytic form in the ghost cells corresponding to that wall.

Inflow Magnetic BC

The inflow magnetic BCs are implemented simply by specifying the value of the magnetic field at the inflow boundary. Since the magnetic field is carried inside the

domain by the plasma this is a realistic assumption and it has been found to work well for a magnetoplasmadynamics thruster simulation. The phenomenon of magnetic field being carried by the plasma can be explained by analogy with the vorticity transport in incompressible fluids. The equation for the magnetic field from the single fluid MHD model given by Eqn (2.60) (in dimensional form) is

$$\frac{\partial \mathbf{B}}{\partial t} = \nabla \times (\mathbf{v} \times \mathbf{B}) + \frac{1}{\sigma \mu_0} \nabla^2 \mathbf{B}, \tag{3.148}$$

where $\sigma = 1/\eta$ is the conductivity of the plasma. Eqn (3.148) is strikingly similar to the vorticity equation in incompressible fluids

$$\frac{\partial \omega}{\partial t} = \nabla \times (\mathbf{v} \times \omega) + \nu \nabla^2 \omega, \tag{3.149}$$

where $\omega = \nabla \times \mathbf{v}$ is the vorticity and ν is the kinematic viscosity.

The first term on the right hand sides of Eqns (3.148) and (3.149) describes the advection of the magnetic field and vorticity respectively. Similarly to the advection of vorticity the magnetic field is advected with the fluid. For an ideal plasma ($\sigma = \infty$) a pair of theorems similar to the Kelvin-Helmholtz theorems for advection of vorticity in an inviscid fluid can be found. The ideal MHD theorems were first derived by Alfvén in 1942 and they state that

1. Flux of magnetic field through any closed contours moving with the fluid is constant.

2. Fluid elements that lie on a magnetic field line remain on the same field line.

The first theorem implies that individual flux tubes must move exactly with the fluid and the second theorem implies that fluid particles and the magnetic field move with the same fluid velocity. The second theorem gives the qualitative description that the fluid appears to be attached to the magnetic field, or "frozen" in the magnetic field.

The second term on the right hand sides of Eqns (3.148) and (3.149) describes the diffusion of the magnetic field and vorticity respectively. In plasmas the finite

conductivity σ results in Ohmic losses and the currents that are responsible for the magnetic field decaying away. During the process of magnetic field decay the magnetic energy initially present in the plasma is converted into internal energy (heat). In the case of incompressible fluids the vorticity diffuses by means of entraping more and more mass in the vortex due to momentum transport (viscosity). In MHD the resistivity has the effect of allowing slippage. The slippage arises because the magnetic field line is no longer frozen in the fluid.

Axisymmetric Magnetic BC

This subroutine is used for axisymmetric simulations. The normal and tangential components of the magnetic field vector are calculated. The normal component is copied and the tangential component is reflected (copied with opposite sign) in the ghost cells. The tangential component reflection consistent with the Phi variation which gives an opposite sign at 180^o.

Outflow Magnetic BC

The outflow BCs are implemented by performing simple copy operations of the magnetic field components from the internal (real) cells into the ghost cells. The rationale between this zero-th order extrapolation is that by performing a copy operation the values of the conserved variables are constant across the cell interface on the boundary. Since this generated zero-strength waves no shock reflections are generated at the boundary. This method works well in regions that have both sub and super-critical flow velocities.

3.4.2 Fluid (Hydro) Boundary Conditions

The fluid boundary conditions are used to specify the values of fluid flow related variables such as density, velocity and pressure at the boundaries of the domain.

Rigid Wall Fluid BC

A rigid wall BC is specified by imposing values of the momentum in the direction normal to the wall such that the normal flux is null. This consists of calculating the normal components of the momentum in the cells next to the wall and copying them with a sign change in the ghost cells. The density and pressure are considered to have zero gradient at the wall so that their values are also copied into the ghost cells.

Through a switch in the input file it is possible to impose a slip or no-slip BC. If a no-slip BC is imposed that the tangential momentum at the cells next to the wall is calculated and its value is copied with a sign change in the ghost cells. The result is that the values of the tangential velocities at the wall cancel. If there is a slip condition at the wall then the tangential component of the momentum is just copied into the ghost cell.

Axisymmetric Fluid BC

This subroutine is used for axisymmetric simulations. The normal and tangential components of the velocity vector are calculated. The normal component is copied and the tangential component is reflected (copied with opposite sign) in the ghost cells.

Inflow Fluid BC

The only inflow fluid BC that is implemented in the code is supersonic inflow used in some of the benchmarks. All the characteristics of the Riemann problem at the inflow boundary are oriented inside the domain so that this BC is simple to implement. Values of the fluid flow variables ($\rho, \rho v_x, \rho v_y, \rho v_z, e$) are specified along the boundaries and they are carried inside the domain by the supersonic flow.

Outflow Fluid BC

Similar to the magnetic field BC the fluid outflow BC is implemented as a simple copy operation of the conserved flow variables $(\rho, \rho v_x, \rho v_y, \rho v_z, e)$.

3.5 Time Step Computation and Convergence Criterion

This section contains the description of the method used to calculate the time step and the residual that used in the convergence analysis.

3.5.1 Time Step Calculation

The time step calculation is based on a modified and extended formula widely used in CFD codes. In the procedure that finds the time step two time steps are actually calculated for each cell. One of the time steps is for the hyperbolic part of the PDEs and the other is for the parabolic part of the PDEs. The minimum of the two is then chosen as time step at a certain cell.

The expression for the time step calculation for the hyperbolic is a modification of

$$\Delta t \leq \frac{\nu}{\frac{|v_x|}{\Delta x} + \frac{|v_y|}{\Delta y} + \frac{|v_z|}{\Delta z} + c\sqrt{\frac{1}{\Delta x} + \frac{1}{\Delta y} + \frac{1}{\Delta z}}} \tag{3.150}$$

where ν is the Courant number c is the speed of sound, and Δx, Δy and Δz are cell sizes in each direction. While appropriate for Cartesian orthogonal grids, Eqns (3.150) cannot be used in the context of curvilinear grids. Thus in WARP3 Eqn (3.150) is replaced by

$$\Delta t_h \leq \frac{\nu}{\frac{|v_x|}{\Delta cc_i} + \frac{|v_y|}{\Delta cc_j} + \frac{|v_z|}{\Delta cc_k} + c_f\sqrt{\frac{1}{\Delta cc_i} + \frac{1}{\Delta cc_j} + \frac{1}{\Delta cc_k}}} \tag{3.151}$$

where c_f is the speed of the fast magnetosonic wave, and Δcc_i, Δcc_j and Δcc_k are the distances between cell centers in the i, j and k directions respectively. The Courant number ν is an input to the code.

The time step calculation for the parabolic part of the PDEs is based on a dimensional analysis of a diffusion equation. For example the magnetic field diffusion equation (written in a non-dimensional form) is

$$\frac{\partial B}{\partial t} = \frac{1}{Lu\ Al}\frac{\partial^2 B}{\partial x^2}, \tag{3.152}$$

and using simple differences instead of differentials an approximate equation is derived

$$\frac{\Delta B}{\Delta t} = \frac{1}{Lu\ Al}\frac{\Delta B}{\Delta x^2}. \tag{3.153}$$

Eqn (3.153) can be solved for Δt to give

$$\Delta t = Lu\ Al \Delta x^2 \tag{3.154}$$

In WARP3 time step calculation procedure Eqn (3.154) is implemented as

$$\Delta t_r = \min(\Delta cc_i^2, \Delta cc_j^2, \Delta cc_k^2) Lu\ Al \tag{3.155}$$

For the momentum diffusion equation the time step is calculated similarly

$$\Delta t_v = \min(\Delta cc_i^2, \Delta cc_j^2, \Delta cc_k^2) Re\ Al \tag{3.156}$$

Note that Eqns (3.154) and (3.156) assume that the resistivity and viscosity have a maximum value of one.

The overall minimum value of the three time steps Δt_h, Δt_r and Δt_v over all cells of all blocks is taken as the global time step. In the case of a parallel run the time step on each processor is calculated and then the global time step is obtained using a reduction operation (implemented through a message passing interface (MPI) procedure.)

3.5.2 Residual and Convergence Criterion

Steady simulations are considered converged when the variation of the conserved variables is in the range of machine zero (about 1×10^{-14}). A method of measuring

convergence is to compute the residual of the conserved variables and study its evolution. The residual in this context is the second term of the right hand side of the update Eqn (3.6). If this term is null then $\mathbf{Q}^{n+1} = \mathbf{Q}^n$ meaning that the solution does not vary in time and a steady state has been reached. In WARP3 the residual used for convergence studies is that of the total energy term. This particular residual has been chosen since the total energy contains all the conserved terms in its expression.

The overall residual of the total energy in all cells is computed and then divided by the number of cell. The expression for the overall residual is

$$\mathrm{res}_{\mathrm{tot}} = \frac{\sqrt{\sum_i \sum_j \sum_k \mathrm{res}_{ijk}}}{\mathrm{cells}_{\mathrm{tot}}}, \tag{3.157}$$

where res_{ijk} is the energy residual in cell ijk and $\mathrm{cells}_{\mathrm{tot}}$ is the total number of cell in the computational domain. In the case of a parallel run the overall residual on each processor is calculated and then the global residual is obtained using a reduction operation (implemented through a message passing interface (MPI) procedure.)

Chapter 4

PARALLEL IMPLEMENTATION

The advantage of a parallel code is rapid turnaround time for the solution of large, complex computational problems. Parallel codes are designed to run on multiple processor computers. Ideally, the speed-up is equal to the number of processors assigned to the task. (For example, for the solution of the same problem a parallel code running on four processors would be four times faster than a serial version running on one of these processors.)

This section describes the methods used to design WARP3 as a parallel code. The parallel code takes advantage of the fact that data dependency of the algorithm used to solve the MHD model is loose. Loose data dependency is illustrated by the stencils needed to calculate the hyperbolic and parabolic fluxes. These stencils are shown in two-dimensional form in Figures 3.12 and 3.13. Only two cells placed symmetrically about the faces of the cell to be updated are needed for flux evaluations. In consequence, if the domain on which the solution is sought is divided into subdomains and the algorithm is applied concurrently to the data on each subdomain, then a limited amount of data is exchanged in order to calculate the fluxes at the boundaries between subdomains and thus to maintain the "physical connection" between them. This programming paradigm is called domain decomposition or coarse grain parallelization, and the type of code that implements it is called single program multiple data (SPMD). The approach presented can be applied in general to the parallel implementation of a solver for a system of PDEs that describe conservation laws.

The hardware parallel codes run on are multiple processor computers. These multiple processor computers can be either symmetric multiprocessor (SMP) computers

or distributed memory computers. SMP computers contain a certain number of CPUs that communicate through a high speed bus and share a common memory. The CPUs and the memory of an SMP computer are normally installed on the same motherboard. The operating system administers shared memory access. The advantage of the SMP computers is that interprocessor communication is fast. The disadvantage is that the number of processors that can be allocated to solve a problem is limited by the the number of CPUs on the motherboard. Distributed memory computers consist of compute nodes connected by a communications network. Each node contains a CPU, memory and storage. Distributed memory computers rather than SMP computers are used for large scale computations since as the problem size gets larger more nodes can be allocated to solve the problem. (The limit would be the ability of the user to purchase more nodes and communication equipment.) The disadvantage of distributed memory computers is that the inter-processor communication is slower than that of an SMP computer and in present systems it is the bottleneck. However vendors such as IBM and Cray designed specialized inter-processor communication networks for their parallel computers that partially alleviate this problem.

Work described here showed that a parallel code implemented using a domain decomposition approach can be developed making abstraction of the architecture of the parallel computer and confirmed the speed-up expectations. The enabling technologies were the Fortran90 programming language and the libraries that implement the message passing interface (MPI) standard. The portability of the code between three parallel computers has been tested and the performance of the parallel is excellent, with the speed-up results confirming the validity of the approach.

4.1 Domain Decomposition and Multiblock Grids

Domain decomposition (DD) or coarse grain parallelization is based on the Roman saying "*divide et impera*" (for divide and rule). The idea behind DD is to divide the

original domain on which a solution is sought into subdomains, distribute the data corresponding to each subdomain and a copy of the code to the processors available to run the task. The code running on each process solves the system of PDEs, updates the solution in the cells of each subdomain, and then it exchanges data with its co-workers.

The formal derivation of domain decomposition starts from the integral form of a conservation law

$$\int_{\Omega} \frac{\partial \mathbf{Q}}{\partial t} d\tau + \int_{\Omega} \nabla \cdot \mathbf{F} d\tau = 0, \tag{4.1}$$

where the integral is taken over the entire domain Ω and the parabolic and hyperbolic fluxes are combined into $\mathbf{F}$. The domain Ω is divided into subdomains so that

$$\Omega = \bigcup_{s}^{\max_s} \Omega_s, \tag{4.2}$$

and the integral in Eqn (4.1) is taken over each subdomain Ω_s

$$\int_{\Omega_s} \frac{\partial \mathbf{Q}}{\partial t} d\tau + \int_{\Omega_s} \nabla \cdot \mathbf{F} d\tau = 0, \quad s = 1, 2, \ldots, \max_s, \tag{4.3}$$

where $\max_s$ is the total number of subdomains. The algorithm described in Section 3 is used to solve each of Eqns (4.3) and advance the solution one time step (SPMD approach.) Once the solution is advanced conserved variables are transferred to the neighboring subdomains so that the physical connection between subdomains is maintained.

Before discussing the coarse grain parallelization it is useful to introduce the concept of multiblock grids. As the name implies multiblock grids are composed of multiple blocks and are widely used by the CFD community since they provide the means of creating high quality discretizations of computational domains. A high quality discretization is a grid that has cells with good orthogonality and smooth volume variation throughout the domain. The blocks can be thought of as the first, finer, level of "grains" of the domain decomposition. The next, coarser, level of "grains"

are the subdomains which can contain one or more blocks. The number of domains is equal to the number of processors available to run the parallel task and there is a one to one mapping of the subdomains to the processors. WARP3 has been designed to allow random distribution of blocks to subdomains. The position of each block relative to its neighbors is specified in a grid topology array that is an input to the code. Grid topology and mapping are discussed in Section 4.2.1.

Since the stencils of the hyperbolic fluxes extend two cells in each direction from a cell face the blocks are "padded" with two layers of cells, similarly to the cells needed to implement the boundary conditions. The double layers also contain the parabolic flux stencil cells. The additional cells are used to store the values of the conserved variables needed to evaluate the hyperbolic and parabolic fluxes at the interblock boundaries. They are called *internal* ghost cells to distinguish them from the boundary conditions ghost cells and their geometry corresponds exactly to that of the real cells of the neighboring block from where they receive data. The conserved variables contained in the two layers of real cells next to a block face are transferred to the internal ghost cells of the neighboring block that corresponds to that face. An illustration of this data transfer is presented in Figure 4.1. To better visualize the data transfers imagine that all the blocks collapse to form the original domain. The internal ghost cells marked with bullets ($\bullet$) lay exactly on the top of their corresponding real cells marked with crosses ($\times$) illustrating the relationship between the real cells and the internal ghost cells that receive data from them. For simplicity Figure 4.1 only shows a layer of internal ghost cells so that a "zoomed in" view of an interblock boundary region is shown in Figure 4.2 to illustrate the details.

An important implementation detail should be introduced at this point. If two neighboring blocks reside on the same subdomain, then no message passing is performed and the data transfer is accomplished using simple copy operations. Otherwise, if two neighboring blocks reside on different subdomains, then the data transfer is accomplished using message passing between the processors that have been allo-

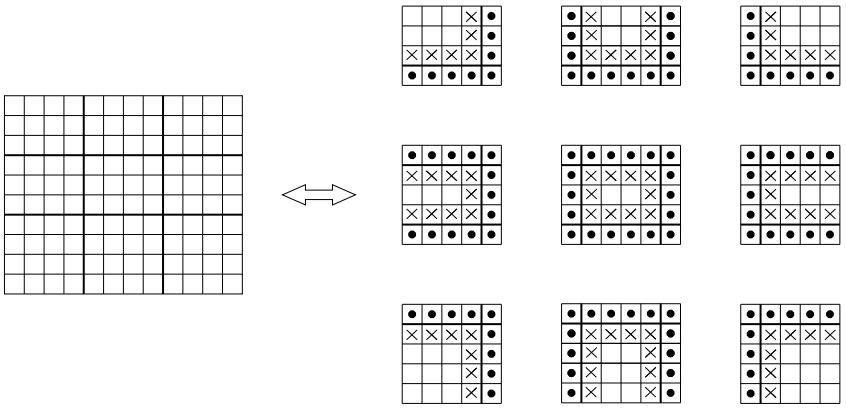

Figure 4.1: Domain decomposition of a two-dimensional grid into nine subdomains. Cells marked by • receive data from neighbors and cells marked by × are send to the neighbors. Note that for the sake of simplicity only one layer of internal ghost cells were drawn and the boundary conditions ghost cells were not included.

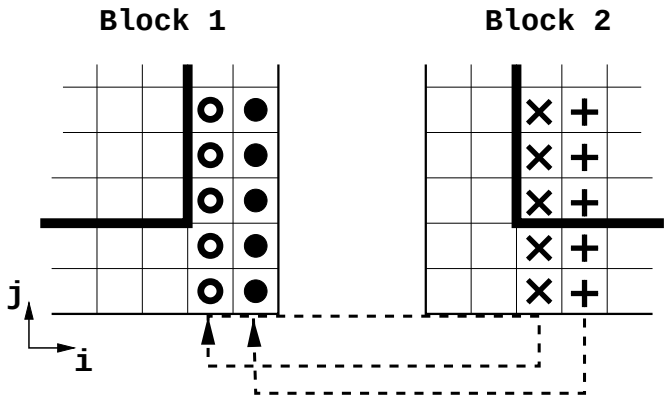

Figure 4.2: Zoomed-in view of the interblock boundary region. Cells outside the thick lines are ghost cells. Ghost cells marked with ∘ and • of block 1 receive their data from real cells marked with × and + respectively of block 2. The data transfer from the real cells of block 1 to the ghost cells of block 2 is not shown.

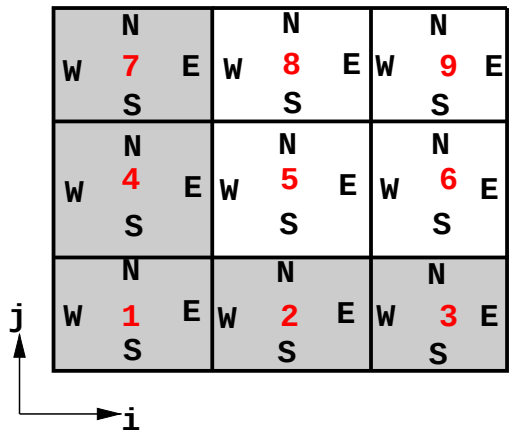

Figure 4.3: Nine blocks of grid from Figure 4.1 distributed to two subdomains showing the global block numbers and the cardinal points used that specify topology. (Blocks on subdomain 0 are shaded.)

cated to each of the subdomains. To illustrate this it is assumed that two processors are available for running the parallel task on the grid shown in Figure 4.1 so that two subdomains are defined. Subdomain notation follows the processor ID notation used by MPI with the first subdomain/processor identified as 0. Blocks are distributed to subdomains such that blocks 1, 2, 3, 4 and 7 are assigned to subdomain 0 and blocks 5, 6, 8 and 9 are assigned to subdomain 1 as shown in Figure 4.3. For example, block 3 exchanges data with its northern neighbor (block 6) and with its western neighbor (block 2). Eastern and southern faces lay on the boundaries of the computational domain so that boundary conditions are applied there. Since blocks 2 and 3 reside on the same subdomain (0), hence on the same processor, the data transfer between them requires copy operations only. Block 6 resides on the other subdomain (processor) so that data transfer is accomplished by message passing.

4.2 Implementation Issues

WARP3 has been designed with the goal to provide a robust and portable parallel solver for the single fluid MHD equations. Numerical experiments and simulations confirmed the robustness of the solver. Portability of the code between different parallel computers architectures was ensured by using Fortran90 and MPI. The same version of the code is used for development and validation on a cluster of DEC Personal Alphastations at the University of Washington Department of Aeronautics and Astronautics and for large scale runs on the massively parallel IBM SP2 at DoD's Maui High Performance Computing Center. As a confirmation of the portability of the code it has to be mentioned that no modifications to the source where necessary when it was first compiled and tested on the SP2. Obviously vendor specific implementations of MPI libraries need to be linked with the code each time it is ported to a different architecture. The SP2 executable uses IBM's proprietary implementation of MPI. More recently Raber *et* al. [55] ported the code to symmetric multiprocessor (SMP) WindowsNT computers, with little modifications, further confirming the portability of the code. The PC-SMP version of WARP3 uses an MPI implementation from the University of Coimbra in Portugal.

WARP3 has been implemented in such a way that a user not familiar with parallel programming should be able to run the code once the operational paradigm is understood. There are only three parallel computing issues a user needs to be familiar with.

1. Blocks are distributed to the processors of a parallel computer as specified in the input file. To be consistent with the concepts presented in the previous section one subdomain and only one is associated with each processor.

2. Processors (their aliases or IP addresses) assigned for the parallel task are specified in a certain file. The location and the name of this file are a function of

the parallel computer used.

3. Each processor that runs the parallel task outputs the results of its subdomain in a file different from that of its co-workers. To obtain the output on the entire domain a UNIX shell script command should be run. (A version of the script for WindowsNT is not yet available.)

The following sections describe some of the important parallel programming issues encountered and solved during the design of the parallel version of WARP3. A user interested only in running the code is referred to Appendix H where code organization details are presented together with a user's guide.

4.2.1 Grid Topology and Local to Global Mapping

In the context of this work grid topology specifies the position of one block relative to its neighbors. The graphic representation of a grid topology is similar to a map where a country's position relative to its neighbors is specified using cardinal points. Similarly, WARP3 uses the cardinal points to specify the position of a block with respect to its neighbors. For example, the neighbors of block 14 from Figure 4.4 are 17 to the North, 15 to the East, 11 to the South and 13 to the West. Additionally, since the grids are three-dimensional, blocks have neighbors above ($k+$ direction) and below ($k-$ direction). Thus the top neighbor of block 14 is block 5 and the bottom neighbor is block 23. Note that this is a direct extension of the convention used for cell notation and orientation which was presented in Section 3.1, Figure 3.1 (Blocks of a structured grid are hexahedra since they are made of hexahedral cells.) This topology specification is common practice for multiblock grids and to this point no parallel computing is involved.

At the initialization of the parallel task the code distributes to each processor the data of the subdomain assigned to it. All grid related data such as cell vertex positions and conserved variables are stored in arrays. In order to optimize memory

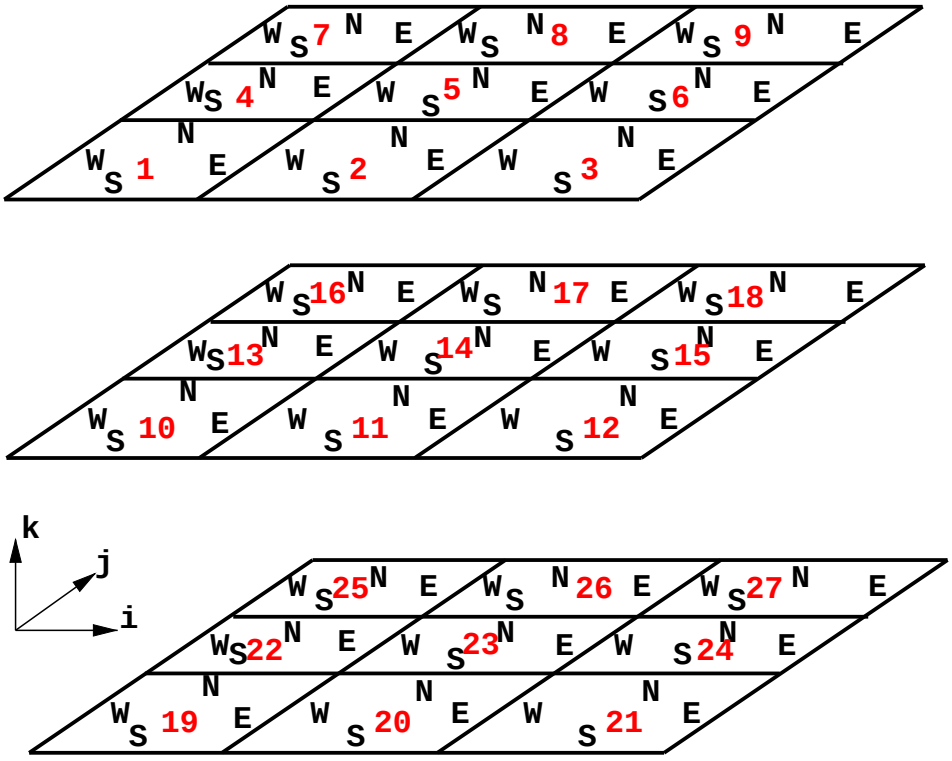

Figure 4.4: Three-dimensional grid divided into twenty-seven blocks. The letters represent the directions and the numbers represent global block IDs. Upward direction is the $k+$ direction and downward direction is the $k-$ direction.

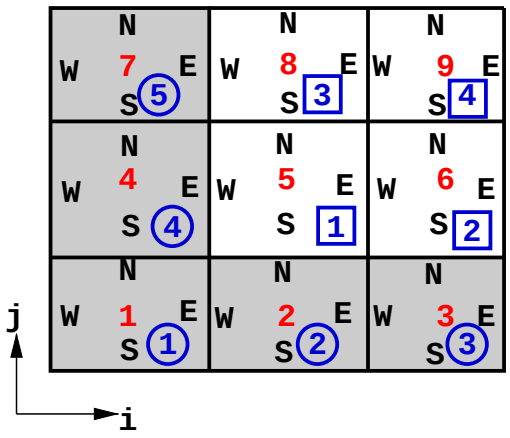

Figure 4.5: Nine-block-grid of Figure 4.3 shown with global and local block IDs. Global block IDs are located near the center of each block and global block IDs are located at the lower right corner of each block. Blocks on subdomain 0 are shaded and their local block IDs are enclosed in a circle. Block IDs on subdomain 1 are enclosed in squares.

use WARP3 employs dynamic arrays. (Dynamic arrays are dimensioned at run time rather than at compile time such as static arrays.) The sizes of these dynamics arrays are given by the maximum number of cells in each direction over all blocks on a processor and the number of blocks per processor.

The blocks on a processor are numbered from 1 to $\max_b(p)$, where $\max_b(p)$ is the total number of blocks distributed to processor p. Local block numbers are different from the global numbers that identify the blocks in the grid topology. To keep track of the relationships between blocks a local to global mapping is determined at the initialization on each processor. This local to global mapping is stored into a vector that at position b, $1 \leq b \leq \max_b(p)$, contains the global block number (or block ID) corresponding to block b on that processor. To illustrate the local to global mapping the grid introduced in Figure 4.3 is shown in Figure 4.5 with local block IDs in addition to the global block IDs. The local to global mapping of the grid in Figure 4.5 is shown in Table 4.1.

Table 4.1: Local and global block IDs for the nine block grid divided into two subdomains as shown in Figure 4.5. This would be a listing of the content of the local to global mapping arrays on each processor.

Processor 0		Processor 1	
Local ID (mapping array index)	Global ID (mapping array value)	Local ID (mapping array index)	Global ID (mapping array value)
1	1	1	5
2	2	2	6
3	3	3	8
4	4	4	9
5	7		

4.2.2 Message Passing and Derived Data Types

Message passing is a means of communication between processes. A process in this context is an executable running on one processor of the parallel computer. Message passing between two processes takes place when one process performs a *send* operation and the other process performs a matching *receive* operation. A message sent to a process by another is uniquely tagged and the message tag and the process ID of the receiving process are given as inputs to the procedure that performs the send. On the receiving process the message tag and the process ID of the sender are given as inputs to the procedure that performs the receive operation. Besides the point-to-point communications global communications such as broadcasts are possible. Global communication procedures are implemented on the top of point-to-point procedures by making the tags and the process IDs wild cards. Both types of inter-process communications are used in WARP3. Point-to-point communications are used for

data transfer from real cells to ghost cells and global communications are used for finding minima and maxima or for reduction operations (such as those needed for data analysis and diagnostics procedures.) The data passed in the messages can be of any type possible in the calling language (C or Fortran) or it can contain a combination of types if derived data types are used.

The message passing procedures used in WARP3 are those from a library that implements the message passing interface (MPI) standard. This standard was widely adopted by academia and industry since its inception in 1994 [23, 24] at the MPI Forum. The Forum required the creation and implementation of a "practical, portable, efficient, and flexible standard for message passing." Implementors followed the MPI standard and created libraries of message passing procedures callable in C and Fortran. The libraries are linked with a code to provide message passing capability. Development and testing of WARP3 is performed with a free version of MPI available as source code from the Argonne National Laboratory at `www-unix.mcs.anl.gov/mpi/` that was compiled on a DEC Personal Alphastation 433au under DEC Unix 4.0 with the DEC C compiler. Large scale runs performed on the IBM SP2 massively parallel computer at Maui High Performance Computing Center use IBM's implementation of MPI. This version is optimized for the high performance switch network that connects the processors on the IBM SP2.

It was mentioned in Section 4.1 that if two neighboring blocks are distributed to different processors the data transfer from the real cells of one block to the corresponding cells of the neighbor is performed with message passing. The message contains the conserved variables (eight for the MHD equations) in each of the two layers of real cells next to the common face of the blocks. (Figure 4.2 shows details of data transfer from block 2 to block 1.) There are two methods to send the values of conserved variables from one block to another. The first and commonly used method is to "pack" the values of the conserved variables into a (long) vector. The packing operation takes place on the sender process. The vector thus generated is

then passed to the receiver process. The receiver process "unpacks" the vector by writing its values in the appropriate conserved variables arrays. The packing and its matching unpacking operation are sometimes called explicit buffering. The advantage of explicit buffering is its relative simplicity and the disadvantage is that packing and unpacking are time consuming. The other method to send data is to define derived data types (DDTs). DDTs allow the definition of complex data structures using basic data types as building blocks and can be thought of as templates that are used to extract the desired values of conserved variables from the arrays that store them. The advantage of DDTs is that they do not require packing and unpacking. DDTs message passing has been implemented in WARP3 after performance tests showed that this method is between three to ten times faster than explicit packing and unpacking on a cluster of DEC Personal Alphastations. (This relatively large range can be explained by the fact that the communications network between the workstations was shared with other users.)

The DDTs used in WARP3 are defined starting from a "zero"- dimensional data structure called a *cell*. A DDT cell contains all eight conserved variables at the (i, j, k) position in a block. A one-dimensional data structure called a *strip* is defined as a vector of cells for each of the i, j and k directions. The next structures in the hierarchy are a *face* and a *frame*. Faces are two-dimensional structures generated as vectors of strips and span a full cross-section of the block, including the ghost cells (both internal and BC.) Frames are special two-dimensional structures made of four strips only. The extent of a face covers the real cells as well as the internal ghost cells around a block. The purpose of frames will be described in Section 4.2.3. The item at the top of the DDTs hierarchy is a *double-face* which is made of two adjacent faces. The double-face is used for transferring conserved variables from one block to another through message passing. A graphical representation of the DDTs discussed here is given in Figure 4.6.

Since MPI [24] does not (yet) support Fortran90 DDTs, the DDTs used in WARP3

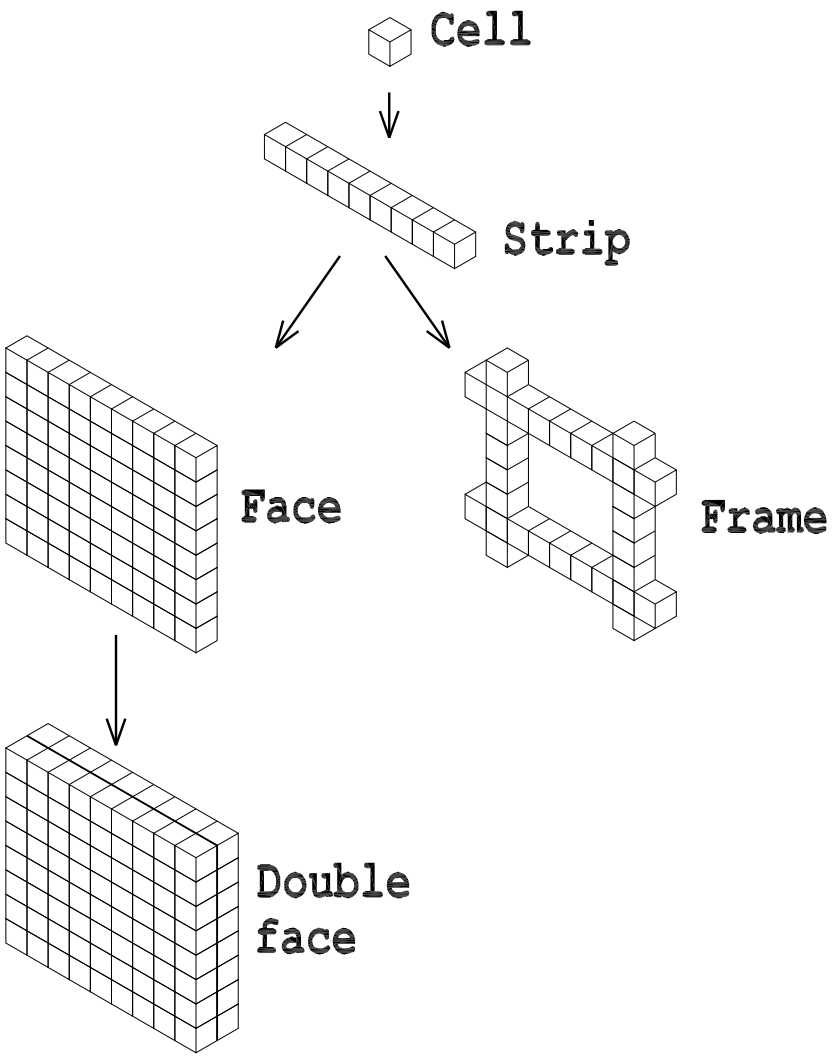

Figure 4.6: Graphical representation of the derived data types used in WARP3. The basic "zero"-dimensional structure is a *cell*. A *strip* is a one-dimensional structure made of cells. A *face* is a two-dimensional structure constructed as a one-dimensional structure of strips. A *frame* is made of four strips only and serves a special purpose as explained in the Section 4.2.3. A *double-face* is made of two adjacent faces and is the structure used to pass conserved variables between blocks that reside on different processors.

were defined using a set of MPI procedures instead of using Fortran90's native DDT declarations.

It was mentioned in the previous section that if two blocks reside on the same process then the data transfer between them is accomplished using copy operations only. Fortran90 aggregate array assignments have been used for block to block copy operations that mimic the message passes. In the future MPI support for Fortran90 DDTs is possible and it might be worthwhile rewriting the DDTs section of WARP3 to use Fortran90 DDTs instead of MPI DDTs so that a unified data transfer method could be implemented.

To improve the performance of WARP3 communication and computation are overlapped. This is accomplished by using nonblocking send and receive operations for point-to-point message passing. A nonblocking send operation initiates a send but does not complete it. The code will return from the send procedure before the message is copied to the send buffer. A separate call to a wait procedure is needed in order to complete the send operation. Nonblocking receives work in a similar manner. The wait procedure for the send operation is posted ahead (in the code stream) of any assignment operations that modify the variables that are sent. For nonblocking receives the wait procedure is posted ahead of any computations that involve the variables that are received. In general the use of nonblocking send and receive operations allows the sender to proceed ahead of the receiver, so that the asynchronicity of the code is increased thus increasing the efficiency. Also the code is more tolerant of fluctuations in the computing speed of the processes. Another advantage of nonblocking communications is that they make WARP3 safe. Snir *et* al. [65] state that in the context of MPI a "portable" code is a "safe" code. Unsafe programs may require system resources that are not guaranteed by MPI, in order to complete succesfully. On systems where resources are not available the program may encounter an error which can manifest itself as an actual program error, or result in deadlock. WARP3 avoids deadlocks and is safe due to the use of nonblocking communications.

4.2.3 Double Message Passing for Parabolic Fluxes

The parabolic flux stencil requires the value of the conserved variables at the internal ghost cells that lie along the edges of a three-dimensional block. For the two-dimensional stencil in Figure 3.14, the internal ghost cells along the edges are those corresponding to position (i, j)=$(0, 0)$. Since the two-dimensional stencil can be thought of as a cross-section section through the three-dimensional stencil the edge in this case would extend in the k direction. The values of the conserved variables in the edge ghost cells $(0, 0)$ are transferred from the real cells at position $(icels, jcels)$ of the block that shares the edge with the target block.

A single data exchange with a double-face message pass if the blocks are on different processors, or its copy equivalent if the blocks are on the same processor, does not guarantee that the correct values of the conserved variables are transferred to the edge ghost cells. However, the real cell data has been transferred to the ghost cells of the other neighbors so that a second message pass will guarantee the correct data transfer. The second data transfer uses the frame DDT structure defined in Section 4.2.2 or its copy equivalent to reduce communication overhead and it is performed only if any of the diffusive fluxes are needed in the computations. The alternative would be to specify sixteen neighbors in the grid topology, six for the faces and twelve for the edges of each block. Data transfer would involve passing double-faces to the blocks that share a face and passing a strip to the blocks that share an edge.

An illustration of the problem is presented in Figure 4.7. For the beginning lets assume that each of the four blocks shown reside on a different processor. The conserved variable of the real cell marked with **a** of block 4 has to be transferred to the shaded cell of block 1. In sequence **A** the first message pass transfers the values of the conserved variables from cell **a** of block 4 to cell **b** of block 2. The second message pass transfers the value from the cell now marked with **a** of block 2 to the edge cell of block 1. The third message pass, presented in frame **C** overwrites the values of

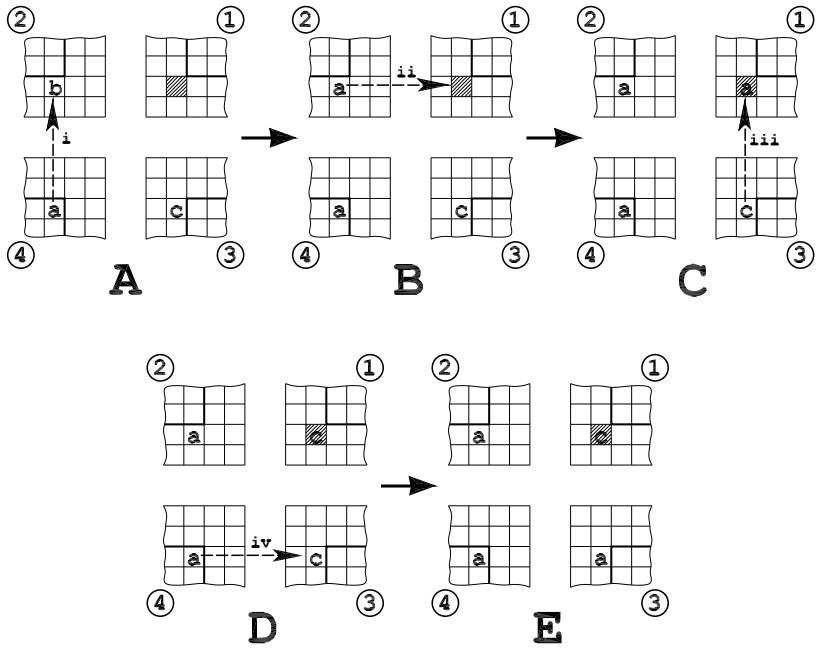

Figure 4.7: Message passing sequence showing that a single message pass does not guarantee that the appropriate values of the conserved variables (denoted with **a**) is transferred to the corresponding edge cell (highlighted). Block numbers are enclosed in circles.

the edge ghost cell of block 1 with the values from ghost cell **c** of block 3. Finally the fourth message pass transfers the value in real cell **a** of block 4 to the ghost cell **c** of block 3. The values of the conserved variables in the edge cell of block 1 are not the correct ones. MPI does not guarantee that the order in which the messages have been sent is the order in which they are received. As a consequence the correct value of the conserved variables might be transferred to the edge ghost cell of block 1 or not. To make sure that the correct value reaches the edge ghost cell a second data transferred is performed. It can be seen in Figure 4.7 that after the completion of the sequence of four data transfers the values of any of the ghost cells that might get transferred to the edge ghost cell is the correct one.

The same problem is encountered if the four blocks in Figure 4.7 are distributed

to the same processor. (Remember that if neighboring blocks are distributed to the same processor the data transfers are performed with copy operations.) In this case the values of the conserved variables that end up in the edge ghost cell are not a function of the order of arrival of the messages but of the block numbering since the copy operations are performed in a loop. To solve the problem in the case the blocks are distributed on the same processor a second copy operation is performed.

4.2.4 I/O Handling

This section briefly discusses data input and output handling in WARP3. The inputs to WARP3 are the data in an input file and the grid that discretizes the domain. The outputs of WARP3 are the cell vertex coordinates and values of the primitive variables at the cell vertices. Details on data input and output and the structure of the input and output files are presented in Appendix H.

Data input in WARP3 is handled by process 0. This process reads the input data from a user specified file and then transfers all the data to its co-workers using message passing. After the input data is transferred, and each process initialized the dynamic arrays it uses to store the grid related data the grid file is read by process 0 and only the grid points corresponding to the blocks distributed to each process are send to the co-workers using message passing. This choice of having process 0 as the input manager avoids any file access contention issues. (If the input and grid files were read by each process that might create file access problems on some systems.)

Data output is handled by each processor and output files can be written to any directory accessible by the owner of the parallel task. The name of each file carries the ID of the process it is output from and an iteration number so that they are uniquely identified. The subdomain data are assembled to retrieve the full computational domain using a Unix shell script that concatenates the files and processes them so that they can be accessed and manipulated with Tecplot.

4.2.5 Load Balancing

The load balancing used in WARP3 is static and it is the responsibility of the user. Given a pool of processors of equal performance static load balancing is achieved by distributing an equal number of cells to each processor. Since the number of floating point operations per cell is constant the load balancing is static implying that once distributed to processors the blocks will reside on their respective processor for the duration of the parallel task. It is up to the user, through the input file, to take care of load balancing. A straightforward modification to WARP3 would be to perform the static load balancing automatically. This would be implemented by a procedure that computes the "ideal" number of cells per processor by dividing the number of cell in the entire domain with the number of processors available for the parallel task. The blocks would then be distributed (by permutations probably) so that the number of cells per processor is as close as possible to the ideal number of cells per processor.

Chapter 5

BENCHMARKS

A standard practice in the CFD community is that once a new code is written or new capabilities are added to an existing code they are validated by putting them through a battery of tests sometimes called benchmarks. It is desirable that the benchmarks compare the results produced by the code with analytical and experimental results that the community at large trusts to be exact solutions of the fluid mechanics problems or accurate observations of fluid physics phenomena. The supersonic flow of air over a wedge is a good example of a benchmark since both analytic solutions and experimental results can be used to validate a CFD code. This section presents the benchmarks used to validate the code and gain confidence in the capability of the code to solve the physics problems described by the single fluid MHD model.

All of the benchmarks shown in this section were performed with the three-dimensional, curvilinear coordinates version of the code. The one-dimensional, two-dimensional and axisymmetric simulations are idealizations of real world phenomena. One and two-dimensional numerical experiments assume that the components of the momentum and the magnetic field in the directions that are left out of the problem are unimportant. The axisymmetric simulations allow the study of problems with axial symmetry with out of plane components of the momentum and magnetic field with the assumption that there is no variation in the azimuthal (ϕ) direction.

The following sections present solutions obtained for a set of one, two and three-dimensional benchmarks. For some of the problems analytic solutions are obtained and they are presented for comparison. For the axisymmetric MPD thruster simulation and the three-dimensional gas dynamics benchmark results produced by WARP3

are compared with other numerical results or experimental observations. The three-dimensional spheromak stability study is compares the results obtained by WARP3 with those generated by linear stability theory and with experimental observations.

5.1 One-Dimensional Riemann Problems

Riemann problems are initial value problems consisting of a system of PDEs that describe conservation laws and a set of initial conditions with a jump in or more of the variables. The Riemann problems studied here are the one-dimensional gas dynamics Riemann problem with initial jumps in density and pressure and a coplanar MHD Riemann problem with initial jump in density, pressure, and y component of the magnetic field, with a background magnetic field in the x direction. The gas dynamics Riemann problem has analytic solutions that give excellent agreement with the numeric results. The MHD Riemann problem does not have analytic solutions but gives excellent agreement with solutions obtained by other researchers. Both types of Riemann problems served as benchmarks for the algorithm that is the core of WARP3 and helped gain confidence in the methods employed.

5.1.1 One-Dimensional Gas Dynamics Riemann Problem

A one-dimensional gas dynamics Riemann problem is an idealization of a shock tube experiment. Shock tube experiments are generally used for hypersonic aerodynamics experiments where a high pressure gas called the driver gas "pushes" a lower pressure gas called the driven gas through a convergent-divergent nozzle. Initially the two gases in the shock tube are separated by a burst diaphragm. Without entering into the details of hypersonic shock tube experiments it is sufficient to say that after the diaphragm bursts the driver gas acts as a piston that pushes the driven gas and a shock wave is generated in the driven gas. The zone where the two gases come in contact is called the contact discontinuity and it is well defined over the duration of

the experiment. An expansion wave propagates in the driver gas so that when steady state is reached the pressures and densities are equalized in the entire tube. The idealization assumes that the gases on each side of the diaphragm are the same, that the gases are in thermal equilibrium so that the pressure ratio across the diaphragm is equal to the density ratio, and that no viscous effects are present.

Analytic solutions for the Riemann problems are found by solving the one-dimensional shock jump relations and they are explained in basic gas dynamics textbooks such as that by Liepmann and Roshko [50]. For example the speed at which the shock is traveling through the low pressure gas is

$$V_s = a_1 \left[1 + \frac{\gamma_1 + 1}{2\gamma_1} \left(\frac{p_2}{p_1} - 1 \right) \right]^{\frac{1}{2}}, \tag{5.1}$$

where a_1 is the speed of sound in the low pressure gas, γ_1 is the specific heat ratios in the low pressure gas, and p_2/p_1 is the ratio of pressures across the shock also called the shock strength. The shock strength is given by

$$\frac{p_4}{p_1} = \frac{p_2}{p_1} \left\{ 1 - \frac{(\gamma_4 - 1)\frac{a_1}{a_4} \left(\frac{p_2}{p_1} - 1 \right)}{\left[4\gamma_1^2 + 2\gamma_1(\gamma_1 + 1) \left(\frac{p_2}{p_1} - 1 \right) \right]^{1/2}} \right\}^{-\frac{2\gamma_4}{\gamma_4 - 1}}, \tag{5.2}$$

where a_4 is the speed of sound in the high pressure gas, and γ_4 is the specific heat ratio in the high pressure gas. Eqn(5.2) is solved numerically with an iterative method for p_2/p_1. The speed of the gas behind the shock wave is given by

$$v_{x,2} = a_1 \left(\frac{p_2}{p_1} - 1 \right) \left[\frac{\frac{2}{\gamma_1}}{(\gamma_1 + 1)\frac{p_2}{p_1} + \gamma_1 - 1} \right]^{\frac{1}{2}}. \tag{5.3}$$

The notation follows standard gas dynamics notation that uses state 4 for the high pressure gas and state 1 for the low pressure gas. State 2 denotes the region through which the shock wave passes and state 3 denotes the region through which the rarefaction wave passes.

Two cases have been investigated with both implicit and explicit schemes. The

Table 5.1: Analytic and numeric (WARP3) results for the Riemann problem with $\rho_l/\rho_r = p_l/p_r = 1.0/0.25 = 4$ shown in Figure 5.1.

	Analytic	Numeric
$\frac{p_2}{p_1}$	1.9342	1.924
V_s	1.5877	1.582
$v_{x,2}$	0.5883	0.585

first Riemann problem was initialized with the following values of the primitive variables

$$\mathbf{w}_1 \equiv \mathbf{w}_r = [0.25, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.25,]^T,$$
$$\mathbf{w}_4 \equiv \mathbf{w}_l = [1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0]^T, \tag{5.4}$$

where $\mathbf{w} = [\rho, v_x, v_y, v_z, B_x, B_y, B_z, p]^T$ is the primitive variables vector and the subscript l and r stand for the *left* and *right* halves of the domain. The ratio of specific heats is $\gamma_1 = \gamma_4 = 1.4$. The grid used for the results shown in Figure 5.1 was divided into 100 cells. For the explicit run the time step was chosen to be $\Delta t = 0.002$ and the simulation was stopped at $t_f = 0.158$. The final time has been chosen so that the shock wave reached the position $x = 0.75$. Solutions obtained with the implicit and explicit schemes lay on the top of each other as can be seen in Figure 5.1. For the implicit scheme pseudo time step Δt^* was chosen to be 1×10^{99} so that the term $1/\Delta t^*$ in Eqn (3.107) is practically null. The analytic and numeric results are presented in Table 5.1.

The second Riemann problem was initialized such that the initial density and pressure ratios were equal to 100.

$$\mathbf{w}_1 \equiv \mathbf{w}_r = [0.01, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.01,]^T,$$
$$\mathbf{w}_4 \equiv \mathbf{w}_l = [1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0]^T. \tag{5.5}$$

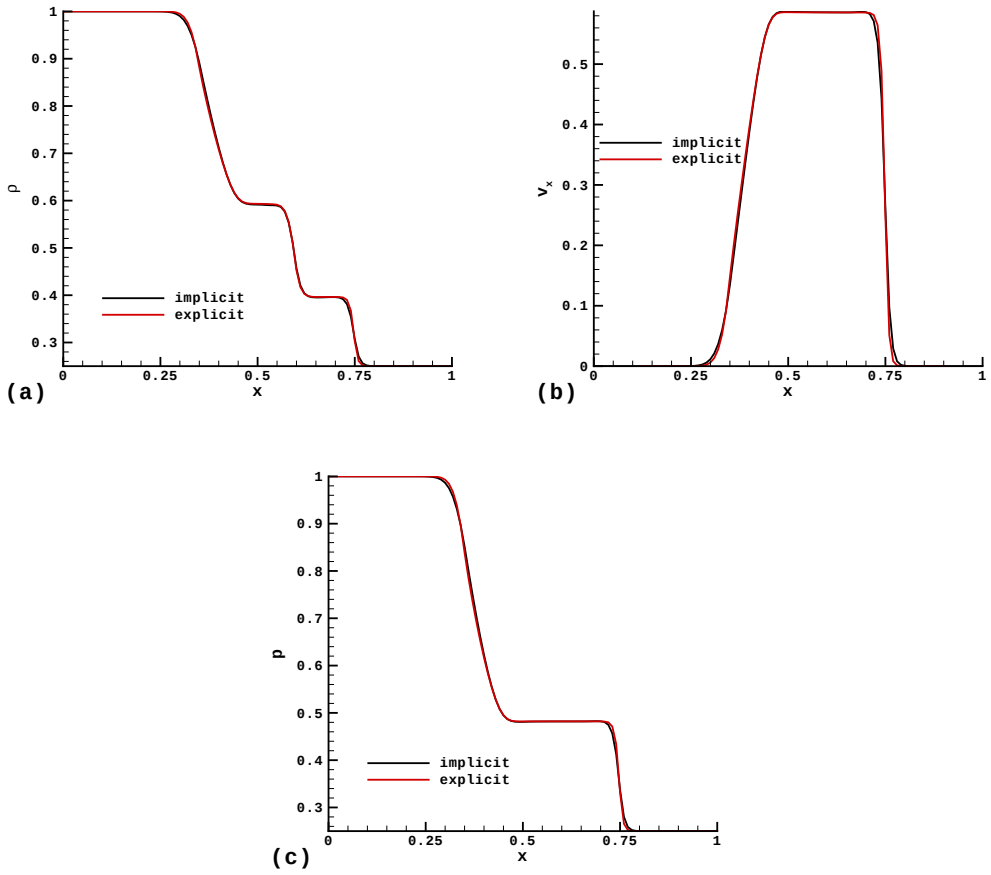

Figure 5.1: Density (a), velocity (b), and pressure (c) of a "mild" Riemann problem solved with the implicit and explicit schemes. Initial pressure and density ratios are 4 (= 1.0/0.25), the number of cells is 100, time step is $\Delta t = 0.02$, and for the implicit scheme $\Delta t^* = 1 \times 10^{99}$, and $m = 5$.

Table 5.2: Analytic and numeric (WARP3) results for the Riemann problem with $\rho_l/\rho_r = p_l/p_r = 1.0/0.01 = 100$ shown in Figure 5.2.

	Analytic	Numeric
$\frac{p_2}{p_1}$	6.3243	6.356
V_s	2.7908	2.75
$v_{x,2}$	1.9087	1.917

This is a rather extreme case and it was suggested by Roe [58] as a means to test the robustness of an approximate Riemann solver and of its sonic fix. The grid used for the results shown in Figure 5.2 was divided into 1000 cells. For both the explicit and implicit runs the time step was chosen to be $\Delta t = 0.0002$ and the simulation was stopped at $t_f = 0.091$. As in the previous case $\Delta t^* = 1 \times 10^{99}$. The analytic and numeric results are shown in Table 5.2.

Numerical results from both the mild and strong Riemann problems show excellent agreement with the analytic results with error less than or equal to 1%. Similar Riemann problems were solved on grids oriented along the Oy and Oz axes and on grids randomly distributed in each of the eight quadrants to test the locally aligned coordinate system approach presented in Section 3.2.4.

During these tests an error has been found in the previous version of the code that used the generalized coordinates approach. The error found is due to incorrectly normalized eigenvectors of the flux Jacobians employed in the Riemann solver. The result of the error is that momentum is "leaking" in the Oy and Oz directions as shown in Figure 5.3. For comparison the results obtained with the new implementation are shown in Figure 5.4. The noise present in the v_y and v_z solutions obtained with the new version of WARP3 is due to the truncation errors in the computation of α_s and α_f of Eqn (C.15). For the case when no magnetic field is present $\alpha_s = 0$ and $\alpha_f = 1$.

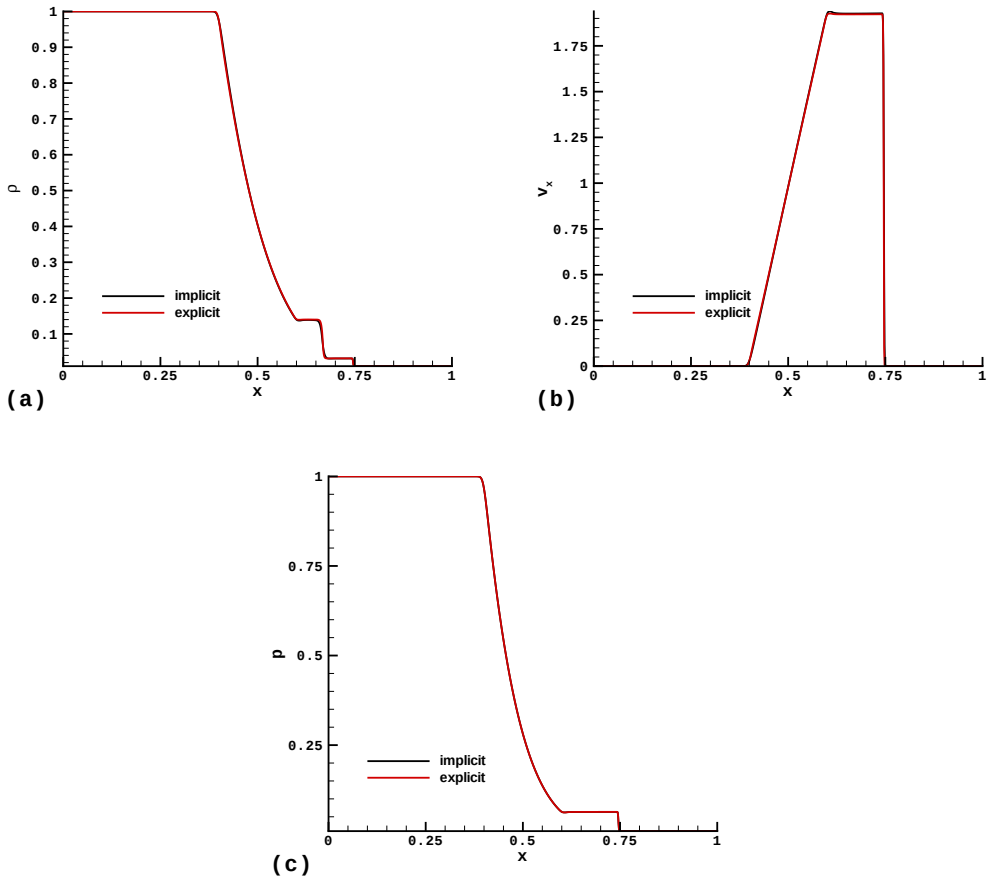

Figure 5.2: Density (a), velocity (b) and pressure (c) of a "strong" Riemann problem obtained with the implicit and explicit schemes. Initial pressure and density ratios are 100 (= 1.0/0.01), the number of cells is 1000, $\Delta t = 0.002$, and for the implicit scheme $\Delta t^* = 1 \times 10^{99}$, and $m = 5$.

Due to truncation errors α_s has a small finite value and for the gas dynamics problems studied in this work the influence of the truncation errors are negligible. The tests were run with the explicit schemes only since the implicit scheme in the old version of the code crashed after the first few time steps.

5.1.2 Coplanar MHD Riemann Problem

This benchmark problem tested the capability of WARP3 to solve MHD Riemann problems. A problem of this type was proposed by Brio and Wu [11] and it is a direct extension of the shock tube test case proposed by Sod [66]. Initial conditions are

$$\begin{aligned} \mathbf{w}_r &= [0.125, 0.0, 0.0, 0.0, 0.75, -1.0, 0.0, 0.1,]^T, \\ \mathbf{w}_l &= [1.0, 0.0, 0.0, 0.0, 0.75, 1.0, 0.0, 1.0]^T. \end{aligned} \tag{5.6}$$

These initial conditions correspond to a jump in the fluid variables combined with a current sheet in the middle of the domain ($d = 0.5$) normal to the domain so that it produces a jump in B_y. Unlike the shock tube problems this problem is unphysical. In a real shock tube filled with a magnetized plasma at the initial conditions given by Eqn (5.6) the magnetic field has to satisfy certain conditions at the walls of the tube. These boundary conditions strongly influence the evolution of the magnetic field and modify the nature of the waves present in the plasma. As mentioned by Zachary and Collela [85] it should not be surprising that the numerical solution appears non-physical. However, it is believed that the test problem is a powerful benchmark for MHD solvers. The solution obtained with the implicit scheme with a time step of $\Delta t = 0.2$ and $\Delta t^* = 1 \times 10^{99}$ after 400 steps is shown in Figure 5.5. The grid consisted of 800 cells and for the case presented here the grid was rotated such that the direction cosines are $(0.8619, 0.2800, 0.4218)$ (first quadrant) corresponding to a rotation about Oz by 18^o and a rotation about the new Oy of 25^o. A note to the researchers that will try to reproduce these results is that the initial components of the magnetic field shown in Eqn (5.6) should be rotated with the grid. There are no

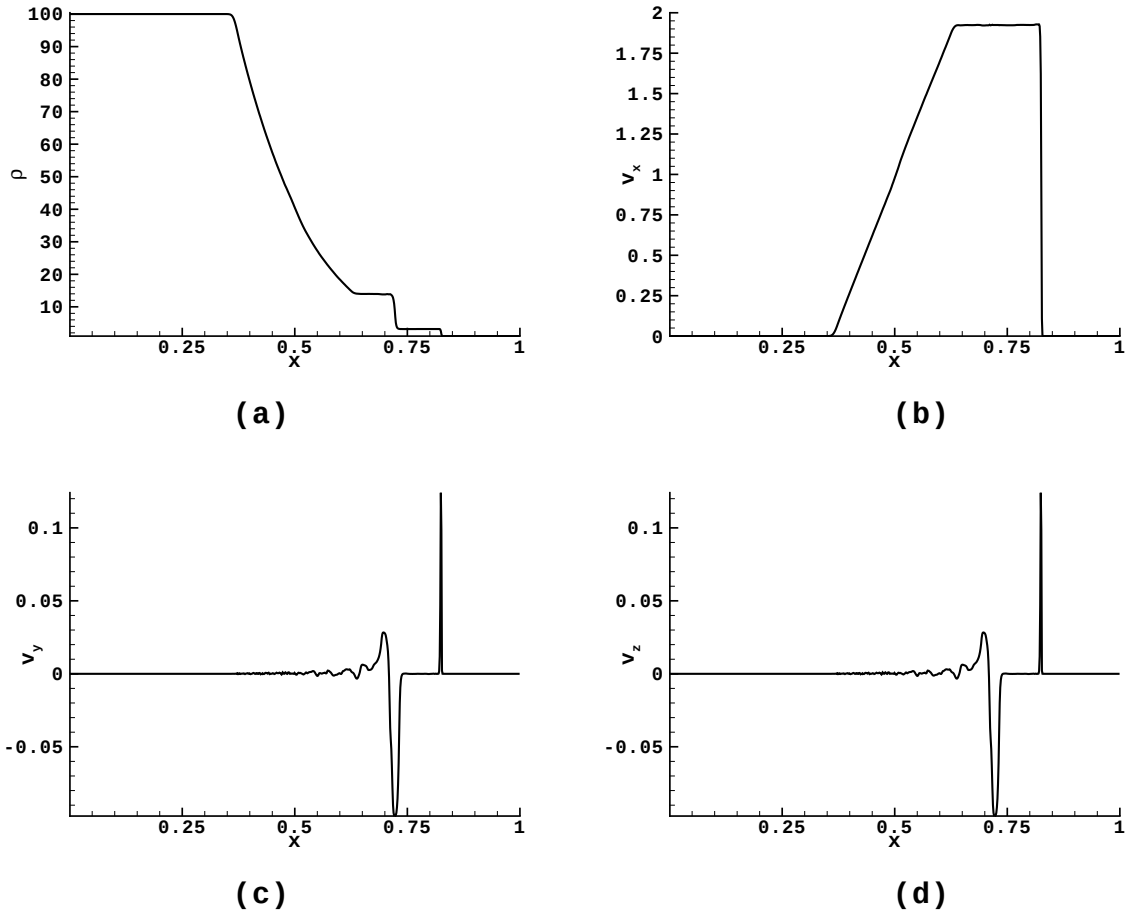

Figure 5.3: Solution of the one-dimensional Riemann problem with initial pressure and density ratios of 100 using the version of the code that employed a generalized coordinates approach. Momentum "leaks" in the Oy and Oz directions.

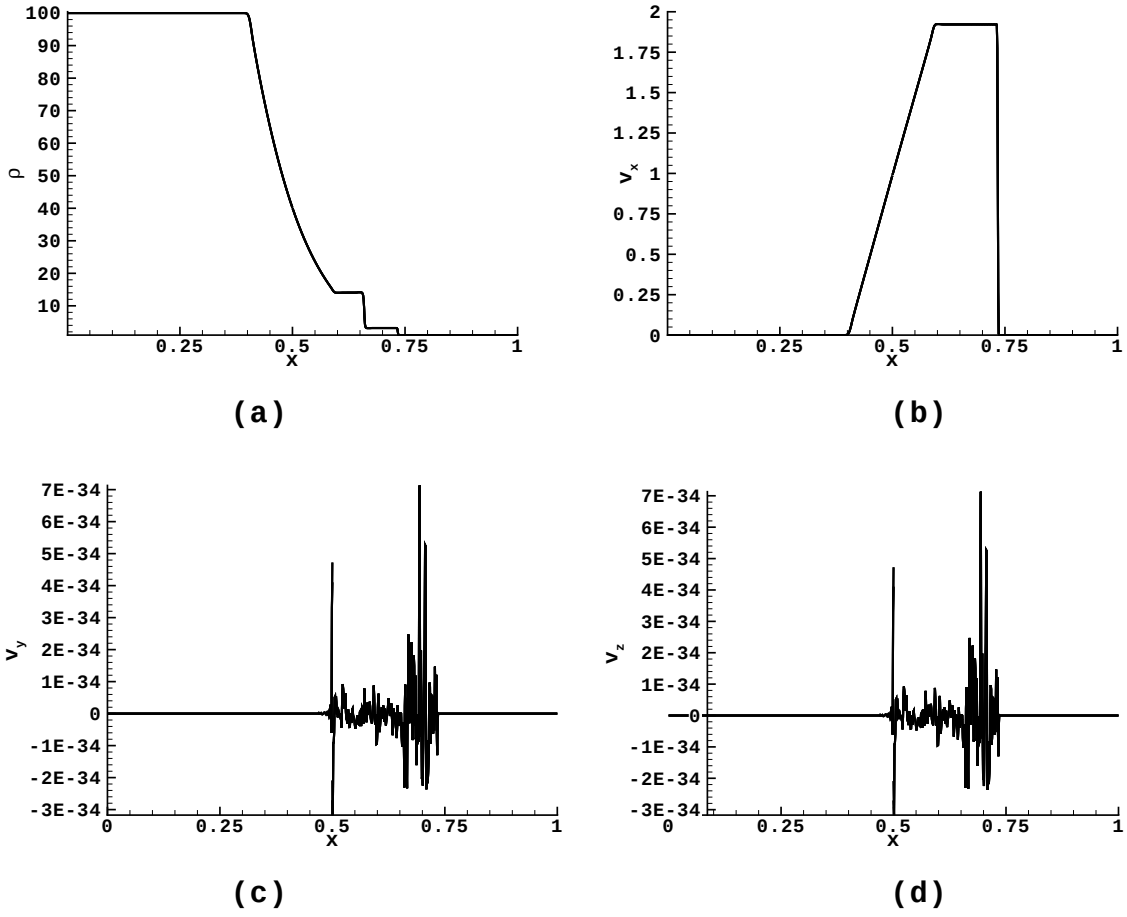

Figure 5.4: Solution of the one-dimensional Riemann problem with initial pressure and density ratios of 100 using the new version of the code that employs a locally aligned coordinate system. The "noise" present in the y and z momenta is due to the truncation errors.

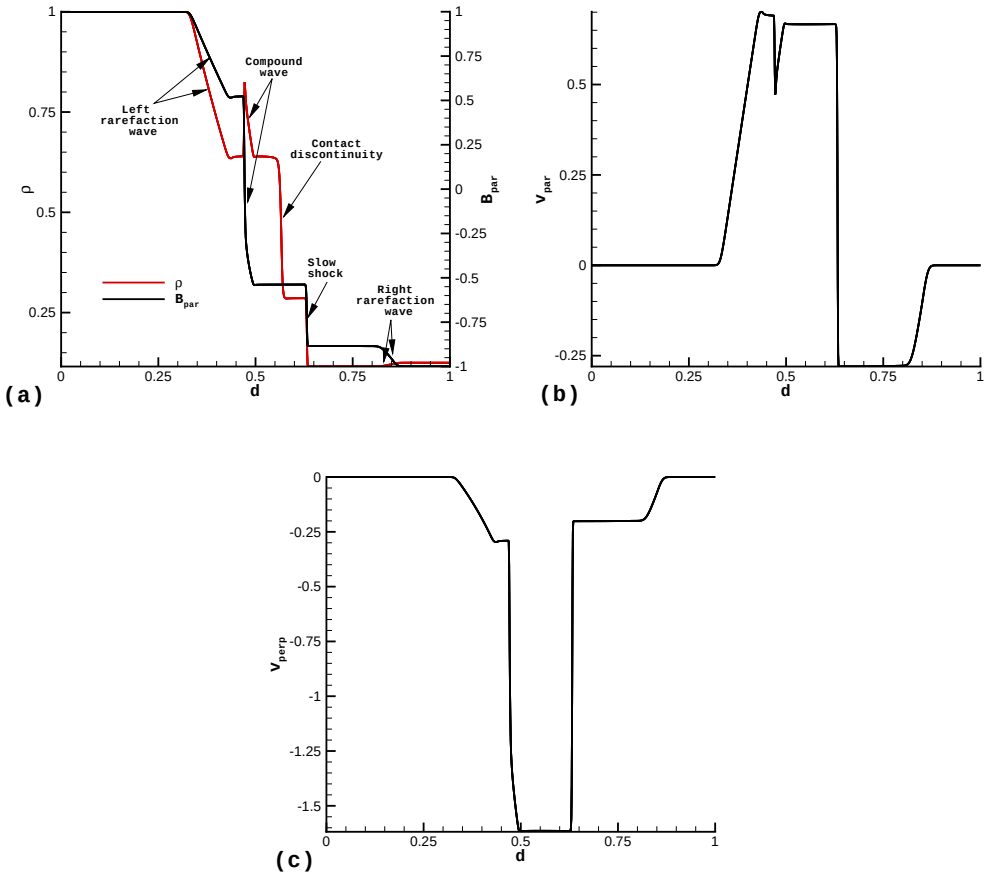

Figure 5.5: Solution of the one-dimensional MHD Riemann problem with initial conditions shown in Eqn (5.6) showing the richness of the single fluid MHD wave solution.

analytic solutions for this Riemann problem but the solution obtained gives excellent agreement with solutions obtained by Brio and Wu [11] and Zachary and Collela [85].

The Riemann problem is called coplanar because only the x and y components of the momentum and magnetic field are allowed, or in the rotated coordinate frame only the n and t_1 components are allowed. The system of PDEs solved is equivalent to that obtained by eliminating the z (t_2) components of the momentum and magnetic field from Eqn (2.62). The effect is that the $v_x \pm c_a$ eigenvalues are eliminated leaving only a system of five waves in the solution (actually six, two of which are collapsed.) The waves that appear in the solution are a left rarefaction wave followed by a compound wave also going to the left. The right going waves are a contact discontinuity, a shock wave and a rarefaction wave. (Some authors refer to the rarefaction waves as fast rarefactions.) These waves are best observed on the density profile in Figure 5.5 (a).

The slight undershoots and overshoots in the profiles are due to the flux limiter method that switches between second order in smooth regions to first order in the regions with sharp gradients of the conserved variables. These "anomalies" diffuse in time and can be eliminated if the eigenvalue cut-off value of ϵ from Eqn 3.21 is increased. For the coplanar Riemann problem the value of ϵ was 1×10^{-4}.

5.2 Diffusion Dominated One-Dimensional Problems

This section presents benchmarks that verified the capability of the code to solve problems where diffusive effects are important. The first case is the flow of a non-conducting gas between two parallel plates one of which is moving. This type of flow is also known as Couette flow. The second case is a time accurate simulation that tested the magnetic field diffusion into a conductor. Third case is an MHD extension to the Couette flow and describes the flow of a magnetized conducting fluid with finite resistivity and viscosity between two parallel plates. In all cases analytic solutions showed excellent agreement with the numeric results.

5.2.1 Couette Flow

Couette flow is the flow of a fluid with zero electric conductivity and with finite viscosity between two parallel plates one of which is at rest, the other moving in its own plane with a certain velocity. This type of flow is one particular case of parallel flow that allows for an exact solution of the Navier-Stokes equations. The assumptions are that the components of the fluid velocity in the x and z directions are null everywhere. With this assumption it follows from the momentum equations that $\partial p/\partial y$ and $\partial p/\partial z$ are null, so that the pressure has only an x dependency. At steady state the momentum equation from Eqn (2.62) becomes

$$\frac{dp}{dx} = \frac{\bar{\mu}}{Re}\frac{d^2 v_x}{dy^2}, \qquad 0 \leq y \leq h, \tag{5.7}$$

where Re is the Reynolds number, h is the distance between the plates, and $\bar{\mu}$ is the dynamic viscosity of the fluid. To define the problem (in the PDE sense) the following boundary conditions are used

$$\begin{cases} y = 0: & v_x = 0 \\ y = h: & v_x = U, \end{cases} \tag{5.8}$$

where U is a specified velocity and the equations have been normalized as explained in Section 2.5.1.

The solution to the problem defined by Eqn (5.7) and Eqn (5.8) is

$$v_x(y) = \frac{y}{h}\left[U - \frac{h^2}{2}\frac{Re}{\bar{\mu}}\frac{dp}{dx}\left(1 - \frac{y}{h}\right)\right]. \tag{5.9}$$

For a null pressure gradient ($dp/dx = 0$) the solution is

$$v_x(y) = \frac{y}{h}U \tag{5.10}$$

and this case is known as simple Couette flow or simple shear flow. Note the linear variation of v_x with y. The general Couette flow is a superposition of this simple

Couette flow over the flow between two parallel walls. The velocity profile for the flow between parallel wall is given by

$$v_x(y) = -\frac{h^2}{2}\frac{Re}{\bar{\mu}}\frac{dp}{dx}\left[\frac{y}{h} - \left(\frac{y}{h}\right)^2\right], \tag{5.11}$$

and it is parabolic. The fact that the two types of flows can be added reveals the linear character of the Navier-Stokes equations in terms of the unknown variables.

Five numeric solutions have been generated corresponding to five values of the pressure gradient and are presented in Figure 5.6. The values of the Reynolds number and of the non-dimensional dynamic viscosity have been chosen such that their ratio is one. For $dp/dx > 0$, i.e., for a pressure decreasing in the direction of the motion, the velocity is positive over the entire width of the channel. For negative values of dp/dx the velocity over a section of the channel can become negative. This result shows that some back-flow might occur near at the wall at rest.

The grid used for the results shown in Figure 5.6 had one hundred cells in the y direction. The implicit scheme was used and tests have been performed with the advective terms on and off. (If the advective terms are turned off then WARP3 is a parabolic PDE solver only.) As for the Riemann problems shown in the previous section the grid was oriented along the x and z axes and randomly in each of the eight quadrants to test the locally aligned coordinate systems. The agreement between the analytic and the numeric results is excellent.

5.2.2 Magnetic Field Diffusion

This benchmark tested the time dependent diffusion of the magnetic field in an electrically conducting fluid in a one-dimensional configuration. The equation that describes this problem is given by the one-dimensional version of the magnetic field equation from system (2.62)

$$\frac{\partial B}{\partial t} = \frac{\bar{\eta}}{Lu}\frac{\partial^2 B}{\partial x^2}, \quad 0 \le x \le L, \quad t \ge 0, \tag{5.12}$$

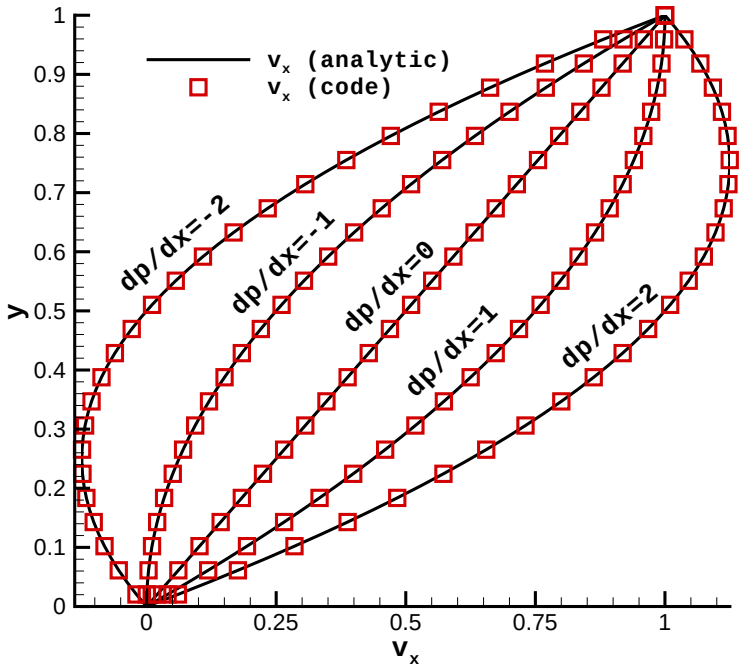

Figure 5.6: Analytic and numeric velocity profiles of a Couette flow. Note the linear profile of the case with null pressure gradient and section of back-flow in cases with negative pressure gradient. Agreement between numeric and analytic results is excellent.

where $\bar{\eta}$ is the normalized resistivity and Lu is the reference Lundquist number. To define the initial-boundary value problem (IBVP) the initial condition specifies that the initial distribution of the magnetic field in the domain

$$B(x,0) = B_0(x), \qquad 0 \leq x \leq L \tag{5.13}$$

and the boundary conditions

$$\begin{cases} x = 0: & B(t) = B_\infty(t), \quad t \geq 0 \\ x = L: & \frac{\partial B}{\partial x} = 0 \end{cases} \tag{5.14}$$

The IBVP defined by Equation (5.12) and conditions (5.13) and (5.14) is solved analytically using a change of variables that makes the equation homogeneous and a separation of variables such that $B(x,t) = X(x)T(t)$ where X and T are the spatial and respectively temporal components of the magnetic field. The analytic solution is

$$B(x,t) = B_\infty + (B_0 + B_\infty)\left[2\sum_{n=1}^{\infty} \frac{\sin\left(\lambda_n \frac{x}{L}\right)}{\lambda_n} e^{-\lambda_n^2 \frac{\alpha t}{L^2}}\right], \tag{5.15}$$

where

$$\begin{aligned} \alpha &= \frac{\bar{\eta}}{Lu} \\ \lambda_n &= \frac{2n-1}{2}\pi. \end{aligned} \tag{5.16}$$

The analytic and numeric solutions are presented in Figure (5.7). The initial condition was $B_0 = 0$, and the boundary condition at the left ($x = 0$) was $B_\infty = 1T$ and the simulation was stopped at $t = 1 \times 10^{-3}$ seconds. The implicit scheme was used on a grid divided into 100 cells. The analytic and numeric solutions show excellent agreement.

5.2.3 Hartmann Flow

This type of flow is an extension of the Couette flow to electrically conducting fluids. The extension is not direct and it should be discussed in some detail. Similar to

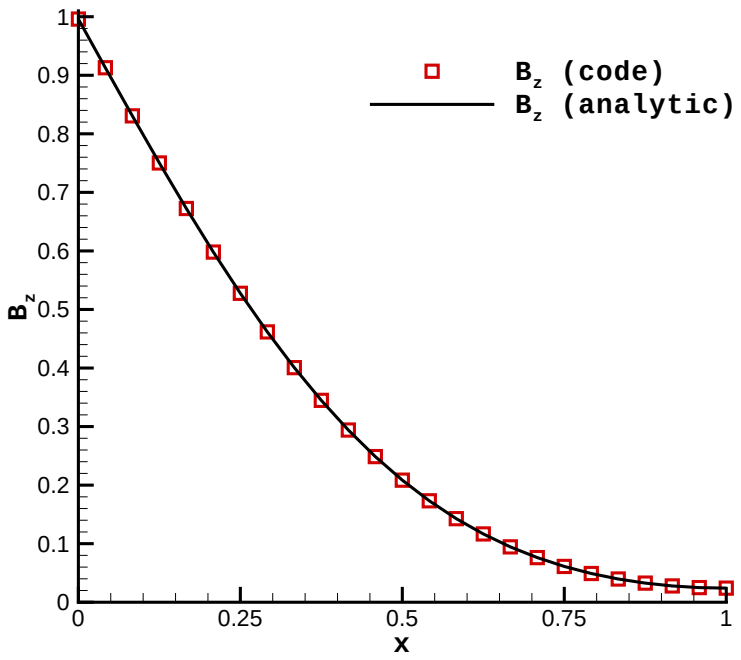

Figure 5.7: Analytic and numeric profile for a magnetic field diffusion problem.

the Navier-Stokes equations the MHD equations are non-linear PDEs because they contain quadratic terms such as $\mathbf{j} \times \mathbf{B}$. However, there are some interesting cases where the equations become linear in terms of the unknown variables and can be solved with analytic methods. These cases constitute a restricted version of MHD from which non-linear phenomena are excluded. They include the interaction of the magnetic and velocity fields ($\mathbf{j} \times \mathbf{B}$ forces) in a degenerate way. The induced field, which is unknown, can enter into equations in linear terms such as $\partial \mathbf{B}/\partial t$, and in quadratic terms where it must not create non-linearities. For example, the induced field should be parallel to the unknown component of the velocity to avoid a product of unknowns $\mathbf{v} \times \mathbf{B}$. To be noted here is that the approach described allows exact solutions which do not rely on approximations or linearizations by neglecting terms that are quadratic in small perturbation quantities (such as in linear stability theory.) The only approximations that are required are that the density (ρ) and the transport coefficients viscosity (ν) and resistivity (η) are constant.

To illustrate the method used to derive, in general, the one dimensional MHD equations the momentum and magnetic field conservation equations are rewritten so that the non-linear terms are emphasized. The momentum equation in vector form is

$$\rho \left[\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} \cdot \nabla)\mathbf{v} \right] + \nabla(p + \frac{B^2}{2}) = (\mathbf{B} \cdot \nabla)\mathbf{B} + \frac{\bar{\mu}}{ReAl} \nabla^2 \mathbf{v}, \tag{5.17}$$

and the magnetic field equation also in vector form is

$$\frac{\partial \mathbf{B}}{\partial t} + (\mathbf{v} \cdot \nabla)\mathbf{B} - (\mathbf{B} \cdot \nabla)\mathbf{v} = \frac{\bar{\eta}}{LuAl} \nabla^2 \mathbf{B}. \tag{5.18}$$

To define the class of problems to which the Hartmann flow belongs it is assumed that v_y is known and that v_x and v_z are unknown. For the term $(\mathbf{v} \cdot \nabla)\mathbf{v}$ to be linear in v_x and v_z their derivatives with respect to x and z should be known. If this is true for v_x and v_z then the term $(\mathbf{B} \cdot \nabla)\mathbf{v}$ is linear only if B_y is known. For the term $(\mathbf{v} \cdot \nabla)\mathbf{B}$ to be linear in v_x, v_z, B_x and B_z, their derivatives with respect to x and z should be known. This last property also makes $(\mathbf{B} \cdot \nabla)\mathbf{B}$ linear. A general one-dimensional problem

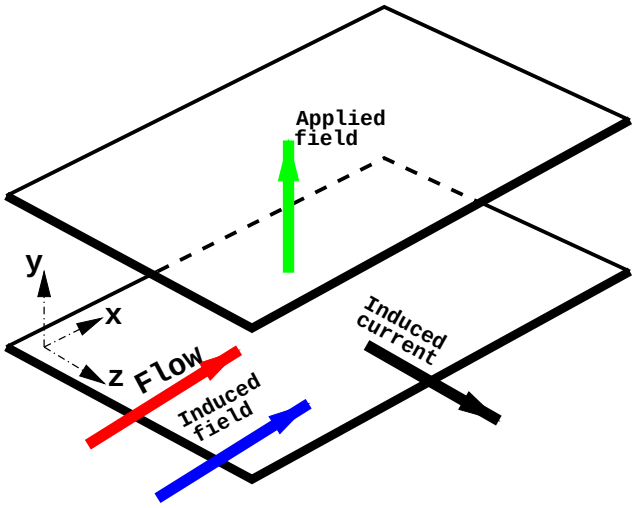

Figure 5.8: Sketch of a Hartmann type of flow.

assumes that the known components of $\mathbf{v}$ and $\mathbf{B}$ are constant and the derivatives of the x and z components with x and z are null. In consequence there are no variations of v_x, v_z, B_x and B_z in the Oxz plane and the components can be decoupled.

The Hartmann flow is a member of the class described above in which v_y is null, B_y is the imposed field and is constant, and v_x and B_x (the induced field) are unknown functions of y only, v_z and B_z also being null. The flow is laminar between two virtually infinite parallel plates. The imposed field B_y is normal to the plates and the flow is produced by motion of the plates. It is assumed in this case that the pressure gradient is null. An illustration of the Hartmann flow is presented in Figure 5.8. In the absence of electric fields in y and z directions currents in the fluid flow only in the direction of $\mathbf{v} \times \mathbf{B}$, i.e. in the x direction. If these currents in the fluid are taken to form closed loops lying in the Oyz plane, each loop belongs to an infinite solenoid composed of similar, parallel loops with the axis in the x direction. Thus the induced field has only an x component, consistent with the assumptions made. In Hartmann flows the viscous diffusion of momentum from the walls (which are moving) competes

with the tendency of the transverse magnetic field to keep the flow field uniform. From another point of view, it can be stated that for the viscous stresses to compete with the magnetic forces, the velocity gradients must be large and this can only occur in the boundary layers as seen in Figure 5.9.

With the assumptions described above the x component of the conservation of momentum and magnetic field equations are

$$\frac{d^2v_x}{dy^2} - \frac{ReLuAl^2}{\bar{\mu}\bar{\eta}\rho}B_y^2 v_x = \frac{ReLuAl^2}{\bar{\mu}\bar{\eta}\rho}B_y E_z,\ 0 \le y \le L \tag{5.19}$$

$$\frac{dB_x}{dy} = \frac{LuAl}{\bar{\eta}}(-E_z - v_x B_y),\ 0 \le y \le L \tag{5.20}$$

where E_z is the electric field in the z direction and it is responsible for the induced current j_z and L is the distance between the plates. Ohm's law ($j_z = (E_z + v_x B_y)/\eta)$) has been used in (5.17) and (5.18) to simplify the equations to a form amenable to solution by ordinary differential equation (ODE) methods. Note that the electric field is an unknown and from an ODE point of view it can be interpreted as an integration constant. Two boundary conditions for each of the equations (5.19) and (5.20) are necessary to define the problem. For the momentum equation the two boundary conditions specify the value of the velocity at the two plates

$$\begin{cases} y = 0: & v_x = v_0 \\ y = L: & v_x = v_L, \end{cases} \tag{5.21}$$

and they are used to evaluate the two integration constants that appear in the solution of the second order ODE (5.19). For the magnetic field equation the two boundary conditions specify the value of the magnetic field at the plates

$$\begin{cases} y = 0: & B_x = 0 \\ y = L: & B_x = 0, \end{cases} \tag{5.22}$$

and are used to evaluate one integration constant and the z component of the electric field. The solutions of equations (5.19) and (5.20) with the boundary condi-

tions (5.21) and (5.22) are

$$v_x(y) = \frac{1}{\sinh(\sqrt{a}L)} \left\{ \left(v_L + \frac{b}{a} \right) \sinh(\sqrt{a}y) + \left(v_0 + \frac{b}{a} \right) \sinh[\sqrt{a}(L-y)] \right\} - \frac{b}{a} \tag{5.23}$$

and

$$B_x(y) = -by \left(\frac{\rho\bar{\eta}}{B_y LuAl} + \frac{B_y^2 LuAl}{\bar{\eta}a} \right) + \frac{B_y^2 LuAl}{\bar{\eta}\sqrt{a}\sinh(\sqrt{a}L)} \left\{ \left(v_L + \frac{b}{a} \right) [\cosh(\sqrt{a}y) - 1] - \left(v_0 + \frac{b}{a} \right) \left\{ \cosh[\sqrt{a}(L-y)] - \cosh(\sqrt{a}L) \right\} \right\}, \tag{5.24}$$

where

$$\begin{cases} a = \frac{B_y^2}{\rho} \frac{ReLuAl^2}{\bar{\eta}} = \frac{B_y^2}{\rho} Ha^2 \bar{\eta} \\ b = \frac{B_y E_z}{\rho} \frac{ReLuAl^2}{\bar{\eta}} = \frac{B_y E_z}{\rho} Ha^2 \bar{\eta} \end{cases} . \tag{5.25}$$

The Hartmann number is defined as

$$Ha = \sqrt{ReLuAl^2} = \frac{BL}{\sqrt{\mu\eta}} \tag{5.26}$$

and it is a measure of the effect of the magnetic force compared to the viscous force. The electric field E_z has not been calculated yet. The simplest solution is the case where the plates move at equal but opposite velocities ($v_0 = -v_L$). In this case $b = 0$ and $E_z = 0$.

The reference transport coefficients where chosen to be unity and tests have been run for a case with $v_0 = v_L = -1$ and at three Hartmann numbers, $1, 10$, and 100. The imposed magnetic field was $B_y = 1$.

The grid was divided into 100 cells and clustering was used to resolve the boundary layers. The length of the domain was $L = 1$. The grids used were generated using an exponential formula such as that presented by Hoffmann [31]. The y coordinate is given by

$$y(\iota) = L \frac{(2\alpha + \beta) \left(\frac{\beta+1}{\beta-1} \right)^{\frac{\iota-\alpha}{1-\alpha}} + 2\alpha - \beta}{(2\alpha + 1) \left[1 + \left(\frac{\beta+1}{\beta-1} \right)^{\frac{\iota-\alpha}{1-\alpha}} \right]}, \tag{5.27}$$

where ι is an equally spaced coordinate between 0 and 1, α is the clustering location parameter (0 for clustering at $y = L$, 1 for clustering at $y = 0$ and 1/2 for clustering at both points), and β is the clustering parameter. By trial and error β has been chosen such that 32 points are located in each of the boundary layers (top and bottom plates.) for the Hartmann number equal to 10 and 100.

The thickness of the boundary layer can be estimated bearing in mind that the x component of $\mathbf{j}\times\mathbf{B}$ force is of order $LuAlv_xB_y^2/\bar{\eta}$ and the viscous force per unit volume in the boundary layers is of order $\bar{\mu}v_x/(\delta^2 ReAl)$. For these forces to be comparable the boundary layer thickness needs to be

$$\delta \approx \frac{\bar{\mu}\bar{\eta}}{HaB_y}. \tag{5.28}$$

With $\bar{\mu} = \bar{\eta} = 1$ and $B_y = 1$ the boundary later thickness of the Hartmann flow is

$$\delta \approx \frac{1}{Ha}, \tag{5.29}$$

not surprisingly similar to the "rule of thumb" in aerodynamics that the thickness of the boundary layer on a wing is inversely proportional to the Reynolds number. From (5.29) it can be noticed that for large Hartmann numbers the effect of the $\mathbf{j}\times\mathbf{B}$ force is that the plate motion is felt by the fluid in a narrow region in the vicinity of the plates. In the case where $Ha = 1$ there are no boundary layers, the effect of the $\mathbf{j}\times\mathbf{B}$ force is negligible, and the velocity profile is linear, similar to that of the simple Couette flow (zero pressure gradient.) These results and the excellent agreement between the analytic and numeric solutions are shown in Figure 5.9. The grids used for the three cases are shown in Figure 5.10.

5.3 Two-Dimensional Rankine-Hugoniot Benchmarks

The benchmarks presented in this section were aimed at testing the capability of WARP3 to solve correctly idealized two-dimensional supersonic flows. Results generated by the code were compared to the analytic solutions of the Rankine-Hugoniot

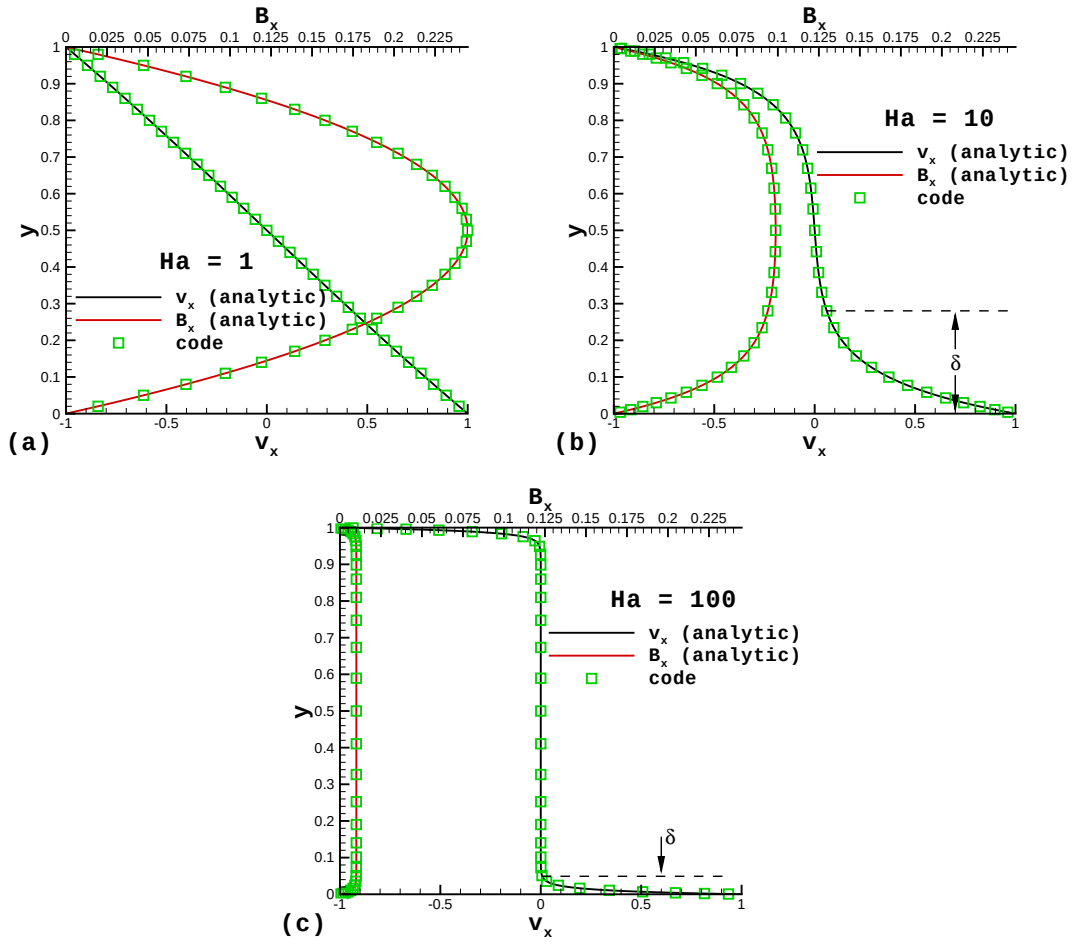

Figure 5.9: Hartmann flows with $Ha = 1$ (a), $Ha = 10$ (b), $Ha = 100$ (c). Note that as pointed by the dimensional analysis for the case with $Ha = 1$ there are no boundary layers and the velocity profile is similar to that of the simple Couette flow. For $Ha = 10$ and $Ha = 100$ the motion of the plates is felt to a distance inversely proportional to the Hartmann number. The values of the induced magnetic field (B_x) have not been scaled so that the relative magnitudes can be compared.

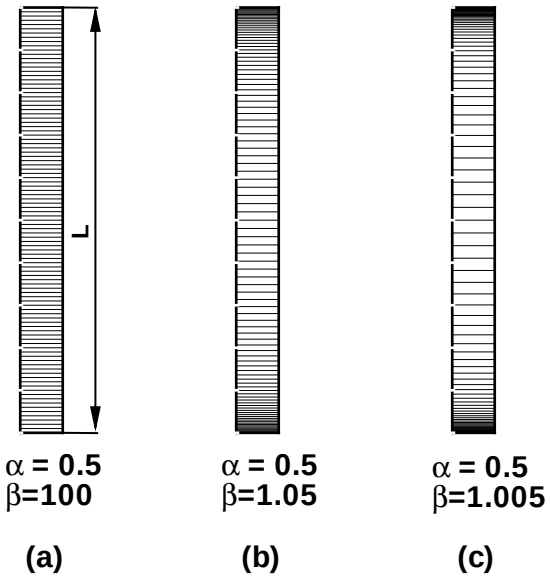

Figure 5.10: Grids used for the Hartmann flows. All grids had 100 points. Grid (a) is equally spaced, grid (b) was generated using a clustering factor $\beta = 1.05$ and grid (c) was used using a clustering factor $\beta = 1.005$.

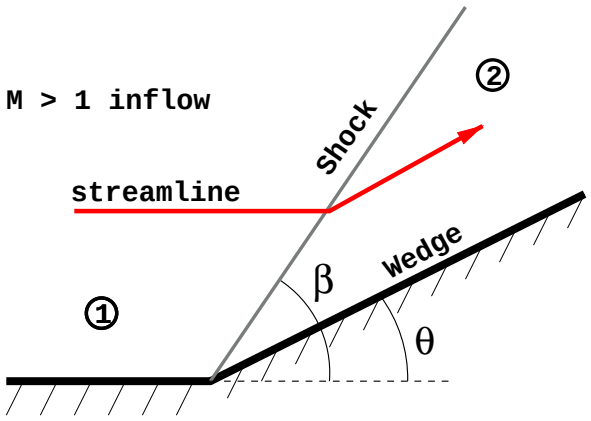

Figure 5.11: Illustration of a supersonic flow over a wedge setup and notation. Angle notation is that used by Liepmann and Roshko [50].

shock relations, that solve the steady state two-dimensional Euler or ideal MHD equations over a control volume aligned with the shock. In real life viscous effects are present but they are considered negligible for the purpose of the benchmarks. The viscous effects are important in the study of the shock-boundary layer interactions, for example.

The physical set-up is presented in Figure 5.11. Incoming flow is supersonic and at the intersection with the wedge an oblique shock forms. The pressure and the density of the gas downstream of the shock are larger than those of the incoming flow so that the effect of the shock is to compress the flow. For this reason the phenomenon is known as *compression by turning*. The angle the wedge makes with the horizontal is denoted θ and the angle of the shock with the horizontal is denoted β, a convention used by Liepmann and Roshko [50]. Following the same convention the state upstream of the shock is denoted 1 and the state downstream of the shock is denoted 2.

Two benchmarks have been performed. The first benchmark assumed that the fluid is a neutral gas (non-conducting) so that the problem is similar to that described by the two-dimensional Euler equations and the appropriate boundary conditions.

The second benchmark assumed that the fluid is a perfect conductor and that it carries with it a magnetic field transverse to the flow. The problem described by the second case is equivalent to the two-dimensional ideal MHD equations plus the boundary conditions. Numeric results are in excellent agreement with the analytic results obtained by solving the Rankine-Hugoniot jump relations. Both benchmarks are steady state problems and the steady state history are similar and the MHD wedge flow convergence is presented.

5.4 Supersonic Gas Dynamics Flow over a Wedge

The first supersonic wedge flow benchmark assumed that the fluid is non-conducting and inviscid. A simple rectangular grid, with equally spaced points, so that in order to simulate the angle of the wedge the incoming flow has to come at an angle with respect to the grid lines. The alternative would have been to use a deformed grid conformal with the wedge geometry and 0^o angle of attack for the inflow. Boundary conditions specified supersonic flow on the North and West sides of the domain, a wall boundary condition at the South side of the subdomain and an outflow boundary condition at the East side, as shown in Figure 5.12. Initial conditions specified null velocity inside the domain. The grid has an aspect ratio (length/height) of 2 divided into four blocks, each block containing 32 cells in the i direction and 16 cells in the j direction for a total number of cells of 2048. The results of a simulation with $M_{\text{in}} = 3.0$ and an angle of incidence $\theta = 25^o$ is shown in Figure 5.13. The inflow density was $\rho_{\text{in}} = 1$ and the inflow pressure was $p_{\text{in}} = 1/\gamma = 0.7143$, where $\gamma(= 1.4)$ is ratio of the specific heats. The reason for choosing $p = 1/\gamma$ was that the speed of sound at inflow is equal to the square root of density hence it is equal to 1. (The speed of sound for an ideal gas is given by $c = \sqrt{\gamma p/\rho}$.) The physical Courant number was $\nu = 100$ and the pseudo time Courant number was $\nu^* = 0.5$. Simulation was stopped when the residual stabilized about a value of 1×10^{-14}. Both limit and complex number

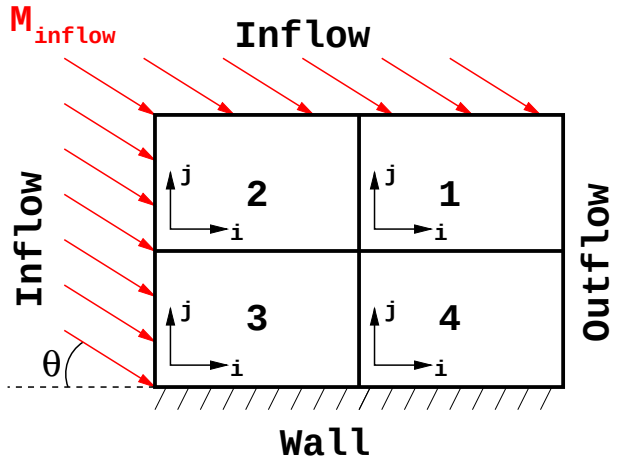

Figure 5.12: Illustration of the boundary conditions for the gas dynamics supersonic flow over a wedge simulation. Block numbers and the $i-j$ orientation is also shown. Note that the incidence angle of the flow is equal to the angle of the wedge.

formulations of the hyperbolic flux Jacobians have been tested and they have been found to work equally well.

To check the numerical results analytically the Rankine-Hugoniot relations are solved. These relations are derived by integrating the steady state two-dimensional Euler equations over a control volume aligned with the shock. The control volume is placed symmetrically on the shock and it extends a distance l along the shock and a distance ϵ normal to the shock, where $\epsilon \ll l$ as illustrated in Figure 5.14. The order of the integral over the control volume of the divergence of the flux is reduced by application of the divergence theorem

$$\int_V \nabla \cdot \overleftrightarrow{f} d\tau = \int_\Sigma \overleftrightarrow{f} \cdot \hat{\mathbf{n}} d\sigma = 0, \tag{5.30}$$

where V is the control volume, Σ is the surface bounding the control volume, $\hat{\mathbf{n}}$ is the

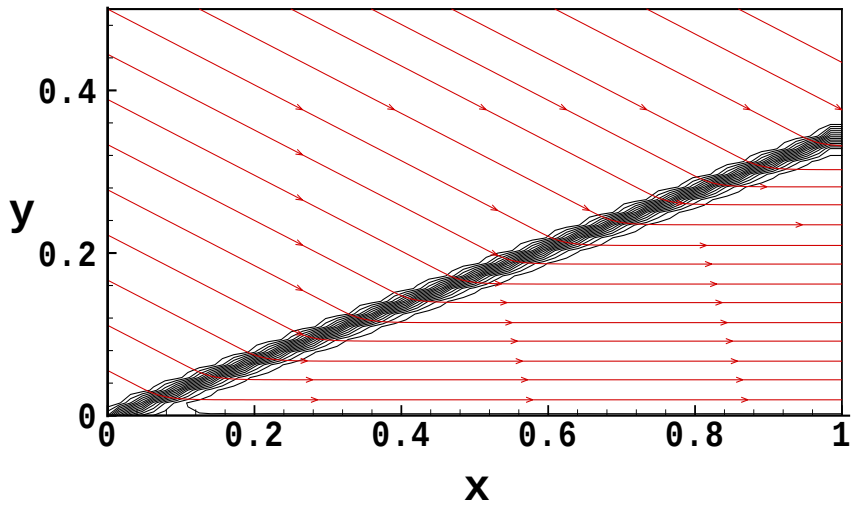

Figure 5.13: Density contours and stream lines for an $M = 3$ flow over a wedge with an angle $\theta = 25^o$.

unit vector normal to Σ, and the flux tensor is $\overset{\leftrightarrow}{f} = \mathbf{f}i + \mathbf{g}j$. The flux vectors are

$$\mathbf{f} = \begin{bmatrix} \rho v_x \\ \rho v_x^2 + p \\ \rho v_x v_y \\ v_x(e+p) \end{bmatrix} \text{ and } \mathbf{g} = \begin{bmatrix} \rho v_y \\ \rho v_x v_y \\ \rho v_y^2 \\ v_y(e+p) \end{bmatrix}, \tag{5.31}$$

and i and j are the unit vectors along the Ox and respectively Oy axes. Integration of the fluxes over the surface of the control volume yields a discrete equation involving the fluxes evaluated upstream (state 1) and downstream (state 2) of the shock. Assuming that $\epsilon \ll l$ the contribution of the end faces is neglected so that the relationship between the fluxes is

$$(\mathbf{f}_1 i + \mathbf{g}_1 j) \cdot \mathbf{n}_1 + (\mathbf{f}_2 i + \mathbf{g}_2 j) \cdot \mathbf{n}_2 = 0, \tag{5.32}$$

where subscripts denote the state at which the fluxes are evaluated. The values of

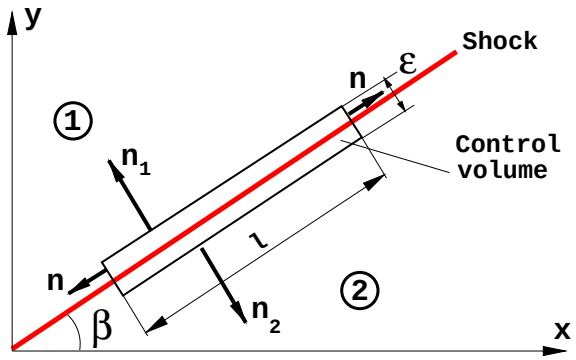

Figure 5.14: Illustration of the control volume and notations used in derivation of the Rankine-Hugoniot relations. The shock angle is β.

the primitive variables at state 1 are known. From geometric considerations

$$\begin{cases} i \cdot \mathbf{n}_1 &= -\sin\beta \\ i \cdot \mathbf{n}_2 &= \sin\beta \end{cases} \quad \text{and} \quad \begin{cases} j \cdot \mathbf{n}_1 &= \cos\beta \\ j \cdot \mathbf{n}_2 &= -\cos\beta \end{cases}, \tag{5.33}$$

so that Eqn (5.32) becomes

$$(\mathbf{f}_2 - \mathbf{f}_1)\tan\beta = \mathbf{g}_2 - \mathbf{g}_1. \tag{5.34}$$

Eqn (5.34) is a non-linear system of four equations with five unknowns. The unknowns are the primitive variables at state 2 and the shock angle b. The fifth equation is obtained from the relationships between v_x and v_y and v_n and v_t, where v_n and v_t are the components of the velocity vector normal and tangential to the shock.

Eqn (5.34) and the additional equation can be solved analytically. Solutions are found in compressible flow textbooks such as that by Oosthuizen and Carscallen [51]. and are not reproduced here. The solutions are normally presented in the form of charts that relate the upstream Mach number (M_1) and the wedge angle (θ) to the wave angle (β) and the downstream Mach number (M_2). For the inflow Mach number considered $M_1 = 3$ and the wedge angle $\theta = 25$ the analytic solution obtained by Oosthuizen and Carscallen [51] and the WARP3 simulation results are in excellent

Table 5.3: Analytic (Reference [51]) and numeric (WARP3) results for the gas dynamics supersonic flow problem with $M_1 = 3$ and $\theta = 25^o$ shown in Figure 5.13.

	Analytic	Numeric
M_2	1.7	1.704
β	44	43.97

agreement. The results are presented in Table 5.3. The Mach number downstream of the shock was calculated based on the values of pressure and density randomly picked from the downstream region and their values were $p_2 = 3.518$ and $\rho_2 = 2.798$, giving a speed of sound $c_2 = 1.338$. The components of the velocity downstream of the shock were $v_x = 2.278$ and $v_y = 1 \times 10^{-4}$. Pressure and density were found to vary at the third decimal place throughout the downstream region and the components of the velocity were found to vary at the fourth decimal place. The shock angle was calculated based on the angle of the density contour plots and was found to be $\beta' = 18.97$. To obtain the true shock angle this angle was added to the flow incidence angle $\theta = 25^o$.

Eqn (5.34) was also solved numerically with Maple (a symbolic/numeric mathematics package). The results obtained are listed together with the WARP3 results in Table 5.4. The agreement between the analytic results and the WARP3 results is excellent.

5.5 Supersonic MHD Flow over a Wedge

Once confidence was gained in the code a more complex wedge flow problem has been solved with the purpose to benchmark the MHD Riemann solver. This benchmark is also a supersonic flow over a wedge with the difference that both the fluid and the wedge material are perfect conductors and a transversal magnetic field is applied at

Table 5.4: Analytic (Maple) and numeric (WARP3) results for the gas dynamics supersonic flow problem with $M_1 = 3$ and $\theta = 25^o$ shown in Figure 5.13.

	ρ_2	$v_{x,2}$	$v_{y,2}$	p_2	β
Analytic	2.796	2.279	0	3.517	44.135
Numeric	2.8	2.28	1×10^{-4}	3.518	43.97

the inflow boundaries. The fluid is perfectly conducting so that the magnetic field is "frozen-in" it. As a result the magnetic field is brought from the boundaries into the domain. Due to the interference with the wedge the flow turns, just as in the case of the purely gas dynamics flow and since the magnetic field is frozen-in it also turns.

The setup of the problem is similar to that of the gas dynamics wedge flow. The inflow Mach number is $M_1 = 3$, the wedge angle is $\theta = 25^o$. In addition, the only non-null component of the magnetic field supplied at the inflow boundaries is $B_y = 0.1$. Similarly with the purely gas dynamics wedge flow the inflow density was $\rho = 1$ and the inflow pressure was $p = 1/\gamma$ where $\gamma = 1.4$. The implicit scheme used a physical Courant number of $\nu = 100$ and a pseudo time Courant number of $\nu^* = 0.5$. Both limit and complex number formulations of the flux Jacobians were tested and it has been found that the complex number formulation does not work well. The formulation produces unphysical variations of the magnetic field in the region downstream of the shock. Since the same formulations have been used succesfully in the purely gas dynamics flow benchmark it is believed that the errors have to do with the normalization of the eigenvectors of the approximate flux Jacobian and they should be subject of further investigation. Results of this simulation (using the limit formulation of the flux Jacobians) are presented in Figure 5.15. Solution of the Rankine-Hugoniot shock jump relations similar to that described in the case of the purely gas dynamics wedge flow show excellent agreement to the WARP3 results

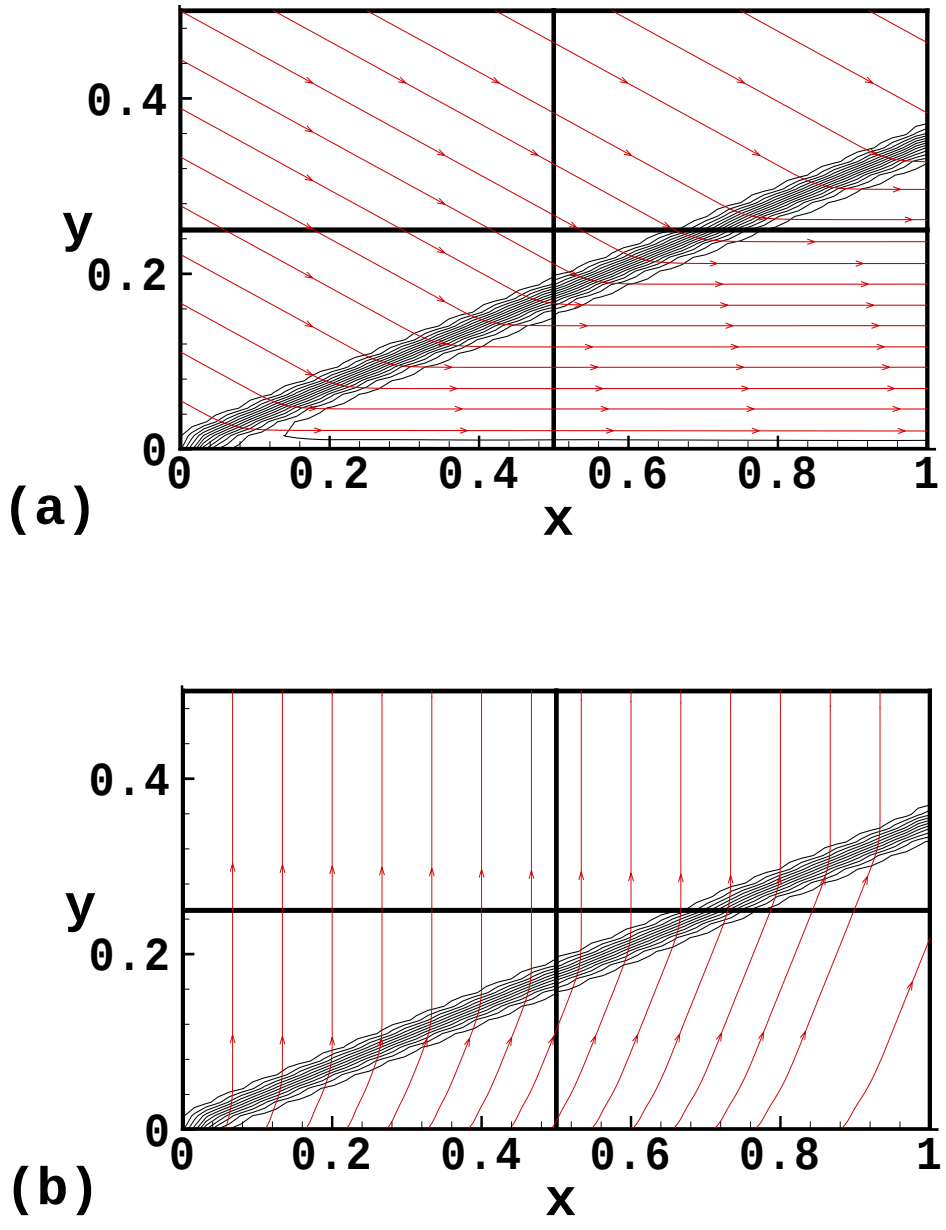

Figure 5.15: Density contours and stream lines (a) and pressure contours and magnetic field lines for an $M = 3$ flow over a wedge with an angle $\theta = 25^o$. Note that the magnetic field lines turn at the shock.

Table 5.5: Analytic (Maple) and numeric (WARP3) results for the magnetohydrodynamics supersonic flow problem with $M_1 = 3$ and $\theta = 25^o$ shown in Figure 5.13.

	ρ_2	$v_{x,2}$	$v_{y,2}$	$B_{x,2}$	$B_{y,2}$	p_2	β
Analytic	2.812	2.27	0	0.122	0.058	3.517	44.135
Numeric	2.8	2.272	1×10^{-5}	0.056	0.119	3.518	43.97

and they are presented in Table 5.5. The analytic results were obtained by solving Eqns (5.34) with Maple with the following fluxes

$$\mathbf{f} = \begin{bmatrix} \rho v_x \\ \rho v_x^2 - B_x^2 + p + B^2/2 \\ \rho v_x v_y - B_x B_y \\ 0 \\ v_x B_y - B_x v_y \\ (e + p + B^2/2)v_x - (\mathbf{B} \cdot \mathbf{v})B_x \end{bmatrix}, \text{ and } \mathbf{g} = \begin{bmatrix} \rho v_y \\ \rho v_y v_x - B_y B_x \\ \rho v_y^2 - B_y^2 + p + B^2/2 \\ v_y B_x - B_y v_x \\ 0 \\ (e + p + B^2/2)v_y - (\mathbf{B} \cdot \mathbf{v})B_y \end{bmatrix}. \tag{5.35}$$

The magnitude of the out-of-plane components of the velocity and magnetic field vector (v_z and B_z) that are null in Eqn (5.35) have values of less than 1×10^{-19} in the results generated by WARP3. Pressure and density were found to vary at the third decimal place throughout the downstream region and the components of the velocity and magnetic field were found to vary at the fourth decimal place. Note that the fluid parameters, such as density, pressure and velocity components, downstream of the shock and the shock angle are the same with those for the regular gas dynamics supersonic flow over a wedge presented in the previous section.

During this benchmark it has been observed that although the solution reached steady state the residual dropped only three orders of magnitude. For the purely gas dynamics benchmark it has been found that the residual dropped twelve orders

of magnitude and a similar behavior was expected from the MHD wedge flow. The reason for the relatively high value of the residual in the MHD solution is that the divergence of the magnetic field ($\nabla \cdot \mathbf{B}$) has values of order unity in the two layers of cells adjacent to the wall and the residual is proportional to $\nabla \cdot \mathbf{B}$. Ideally $\nabla \cdot \mathbf{B} = 0$, and away from the wall the values of $\nabla \cdot \mathbf{B}$ are less than 1×10^{-10}. The explanation for the large values of $\nabla \cdot \mathbf{B}$ in the cell layers next to the wall is that the normal component of the magnetic field at the wall (B_y) is canceled by the perfectly conducting boundary condition. The problem is alleviated if the grid resolution is increased. For example, if the grid resolution is doubled in each direction the maximum magnitude of $\nabla \cdot \mathbf{B}$ is of order 10^{-1}. As shown by the analytic results the solution in the bulk of the domain is not influenced by this non-physical value of $\nabla \cdot \mathbf{B}$ and the two rows of cells next to the wall have not been included in the evaluation of the residual for the convergence studies. A comparison between the residual obtained with the two cells next to the wall and without them is presented in Figure 5.16.

Convergence to steady state has been investigated for this benchmark for both implicit and explicit schemes. For these studies the first two layers of cells next to the wall have been removed from the residual evaluation. For the explicit scheme a range of Courant numbers has been tested, as shown in Figure 5.17. It can be observed that for a Courant number $\nu = 0.8$ the residual drops only one order of magnitude. As the Courant number is reduced the residual drops more but only at small values, e.g. $\nu = 0.1$ and $\nu = 0.05$, a significant improvement is obtained. For Courant numbers close to unity the solution has reached a "pseudo" steady state. The values of the conserved variables downstream of the shock are constant and correlate very well with the solution of the Rankine-Hugoniot jump relations, however, the shock is seen to oscillate about the position predicted analytically. This behavior is peculiar to explicit schemes in which the flux limiters are too compressional, which is equivalent to saying that insufficient numerical damping has been added to the algorithm. The fact that the residual drops for small values of the Courant number is consistent with

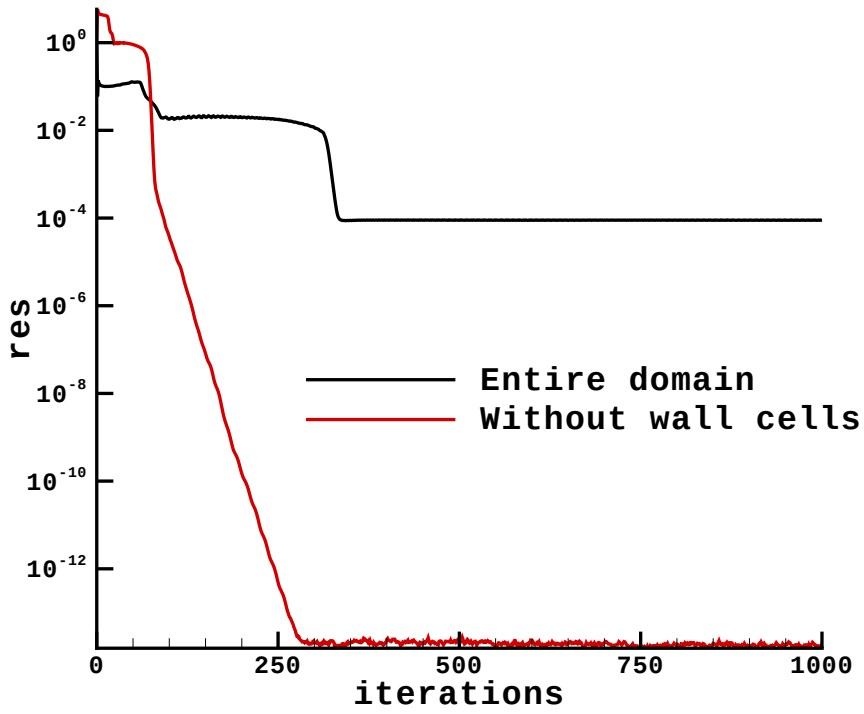

Figure 5.16: Convergence history for the MHD wedge flow problem. The residual evaluated over the entire domain drops only three orders of magnitude due to high values of $\nabla \cdot \mathbf{B}$ next to the perfectly conducting wall. If the first two layers of cells next to the wall are removed from the evaluation the residual drops twelve orders of magnitude.

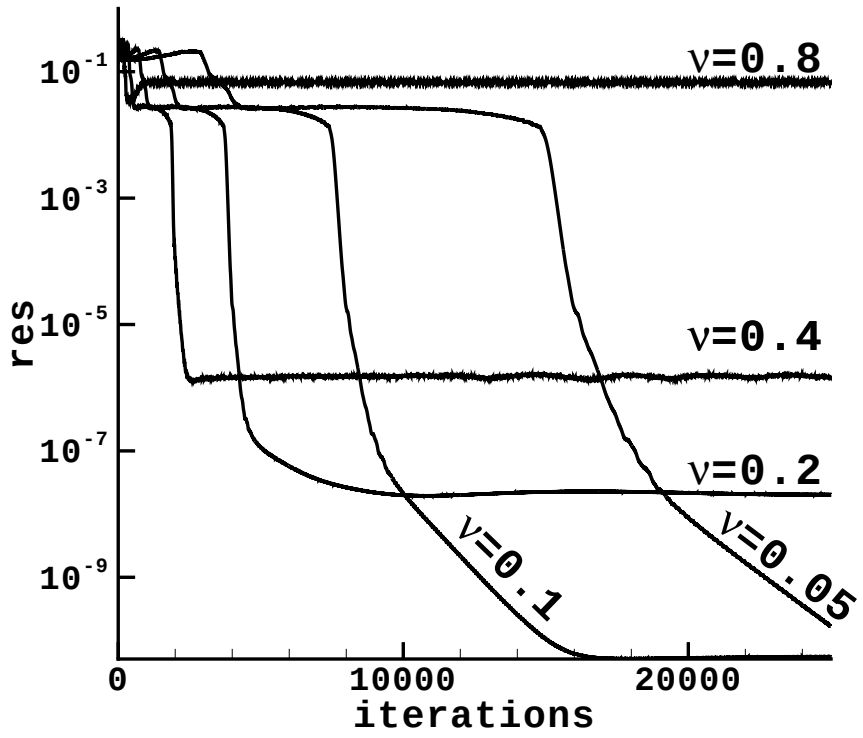

Figure 5.17: Convergence histories of the explicit scheme for supersonic MHD flow over a wedge. Only for small Courant numbers the algorithm has enough numerical damping that allows a reduction of the residual of six orders of magnitude.

the general observation from CFD practice that lower Courant numbers provide more damping to the algorithm. The implicit convergence studies have been performed with a constant physical Courant number $\nu = 100$ and a constant pseudo time Courant number $\nu^* = 0.5$. To compare the convergence rates to those of Whitfield [82] the tests have been performed with a varying number of pseudo time steps (m) and a varying number of the passes of the symmetric Gauss-Seidel procedure (sgs). Remember that the symmetric Gauss-Seidel iterative method is employed to solve the large sparse linear system of equations that results from the discretization of the implicit scheme

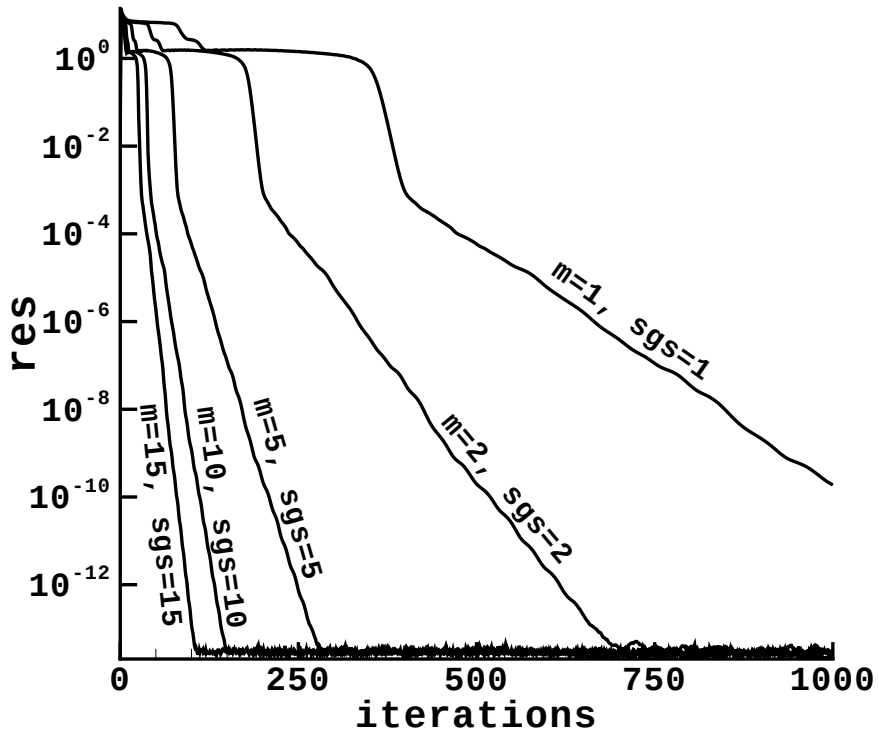

Figure 5.18: Convergence histories of the implicit scheme for supersonic MHD flow over a wedge. Physical Courant number was $\nu = 100$ and pseudo time Courant number was $\nu^* = 0.5$.

(see Section 3.3.) For all the applications presented in this work only five or six passes of the Gauss-Seidel procedure were required to reduce the error in the solution of the linear system to less than 1×10^{-10} and it is expected that this is a general trait. However, for the sake of comparison the number of symmetric Gauss-Seidel passes varied from 1 to 15. As it can be observed in Figure 5.18 the residual drops twelve orders of magnitude in all cases. The convergence histories for $m = 15$, $sgs = 15$ and $m = 10$, $sgs = 10$ do not show a large improvement compared to the $m = 5$, $sgs = 5$ history. It is important to note that the flux Jacobians are not recalculated during

the sweeps of the SGS scheme nor does data get transferred between blocks. As a consequence the ovearall time required for an iteration with $m = 5$ and $sgs = 5$ is less than 25 iterations with $m = 1$, $sgs = 1$, for example. The CPU time necessary for the implicit scheme to converge the solution to machine precision (1×10^{-14}) for the $m = 5$, $sgs = 5$ case was 2761 seconds and the CPU time necessary by the explicit scheme with $\nu = 0.1$ to reduce the residual by eight orders of magnitude was 2874 seconds. The advantages of the implicit scheme for steady state simulations are obvious. Besides the CPU time savings realized with the implicit scheme, another advantage is that it provides the correct amount of numerical damping so that steady state solutions are converged to machine zero. Comparatively, the explicit scheme that uses low Courant numbers to obtain comparable reductions of the residual adds numerical damping indiscriminately, everywhere in the computational domain.

A similar implicit versus explicit behavior has been observed by Batten *et* al. [6] who investigated the convergence of the solution of supersonic flow over a bump. Batten *et* al. compared the convergence of an explicit Runge-Kutta scheme with their own implicit scheme and found that the explicit scheme (with a Courant number $\nu = 0.6$) manages to reduce the residual by one order of magnitude only compared to the implicit scheme that reduces the residual by at least nine orders of magnitude.

5.6 Axisymmetric Simulation - Magnetoplasmadynamic Thruster

The purpose of this benchmark was to solve the MHD equations on an axisymmetric geometry and to compare the solution obtained with experimental data and other numerical investigations of continuous flow coaxial magnetoplasmadynamic (MPD) thrusters. Development of coaxial plasma accelerators has started in the 1950's and they were used to generate high energy plasmas for fusion experiments. The first such device was built by Marshall [38] and operated in a pulsed mode (hence the term Marshall guns for pulsed MPD accelerators.)

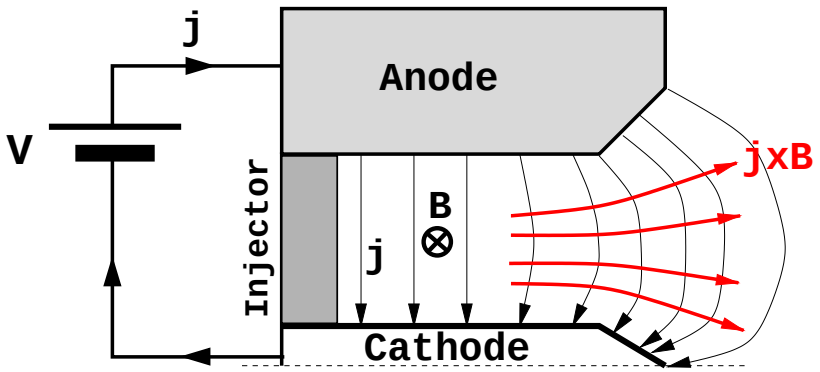

Figure 5.19: Coaxial magnetoplasmadynamic thruster. The voltage applied across the electrodes ionizes the gas fed through the insulating injector region. The current through the plasma ($\mathbf{j}$) produces an azimuthal magnetic field sometimes called a self field. The Lorenz force ($\mathbf{j} \times \mathbf{B}$) accelerates the plasma thus producing the momentum ($\dot{m}v$) component of the thrust. Heating of the plasma due to resistive effects raises the plasma thermal energy contributing to the pressure (pA) component of the thrust.

Coaxial MPD thrusters are a class of plasma propulsion devices that use the Lorenz force ($\mathbf{j} \times \mathbf{B}$) to accelerate a neutral plasma at high velocities. The configuration of an MPD device is shown in Figure 5.19. A continuous voltage is applied across the electrodes. The electrodes are mounted coaxially and are separated by an insulating region through which a neutral gas is injected. A breakdown of the neutral gas occurs in the vicinity of the injector and current starts flowing through the electrodes. The current produces an azimuthal field and under the influence of the Lorenz force the quasi-neutral plasma is accelerated through the region between the two electrodes. The momentum gained by the plasma produces part of the thrust ($\dot{m}v$ component) of an MPD engine. Heating of the plasma due to resistive effects raises the thermal energy of the plasma and produces the pressure (pA component) of the thrust. The power at which plasma thrusters typically operate is between 10 kW and 100 kW, the currents are in the kiloAmpère region and the thrusts are in the range of Newtons to tens of Newtons. Exhaust velocities between 10 and 45 km/s have been reported in

literature [20]. The advantages of MPD thrusters is that they operate at a relatively high pressure level (10^{-2} to 100 $mmHg \approx 1.3 \times 10^{-5}$ to 0.13 bar) so that they attain the highest thrust density (in the region of 0.1 N/cm^2) and the lowest power density (2 to 0.2 kg/kW) of all electric thrusters. The disadvantages are overall efficiencies of less than 50% and low operational life due to cathode erosion from sputtering.

The benchmark was aimed at performing a qualitative comparison between the solutions generated with WARP3 and experimental and numerical results presented by Sleziona *et* al. [64]. A simple geometry has been used, similar to that of the ZT2-IRS thruster shown in Sleziona's paper. The thruster is a typical coaxial MPD thruster with a length of 120 mm, a cathode (outer) diameter of 16 mm and an anode (internal) diameter of 34 mm. The only difference between the geometry of the thruster simulated with WARP3 (from now on called WT) and ZT2 is that the cathode and anode of WT have sharp edges (viz. Figure 5.20) at the exit region instead of chamfered ones such as those of ZT2 to simplify the grid generation. The rounded edges on the real thruster provide a zone where the work fluid (plasma) can be accelerated further, similar to the divergent section of a deLaval nozzle, and they reduce the probability of arcs occurring in that region.

The simulation was setup such that it closely follows the description of the numerical procedure of Sleziona and co-workers [64]. The work fluid is Argon (Ar) injected at 2 g/s such that a cold gas thrust of 1 N is obtained with the thruster as observed experimentally. In practice a cold gas test is performed without voltage applied to the electrodes and the thrust is obtained only by gas dynamics means. The thrust in this case was computed using an integral formula that takes into account only the pressure and momentum contributions

$$T = \int_{\Sigma_s} (\rho \mathbf{v}\mathbf{v} + \overleftrightarrow{P}) \cdot d\sigma, \tag{5.36}$$

where Σ_s is the surface of all internal walls. A few iterations have been performed manually and it has been found that the values of the gas inflow that produce $1N$ of

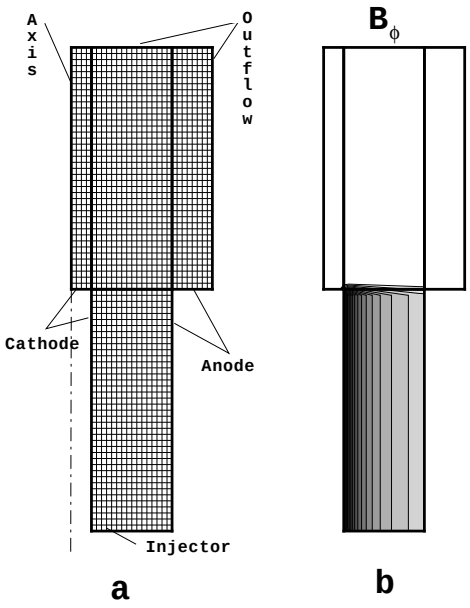

Figure 5.20: Grid used in the axisymmetric MPD simulation (a) and the initial magnetic azimuthal magnetic field (b)

thrust are $\rho_{in} = 10\ g/m^3$, $v_{in} = 442.1\ m/s$ and $p_{in} = 1.035 \times 10^{-2}\ bar = 7.762\ mmHg$. The area of the inlet was inferred from the drawings in Sleziona's paper to be $A \approx 452.4\ mm^2$. Note that the injection velocity is supersonic since the speed of sound in Ar at the above conditions is $c = 406\ m/s$. A current of 6 kA (equal to that of the ZT2 experiment) produces a maximum value of the magnetic field equal to 0.15 T at the cathode. Viscosity and resistivity of the plasma were calculated using Eqns 2.54 and 2.55. The grid consisted of 10 blocks distributed as shown in Figure 5.20. Blocks 1 to 4 were used to discretize the region between electrodes and blocks 5 to 10 were used to discretize what in the experimental setup would be a dump tank. Since WARP3 is a fluid code a true vacuum is impossible to simulate so that in the external region the pressure was initialized to be equal to the pressure inside the

thruster. The magnetic field has been initialized only between the electrodes, it has only an azimuthal component (B_Φ) that varies inversely proportional to the radius, and it is constant from cross-section to cross-section. The boundary conditions were specified to be consistent with the MPD setup so that the values of the density, velocity and pressure of the gas at the injector were those described above. Also at the injector region an inflow of magnetic field was specified (azimuthal component only) consistent with the initial conditions and the simulation of current flow between the electrodes. The cathode and the anode are rigid walls and the value of the magnetic field at the walls is also given by the $1/r$ variation and it is held constant (soaked magnetic BCs.) The exterior region had outflow boundary conditions everywhere except at the axis where axisymmetric boundary conditions were specified.

For this simulation both limit and complex number formulations have been tested and found to work equally well and the implicit run parameters were a physical Courant number $\nu = 100$, a pseudo time Courant number $\nu^* = 0.5$ and the number of pseudo times steps were $m = 10$. The divergence of the magnetic field was observed to be less than 1×10^{-11} at all points in the domain. Results of this simulation in the form of the velocity vectors and azimuthal magnetic field are presented in Figure 5.21. The radial variation of the velocity vector is consistent with the fact that the electromagnetic force has a $1/r^2$ variation and is similar to that seen by Sleziona and co-workers [64] in their simulations. The effect of the viscosity is not present at the walls since the grid used was too coarse to capture any boundary layer effects. However the viscous effects are noticeable in the external region were it can be seen that the exhaust gas drives a vortex and a recirculation region at the axis (downstream of the cathode) and a larger vortex at the anode. These effects have not been observed by Sleziona and his co-workers, however it has to be noted that their codes did not include any viscous effects. Since the plasma has a finite resistivity the magnetic field is no longer frozen-in it and there is an amount of slippage between the magnetic field and the plasma, however some magnetic field is still "blown-out"

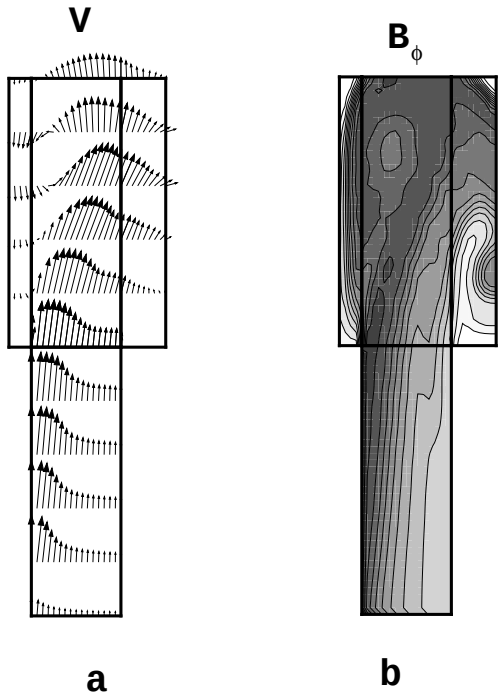

Figure 5.21: Velocity field (a) and contour plots of the azimuthal magnetic field (b)

of the thruster as seen in Figure 5.21 (b).

The thrust of the MPD device has been calculated with

$$T = \int_{\Sigma_s} (\rho \mathbf{v}\mathbf{v} + \overleftrightarrow{P}) \cdot d\sigma - \int_V \mathbf{j} \times \mathbf{B} d\tau, \tag{5.37}$$

where V is the volume of the thruster. Eqn (5.37) of the thrust of the MPD contains a gas dynamics part, first term of the right hand side, which is equivalent to Eqn (5.36), and an electromagnetic part, the second term of the right hand side. Eqn (5.37) emphasizes the fact the thrust produced by gas dynamics effects only, such as that produced by a chemical rocket, is transferred to the engine by surface forces while the electromagnetic effects produce thrust by accelerating the work-fluid in bulk.

The WARP3 simulation never reached steady state due to the periodic shedding

of the vortices and the thrust was found to vary with 6% about a nominal value of $T = 11.83N$. An average exhaust velocity $\bar{v}_e$ of the work fluid can be obtained using the formula for thrust as a function of the mass flow rate $T = \dot{m}\bar{v}_e$. With a mass flow rate of $\dot{m} = 2\ g/s$ the average exhaust velocity is $\bar{v}_e = 5.915\ km/s$ which is consistent to that computed by WARP3. The exhaust velocity falls short of the range of exhaust velocities (from 10 to 50 km/s) found in the literature (e.g. Löb [20]) probably due to the missing divergent region of the thruster, which would allow gains in the $\rho\mathbf{vv}$ term. It has to be mentioned here that Sleziona and co-workers did not publish a thrust level for the ZT2 thruster but the average exhaust velocity can be evaluated from their velocity field plots. The exhaust velocity of the ZT2 thruster simulation of 6.2 km/s is of the same order of magnitude and slightly higher than that obtained with WARP3.

5.7 Three-Dimensional Gas-Dynamics Benchmark - Supersonic Gas-Dynamics Interference Flow Along Intersecting Wedges

The aim of the benchmark presented in this section was to test the Riemann solver in three-dimensions. A three-dimensional gas dynamics simulation that can be qualitatively compared to experiments and computational results is setup to reproduce supersonic flow over intersecting wedges as shown schematically in Figure 5.22. A supersonic flow encounters a structure made of two wedges assembled at a right angle. The base wedge has an angle δ_b and the wedge perpendicular to it, called the interference wedge, has an angle δ_i. At the leading edge of each wedge a regular, two-dimensional shock wave is generated. The two-dimensional shocks intersect each other and the region between their intersection and the wedges contains an intricate wave structure consisting of an outer compression, an inner shock and an entropy wave. Configurations such as the intersecting wedges frequently occur in practice, for example at wing roots and at the engine inlets of supersonic aircraft. Exper-

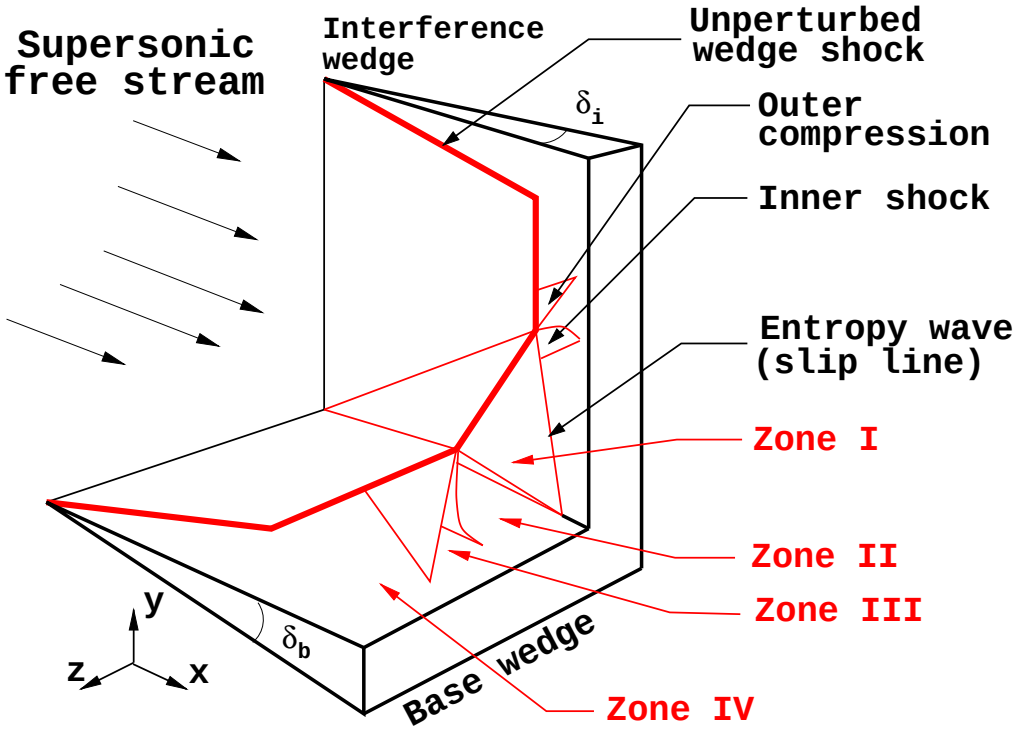

Figure 5.22: Illustration of a supersonic flow over a double wedge and schematic of the characteristic wave structure. The wave structure is symmetric and only one set of waves are labeled for simplicity.

imental investigations of the wave structure were performed in the mid 1960's by Charwat and Redekeopp [17] and the flow is a good benchmark for compressible CFD codes as shown by Wang and Taylor [80] and by Fyfe *et* al. [26]. The complexity of the flow structure makes a rigorous description difficult, however, four principal zones can be distinguished, all supersonic, according to Charwat and Redekeopp [17] and confirmed by computations. The flow in the central zone (Zone I) is similar to conical flow. The adjustment between the conical flow and the purely two-dimensional flow at the far field takes place in two distinct zones. The inner zone, Zone II, is separated from the central flow by slip lines (entropy waves). (An entropy wave is characterized by a jump in density only.) The outer boundary of Zone II is a relatively

strong, curved inner shock. Zone III, the outer interaction region, is characterized by a compressive fan that appears to be centered at the principal wave-intersection line. Finally, Zone IV is the unperturbed region.

For the simulation presented here the wedge angles were $\delta_b = \delta_i = 12.2^o$ and the free stream Mach number was $Ma = 2.5$, similar to an experiment performed by Charwat and Redekeopp. The grid used was a cube of side one, divided into 64 blocks, four blocks along each coordinate axis. The grid spacing was uniform with a resolution of 40 cells, 10 cells in each block for a total of $64,000$ cells. Both limit and complex number formulations of the flux Jacobians have been used in this benchmark and both proved to work equally well. Comparisons between the performance of the multiblock parallel code and that of a single block sequential code required that the sequential code used a grid with the same resolution ($64,000$ cells) but made of a single block. Due to computational time restrictions higher resolution runs have not been performed so that slip lines of Zone I can only be inferred from density and pressure variations. For example, similar simulations (performed by Wang and Johnson [80]) that clearly show the region delimited by the slip lines in Zone I used grids with a total of $840,000$ points. Convergence histories have been studied for this application using a method similar to that used for the study of convergence histories for the two-dimensional supersonic wedge MHD flow. The physical and pseudo time Courant numbers were kept at constant values of $\nu = 100$ and $\nu^* = 0.5$ respectively and the number of pseudo time steps (m) was varied between 1 and 10 and the number of symmetric Gauss-Seidel (SGS) sweeps was also varied between 1 and 10. This problem was run on multiple processors to study convergence performance of the parallel multiblock code. Results are displayed in Figure 5.24. It can be observed that the algorithm reduces the residual by 13 orders of magnitude and the convergence histories are similar to those observed in the two-dimensional wedge flows. The simulation was performed on one, four and eight processors with a grid of constant size. The one processor run used a single block grid of $64,000$ cells and the parallel runs used a

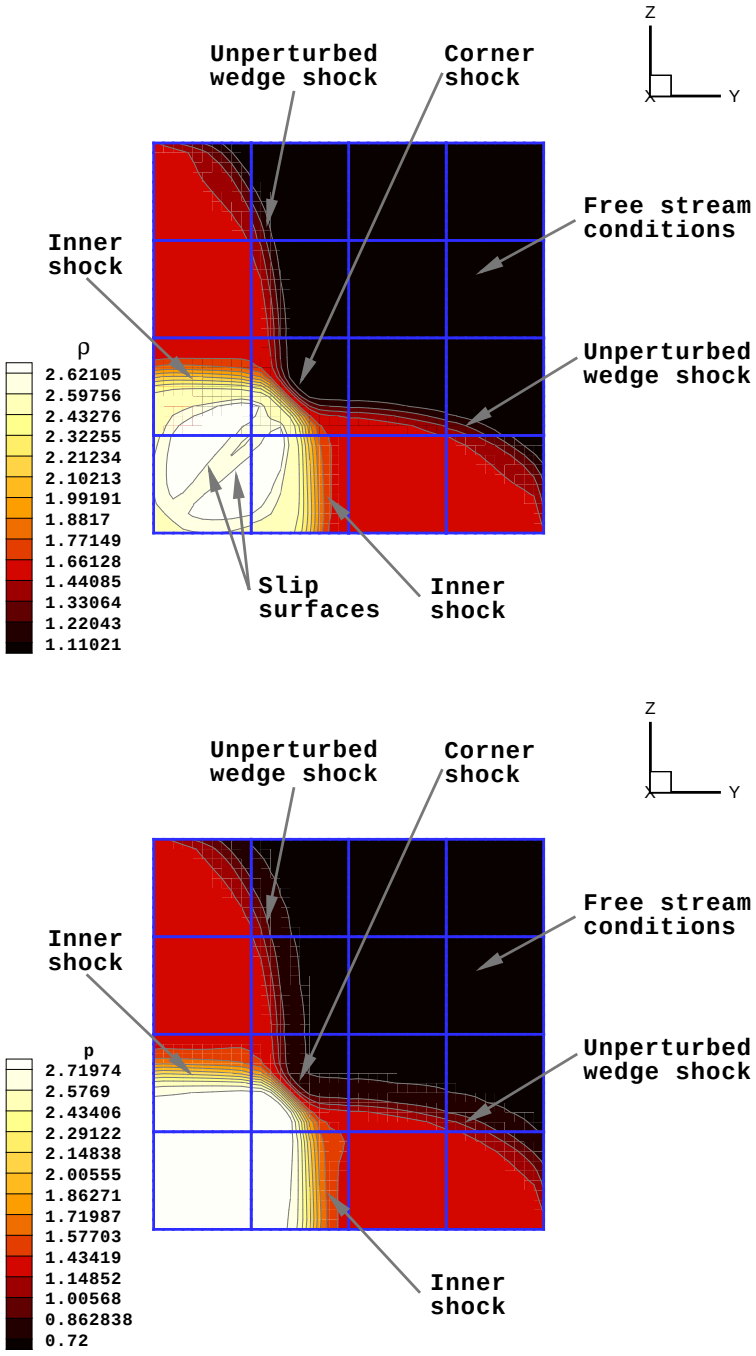

Figure 5.23: Pressure and density plots at station $x = 0.75$. Since the resolution of this run was relatively small (64, 000 cells) the slip lines of Zone I can only be inferred from density and pressure variations.

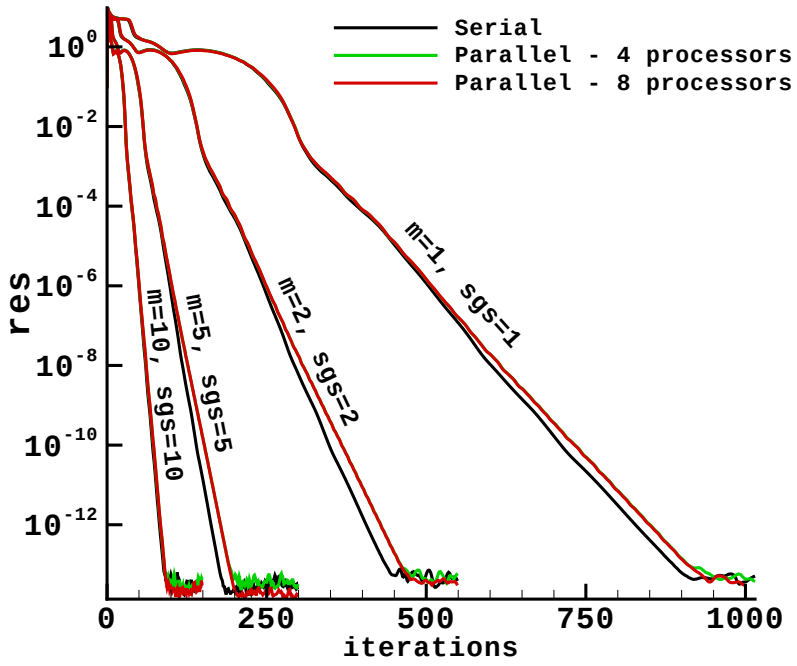

Figure 5.24: Convergence histories of the supersonic flow over intersecting wedges problem. The single processor run was performed on a single block grid of 64,000 cells and the multiprocessor runs were perfromed on a sixtyfour-block grid. Note that the residual drops thirteen orders of magnitude in each case but the convergence is slighlty slower for the multiblock multiprocessor runs.

grid with the same number of cells equally distributed over sixtyfour blocks. It can be observed that the parallel runs converge at slightly lower rates than the sequential run. This effect is not directly attributable to the code running on multiple processors but to the fact that the grid for the multiprocessor run was divided in blocks. The symmetric Gauss-Seidel algorithm that finds the solution of the linear system resulting from the discretization of the the implicit scheme is no longer a truly Gauss-Seidel algorithm but what is sometimes called a block Gauss-Seidel algorithm that finds the solution of the linear system inside each block rather than in the entire domain. Thus the convergence rate is not dependent on the number of processor as it can be observed in Figure 5.24 but it is probably a function of the number of blocks in the grid. It is expected that the larger the number of the blocks the lower the convergence rate of the implicit algorithm will be. However, it can be observed that for sixtyfour blocks the multiblock runs converge only a few tens of iterations later than the single block runs. To alleviate this problem similar parallel implementations such as that for incompressible Navier-Stokes by Pankajakshan and Briley [53] perform the data exchange between blocks after each SGS sweep. (WARP3 exchanges data bewteen blocks only after each pseudo time iteration.)

5.8 Three-Dimensional MHD Benchmark - Spheromak Tilt Stability Study

This section presents a three-dimensional MHD benchmark of WARP3 against linear stability theory predictions for tilt motion of prolate (aspect ratios $L/R > 1$) spheromaks and against experimental observations. Prolate spheromaks in initially minimum energy configurations (constant λ profiles) have been perturbed with a velocity field obtained from an axisymmetric linear stability analysis code. The histories of the kinetic energy for three aspect ratios ($L/R = 2, 3$ and 4) show an initial growth of the kinetic energy predicted by linear theory as shown by Finn *et* al. [22] and Bon-

deson *et* al. [9]. Compact toroid motion inside prolate cylindrical flux conservers has been investigated experimentally by Jarboe *et* al. [41] and Armstrong *et* al. [21] and the compact toroids have been observed to tilt. (Aspect ratio of the flux conserver in Jarboe's experiments was $L/R = 5.22$.) This benchmark showed that WARP3 can accurately perform time-dependent simulations of plasma physics of fusion interest and it also showed how a tilt mode saturates in prolate spheromaks. Experiments showed that oblate spheromaks in flux conservers with $L/R = 1$ are stable [21] and linear theory predicts that the threshold aspect ratio for stability to tilt modes is $L/R = 1.67$ [9, 22]. A spheromak with an aspect ratio unity has been perturbed and shown to be stable. This benchmark showed that WARP3 has the potential to address a few of the theoretical and computational issues Hooper [33] identified as requirements for the advance of the state of the art in spheromak research. This is the first time the stability of a spheromak is investigated in a three dimensional geometry with a non-linear code.

The following sections give a brief description of the spheromaks and why they are interesting from the point of view of controlled fusion reactors, discuss a few of the aspects of spheromak formation and sustainment and present the results of the WARP3 simulations. The descriptions of spheromak theory and practice presented below are not exhaustive and are given to provide familiarity with the concepts of magnetic confinement fusion in general and spheromak devices in particular. The interested reader is encouraged to refer to the review article by Jarboe [40] and the spheromak community white paper compiled and edited by Hooper [33] and the wealth of references listed in them.

5.8.1 Spheromaks as Magnetic Confinement Fusion Devices

In a magnetic confinement fusion reactor the combination of the hydrogen isotopes deuterium and tritium, for example, produces heavier nuclei (helium) and releases energy in the form of an energetic neutron. In order to contain and control the

reaction process the deuterium and tritium isotopes are heated and ionized and kept away from the walls of the reactor my magnetic fields. If the temperature of the D-T plasma is high enough the force of the collisions between the ions will exceed the Coulomb force and bring the nuclei in the region of strong atomic forces leading to their fusion and the corresponding release of energy. Various methods of creating the magnetic fields that confine the plasma and to heat it to the fusion temperatures have been studied over the last half a century. Probably the best known fusion reactor experiments are the tokamaks which have toroidal geometry [81].

Spheromaks have been studied first by astrophysicists in the context of space plasmas where they have been observed to be force-free configurations with closed flux surfaces similar to a Hill's vortex [76]. In laboratory experiments spheromaks are also force-free plasma objects in which confinement is produced by magnetic fields generated internally by plasma currents. For longer duration experiments and for planned fusion reactors external coils are required to produce an equilibrium field. Compared to tokamaks however the external coils of a spheromak device do not penetrate the first wall. Spheromaks were extensively studied in the US, UK and Japan in the 1980's with work continuing in the 1990's in Japan and the UK. An excellent review article on spheromak research that compiles the results of theoretical and experimental work was published by Jarboe [40]. At the end of the 1990's there is a revamped interest in spheromak research as illustrated by the sustained spheromak physics experiment (SSPX) under construction at the Lawrence Livermore National Laboratory [33, 34, 35]. The renewed interest comes from the appeal of spheromaks as fusion devices. Carruthers [16] describes a few of the relevant characteristics a realistic fusion reactor should have. The characteristics are:

1. **Design** - the reactor should be "not too complex" and suitable for industrial manufacture and quality control and should not require excessive high technology effort on the plant site.

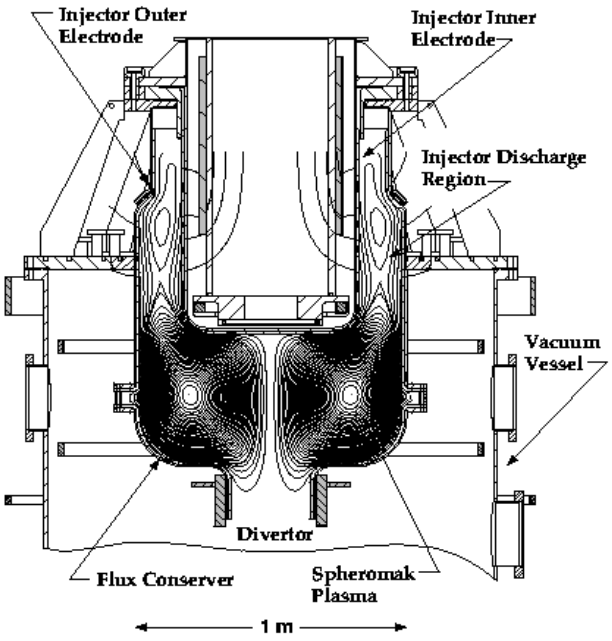

Figure 5.25: Longitudinal section through the proposed sustained spheromak experiment (SSPX) at Lawrence Livermore Laboratory. Reproduced from Hooper *et* al. [35]

2. **Size** - the reactor should produce electric power in the range of 1-2 GW or less.

3. **Economics** - the capital costs should not be too far from those of possible competitors such as the coal fired boilers or turbo-generators powered by natural gas.

Potential spheromak fusion reactors match all the criteria above. The geometry of a spheromak device is particularly simple, the first wall is topologically similar to a sphere and coils do not penetrate it. The external coils that provide an equilibrium magnetic field in a fusion reactor are circular and coaxial with the flux conserver. A longitudinal section through SSPX as presented by Hooper *et* al. [35] is reproduced in Figure 5.25 to illustrate the simplicity of the spheromak geometry. Spheromaks are compact because the plasma object has toroidal and poloidal magnetic fields of comparable magnitudes produced by internal currents. For purposes of comparison to

tokamaks two plasma betas are introduced. Engineering beta is defined as the ratio between the average plasma pressure $< p >$ and the value of the poloidal magnetic field at the plasma edge B_e

$$\beta_e = \frac{\sqrt{< p^2 >}}{\frac{B_e^2}{2\mu_0}}. \tag{5.38}$$

Since the poloidal field strength (B_e) is close to the poloidal magnetic field supplied by the external equilibrium coils β_e is a measure for engineering efficiency. It compares the energy "invested" in the system in the form of magnetic energy ($\mu B_e^2/2$) to keep the plasma stable with the (average) thermal energy of the plasma $< p >$. A second beta is defined using the conventional expression

$$\beta_0 = \frac{p_0}{\frac{B_{t,0}^2}{2\mu_0}}, \tag{5.39}$$

where p_0 is the plasma pressure at the magnetic axis and B_{t_0} is the toroidal magnetic field at the magnetic axis. The advantage of a spheromak over a tokamak stems from the fact that in a spheromak the field strength at the coils ($B_{coil} \approx B_e$) is roughly a third of the toroidal field at the magnetic axis ($B_{t,0}$). In contrast in a tokamak the same ratio is between two to three. In consequence for a given tolerable maximum field at the coils (limited by the strength of the coil material) the achievable plasma pressure can be significantly higher in spheromaks. For example engineering betas in spheromaks are in the range of 15% and in tokamaks they are in the range of $2-3\%$. Magnetic axis betas (β_0) are about 2% in spheromaks and $5-7\%$ in tokamaks. This apparent advantage of tokamaks is offset by the fact that the toroidal field is provided by external coils that "link" the first wall and they are serious challenges to maintenance operations.

Another advantage of spheromaks is that steady state operation can be accomplished by helicity injection. Sustainment of a spheromak by helicity injection from a coaxial plasma gun into a flux conserver was first demonstrated on the compact toroid experiment (CTX) by Jarboe *et* al. [42].

Last but not least, in a spheromak the closed field lines are surrounded by open field lines thus providing a natural diverter. From a reactor design point of view this an advantage over tokamaks since in tokamaks field lines need to be "pulled" out in a diverter. (Diverters are used to collect impurities from the fusion plasma. The impurities consist mainly of heavy ionized atoms that radiate energy away from the plasma at high rates and are detrimental to the plasma heating process.)

5.9 Formation and Sustainment of Spheromaks

This section provides a brief description of a method of forming and sustaining spheromaks that has been succesfully proven by the CTX [42] experiment.

The spheromak formation method presented here is not unique and the interested reader is referenced to Jarboe's review article [40] for discussion of other methods. At $t = 0$ a coil that is coaxial with a plasma gun is energized so that it provides the initial magnetic flux sometimes called bias flux. A neutral gas is injected between the electrodes and a voltage is applied across the electrodes as illustrated in Figure 5.26 (a). The gas is ionized by the applied voltage and heated by the current that starts flowing between the electrodes. The plasma and the field are ejected from the gun and the bias field is stretched into the flux conserver becoming the poloidal flux of the spheromak. The fields generated by the electrode currents become the toroidal fields of the spheromak as shown in Figure 5.26 (b). If the voltage applied across the electrodes is cut-off the fields between the gun and the spheromak reconnect and an isolated spheromak is formed [40]. Experimental observations show that the isolated spheromak relaxes to a force-free minimum energy state (also called a Taylor state [75]) while conserving helicity. However, if the voltage applied across the electrodes is maintained above a certain threshold value [5] helicity is injected into the system at a rate $\dot{K} = 2\dot{\Phi}_1\Phi_2$ where $\dot{\Phi}_1 = \dot{\Phi}_{tor} = V_{gun}$ and $\Phi_2 = \Phi_{pol} = \Phi_{gun}$. Helicity injection can thus be thought of as a mode of energy transfer from the toroidal flux

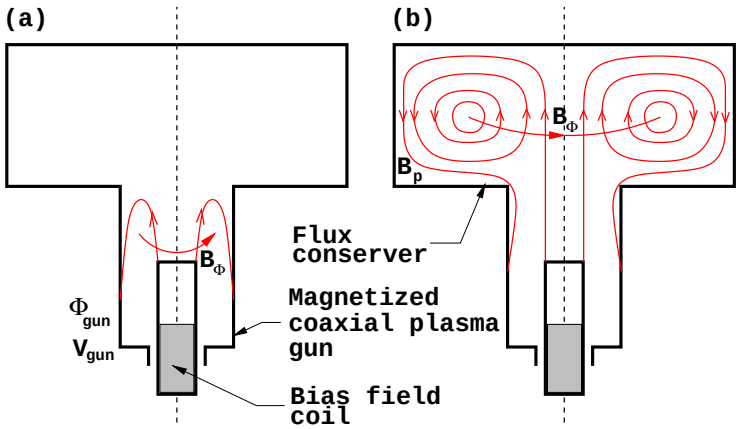

Figure 5.26: Illustration of a spheromak formed by helicity injection. Initially a coil installed coaxial with the gun is energized to provide a bias magnetic field (a). A neutral gas is injected between the electrodes and a voltage is applied across the electrodes. The neutral gas is ionized and heated by the radial current the flows through it. Lorenz forces expel the plasma into the flux conserver and the bias field becomes the poloidal field of the spheromak. The fields generated by the electrode currents become the toroidal fields of the spheromak (b).

to the poloidal flux. Toroidal flux is generated by the currents flowing in poloidal direction at the edge of the spheromak and sustained by the applied voltage across the gun electrodes. Poloidal flux is replenished by reconnections between field lines that snap open and close so that poloidal and toroidal flux surfaces can connect.

As Jarboe mentions in [40] some of the most important scientific results to come out of spheromak research have come from generalizing and verifying the Taylor minimum-energy principle. This principle states that a magnetized plasma configuration relaxes to a state of minimum energy while conserving helicity. Helicity is the linkage of magnetic flux with magnetic flux and its mathematical definition is

$$K = \int_V \mathbf{A} \cdot \mathbf{B} d\tau, \tag{5.40}$$

where $\mathbf{A}$ is the vector potential ($\mathbf{A} = \nabla \times \mathbf{B}$) and $\mathbf{B}$ is the magnetic field. For a simple system of two linked flux tubes of strengths Φ_1 and Φ_2 the helicity is $K = 2\Phi_1\Phi_2$.

(Moffat gives a detailed explanation of the concept of helicity in [49].) In a closed ideal MHD system (with $\mathbf{B} \cdot \mathbf{n} = 0$ at the boundaries) and with the helicity defined as in Eqn (5.40) the minimum-energy state equilibrium is satisfied by the equation

$$\nabla \times \mathbf{B} = \lambda \mathbf{B}, \tag{5.41}$$

where $\lambda = \mu_0 j_{||}/B =$ a global constant.

5.9.1 Spheromak Stability

Stability is an important issue for magnetic confinement fusion devices because, simply put, an unstable plasma object has poor confinement properties, if any. Instabilities can be classified by the sources of energy that drive them. Thus two broad classes of instabilities can be defined. The energy for the first class comes from the interaction of the spheromak with the outside world and the instabilities are called external modes. The second class of instabilities are the current drive modes that obtain their energy from the variation of the current profile. The instabilities in the second class are called internal modes.

WARP3 was benchmarked against linear stability theory results and against experimental observations of compact toroids in prolate flux conservers. The prolate spheromaks are initially in a minimum energy equilibrium state, with λ a global constant in Eqn (5.41). Analysis of spheromaks in oblate flux conservers was then performed. Initially the oblate spheromaks were in a state of equilibrium other than minimum energy, particularly an equilibrium dictated by a linear profile of λ, with a slope parameter $\alpha = -0.6$. It has been found that in the limits of the ideal MHD model the same higher compound mode becomes dominant and saturates indifferent of the initial perturbations that were a mixture of modes that ranged from purely $n = 1$ to purely $n = 2$.

5.9.2 Gross Tilt Modes in Prolate Spheromaks

This section describes the results of WARP3 simulations that were used to perform three-dimensional benchmarks of the code. The external modes investigated here can be categorized as external modes. Simulations were set-up to study the gross tilt modes of initially force-free spheromaks in prolate cylindrical flux conservers. Rosenbluth and Bussac [59] studied analytically spheromak stability in spherical flux conservers using the energy principle. They showed that spheromaks are stable and the minimum energy state is axially symmetric in slightly oblate flux conservers. They also showed that spheromaks enclosed in prolate flux conservers have minimum energy in a non-axially symmetric (tilted) state. Bondeson *et* al. and Finn *et* al. have shown (also theoretically) that for spheromaks contained in closed cylindrical flux conservers the minimum energy state is axisymmetric if the aspect ratio (L/R) of the flux conserver is less than 1.67 and tilted if the aspect ratio is larger than 1.67. Experiments by Armstrong [21] confirmed the theoretical findings.

WARP3 simulations were setup to verify the numerical solutions against the linear theory predictions and experimental results. Spheromaks were initialized as force-free minimum energy configurations given by the solution of Eqn (5.41) in cylindrical coordinates

$$\begin{aligned} B_r &= -k_z J_1(k_r r)\cos(k_z z) \\ B_\phi &= \nu_0 J_1(k_r r)\sin(k_z z) \\ B_z &= k_r J_0(k_r r)\sin(k_z z) \end{aligned} \tag{5.42}$$

where $0 \le z \le L$, $0 \le r \le R$, L is the length of the spheromak, and R is the radius of the spheromak. B_r, B_ϕ and B_z are the components of the magnetic field in polar cylindrical coordinates, J_0 and J_1 are the zeroth and first order Bessel functions, $k_r R = 3.8317$ (first zero of J_1), $k_z L = \pi$ and $\nu_0 = \sqrt{k_r^2 + k_z^2}$. This axisymmetric equilibrium was mapped onto the three-dimensional grid shown in Figure 5.27 by a coordinate transformation. One oblate ($L/R = 1$) and three prolate ($L/R = 2$, 3

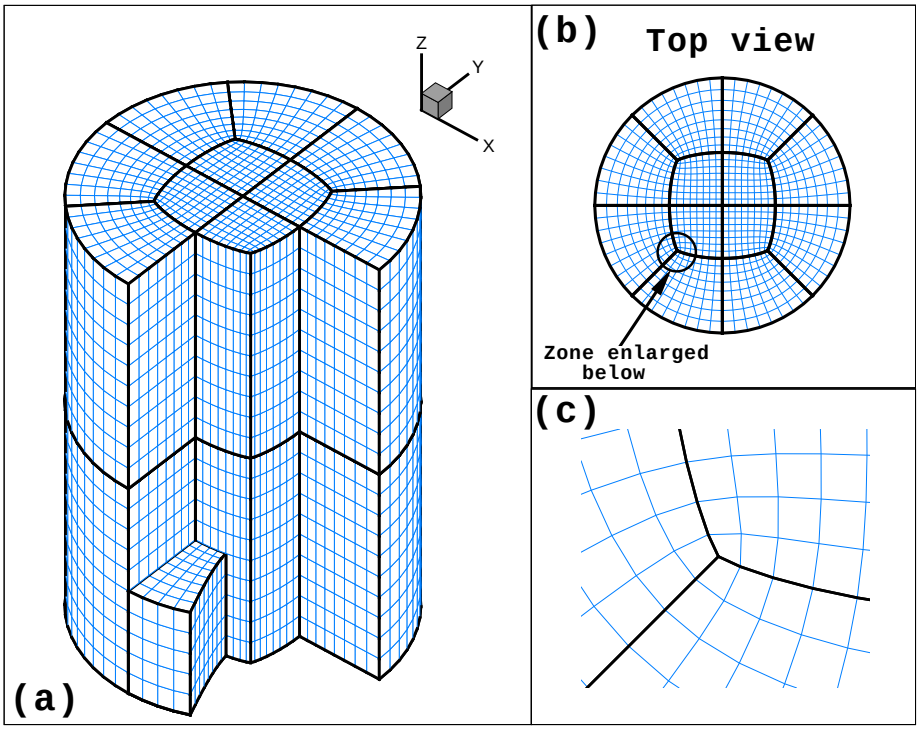

Figure 5.27: Grid used for the spheromak runs. This grid is made of two layers of twelve blocks each, placed such that a cylindrical volume is discretized without singularities such as those present in "pie-slice" grids.

and 4) spheromaks have been studied. The spheromaks were perturbed with modes obtained from linear stability codes to ensure that simulations were initialized with "pure" modes. The perturbations for the prolate spheromaks were obtained from an axisymmetric linear stability code in the form of complex velocity vectors [62]. The linear code was run until the growth rate of the kinetic energy stabilized at the fifth decimal place. The velocity field obtained from the linear stability code was then interpolated and mapped to the three-dimensional grid. The ϕ (toroidal angle dependence was)

$$\mathbf{v} = \mathrm{Re}(\mathcal{V})\cos(n\phi) - \mathrm{Im}(\mathcal{V})\sin(n\phi) \tag{5.43}$$

where $\mathcal{V}$ is the complex velocity vector obtained from the linear code, and n is the toroidal mode number which was unity for these simulations to correspond to a gross tilt motion. The contour plots of the velocity components and the velocity vector fields are shown in Figures 5.28 and 5.29 for two longitudinal sections, at $\phi = 0^o$ and 90^o respectively, of the $L/R = 3$ spheromak. The toroidal magnetic field at $\phi = 0^o$ can be observed in Figure 5.34. For the oblate spheromak the perturbations were obtained in the form of displacements from another linear stability code [63, 45] based on an energy formulation approach. The displacements, also obtained in the form of complex three dimensional vectors, were mapped to the velocity vectors on the three-dimensional grid with a formula similar to Eqn (5.43). The perturbations for the $L/R = 1$ spheromak are shown in Figures 5.30 and 5.31.

It is expected that if the perturbations given to the spheromaks, initially in equilibrium, were small the initial growth of the unstable modes is linear. Perturbations were normalized so that they had a maximum magnitude of the velocity of 1% of the Alfvén speed were applied to the spheromaks. Since WARP3 is a fluid code a finite pressure is specified in the form of $\beta = p/(B^2/2\mu_0)$ which for the simulations presented here was 4%. The kinetic energy of the spheromak was computed at each time step. Results of explicit simulations with a Courant number $\nu = 0.85$ are shown in

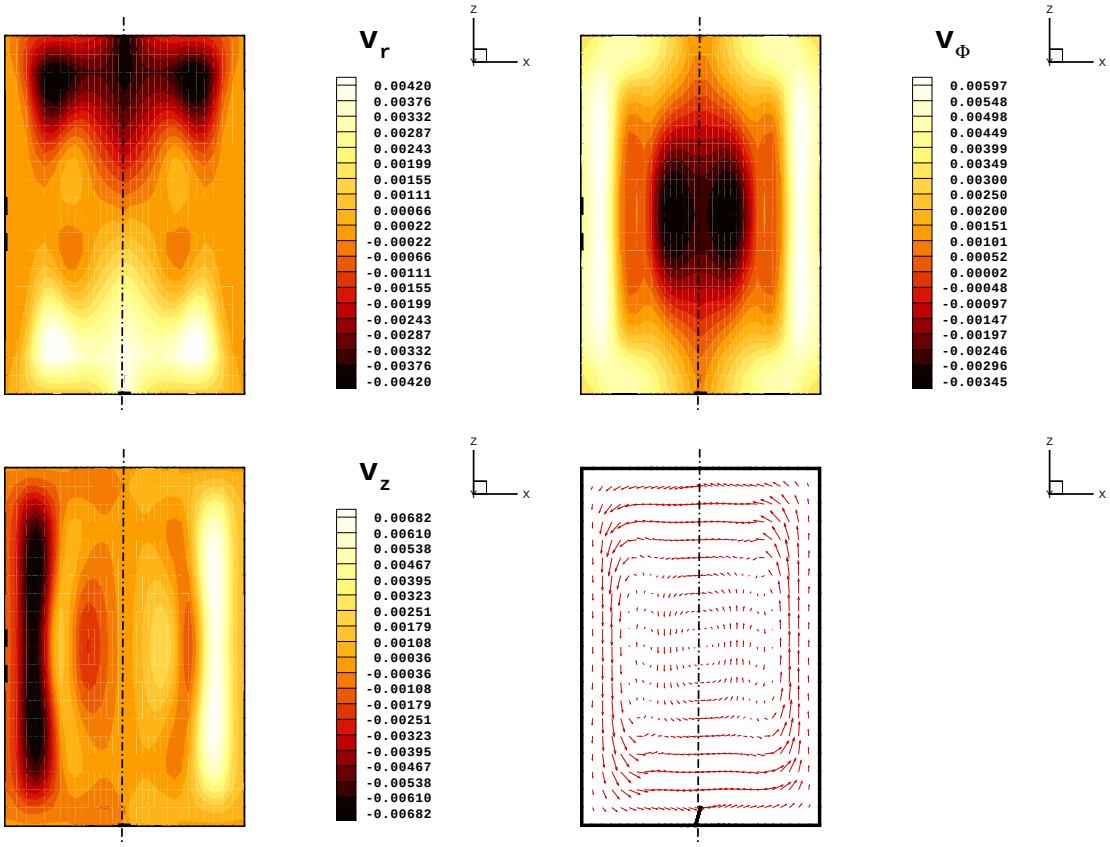

Figure 5.28: Contours of the velocity components and vector field at azimuthal angle $\phi = 0^o$ for a spheromak with $L/R = 3$.

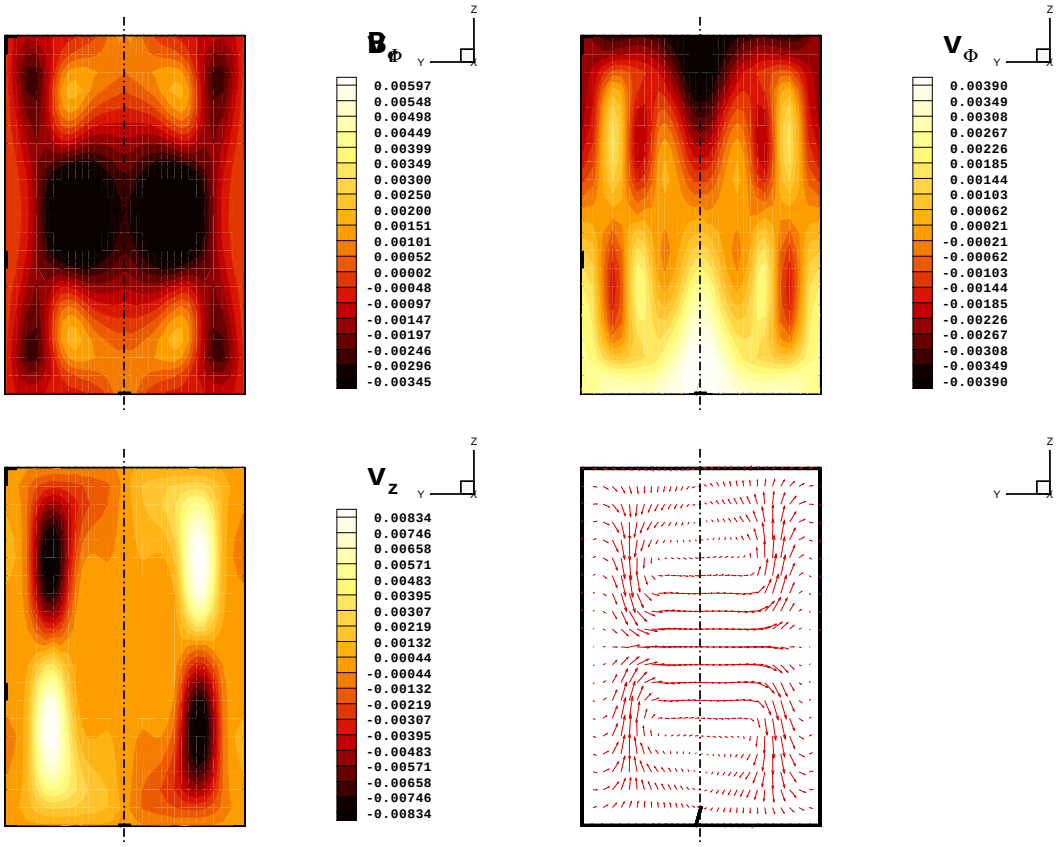

Figure 5.29: Contours of the velocity components and vector field at azimuthal angle $\phi = 90^o$ for a spheromak with $L/R = 3$.

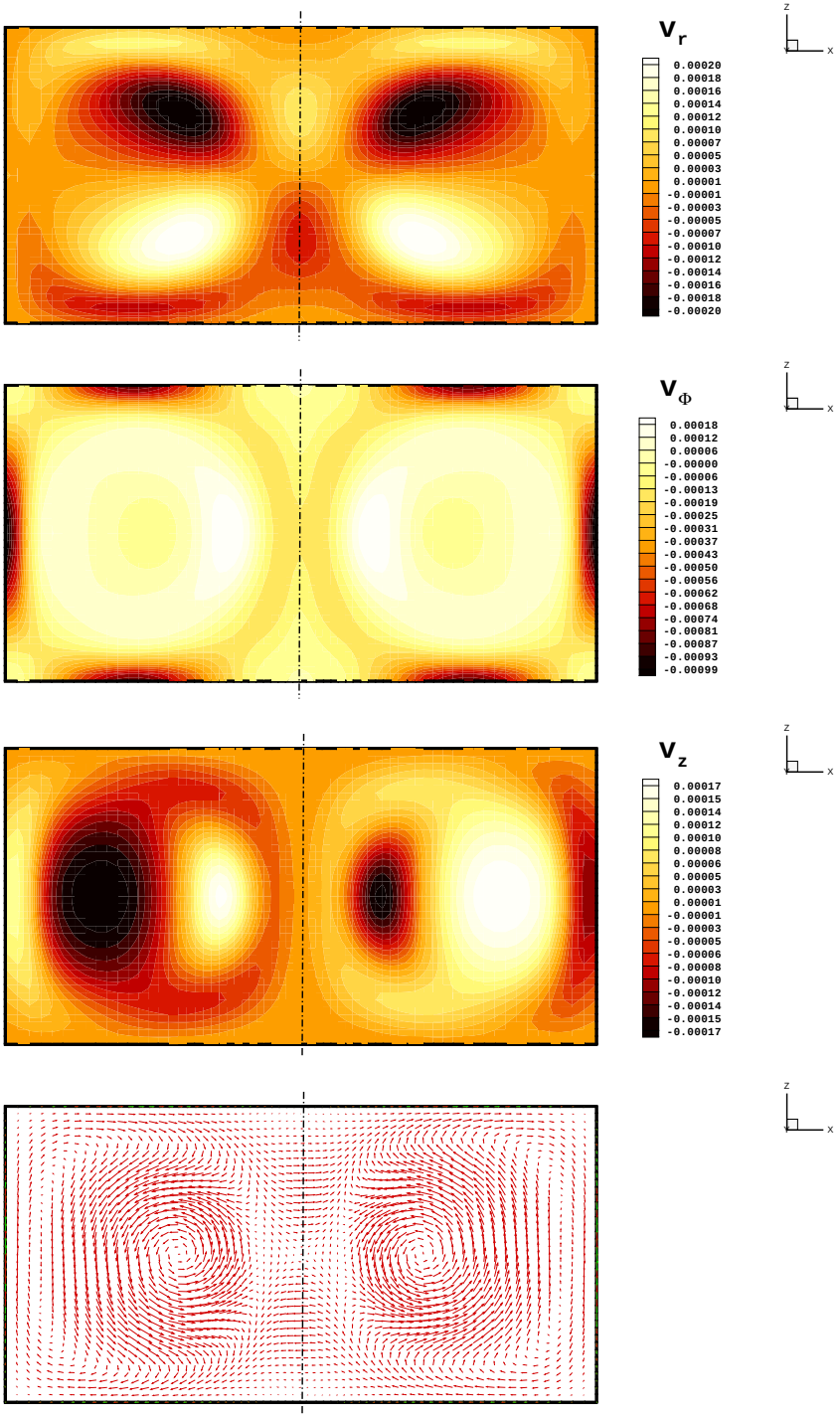

Figure 5.30: Contours of the velocity components and vector field at azimuthal angle $\phi = 0^o$ for a spheromak with $L/R = 1$.

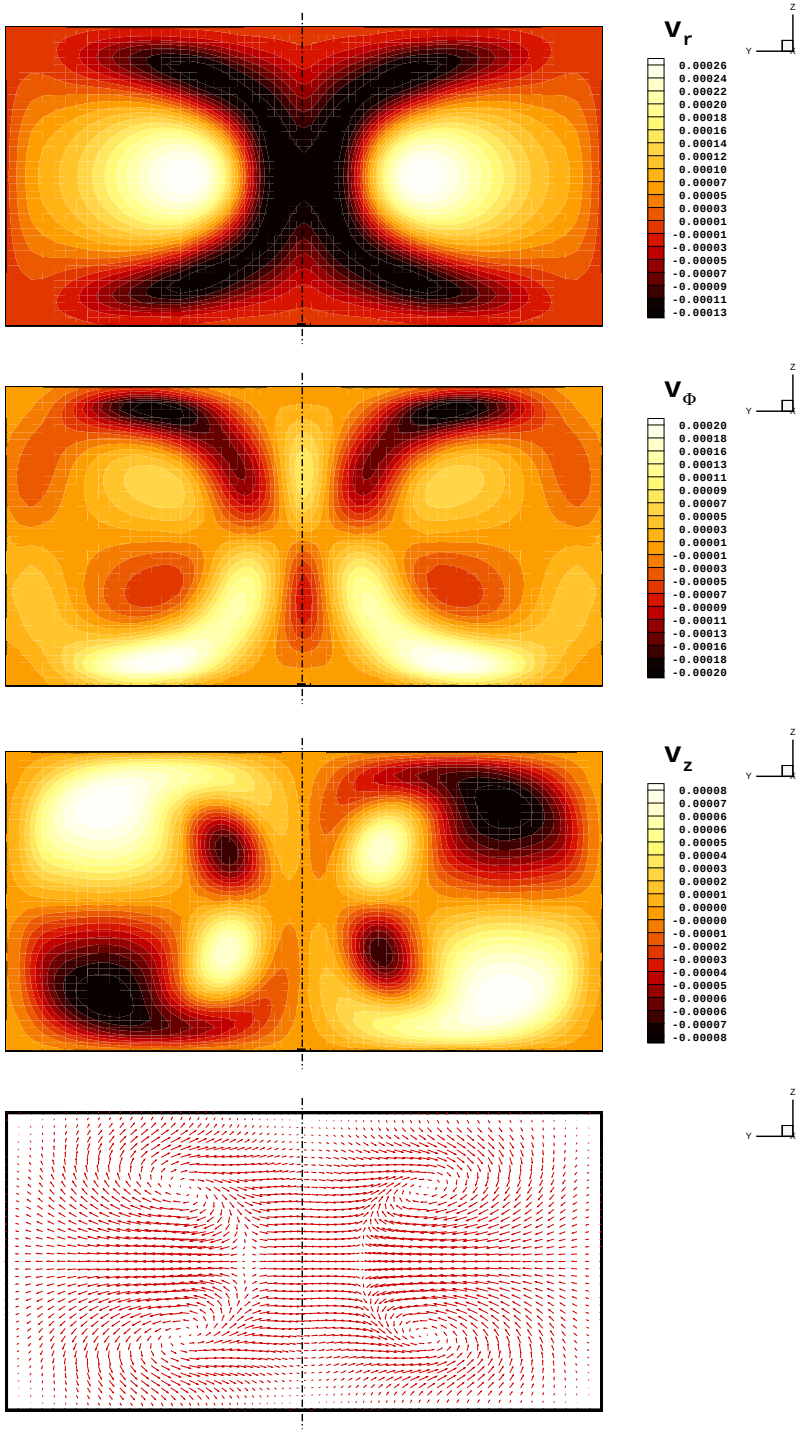

Figure 5.31: Contours of the velocity components and vector field at azimuthal angle $\phi = 90^o$ for a spheromak with $L/R = 1$.

Figure 5.9.2. Once the implicit scheme was benchmarked for the simpler geometries described in the previous sections it has been applied to the prolate spheromaks cases described above. The growth of the kinetic energy for a spheromak with an aspect ratio $L/R = 3$ is presented in Figure 5.33. It can be noticed that consistent with the observations made in the case of supersonic MHD flow the explicit scheme is slightly undamped and as a consequence some oscillations are present in the kinetic energy for the first two Alfvèn transit times. It has been noted qualitatively that there are no differences in the bulk motion of the spheromak between the explicit and implicit schemes. It has been found that for the time dependent simulations the number of pseudo time steps m has to be at least double than the ratio of the physical Courant number to the pseudo time Courant number ($m \approx 2\nu/\nu^*$). For the simulations presented in this section it has been observed that if the implicit scheme is run with physical Courant numbers (ν) in excess of 10 at a fixed pseudo time Courant number ($\nu^* = 0.85$) the minimum number of pseudo time steps is $m = 25$. The implicit scheme run with these parameters is prohibitive from the point of view of CPU time and a lower physical Courant number ($\nu = 5$) and pseudo time steps ($m = 12$) have been used for the simulations presented here. With the lower ν and m the implicit scheme proved marginally less expensive to run than the explicit scheme. For example for a grid of a total of $15,360$ cells run to 10 Alfvèn transit times on a cluster of 12 DEC Alpha processors the total CPU time was 6325 seconds for the implicit scheme with $\nu = 5$, $\nu^* = 0.85$, and $m = 12$ and for the explicit scheme run with $\nu = 0.85$ the total CPU time was 6654 seconds.

Results of the simulations for a prolate spheromak with an aspect ratio $L/R = 3$ are presented in Figures 5.34 to 5.36 as a time sequences of longitudinal sections showing contour plots of the toroidal component of the magnetic field B_ϕ. Along with the plots of the magnetic field the kinetic energy of the spheromak, normalized with the initial kinetic energy is shown. After 15 Alfvén transit times the net motion of the spheromak is evident and the spheromak reaches a maximum kinetic energy at

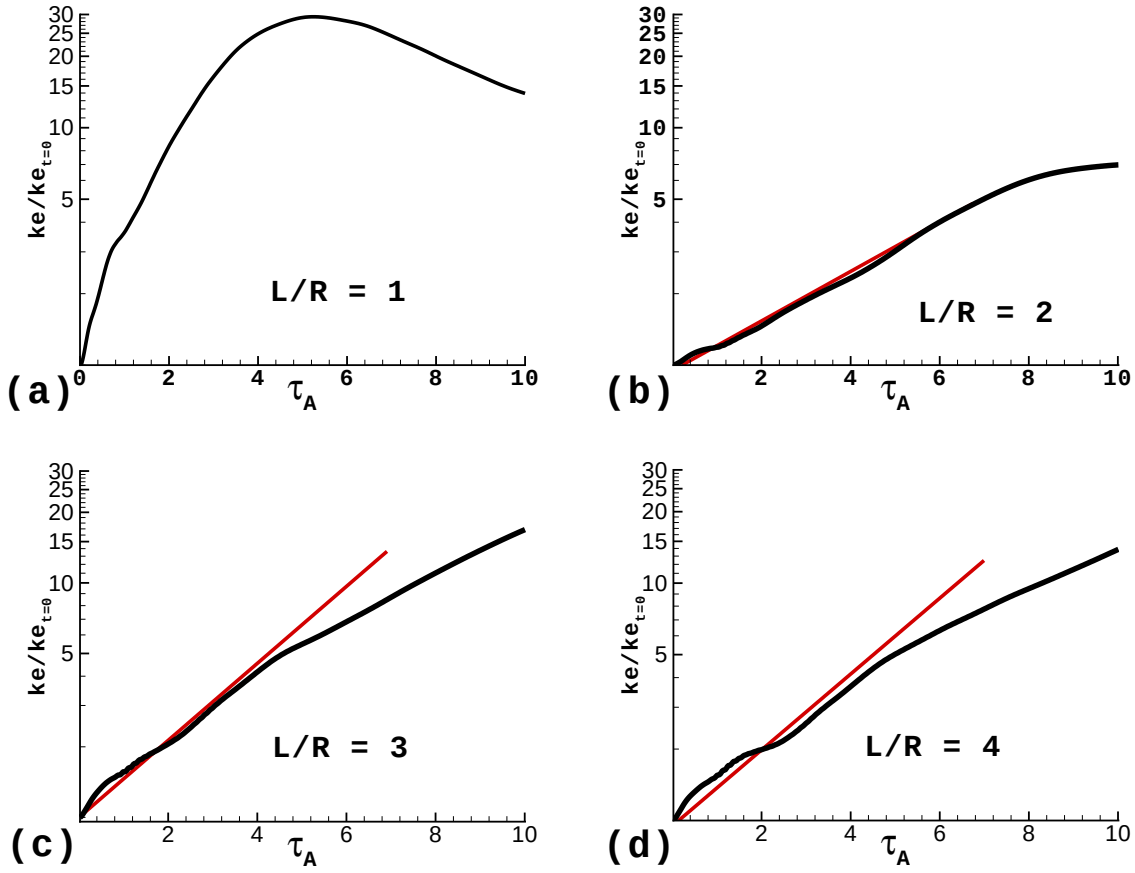

Figure 5.32: Histories of the kinetic energy for one oblate spheromak ($L/R = 1$) and three oblate spheromaks ($L/R = 2$, 3, and 4.) These histories were obtained with the explicit scheme with a Courant number $\nu = 0.85$ and reproduced with the implicit scheme.

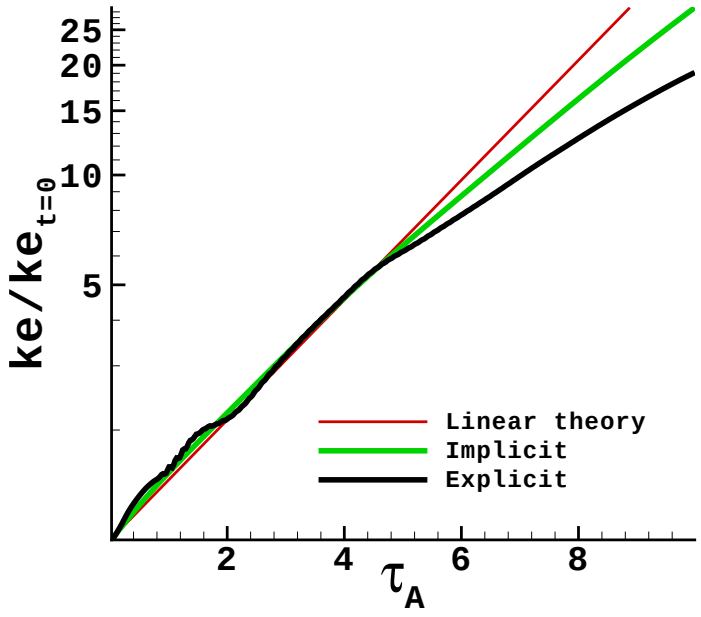

Figure 5.33: Growth rates of kinetic energy for a spheromak with aspect ratio $L/R = 3$ predicted by linear theory and the explicit and implicit schemes.

about 29 Alfvèn times where the plasma object is expanding in the corners of the flux conserver. The kinetic energy then drops and the mode saturates with the spheromak stabilizing into the corners of the flux conserver which would be consistent with the minimum energy observations of the experiments. After 65 Alfvèn transit times the numerical diffusion destroys the spheromak structure and the simulation is stopped.

Since the system of PDEs solved by WARP3 is non-dimensional the results of the simulations are self-similar. To obtain values of the plasma parameters that relate to experiments reference values such as a characteristic length, reference values of the density, pressure and magnetic field have to be chosen. For this simulation the radius of the spheromak is chosen as the characteristic length, $R = 1.0\ m$, and the values of the density of the plasma $n = 1 \times 10^{20}\ m^{-3}$, temperature $T = 400\ eV$, and magnetic field $B = 1\ T$ are those in the proposed SSPX experiment [35]. The resulting plasma pressure (considered uniform for similarity with the simulations) is $p = nkT = 16016\ N/m^2$, where $k = 1.3807 \times 10^{-23}\ J/K$ is the Boltzmann constant, from where $\beta = 4.02\%$ as mentioned above. The Alfvèn speed is $c_A = B/\sqrt{\mu_0 \rho} = 2.18 \times 10^6\ m/s$ and the Alfvèn transit time is $\tau_A = R/c_A = 0.45\ \mu s$. With these parameters it can be determined that the WARP3 simulation shows that the spheromak tilted after about $13.05\ \mu s$ which is consistent with the experimental observations of Jarboe *et* al. [41].

5.10 Parallel Code Performance

This section describes the performance of the parallel code and discusses some of the peculiarities of WARP3. To start on a historical note, the first parallel version of WARP3 was implemented and tested when the code was able to handle only single block two-dimensional Cartesian orthogonal grids but was solving for all eight conserved variables. The implicit scheme used in the old version employed approximated Jacobians that did not work in three dimensions. The domain decomposition was implemented by dividing the single block grid into blocks that were equal in num-

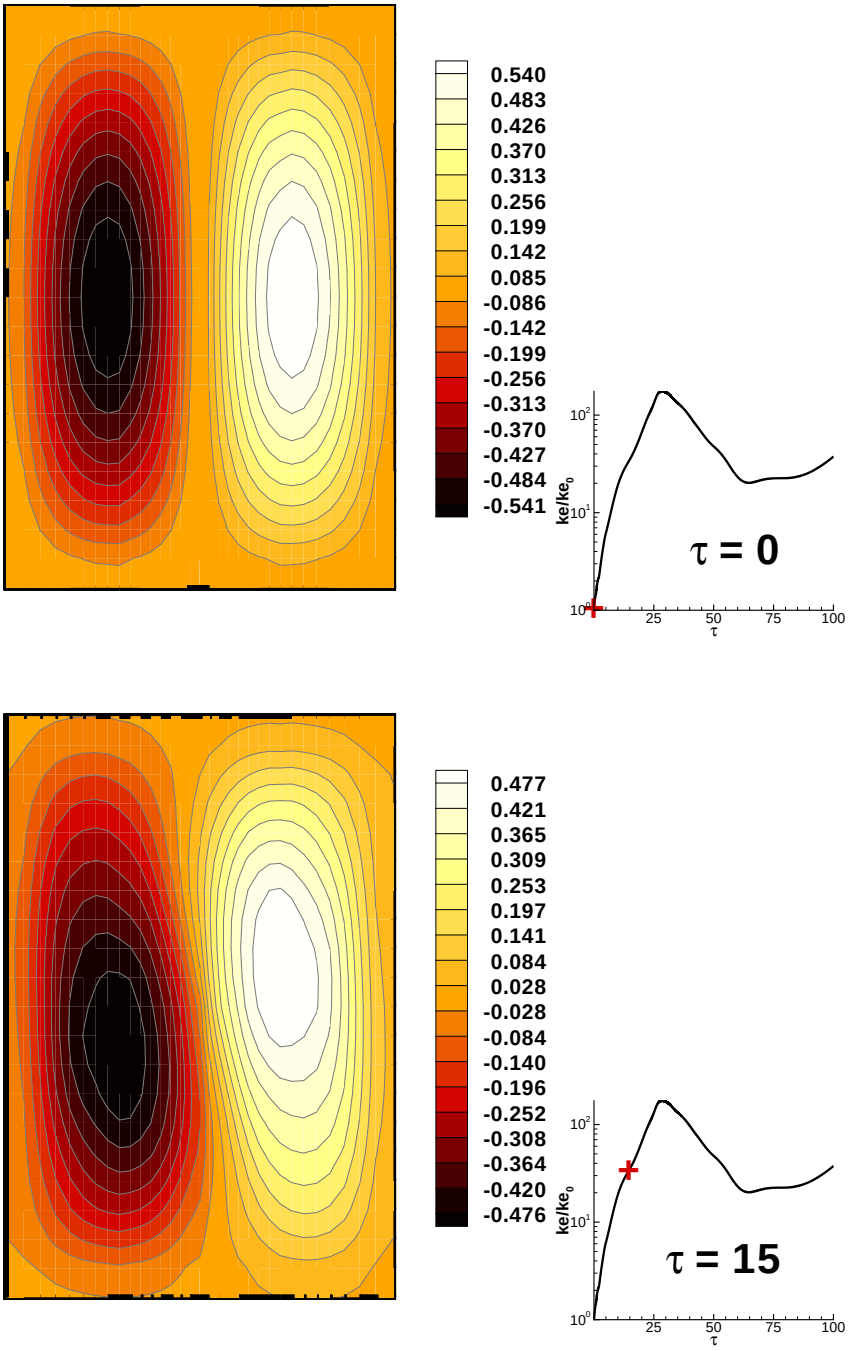

Figure 5.34: Contours of the toroidal magnetic field at $\phi = 0^o$ at $\tau = 0$ and $\tau = 15$. Note that at $\tau = 15$ the bulk motion of the spheromak is obvious. The kinetic energy of the spheromak is increasing.

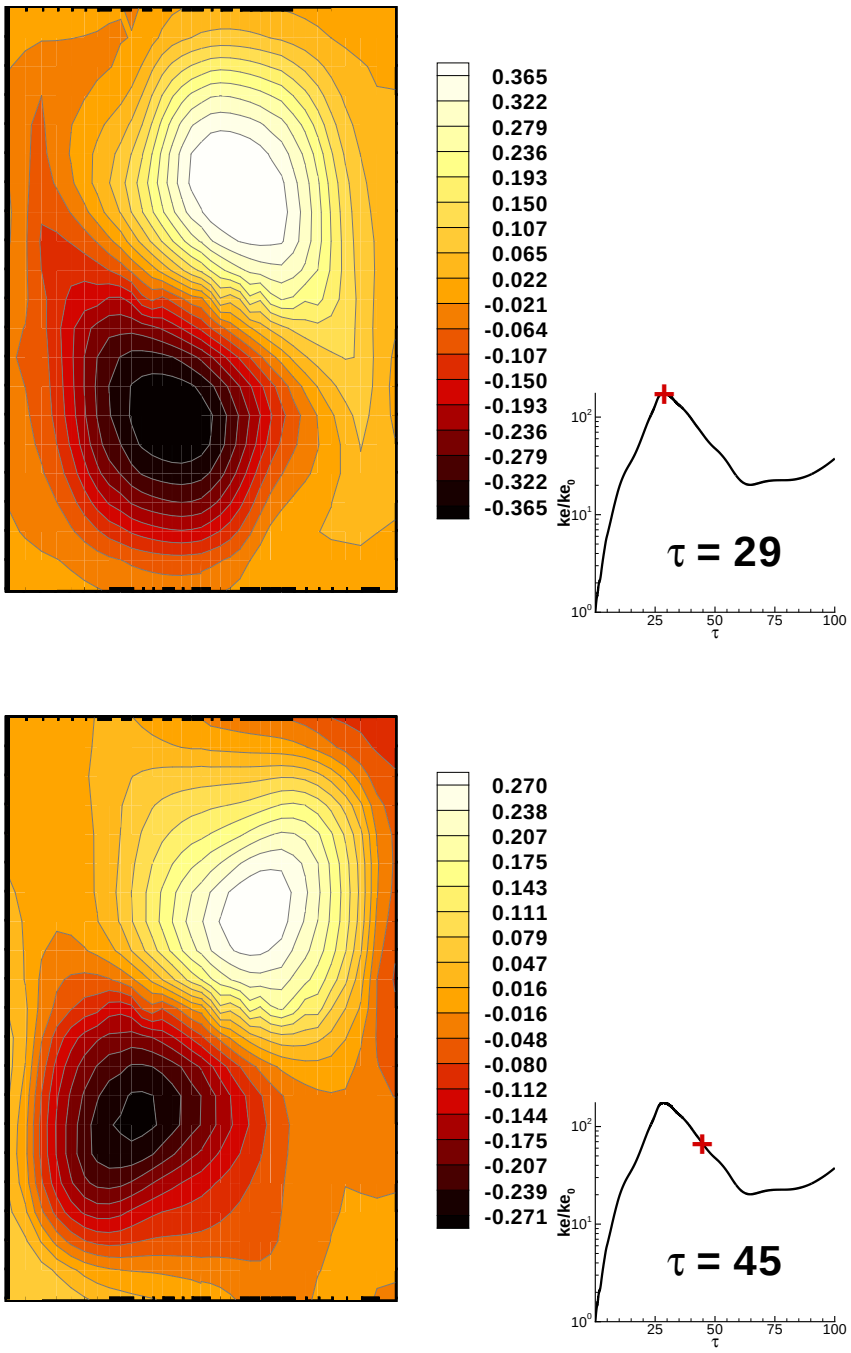

Figure 5.35: Contours of the toroidal magnetic field at $\phi = 0^o$ at $\tau = 29$ and $\tau = 45$. The kinetic energy of the spheromak reaches a maximum at $\tau - 29$ corresponding to the spheromak reaching in the corners of the flux conserver.

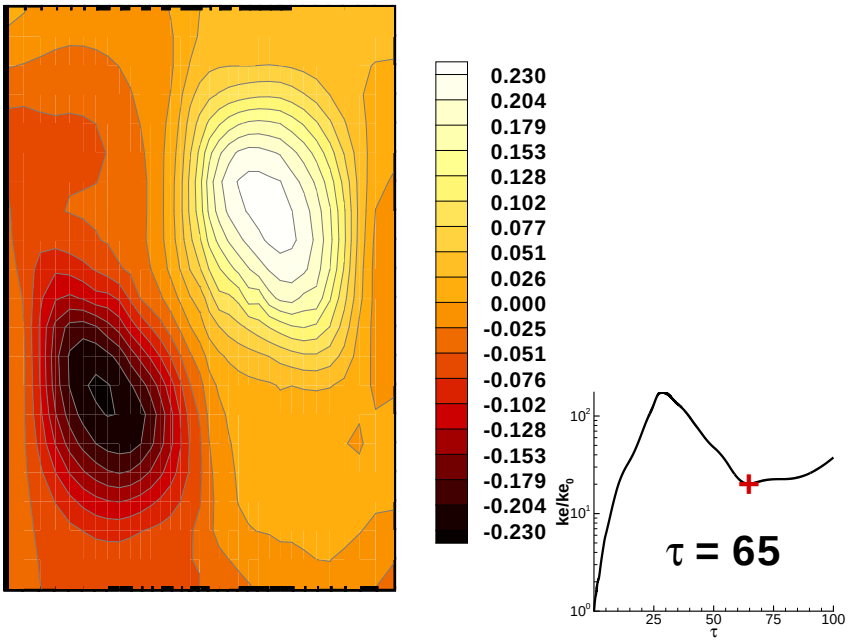

Figure 5.36: Contours of the toroidal magnetic field at $\phi = 0^o$ at $\tau = 65$. The spheromak stabilized in the corners of the flux conserver and decays due to numerical diffusion.

ber with the number of processors assigned to the parallel job. Since the algorithm performs a constant number of operations per cell the load balancing was static and was accomplished by distributing an equal number of cells per block. If the division (between the total number of cells and the number of blocks) was not exact, then some of the blocks were distributed a slightly larger number of cells. Results obtained with this first parallel implementation were excellent and showed that the coarse grain parallelization approach was valid and performed well. The message passing between the blocks used explicit buffering instead of derived data types that are used in the current version of WARP3. Results of the first parallel implementation of WARP3 were presented by Udrea *et* al. in [77].

The problem solved was a channel thruster and the plasma model included viscosity and resistivity. The steady state problem was run on one, four, eight, sixteen and sixtyfour processors and the solution was marched until the residual dropped three orders of magnitude. The runs were performed on the MHPCC SP2. Since the clock speeds of the processors have slight fluctuations and since the inter-processor communication network is a shared resource the problem was run twice, both in the implicit and explicit modes, to observe if many major fluctuations took place. The CPU times for each run were recorded and averaged.

Performance of the parallel code was assessed using a parameter called the parallel code speedup. The speedup is defined as the rate between the time needed by the serial (sequential) code to solve the problem and the time needed by the parallel code to solve the same problem $S_p = T_{seq}/T_p$.

Two sets of tests were performed. The first test was called the fixed grid test to emphasize that a grid with a constant number of cells (400×80) was used to discretize the domain. This grid was divided into as many blocks as processors (from 1 to 64). The second test was performed on scaled grids. In the scaled grids test the processors were assigned grids of equal sizes independent of the number of the processors assigned to the job. For example the sequential run used a grid of 50×10

cells and the four processor run used an overall grid of 200×40 cells divided into four blocks of equal size (50×10).

Speedup results for the fixed grid test are shown in Figure 5.37. As expected the the speedup increases with the number of processors assigned to run the code. It can noted that for the explicit mode the speedup is super-linear which seems to contradict Amdahl's law, which is actually a breakdown of the speedup formula

$$S_p = \frac{T_{seq}}{T_{comm} + \frac{\sum_{i=1}^{p} T_{comp,i}}{p}}, \tag{5.44}$$

where the parallel computer time in the denominator has been parsed into the time spent in inter-processor communications (T_{comm}) and the CPU time is calculated as a sum over all processors of their respective CPU times (T_{comp}) divided by the number of processors. Assuming that the communications time were null and that the sequential time is equal to the sum of the parallel times, then the ideal speedup would be equal to the number of processors. However, Amdahl's law does not in consideration the architecture of the CPU system used, in particular the cache effects. On the IBM RS/6000 processors that constitute the nodes of the SP2, the data is passed from the main memory to the CPU through a data cache (as is the case with all modern CPUs.) A data cache miss involves a delay of eight CPU cycles while the data in the cache can be accessed in one cycle [36]. Noting that a FLOP (add and multiply operation) takes one CPU cycle the conclusion is that a data cache miss decreases the performance significantly. By increasing the number of processors assigned to the task and keeping the overall problem size constant, the amount of data assigned to each processor is reduced. In consequence the number of cache misses is reduced or completely eliminated hence the super-linear speedup. The implicit mode shows a normal (less than linear) speedup because it is computationally more intensive and the effects of the cache misses are less important. A super-linear speedup similar to that of the explicit case has been observed by Michl *et* al. [48] on a cluster of IBM RS/6000-500 workstations. It also interesting to note the trend of the speedup which

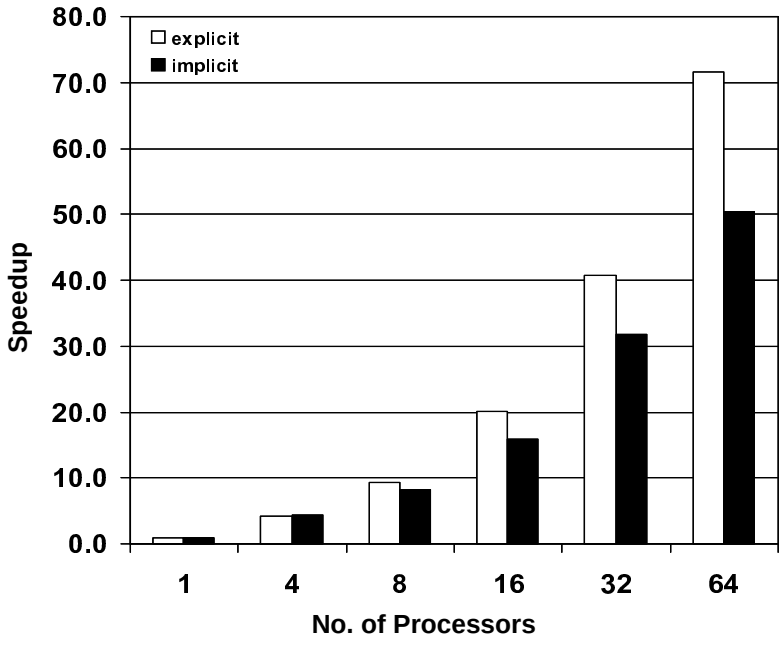

Figure 5.37: Fixed grid speed-up results of an older verison of WARP3 on a two-dimensional grid.i Note the super-linear speedup of the explicit scheme that is explained by system architecture effects.

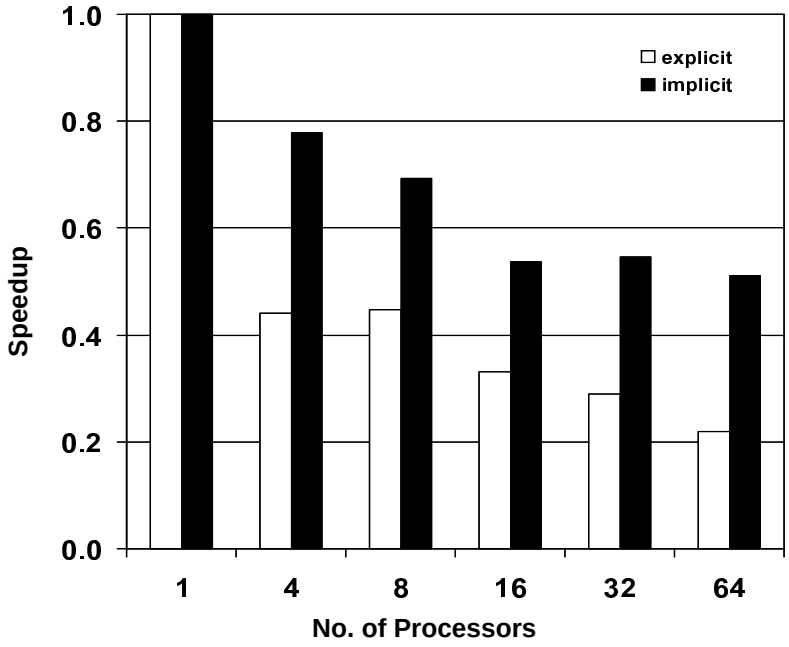

Figure 5.38: Scaled grid speed-up results of an older version of WARP3 on a two-dimensional grid. Ideal speedup on scaled grids is unity. This tests eliminates the effects of system architecture from the analysis and shows how the communication time affects the performance of the parallel code. The more processors are used for a job the farther the speedup is from unity.

shows an increasing slope for both explicit and implicit modes, showing that the code is far from saturation from communications. Saturation from communications takes place when the amount of time spent on communications becomes comparable to that used for computations.

The scaled grid test was performed in order to eliminate the system architecture effects from the performance analysis. The ideal speedup in this case is constant and equal to one. This result is obtained taking $T_{comm} = 0$ and noticing that for scaled up grids $T_{seq} = T_p$. Replacing these terms in Amdahl's law in Eqn (5.44) results in $S_p = p/p = 1$. The test showed that the speedup is less than one and

that it is decreasing with the number of processors. The results of the scaled grid test are shown in Figure 5.38. The fact that speedup is inversely proportional to the number of processors (for scaled-grids) is an expected result and shows that the total communication time increases with the number of processors. It has to be noted that since the negative slope of the speedup is gentle and it seems to be flattening out.

After the modifications of the algorithm were implemented the parallel performance has been tested again to make certain that the current version of WARP3 delivers the same parallel performance as the old version. The algorithm modifications include not only the improvements to the Riemann solver and the new implicit scheme but also the use of derived data types (DDT) for message passing. The parallel performance tests of the new version were conducted on a spheromak configuration similar to that presented in Section 5.8. Only a scaled grid test was performed since it was shown that this type of test eliminates the system architecture effects from the analysis. The spheromak grid of the type used in this test is shown in Figure 5.27 where it can be seen that it is made of twelve blocks. The total number of processors available for running the parallel code at the Aeronautics and Astronautics Department is sixteen and the smallest common multiple between sixteen and twelve is fortyeight so that the grids used had fortyeight blocks organized in four layers of twelve block each. The total number of cells for the one processor run was $12,288$ distributed equally over the blocks and the number of cells increased proportionally to the number of processors for the run. For example the sixteen processor run was performed on a grid of $196,608 = (12,288 \times 16)$ cells. The code was run in the ideal MHD mode and it was stopped after 500 iterations in the explicit case and 250 iterations in the implicit case. Four runs were performed for each of the cases and the timing results were averaged. This time instead of using the CPU time as a measure of performance the total wall clock time was used. The wall clock time is the overall amount of time that takes a job to complete and is a realistic index because it shows effectively how much time a user has to wait for his code to complete a run. The

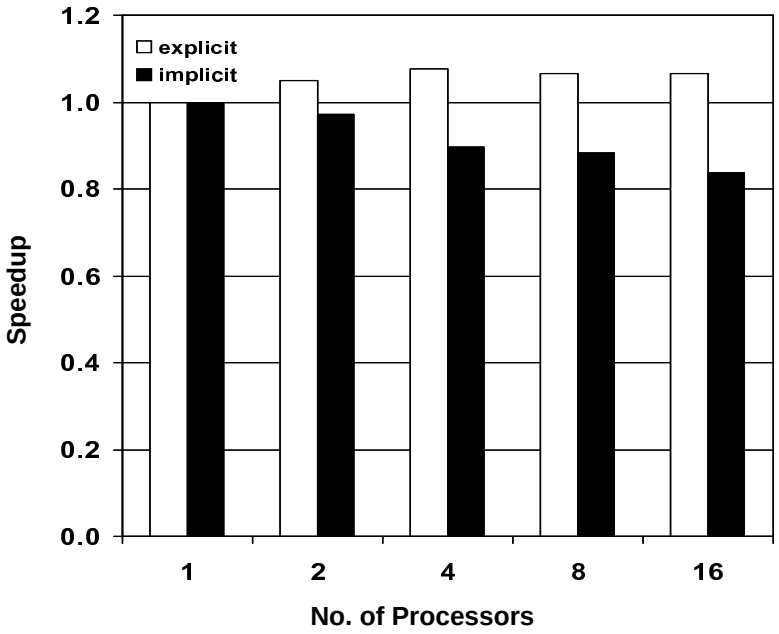

Figure 5.39: New version scaled-grid speed-up results on a three-dimensional grid on an IBM SP2 results. Note the slighlty larger than unity speedup of the explicit parallel runs. The explanation is that for the single processor run the message exchange between the fortyeight blocks of the grid hurts the performance of the code. As the number of processors goes up the number of blocks per processor decreases, which combined with the asynchronicity of the parallel code gives the net result that speedups are slightly better than ideal.

test was run on the DEC Alpha cluster at the Department and on the IBM SP2 at MHPCC for comparison purposes. The results of the test are shown in Figure 5.39 for the IBM SP2 and in Figure 5.40 for the Alpha cluster. As mentioned above the ideal speedup for scaled grid runs in equal to one and it can be noticed that for the explicit runs the speedup obtained is larger than one on both systems. An explanation for this behavior is found if the message exchange between blocks is analyzed. If WARP3 is run on a single processor, the code runs in lockstep mode. Specifically the code can perform computations only when all data has been exchnage between

all blocks. If WARP3 is run on multiple processors the message exchange between blocks on different processors is asynchronous. The code does not wait for a message pass to complete and proceeds ahead with other message passes or computations. The code only waits for a message pass to be completed when the data is needed by the code. The performance of the code run in the implicit mode is as expected because is computationally more intensive and it can be seen that it is better than the two-dimensional runs performed with the old version of the code. The better parallel performance of the three-dimensional runs is attributed to the fact that the amount of computation per processor is increased compared to two-dimensional runs so that the time spent on inter-processor communications has a lower contribution to the parallel performance of WARP3.

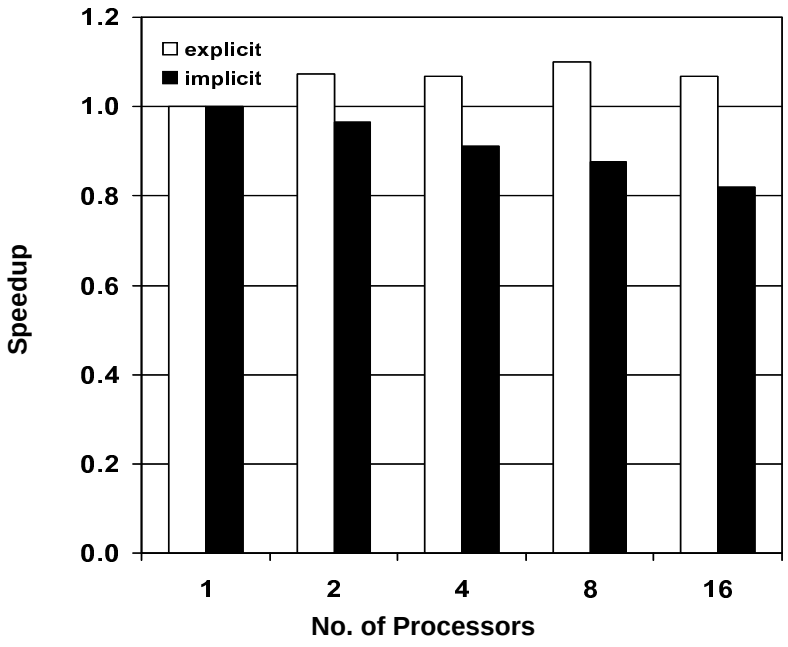

Figure 5.40: New version scaled-grid speed-up results on a three-dimensional grid on the Departmental Alpha cluster results. Similar to the IBM SP2 results the explicit mode displays speedup slightly larger than unity.

Chapter 6

CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK

This section summarizes the contributions to the state of the art in computational MHD algorithms and presents some suggestions for researchers that will be involved with computational MHD in general and WARP3 in particular.

6.1 Locally Aligned Coordinate System

The traditional CFD method of generalized coordinates transformation implemented in a previous version of the code by Jones [43] proved to be extremely complicated and prone to errors. The generalized coordinate transformation method is not complicated *per se* and is applied to many three-dimensional CFD codes that are in use today. However, the flux formulas used in the approximate Riemann solver for MHD are complicated by the fact that they have to be normalized to avoid degeneracies in certain cases. The normalization and treatment of degeneracies were discussed in Section 3.2.3.

An original approach has been developed to reduce the complexity of the flux formulas and was implemented succesfully. The new approach projects the vector conserved variables (momentum and magnetic field) on a coordinate system aligned with the cell face where the Riemann problem is to be solved. The approximate Riemann solver thus uses the relatively simple one-dimensional flux formulas and evaluates the fluxes at the respective face in the locally aligned coordinate system. The fluxes are projected back on the $Oxyz$ coordinate system and the conserved

variables are updated. The amount of computation is minimal, a few FLOPs per conserved variable and is offset by the simpler flux formulas (compared to generalized coordinates transformation)

A simple one-dimensional gas dynamics Riemann problem test presented in Figure 5.3 shows that the generalized coordinate transformation had at least one error that resulted in momentum leaking in the other two spatial directions. The solution to the same problem obtained with the new algorithm that uses a locally aligned coordinate system is presented in Figure 5.4 and it shows that the leakage problem has been fixed.

6.2 Explicit and Implicit Approximate Riemann Solvers

WARP3 incorporates both an explicit and an implicit scheme that solve the PDE system that describes the viscous and resistive single fluid MHD model. Both schemes use the same approximate Riemann solver that is second order accurate in space in regions where the solution is smooth and first order accurate in the regions with sharp gradients of the conserved variables. The implicit scheme is second order accurate in time and the explicit is first order accurate in time. Since both schemes use a common algorithm (Riemann solver) the effort of maintaining both schemes is minimal. The dual mode capability is justified by the fact that development and testing of the simpler explicit scheme is easier to understand, modify and debug.

Original contributions to and advances in the state of the art of computational fluid dynamics have been incorporated in WARP3. The implicit scheme uses a dual level discretization of the conservation equations so that both steady state and time dependent simulations can be performed with WARP3. The discretization of the implicit formulation results in a large sparse linear system that is solved using a symmetric Gauss-Seidel scheme. The flux Jacobians present in the implicit discretization are calculated using two methods. The first method is the traditional single sided,

first order accurate approximation and the second method uses a complex number formulation that is second order accurate. The only caveat is that the complex number formulation gave unphysical behavior of the magnetic field for the supersonic MHD wedge flow. However, since the complex number formulation works well for all the other benchmarks it is assumed that the problem comes from the normalization of the eigenvectors. It is hoped that the problem with the normalization can be found by performing more MHD supersonic wedge flow tests and analyzing how the magnitudes of the components of the numerical fluxes vary with the Mach number. Compared to the previous version of WARP3 the new version uses full flux Jacobians rather than the diagonal matrices that approximated the flux Jacobians. As a consequence the new version of WARP3 is a robust code that works well and has been benchmarked for one, two and three-dimensional simulations.

There are a few advantages of the implicit scheme over the explicit scheme. The most obvious one is for steady state calculations. It has been showed that the implicit scheme converges the steady state solution to machine precision independent of the value of the time step. To converge to a similar solution the explicit scheme requires small time steps that add numerical diffusion indiscriminately in the entire domain as shown in Section 5.3. The combination of symmetric Gauss-Seidel passes (sgs) and pseudo time steps (m) that give the lowest CPU time for convergence of the implicit scheme seems to be problem dependent and has been observed to be around $sgs = 5$ and $m = 5$ for the two-dimensional and three-dimensional gas dynamics problems studied. For time-dependent simulations the CPU efficiency of the implicit scheme is less impressive, however the second order time accuracy of the implicit scheme and the fact that it is less compressive than the explicit scheme helps tracking time dependent data with more accuracy.

6.3 Parallel Code Performance

Another major contribution to the advance of the state of the art is the implementation of the parallel algorithm. Being able to run WARP3 on multiple processor computers speeds up the simulations thus providing significant reductions in job run times. The interprocessor communication is based on a widely adopted communications standard (MPI) and the code is written in Fortran90 and ANSI C so that WARP3 is portable between various parallel computer architectures. The methods used to exchange data between blocks are highly optimized by using derived data types (DDTs). Use of DDTs eliminates the overhead associated with explicit buffering and has proven to be robust and relatively easy to implement. The portability of the code was proven almost on the daily basis by the fact that the code was developed and tested on a cluster of DEC Alpha workstations at the Department of Aeronautics and Astronautics and large scale runs were performed on an IBM SP2 without any modifications to the code. Last but not least, with slight modifications WARP3 was ported and succesfully run on multiprocessor WindowsNT computers. The excellent performance of the parallel code has been demonstrated by the speed-up results obtained.

One hardware related issue encountered during parallel runs was that the reliability of a cluster of more than one hundred processors is rather low. The culprit in the cases described here were communication network nodes that failed and as a result the code stopped due to MPI time-out errors. The restart capability of WARP3 was designed with this in mind so that a run can be restarted from data written to disk before a system crash. It is up to the user to specify how often WARP3 writes restart data to disk and caution should be exercised since writing to disk to often slows down the code considerably. However it has been observed that over the last two years overall hardware capability has improved considerably and runs with eighty to ninety processors worked without any problems.

6.4 Benchmarks and Simulations

The benchmarks performed with WARP3 where extensive and they started with one-dimensional gas dynamics and MHD Riemann problems that tested the approximate Riemann solver. Diffusion dominated one-dimensional problems tested the algorithms used to evaluate the parabolic fluxes. Two-dimensional supersonic wedge flow problems, for both gas dynamics and MHD were the next steps took to validate the code. The steady state supersonic wedge flow problems showed that the implicit scheme is more efficient than the explicit scheme in driving the solution to steady state. A three-dimensional gas dynamics intersecting wedge flow simulation compared well with experimental results and results obtained with other CFD codes. Stability studies were performed for spheromaks enclosed in cylindrical flux conservers. The spheromak simulations compared well linear theory predictions for the three cases studied and matched the growth rate of the kinetic energy predicted by linear stability theory.

6.5 Suggestions for Future Work

It is believed that in the current state WARP3 is a robust approximate Riemann solver for the resistive and viscous single fluid MHD model and it can be used as research and design tool for plasma physics by future workers.

As mentioned in Section 2 the treatment of the Hall and electron pressure gradient terms is not present at this time in the code. These terms are important for plasmas of fusion interest and for electric thrusters. Attempts by Becerra-Sagredo [7] to include the Hall terms in the explicit version of the code as part of the parabolic fluxes showed mixed results and he suggested that future attempts at including the Hall terms should consider including them in the hyperbolic part of the system of PDEs. As for the electron pressure gradient current development efforts by another graduate student, Ward Vuillemot, might lead to to their implementation in the code as well.

The complex number formulation, in particular its behavior at high speed, high

density and pressure MHD flows should be investigated further. It is rather difficult to debug the complex formulation since the complex number eigenvectors loose any physical meaning. Since only for the supersonic MHD flows a problem has been encountered with the complex number formulation of the flux Jacobians it is likely that the error will be found performing more similar tests. Probably the best way to proceed is to perform supersonic MHD wedge flow simulations with increasing inflow Mach numbers and keeping the other boundary and initial conditions the same. Studying the relative variations of the elements of the eigenvectors and the eigenvalues will probably help identify any erroneous terms that appear in the eigenvectors.

It is suggested that once WARP3 is frozen (no major modifications to the algorithm are to be implemented) be optimized using profiling and reorganization of the procedures so that its CPU efficiency is increased.

The parallel code has been proven to be robust and the software components that deal with information exchanged have been frozen. It might be worth while writing a load balancing procedure that automates block distribution to processors. Given a certain number of processors the procedure would distribute the blocks per processors so that an equal (or approximately equal) number of cells is allocated to each processor.

During the development of WARP3 it has been learned that having a good debugger is essential to generating quality code. In the last year and a half an excellent debugging tool has been acquired by the Department and used intensively on WARP3. The debugger is **totalview 3.8** and it is recommended that future workers spend some time learning how to use this tool - it will save large amounts of time off debugging work.

BIBLIOGRAPHY

[1] A. Anders. *A Formulary for Plasma Physics.* Akademie-Verlag Berlin, 1990.

[2] N. Aslan. Numerical Solutions of One-Dimensional MHD Equations by a Fluctuation Approach. *Int. J. Num. Meth. Fluids*, 22:569–580, 1996.

[3] N. Aslan. Two-Dimensional Solutions of MHD Equations with an Adapted Roe Method. *Int. J. Num. Meth. Fluids*, 23:1211–1222, 1996.

[4] N. Aslan and T. Kammash. Developing Numerical Fluxes with New Sonic Fix for MHD Equations. *J. Comput. Phys.*, 133:43–55, 1997.

[5] C.W. Barnes, T.R. Jarboe, G.J. Marklin, S.O. Knox, and I. Hennins. The Impedenace and Energy Efficiency of a Coaxial Magnetized Plasma Source Used for Spheromak Formation and Sustainment. *Phys. Fluids*, 2(B):1871–1878, 1990.

[6] P. Batten, M.A. Leschziner, and U.C. Goldberg. Average-State Jacobians and Implicit Methods for Compressible Viscous and Turbulent Flows. *J. Comput. Phys.*, 137:37–78, 1997.

[7] J. Becerra-Sagredo. *Semi-implicit Treatment of the Hall Term in Finite Volume, MHD Computations.* PhD thesis, University of Washington, 1998.

[8] C. K. Birdsall and D. Fuss. Clouds-in-Clouds, Clouds-in-Cells Physics for Many-Body Plasma Simulation. *J. Comput. Phys.*, 135:141–148, 1997.

[9] A. Bondeson, G.J. Marklin, Z.G. An, H.H. Chen, Y.C. Lee, and C.S. Liu. Tilting instability of a cylindrical spheromak. *Phys. Fluids*, 24(9):1682–1688, 1981.

[10] J.U. Brackbill and D.C. Barnes. The Effect of Nonzero $\nabla \cdot \mathbf{B}$ on the Numerical Solution of Magnetohydrodynamics Equation. *J. Comput. Phys.*, 35:426–430, 1980.

[11] M. Brio and C.C. Wu. An Upwind Differencing Scheme for the Equations of Magnetohydrodyamics. *J. Comput. Phys.*, 45:400–422, 1988.

[12] P. Cargo and G. Gallice. Construction of a Roe matrix for the ideal magnetoydrodynamics equations. *Z. angew. Math. Mech.*, 76:369–370, 1995.

[13] P. Cargo and G. Gallice. Matrices de Roe pour de lois de conservation générales sous forme eulérienne ou lagrangienne: application à la dynamique de gaz et à la magnétohydrodynamique. *C. R. Acad. Sci. Paris*, 321(I):1069–1072, 1995.

[14] P. Cargo and G. Gallice. Un Solveur de Roe pour les équations de la magnétohydrodynamique. *C. R. Acad. Sci. Paris*, 320(I):1269–1272, 1995.

[15] P. Cargo, G. Gallice, and P.A. Raviart. Construction d'une linéarisée de Roe pour les équations de la MHD idéale. *C. R. Acad. Sci. Paris*, 323(I):951–955, 1996.

[16] R. Carruthers. Criteria for the Assessment of Reactor Potential. In B. Brunelli and G.G. Leota, editors, *Unconventional Approaches to Fusion*, pages 39–46, 1994.

[17] A.F. Charwat and L.G. Redekeopp. Supersonic Interference Flow along the Corner of Intersecting Wedges. *AIAA. J.*, 5(3):480–488, 1966.

[18] W. Dai and P.R. Woodward. An Approximate Riemann Solver for Ideal Magnetohydrodynamics. *J. Comput. Phys.*, 111:354–372, 1994.

[19] L. Dubuc, F. Cantariti, M. Woodgate, B. Gribben, K.J. Badcock, and B.E. Richards. Solution of the Unsteady Euler Equations Using an Implicit Dual-Time Method. *AIAA. J.*, 36(8):1417–1424, 1998.

[20] H. Löb (editor). *Kerntechnik bei Satelliten und Raketen (Nuclear Engineering for Satellites and Rockets).* Verlag Karl Thiemig KG, 1970.

[21] W.T. Armstrong *et* al. Compact Toroid Experiments and Theory. In *Plasma Physics and Controlled Nuclear Fusion Research*, volume I of *IAEA-1980*, pages 481–488, 1980.

[22] J.M. Finn, W.M. Manheimer, and E. Ott. Spheromak tilting instability in cylindrical geometry. *Phys. Fluids*, 24(7):1336–1341, 1981.

[23] Message Passing Interface Forum. *MPI: A Message Passing Interface Standard 1.1.* 1994.

[24] Message Passing Interface Forum. *MPI2: Extensions to the Message Passing Interface.* 1997.

[25] J. P. Freidberg. *Ideal Magnetohydrodynamics.* Plenum Press, 1987.

[26] A. Fyfe, S. Taylor, and R. Williams. A Concurrent, Tetrahedral Fluid Solver, Based on Wave Propagation. In N. Satofuka, J. Periaux, and A. Ecer, editors, *Parallel Computational Fluid Dynamics Conference - New Algorithmas and Applications*, pages 305–312, 1994.

[27] S. K. Godunov. A Difference Method for the Numerical Calculation of Discontinous Solutions of Hydrodynamics Equations. *Mat. Sb.*, 47:271–282, 1959.

[28] A. Harten. On a Class of High Resolution Total-Variation-Stable Finite Difference Schemes. Technical report, NYU, October 1982.

[29] A. Harten. High resolution schemes for hyperbolic conservation laws. *J. Comput. Phys.*, 49:357–250, 1983.

[30] M.T. Heath. *Scientific Computing. An Introductory Survey.* McGraw-Hill Book Company, 1996.

[31] K. Hoffmann and S. T. Chiang. *Computational Fluid Dynamics for Engineers.* Engineering Education System, 1998.

[32] M. Holt. *Numerical Methods in Fluid Dynamics.* Springer-Verlag, 1977.

[33] E.B. Hooper. The Spheromak Path to Fusion Energy. Technical Report UCRL-ID-130429, Lawrence Livermore National Laboratory, April 1998.

[34] E.B. Hooper, J.H. Hammer, C.W. Barnes, J.C. Fernandez, and F.J. Wysocki. A re-examination of spheromak experiments and opportunities. *Fusion Techn.*, 29:191–205, 1996.

[35] E.B. Hooper, L.D. Pearlstein, and R.H. Bulmer. MHD Equilibria in a Spheromak Sustained by Coaxial Helicity Injection. *LLNL Paper*, 1998.

[36] IBM Corp. *Optimization/Tuning Guide for XL Fortran and XL C.*

[37] J.C. Newman III, W.K. Anderson, and D.L. Whitfield. Multidisciplinary Sensitivity Derivatives Using Complex Variables. Technical Report MSSU-EIRS-ERC-98-08, NSF Engineering Research Center for Computational Field Simulation, July 1998.

[38] Jr. J. Marshall. Performance of a Hydromagnetic Plasma Gun. *Phys. Fluids*, 3:134–140, 1960.

[39] A. Jameson. Time Dependent Calculations Using Multigrid with Applications to Unsteady Flows Past Airfoils and Wings. AIAA 91-1596, June 1991.

[40] T.R. Jarboe. Review of spheromak research. *Plasma Phys. Control. Fusion*, 36:945–990, 1994.

[41] T.R. Jarboe, I. Hennins, H.W. Hoida, R.K. Linford, J. Marshall, D.A. Platts, and A.R. SHerwood. Motion of a Compact Toroid inside a Cylindrical Flux Conserver. *Phys. Rev. Lett.*, 45(15):1264–1267, 1980.

[42] T.R. Jarboe, I. Hennins, A.R. Sherwood, C.W. Barnes, and H.W. Hoida. Slow Formation and Sustainment of Spheromaks by a Coaxial Magnetized Plasma Source. *Phys. Rev. Lett.*, 51:39–45, 1983.

[43] O.S. Jones. *Study of Magnetic Relaxation in Plasmas Using a Parallel Implicit MHD Solver*. PhD thesis, University of Washington, 1997.

[44] O.S. Jones, D.S. Eberhardt, and U. Shumlak. An Implicit Scheme for Nonideal Magnetohydrodynamics. *J. Comput. Phys.*, 130:231–242, 1997.

[45] S.O. Knox, C.W. Barnes, G.J. Marklin, T.R. Jarboe, I. Henins, H.W. Hoida, and B.L. Wright. Observations of Spheromak Equilibria Which Differ from the Minimum-Energy State and Have Internal Kink Distortions. *Phys. Rev. Lett.*, 56(8):842–845, 1986.

[46] R.J. LeVeque. *Numerical Methods for Conservation Laws*. Birkhäuser Verlag, 1992.

[47] A.M. Messner and G.Q. Taylor. Solid Polyhedron Measures. *Assoc. for Comp. Mach. Trans. on Math. Soft.*, 6:121–130, 1980.

[48] Thomas Michl, Siegfried Wagnerand Michael, and Lenke Arnt Bode. Dataparallel Implicit Navier-Stokes Solver on Different Multiprocessors. In A. Ecer, J. Häuser, P. Leca, and J. Periaux, editors, *Parallel Computational Fluid Dynamics New Trends and Advances, Proceedings of the Parallel CFD '92 Conference, Paris - France May 10 -12, 1993*, pages 133–140. Elsevier Science Publisers, 1995.

[49] H.K. Moffat. *Magnetic Field Generation in Electrically Conducting Fluids.* Cambridge University Press, 1978.

[50] H.W. Liepmann nad A. Roshko. *Elements of Gasdynamics.* John Wiley and Sons, Inc., 1967.

[51] P.H. Oosthuizen and W.E. Carscallen. *Compressible Fluid Flow.* McGraw-Hill Book Company, 1996.

[52] P.D. Orkwis and K.J. Vanden. On the Accuracy of Numerical Versus Analytical Jacobians. AIAA 94-0176, January 1994.

[53] R. Pankajakshan and W. Roger Briley. Parallel Solution of Viscous Incompressible Flow on Multi-Block Structured Grids Using MPI. In A. Ecer, J. Periaux, N. Satofuka, and S. Taylor, editors, *Parallel Computational Fluid Dynamics Conference - Implementation and Results Using Parallel Computers*, pages 601–608, 1995.

[54] K.G. Powell, P.L. Roe, R.S. Myong, T. Gombosi, and D. De Zeeuw. An Upwind Scheme for Magnetohydrodynamics. In K.W. Morton and M.J. Baines, editors, *Numerical Methods for Fluid Dynamics V*, pages 163–180, 1995.

[55] S. Raber, B. Udrea, and U. Shumlak. Computational MHD on an Intel Based Windows NT Platform. In *American Physical Society Division of Plasma Physics Conference Proceedings*, New Orleans, LA, November 1998.

[56] R. D. Richtmyer and K. W. Morton. *Difference Methods for Initial-Value Problems*. Interscience Publishers, 1967.

[57] P. L. Roe. Approximate Riemann Solvers, Parameter Vectors and Difference Schemes. *J. Comput. Phys.*, 43:357–372, 1981.

[58] P.L. Roe and D.S. Balsara. Notes on the Eigensystem of Magnetohydrodynamics. *SIAM. J. App. Math.*, 56:57–67, 1996.

[59] M.N. Rosenbluth and M.N. Bussac. MHD Stability of Spheromak. *Nuc. Fusion*, 19(4):489–498, 1979.

[60] R.I. Roy, D.E. Hastings, and S. Taylor. Three-Dimensional Plasma Particle-in-Cell Calculations of Ion Thruster Backflow Contamination. *J. Comput. Phys.*, 128:6–18, 1996.

[61] G.E. Sasser, L. Bowers, J.W. Luginsland, M. Stum, and J.J. Watrous. 3D PIC Simulations of the Relativistic Klystron Oscillator. In *American Physical Society Division of Plasma Physics Conference Proceedings*, New Orleans, LA, November 1998.

[62] U. Shumlak. *MHD Tilt Stability of Dynamic Spheromaks*. PhD thesis, University of California at Berkeley, 1992.

[63] U. Shumlak and T.R. Jarboe. Higher Mode Stability in Spheromak Equilibria. *Submitted for publication*, 1999.

[64] P.C. Sleziona, M. Auweter-Kurtz, and H.O. Schrade. Computation of MPD Flows and Comparison with Experimental Results. *Int. J. Num. Meth. Eng.*, 34:759–771, 1992.

[65] M. Snir, S.W. Otto, S. Huss-Lederman, D.W. Walker, and J. Dongarra. *MPI: The complete Reference*. MIT Press, 1996.

[66] G. Sod. A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. *J. Comput. Phys.*, 27(1):611–630, 1978.

[67] V. Spitzer. *Physics of Fully Ionized Gases*. Interscience Publishers New York, 1956.

[68] W. Squire and G. Trapp. Using Complex Variables to Estimate Derivatives of Real Functions. *SIAM Review*, 40(1):110–112, 1998.

[69] J.L. Steger and R.F. Warming. Flux Vector Splitting of the Inviscid Gasdynamic Equations with Applications to Finite-Difference Methods. *J. Comput. Phys.*, 40:263–293, 1989.

[70] A.H. Stroud. *Approximate Calculation of Multiple Integrals*. Prentice-Hall, 1971.

[71] G. W. Sutton and A. Sherman. *Engineering Magnetohydrodynamics*. McGraw-Hill Book Company, 1965.

[72] P.K. Sweeby. High Resolution Schemes Using Flux Limiters for Hyperbolic Conservation Laws. *SIAM J. on Num. Anal.*, 21:995–1011, 1984.

[73] D.W. Swift. Use of a Hybrid Code for Global-Scale Plasma Simulation. *J. Comput. Phys.*, 126:109–121, 1996.

[74] G. Tannehill. *Computational Fluid Mechanics and Heat Transfer*. Taylor and Francis, 1997.

[75] J.B. Taylor. Relaxation of Toroidal Plasma and Generation of Reverse Magnetic Fields. *Phys. Rev. Lett.*, 33:1139–1145, 1974.

[76] W.B. Thompson. *An Introduction to Plasma Physics*. Pergamon Press, 1962.

[77] B. Udrea, O.S. Jones, U. Shumlak, and D.S. Eberhardt. A portable parallel implicit approximate Riemann solver for non-ideal MHD. AIAA 97-0366, January 1997.

[78] B. van Leer, W.T. Lee, and K.G. Powell. Sonic Point Capturing. *AIAA. J.*, 49:235–250, 1983.

[79] K. J. Vanden and D.L. Whitfield. Direct and Iterative Algorithms for the Three-Dimensional Euler Equations. *AIAA. J.*, 33(5):851–858, 1995.

[80] J.C.T. Wang and S. Taylor. A Concurrent Navier-Stokes Solver for Implicit Multibody Calculations. In A. Ecer, J. Hauser, P. Leca, and J. Periaux, editors, *Parallel Computational Fluid Dynamics Conference - New Trands and Advances*, pages 295–305, 1993.

[81] J. Wesson. *Tokamaks*. Oxford Science Publications, 1997.

[82] D.L. Whitfield. Numerical Solution of the Shallow Water Equations. Technical Report MSSU-EIRS-ERC-96-4, NSF Engineering Research Center for Computational Field Simulation, July 1996.

[83] D.L. Whitfield and L.K. Taylor. Variants of a Two-Level Method for the Approximate Numerical Solution of Field Simulation Equations. Technical Report MSSU-EIRS-ERC-98-09, NSF Engineering Research Center for Computational Field Simulation, July 1998.

[84] H.C. Yee, R.F. Warming, and A. Harten. Implicit Total Variation Diminishing (TVD) Schemes for Steady-State Calculations. *J. Comput. Phys.*, 57:327–360, 1985.

[85] A.L. Zachary and P. Collela. A Higher-Order Godunov Method for the Equations of Ideal Magnetohydrodynamics. *J. Comput. Phys.*, 99:341–347, 1992.

[86] A.L. Zachary, A. Malagoli, and P. Collela. A Higher-Order Godunov Method for Multidimensional Ideal Magnetohydrodynamics. *SIAM J. on Sci. Comput.*, 15-N2:263–284, 1994.

Appendix A

CONSERVED-PRIMITIVE VARIABLE TRANSFORMATION

The conserved variables of the single fluid model MHD written in vector form are

$$\mathbf{q} = [\rho, \rho v_x, \rho v_y, \rho v_z, B_x, B_y, B_z, e]^T, \tag{A.1}$$

and the primitive variables are

$$\mathbf{w} = [\rho, v_x, v_y, v_z, B_x, B_y, B_z, p]^T. \tag{A.2}$$

The formula for the transformation of the flux Jacobians from conserved to primitive variables and vice-versa is obtained starting with the one-dimensional conservation law

$$\frac{\partial \mathbf{q}}{\partial t} + \frac{\partial \mathbf{f}}{\partial x} = \frac{\partial \mathbf{q}}{\partial t} + A_c \frac{\partial \mathbf{q}}{\partial x} = 0. \tag{A.3}$$

The conservation equation is written in terms of primitive variables by applying the chain rule to Eqn (A.3)

$$\frac{\partial \mathbf{q}}{\partial \mathbf{w}} \frac{\partial \mathbf{w}}{\partial t} + A_c \frac{\partial \mathbf{q}}{\partial \mathbf{w}} \frac{\partial \mathbf{w}}{\partial x} = 0, \tag{A.4}$$

where $A_c = \partial \mathbf{f} / \partial \mathbf{q}$ is the flux Jacobian derived in terms of conserved variables.

Premultiplication of Eqn (A.4) with the inverse of $\partial \mathbf{q} / \partial \mathbf{w}$ yields

$$\frac{\partial \mathbf{w}}{\partial t} + \left(\frac{\partial \mathbf{q}}{\partial \mathbf{w}} \right)^{-1} A_c \frac{\partial \mathbf{q}}{\partial \mathbf{w}} \frac{\partial \mathbf{w}}{\partial x} = \frac{\partial \mathbf{w}}{\partial t} + A_p \frac{\partial \mathbf{w}}{\partial x} = 0, \tag{A.5}$$

where $A_p = \partial \mathbf{f} / \partial \mathbf{w}$ is the flux Jacobian derived in terms of primitive variables.

The transformation from the primitive to conserved form of the flux Jacobian is

$$A_p = \left(\frac{\partial \mathbf{q}}{\partial \mathbf{w}}\right)^{-1} A_c \frac{\partial \mathbf{q}}{\partial \mathbf{w}} = \frac{\partial \mathbf{w}}{\partial \mathbf{q}} A_c \frac{\partial \mathbf{q}}{\partial \mathbf{w}}, \tag{A.6}$$

and the transformation matrices are

$$\frac{\partial \mathbf{w}}{\partial \mathbf{q}} = \left(\frac{\partial \mathbf{q}}{\partial \mathbf{w}}\right)^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ v_x & \rho & 0 & 0 & 0 & 0 & 0 & 0 \\ v_y & 0 & \rho & 0 & 0 & 0 & 0 & 0 \\ v_z & 0 & 0 & \rho & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \frac{v^2}{2} & \rho v_x & \rho v_y & \rho v_z & B_x & B_y & B_z & \frac{1}{\gamma'} \end{bmatrix}, \tag{A.7}$$

and

$$\frac{\partial \mathbf{q}}{\partial \mathbf{w}} = \left(\frac{\partial \mathbf{w}}{\partial \mathbf{q}}\right)^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{v_x}{\rho} & \frac{1}{\rho} & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{v_y}{\rho} & 0 & \frac{1}{\rho} & 0 & 0 & 0 & 0 & 0 \\ -\frac{v_z}{\rho} & 0 & 0 & \frac{1}{\rho} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \frac{\gamma'}{2} v^2 & \gamma' v_x & \gamma' v_y & \gamma' v_z & \gamma' B_x & \gamma' B_y & \gamma' B_z & \gamma' \end{bmatrix}, \tag{A.8}$$

where $v^2 = v_x^2 + v_y^2 + v_z^2$ and $\gamma' = \gamma - 1$.

The left and right eigenvectors of the flux Jacobians can also be transformed from primitive to conserved form and vice-versa. The transformation formulas for the eigenvector matrices are obtained starting from the eigenvector decomposition of the primitive flux Jacobian

$$A_p = R_p \Lambda L_p = \frac{\partial \mathbf{w}}{\partial \mathbf{q}} A_c \frac{\partial \mathbf{q}}{\partial \mathbf{w}} = \frac{\partial \mathbf{w}}{\partial \mathbf{q}} R_c \Lambda L_c \frac{\partial \mathbf{q}}{\partial \mathbf{w}}, \tag{A.9}$$

where Λ is a diagonal matrix with the eigenvalues of the flux Jacobian as elements.

Identyfing the terms in Eqn (A.9) yields the transformation formulas

$$R_p = \frac{\partial \mathbf{w}}{\partial \mathbf{q}} R_c \quad \text{and} \quad R_c = \frac{\partial \mathbf{q}}{\partial \mathbf{w}} R_p \tag{A.10}$$

$$L_p = L_c \frac{\partial \mathbf{q}}{\partial \mathbf{w}} \quad \text{and} \quad L_c = L_p \frac{\partial \mathbf{w}}{\partial \mathbf{q}} \tag{A.11}$$

Appendix B

EIGENSYSTEM OF THE MODIFIED SINGLE FLUID MHD FLUX JACOBIAN

This appendix contains the expressions of the flux Jacobian in the direction parallel to the Ox axis, and the expressions of its eigenvalues and eigenvectors. The eigenvectors are not normalized and are prone to degeneracies as explained in Section 3.2.3. (The eigensystem is given for reference as it was first published by Powell *et* al. [54].)

The flux Jacobian in the direction normal to the cell face is calculated in terms of primitive variables

$$\mathbf{w} = [\rho,\ v_x,\ v_y,\ v_z,\ B_x,\ B_y,\ B_z,\ p]^T, \tag{B.1}$$

and it is

$$A_p = \begin{bmatrix} v_x & \rho & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & v_x & 0 & 0 & 0 & \frac{B_y}{\rho} & \frac{B_z}{\rho} & \frac{1}{\rho} \\ 0 & 0 & v_x & 0 & 0 & -\frac{B_x}{\rho} & 0 & 0 \\ 0 & 0 & 0 & v_x & 0 & 0 & -\frac{B_x}{\rho} & 0 \\ 0 & 0 & 0 & 0 & v_x & 0 & 0 & 0 \\ 0 & B_y & -B_x & 0 & 0 & v_x & 0 & 0 \\ 0 & B_z & 0 & -B_x & 0 & 0 & v_x & 0 \\ 0 & \gamma p & 0 & 0 & 0 & 0 & 0 & 0 & v_x \end{bmatrix}, \tag{B.2}$$

The flux Jacobian derived using primitive variables has a simpler expression than the flux Jacobian calculated using conserved variables.

The set of eigenvalues and eigenvectors of the flux Jacobian in Eqn (B.2) are

Entropy wave: $\lambda_e = v_x$

$$r_e = \left[1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0\right]^T \tag{B.3}$$

$$l_e = \left[1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ -\tfrac{1}{c^2}\right] \tag{B.4}$$

Alfvén waves: $\lambda_a = v_x \pm c_a$

$$r_A^{\pm} = \left[0,\ 0,\ -B_z,\ B_y\ 0,\ \pm B_z\sqrt{\rho},\ \mp B_y\sqrt{\rho}\right]^T \tag{B.5}$$

$$l_A^{\pm} = \left[0,\ 0,\ -B_z,\ B_y\ 0,\ \pm\tfrac{B_z}{\sqrt{\rho}},\ \mp\tfrac{B_y}{\sqrt{\rho}}\right]^T \tag{B.6}$$

$$\tag{B.7}$$

Fast magneto-acoustic waves: $\lambda_f = v_x \pm c_f$

$$r_f^{\pm} = \left[\rho\ \pm c_f,\ \mp\tfrac{B_x B_y c_f}{\rho c_f^2 - B_x^2},\ \mp\tfrac{B_x B_z c_f}{\rho c_f^2 - B_x^2},\ 0,\ \tfrac{B_y \rho c_f^2}{\rho c_f^2 - B_x^2},\ \tfrac{B_z \rho c_f^2}{\rho c_f^2 - B_x^2},\ \gamma p\right]^T \tag{B.8}$$

$$l_f^{\pm} = \left[0\ \pm \rho c_f,\ \mp\tfrac{B_x B_y \rho c_f}{\rho c_f^2 - B_x^2},\ \mp\tfrac{B_x B_z \rho c_f}{\rho c_f^2 - B_x^2},\ 0,\ \tfrac{B_y \rho c_f^2}{\rho c_f^2 - B_x^2},\ \tfrac{B_z \rho c_f^2}{\rho c_f^2 - B_x^2},\ 1\right]^T \tag{B.9}$$

Slow magneto-acoustic waves: $\lambda_s = v_x \pm c_s$

$$r_s^{\pm} = \left[\rho\ \pm c_s,\ \mp\tfrac{B_x B_y c_s}{\rho c_s^2 - B_x^2},\ \mp\tfrac{B_x B_z c_s}{\rho c_s^2 - B_x^2},\ 0,\ \tfrac{B_y \rho c_s^2}{\rho c_s^2 - B_x^2},\ \tfrac{B_z \rho c_s^2}{\rho c_s^2 - B_x^2},\ \gamma p\right]^T \tag{B.10}$$

$$l_s^{\pm} = \left[0\ \pm \rho c_s,\ \mp\tfrac{B_x B_y \rho c_s}{\rho c_s^2 - B_x^2},\ \mp\tfrac{B_x B_z \rho c_s}{\rho c_s^2 - B_x^2},\ 0,\ \tfrac{B_y \rho c_s^2}{\rho c_s^2 - B_x^2},\ \tfrac{B_z \rho c_s^2}{\rho c_s^2 - B_x^2},\ 1\right]^T \tag{B.11}$$

Magnetic flux wave: $\lambda_d = v_n$

$$r_f = \left[0,\ 0,\ 0,\ 0,\ 1,\ 0,\ 0,\ 0\right]^T \tag{B.12}$$

$$l_f = \left[0,\ 0,\ 0,\ 0,\ 1,\ 0,\ 0,\ 0\right], \tag{B.13}$$

where c_a is the Alfvén speed, and $c_{f,s}$ are the fast and respectively slow magento-acoustic speeds. The formulas for these speed are

$$c_a = \frac{B_x}{\sqrt{\rho}} \tag{B.14}$$

$$c_{f,s}^2 = \frac{1}{2}\left[\frac{\gamma p + B^2}{\rho} \pm \sqrt{\left(\frac{\gamma p + B^2}{\rho}\right)^2 - 4\frac{\gamma p B_x^2}{\rho^2}}\right], \tag{B.15}$$

where the fast magneto-acoustic speed corresponds to the plus in front of the square root, and $B^2 = B_x^2 + B_y^2 + B_z^2$.

Appendix C

NORMALIZED EIGENSYSTEM OF THE SINGLE FLUID MHD FLUX JACOBIAN IN A LOCALLY ALIGNED REFERENCE FRAME

This appendix contains the expressions of the flux Jacobian in the direction normal to a cell interface, and the expressions of its eigenvalues and eigenvectors.

The flux Jacobian in the direction normal to the cell face is calculated in terms of primitive variables

$$\mathbf{w} = [\rho,\ v_n,\ v_{t1},\ v_{t2},\ B_n,\ B_{t1},\ B_{t2},\ p]^T, \tag{C.1}$$

and it is given by

$$A_p = \begin{bmatrix} v_n & \rho & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & v_n & 0 & 0 & 0 & \frac{B_{t1}}{\rho} & \frac{B_{t2}}{\rho} & \frac{1}{\rho} \\ 0 & 0 & v_n & 0 & 0 & -\frac{B_n}{\rho} & 0 & 0 \\ 0 & 0 & 0 & v_n & 0 & 0 & -\frac{B_n}{\rho} & 0 \\ 0 & 0 & 0 & 0 & v_n & 0 & 0 & 0 \\ 0 & B_{t1} & -B_n & 0 & 0 & v_n & 0 & 0 \\ 0 & B_{t2} & 0 & -B_n & 0 & 0 & v_n & 0 \\ 0 & \gamma p & 0 & 0 & 0 & 0 & 0 & 0 & v_n \end{bmatrix}. \tag{C.2}$$

The eigenvalues of the flux Jacobian in Eqn (C.2) yield expressions for the Alfvèn

wave speed and the fast and slow magneto-acoustic wave speeds

$$c_{A,n} = \frac{|B_n|}{\sqrt{\rho}} \tag{C.3}$$

$$c_{f,s}^2 = \frac{1}{2}\left(\frac{\gamma p + B^2}{\rho} \pm \sqrt{\left(\frac{\gamma p + B^2}{\rho}\right)^2 - 4\frac{\gamma p B_n^2}{\rho^2}}\right). \tag{C.4}$$

where B_n is the component of the magnetic field normal to the cell interface, and the fast magneto-acoustic speed c_f corresponds to the plus sign in front of the square root.

The eight normalized left and right eigenvectors are

Entropy wave: $\lambda_e = v_n$

$$r_e = \left[1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0\right]^T \tag{C.5}$$

$$l_e = \left[1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ -\tfrac{1}{c^2}\right] \tag{C.6}$$

Alfvén waves: $\lambda_a = v_n \pm c_{A,n}$

$$r_A^{\pm} = \left[0,\ 0,\ -\tfrac{\beta_{t2}}{\sqrt{2}},\ \tfrac{\beta_{t1}}{\sqrt{2}},\ 0,\ \pm\sqrt{\tfrac{\rho}{2}}\beta_{t2},\ \mp\sqrt{\tfrac{\rho}{2}}\beta_{t1}\right]^T \tag{C.7}$$

$$l_A^{\pm} = \left[0,\ 0,\ -\tfrac{\beta_{t2}}{\sqrt{2}},\ \tfrac{\beta_{t1}}{\sqrt{2}},\ 0,\ \pm\tfrac{\beta_{t2}}{\sqrt{2\rho}},\ \mp\tfrac{\beta_{t1}}{\sqrt{2\rho}}\right] \tag{C.8}$$

Fast magneto-acoustic waves: $\lambda_f = v_n \pm c_f$

$$r_f^{\pm} = \left[\rho\alpha_f,\ \pm\alpha_f c_f,\ \mp\alpha_s c_s \beta_{t1} sgn B_n,\ \mp\alpha_s c_s \beta_{t2} sgn B_n,\ 0,\ \alpha_s\sqrt{\rho}c\beta_{t1},\ \alpha_s\sqrt{\rho}c\beta_{t2},\ \alpha_f\gamma p\right]^T \tag{C.9}$$

$$l_f^{\pm} = \left[0,\ \pm\tfrac{\alpha_f c_f}{2c^2},\ \mp\tfrac{\alpha_s}{2c^2}c_s\beta_{t1}sgnB_n,\ \mp\tfrac{\alpha_s}{2c^2}c_s\beta_{t2}sgnB_n,\ 0,\ \tfrac{\alpha_s}{2\sqrt{\rho}c}\beta_{t1},\ \tfrac{\alpha_s}{2\sqrt{\rho}c}\beta_{t2},\ \tfrac{\alpha_f}{2\rho c^2}\right]^T \tag{C.10}$$

Slow magneto-acoustic waves: $\lambda_s = v_n \pm c_s$

$$r_f^{\pm} = \left[\rho\alpha_s,\ \pm\alpha_s c_s,\ \pm\alpha_f c_f \beta_{t1} sgn B_n,\ \pm\alpha_f c_f \beta_{t2} sgn B_n,\ 0,\ -\alpha_f\sqrt{\rho}c\beta_{t1},\ -\alpha_f\sqrt{\rho}c\beta_{t2},\ \alpha_s\gamma p\right]^T \tag{C.11}$$

$$l_f^{\pm} = \left[0,\ \pm\tfrac{\alpha_s c_s}{2c^2},\ \pm\tfrac{\alpha_f}{2c^2} c_f \beta_{t1} sgn B_n,\ \pm\tfrac{\alpha_f}{2c^2} c_f \beta_{t2} sgn B_n,\ 0,\ -\tfrac{\alpha_f}{2\sqrt{\rho}c}\beta_{t1},\ \tfrac{\alpha_f}{2\sqrt{\rho}c}\beta_{t2},\ \tfrac{\alpha_s}{2\rho c^2}\right]^T \tag{C.12}$$

Magnetic flux wave: $\lambda_d = v_n$

$$r_f = \left[0,\ 0,\ 0,\ 0,\ 1,\ 0,\ 0,\ 0\right]^T \tag{C.13}$$

$$l_f = \left[0,\ 0,\ 0,\ 0,\ 1,\ 0,\ 0,\ 0\right]. \tag{C.14}$$

In the formulas for the eigenvectors the following expressions have been used

$$\alpha_f^2 = \frac{c^2 - c_s^2}{c_f^2 - c_s^2} \qquad \alpha_s^2 = \frac{c_f^2 - c^2}{c_f^2 - c_s^2} \tag{C.15}$$

and

$$\beta_{t1} = \begin{cases} \frac{B_{t1}}{B_\perp} & \text{if } B_\perp \neq 0 \\ \frac{1}{\sqrt{2}} & \text{otherwise} \end{cases} \qquad \beta_{t2} = \begin{cases} \frac{B_{t2}}{B_\perp} & \text{if } B_\perp \neq 0 \\ \frac{1}{\sqrt{2}} & \text{otherwise} \end{cases}, \tag{C.16}$$

where $B_\perp = \sqrt{B_{t1}^2 + B_{t2}^2}$.

Appendix D

ROE AVERAGES OF THE MHD EQUATIONS

In this appendix two methods for obtaining Roe averages for the single fluid MHD equations are described. First method, partially described by Powell *et* al. [54], has two disadvantages. One of the disadvantages is that the eigenvalue corresponding to the Alfvén wave of the new flux Jacobian can become complex in certain situations. The other disadvantage is that the eigenvectors of the flux Jacobian cannot be normalized due to their complicated expressions. The second method due to Cargo and Gallice [14, 13, 15, 12] treats the one-dimensional MHD system of equations only ($B_x = const.$). Extension of their method to three dimensions yields complicated formulas for the fast and slow magnetosonic speeds and for the eigenvectors.

Last but not least a method proposed by Aslan [2, 3] to find Roe averages works only for the cases when the magnetic field is constant in the direction parallel to that in which the Riemann problem is solved. The method is described in detail by Aslan and is not reproduced here.

D.1 Roe Averages of Powell

Powell *et* al. [54] define the parameter vector

$$\mathbf{u} = \rho^{1/2}[1,\ v_x,\ v_y,\ v_z,\ B_x,\ B_y,\ B_z,\ h]^T, \tag{D.1}$$

where $\rho h = \gamma p/(\gamma - 1) + \rho(v_x^2 + v_y^2 + v_z^2)/2 + B_x^2 + B_y^2 + B_z^2$ is the enthalpy per unit volume. The Roe flux Jacobian $\tilde{A}$ for the ideal three-dimensional MHD equations (not reproduced here for brevity) was found by Powell. Its eigenvalues yield the Alfvén

speed (c_a) and the fast (c_f) and slow (c_s) magnetosonic speeds

$$c_a = \sqrt{\widehat{\left(\frac{B_x}{\rho}\right)}\tilde{B}_x}$$

$$c_{f,s}^2 = \frac{1}{2}\frac{B^2+\gamma\hat{p}}{\hat{\rho}} \pm \sqrt{\left(\frac{B^2+\gamma\hat{p}}{\hat{\rho}}\right)^2 - 4\frac{\tilde{B}_x}{\hat{\rho}}\left[\widehat{\left(\frac{B_x}{\rho}\right)}\gamma\hat{p} + \widehat{\left(\frac{B_y}{\rho}\right)}\beta_3 + \widehat{\left(\frac{B_z}{\rho}\right)}\beta_4\right]}$$

where

$$\beta_3 = (\gamma-1)\hat{\rho}\left[\tilde{B}_y\widehat{\left(\frac{B_x}{\rho}\right)} - \widehat{\left(\frac{B_xB_y}{\rho}\right)}\right]$$

$$\beta_4 = (\gamma-1)\hat{\rho}\left[\tilde{B}_z\widehat{\left(\frac{B_x}{\rho}\right)} - \widehat{\left(\frac{B_xB_z}{\rho}\right)}\right].$$

In the above equations $\hat{C}$ and $\tilde{C}$ represent two different averages of a quantity C. The Roe averages of the flow variables are

$$\hat{\rho} = \sqrt{\rho_L\rho_R}$$

$$\hat{v}_x = \frac{\sqrt{\rho_L}v_{x,L} + \sqrt{\rho_R}v_{x,R}}{\sqrt{\rho_L}+\sqrt{\rho_R}}$$

$$\hat{h} = \frac{\sqrt{\rho_L}h_L + \sqrt{\rho_R}h_R}{\sqrt{\rho_L}+\sqrt{\rho_R}}$$

$$\widehat{\left(\frac{B_x}{\rho}\right)} = \frac{\sqrt{\rho_R}B_{x,L} + \sqrt{\rho_L}B_{x,R}}{\hat{\rho}(\sqrt{\rho_L}+\sqrt{\rho_R})}$$

$$\widehat{\left(\frac{B_xB_x}{\rho}\right)} = \frac{\sqrt{\rho_R}B_{x,L}B_{x,L} + \sqrt{\rho_L}B_{x,R}B_{x,R}}{\hat{\rho}(\sqrt{\rho_L}+\sqrt{\rho_R})}$$

$$\tilde{B}_x = \frac{B_{x,L}+B_{x,R}}{2}$$

$$B^2 = \hat{\rho}\left[\tilde{B}_x\widehat{\left(\frac{B_x}{\rho}\right)} + \tilde{B}_y\widehat{\left(\frac{B_y}{\rho}\right)} + \tilde{B}_z\widehat{\left(\frac{B_z}{\rho}\right)}\right]$$

$$\hat{p} = \frac{\gamma-1}{\gamma}\hat{\rho}\left\{\hat{h} - \frac{\hat{u}^2+\hat{v}^2+\hat{w}^2}{2} - \frac{B^2}{\hat{\rho}} + \left[\tilde{B}_x\widehat{\left(\frac{B_x}{\rho}\right)} - \widehat{\left(\frac{B_xB_x}{\rho}\right)}\right]\right\}$$

with similar definitions for $\hat{v}$, $\hat{w}$, $\widehat{\left(\frac{B_y}{\rho}\right)}$, $\widehat{\left(\frac{B_z}{\rho}\right)}$, $\widehat{\left(\frac{B_xB_y}{\rho}\right)}$, $\widehat{\left(\frac{B_xB_z}{\rho}\right)}$, $\tilde{B}_y$, and $\tilde{B}_z$.

Powell and his co-workers did not publish the eigenvectors of $\tilde{A}$ so that they were calculated by hand and checked with Mathematica. The eigenvectors are

Entropy wave: $\lambda_e = \hat{v}_x$

$$r_e = \left[1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0\right]^T$$

$$l_e = \left[1,\ 0,\ 0,\ 0,\ 0,\ -\frac{\frac{\beta_3}{\hat{\rho}}}{c^2+\widehat{\left(\frac{B_y}{\rho}\right)}\frac{\beta_3}{\hat{\rho}}+\widehat{\left(\frac{B_z}{\rho}\right)}\frac{\beta_4}{\hat{\rho}}},\ -\frac{\frac{\beta_4}{\hat{\rho}}}{c^2+\widehat{\left(\frac{B_y}{\rho}\right)}\frac{\beta_3}{\hat{\rho}}+\widehat{\left(\frac{B_z}{\rho}\right)}\frac{\beta_4}{\hat{\rho}}},\ -\frac{\widehat{\left(\frac{B_x}{\rho}\right)}}{c^2+\widehat{\left(\frac{B_y}{\rho}\right)}\frac{\beta_3}{\hat{\rho}}+\widehat{\left(\frac{B_z}{\rho}\right)}\frac{\beta_4}{\hat{\rho}}}\right]$$

Magnetic flux wave: $\lambda_d = v_x$

$$r_d = \left[0,\ 0,\ 0,\ 0,\ 1,\ 0,\ 0,\ 0\right]^T$$

$$l_d = \left[0,\ 0,\ 0,\ 0,\ 1,\ 0,\ 0,\ 0\right]$$

Alfvén wave: $\lambda_a = \hat{v}_x \pm c_a$

$$r_A^{\pm} = \begin{bmatrix} 0 \\ 0 \\ -\left[\tilde{B}_z + \frac{\beta_4}{\hat{\rho}\widehat{\left(\frac{B_x}{\rho}\right)}}\right] \\ \tilde{B}_y + \frac{\beta_3}{\hat{\rho}\widehat{\left(\frac{B_x}{\rho}\right)}} \\ 0 \\ \pm c_a \frac{\hat{\rho}}{\tilde{B}_x}\left[\tilde{B}_z + \frac{\beta_4}{\hat{\rho}\widehat{\left(\frac{B_x}{\rho}\right)}}\right] \\ \mp c_a \frac{\hat{\rho}}{\tilde{B}_x}\left[\tilde{B}_y + \frac{\beta_3}{\hat{\rho}\widehat{\left(\frac{B_x}{\rho}\right)}}\right] \\ \pm\frac{\tilde{B}_z\beta_3 - \tilde{B}_y\beta_4}{c_a} \end{bmatrix} \qquad l_A^{\pm} = \begin{bmatrix} 0 \\ 0 \\ -\left[\tilde{B}_z + \frac{\beta_4}{\hat{\rho}\widehat{\left(\frac{B_x}{\rho}\right)}}\right] \\ \tilde{B}_y + \frac{\beta_3}{\hat{\rho}\widehat{\left(\frac{B_x}{\rho}\right)}} \\ 0 \\ \pm\frac{c_a}{\hat{\rho}\widehat{\left(\frac{B_x}{\rho}\right)}}\left[\tilde{B}_z + \frac{\beta_4}{\hat{\rho}\widehat{\left(\frac{B_x}{\rho}\right)}}\right] \\ \mp\frac{c_a}{\hat{\rho}\widehat{\left(\frac{B_x}{\rho}\right)}}\left[\tilde{B}_y + \frac{\beta_3}{\hat{\rho}\widehat{\left(\frac{B_x}{\rho}\right)}}\right] \\ 0 \end{bmatrix}^T$$

Fast and slow magnetosonic waves: $\lambda_{f,s} = \hat{v}_x \pm c_{f,s}$

$$r^{\pm}_{f,s} = \begin{bmatrix} \hat{\rho} \\ \pm c_{f,s} \\ \mp \frac{c_{f,s}\tilde{B}_x \widehat{\left(\frac{B_y}{\rho}\right)}}{c^2_{f,s}-c^2_a} \\ \mp \frac{c_{f,s}\tilde{B}_x \widehat{\left(\frac{B_z}{\rho}\right)}}{c^2_{f,s}-c^2_a} \\ 0 \\ \frac{c_{f,s}\hat{\rho}\widehat{\left(\frac{B_y}{\rho}\right)}}{c^2_{f,s}-c^2_a} \\ \frac{c_{f,s}\hat{\rho}\widehat{\left(\frac{B_z}{\rho}\right)}}{c^2_{f,s}-c^2_a} \\ \gamma\hat{p} - \beta_3 \frac{\tilde{B}_x \widehat{\left(\frac{B_y}{\rho}\right)}}{c^2_{f,s}-c^2_a} - \beta_4 \frac{\tilde{B}_x \widehat{\left(\frac{B_z}{\rho}\right)}}{c^2_{f,s}-c^2_a} \end{bmatrix} \quad l^{\pm}_{f,s} = \begin{bmatrix} 0 \\ \pm\hat{\rho}c_{f,s} \\ \frac{\mp c_{f,s}\hat{\rho}\tilde{B}_y \widehat{\left(\frac{B_x}{\rho}\right)} \mp \beta_3 c_{f,s}}{c^2_{f,s}-c^2_a} \\ \frac{\mp c_{f,s}\hat{\rho}\tilde{B}_z \widehat{\left(\frac{B_x}{\rho}\right)} \mp \beta_4 c_{f,s}}{c^2_{f,s}-c^2_a} \\ 0 \\ \frac{c^2_{f,s}\tilde{B}_y - \beta_3 \frac{c^2_{f,s}}{\hat{\rho}\widehat{\left(\frac{B_x}{\rho}\right)}}}{c^2_{f,s}-c^2_a} \\ \frac{c^2_{f,s}\tilde{B}_z - \beta_4 \frac{c^2_{f,s}}{\hat{\rho}\widehat{\left(\frac{B_x}{\rho}\right)}}}{c^2_{f,s}-c^2_a} \\ 1 \end{bmatrix}^T \tag{D.2}$$

It can be easily checked that these eigenvectors correspond to those for the basic (non Roe-averaged) flux Jacobian by making $\widehat{\left(\frac{B_x}{\rho}\right)} = \frac{B_x}{\rho}$, $\tilde{B}_x = B_x$, and similarly for the other averages, and $\beta_3 = \beta_4 = 0$.

The eigenvectors shown above are not normalized and are prone to scaling problems in a few cases, as shown in Section 3.2.3. The structure of the new eigensystem is complicated and does not allow the same elegant constructions as in the non Roe-averaged case. For example, the relationship between the eigenavalues yielding the fast and slow magnetosonic wavespeeds are

Non-Roe averaged

$$(v^2 - c^2)(v^2 - c^2_{A,x}) = v^2 \frac{B^2_\perp}{\rho}, \qquad B^2_\perp = B^2_y + B^2_z,$$

Roe averaged

$$(v^2 - c^2)(v^2 - c^2_{A,x}) = v^2 \frac{B^2_\perp}{\hat{\rho}} + P, \qquad B^2_\perp = \hat{\rho}\left[\tilde{B}_y \widehat{\left(\frac{B_y}{\rho}\right)} + \tilde{B}_z \widehat{\left(\frac{B_z}{\rho}\right)}\right],$$

where $v = c_{f,s}$, and $P = \tilde{B}_x \widehat{\left(\frac{B_y}{\rho}\right)} \frac{\beta_3}{\hat{\rho}} + \tilde{B}_x \widehat{\left(\frac{B_z}{\rho}\right)} \frac{\beta_4}{\hat{\rho}}$.

Since these complications do not allow the construction of a robust algorithm a different method for finding the Roe averages was sought.

D.2 Roe Averages of Cargo and Gallice

Cargo and Gallice [14, 13] found a flux Jacobian $\tilde{A}$ for the one dimensional MHD equations ($B_x = const.$). They used the primitive variables vector as the parameter vector

$$\mathbf{u} \equiv \mathbf{w} = [\rho,\ v_x,\ v_y,\ v_z,\ B_x,\ B_y,\ B_z,\ p]^T. \tag{D.3}$$

Their work has been extended to the three-dimensional MHD equations and a new Roe-averaged flux Jacobian was obtained. The Roe averaged flux Jacobian is

$$\tilde{A} = \begin{bmatrix} \overline{v}_x & \underline{\rho} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{(1-\gamma')X-2r_x}{\underline{\rho}} & \overline{v}_x & 0 & 0 & 0 & \frac{B_y}{\underline{\rho}} & \frac{B_z}{\underline{\rho}} & \frac{1}{\underline{\rho}} \\ -\frac{\alpha}{\underline{\rho}} & 0 & \overline{v}_x & 0 & 0 & -\frac{B_x}{\underline{\rho}} & 0 & 0 \\ -\frac{\beta}{\underline{\rho}} & 0 & 0 & \overline{v}_x & 0 & 0 & -\frac{B_x}{\underline{\rho}} & 0 \\ 0 & 0 & 0 & 0 & \overline{v}_x & 0 & 0 & 0 \\ 0 & \underline{B_y} & -\underline{B_x} & 0 & 0 & \overline{v}_x & 0 & 0 \\ 0 & \underline{B_z} & 0 & -\underline{B_x} & 0 & 0 & \overline{v}_x & 0 \\ 0 & \gamma P + \gamma'(\rho X - 2Rr_x) & 0 & -\gamma'\alpha R & -\gamma'\beta R & 0 & 0 & 0 \quad \overline{v}_x \end{bmatrix}, \tag{D.4}$$

where $\gamma' = \gamma - 1$ and the averages of a quantity C ($\overline{C}$ and $\underline{C}$) are defined as

$$\overline{C} = \frac{\sqrt{\rho_L}C_L + \sqrt{\rho_R}C_R}{\sqrt{\rho_L} + \sqrt{\rho_R}} \tag{D.5}$$

$$\underline{C} = \frac{\sqrt{\rho_R}C_L + \sqrt{\rho_L}C_R}{\sqrt{\rho_L} + \sqrt{\rho_R}}. \tag{D.6}$$

The other quantities shown in $\tilde{A}$ are defined as

$$X = \frac{1}{2}\frac{(\Delta B_x)^2 + (\Delta B_y)^2 + (\Delta B_z)^2}{(\sqrt{\rho_L} + \sqrt{\rho_R})^2} \tag{D.7}$$

$$r_x = \frac{1}{2}\left(\frac{\Delta B_x}{\sqrt{\rho_L} + \sqrt{\rho_R}}\right)^2 \tag{D.8}$$

$$\alpha = \frac{\Delta B_x \Delta B_y}{(\sqrt{\rho_L} + \sqrt{\rho_R})^2} \tag{D.9}$$

$$\beta = \frac{\Delta B_x \Delta B_z}{(\sqrt{\rho_L} + \sqrt{\rho_R})^2} \tag{D.10}$$

$$R = \sqrt{\rho_L \rho_R} \tag{D.11}$$

$$P = \frac{\gamma - 1}{\gamma}\underline{\rho}\{\overline{h} - \frac{1}{2}(\overline{v}_x^2 + \overline{v}_y^2 + \overline{v}_z^2) - \frac{1}{\underline{\rho}}\left[(\underline{B}_x)^2 + (\underline{B}_z)^2 + (\underline{B}_z)^2\right] - X\}, \tag{D.12}$$

where $\Delta B_x = B_{x,L} - B_{x,R}$ for example.

The expression for the Alfvén speed is

$$c_{A,x} = \sqrt{\frac{\underline{B}_x(\underline{B}_x + \overline{B}_x)}{2\underline{\rho}}}. \tag{D.13}$$

The expressions for the fast and slow magnetosonic speeds are

$$c_{f,s}^2 = \frac{1}{2}(c^2 + c_A^2) - r_x\left[(\gamma - 1)\frac{R}{\underline{\rho}} + 1\right] \pm \Bigg\{(c^2 + c_A^2)^2 - 4c^2 c_{A,x}^2 + 4r_x^2\left[(\gamma - 1)\frac{R}{\underline{\rho}} + 1\right]^2 +$$

$$4\left[(\gamma - 1)\frac{R}{\underline{\rho}} + 1\right]\left[r_x(c_{A,x}^2 - c_{A,y}^2 - c_{A,z}^2 - c^2) + c_{A,x}(\alpha c_{A,y} + \beta c_{A,z})\right] + 4(\gamma - 1)(\alpha^2 + \beta^2)\frac{R}{\rho}\Bigg\}^{\frac{1}{2}}, \tag{D.14}$$

where

$$c^2 = \gamma\frac{P}{\underline{\rho}} + X$$

$$c_A^2 = c_{A,x}^2 + c_{A,y}^2 + c_{A,z}^2$$

and $c_{A,y}$ and $c_{A,z}$ are calculated similarly to $c_{A,x}$.

Due to these expressions of the flux Jacobian and of the signal speeds it is extremely complicated to derive the eigenvectors. Computation of the eigenvectors has been attempted and it was abandoned when it was obvious that there are no benefits in completing it, since the normalization would have been impossible.

Appendix E

AXISYMMETRIC SOURCE TERMS

The purpose of this appendix is to describe the modifications to the system of PDEs that allow axisymmetric simulations. An axisymmetric simulation models problems that have axial symmetry. The coordinate system that characterizes such problems is polar cylindrical. The assumption made in deriving the axisymmetric model is that although out of plane conserved variables are allowed their azimuthal variation (ϕ direction) is neglected. The modifications to the system of PDEs allow the use of two-dimensional grids that span the Orz plane to be used for these type of simulations. These modifications account for the radial variation of cell volumes and for the cancellation of some of the fluxes present in the three-dimensional model so that no azimuthal variation of the conserved variables is allowed. They are implemented using a source term that is added to the right hand side of the system of conservation laws.

It has to be emphasised that both the grid that discretizes the domain of interest and the flow should have axial symmetry for the axisymmetric assumptions to be valid. To illustrate, the flow over an axisymmetric body (e.g. a cone) at zero angle of attack and no sideslip is an example of an axisymmetric problem. If the flow has side-slip or the body has an angle of attack the flow is no longer axisymmetric.

Since WARP3 is written so that the conserved variables are stored in a Cartesian reference frame a mapping is used for axisymmetric simulations. Thus the radial direction and vector components are mapped to the Ox direction and components and the axial direction and vector components are mapped to the Oy direction. The out of plane (azimuthal) components are mapped to negative Oz direction. To summa-

rize the mapping can be symbolically represented as $(r, \phi, z) \rightarrow (x, -z, y)$. Regular two dimensional grids (with one cell in the Oz direction) are used for axisymmetric simulations with size of the cell in the Oz direction equal to **unity**.

The original (three dimensional) system of PDEs in Eqn (2.62) is modified so that it becomes

$$\frac{\partial \mathbf{Q}}{\partial t} + \nabla \cdot \overleftrightarrow{f} = \mathbf{S}_a \tag{E.1}$$

where $\mathbf{S}_a$ is the axisymmetric source term derived such that

$$\nabla \cdot \overleftrightarrow{f} = (\frac{\partial}{\partial x} + \frac{\partial}{\partial y})\overleftrightarrow{f} - \mathbf{S}_a, \tag{E.2}$$

where the mapping notation described above has been used.

After some algebra the expression of the source term for axisymmetric flows is

$$\mathbf{S}_a = -\frac{1}{x}\begin{bmatrix} \rho v_x \\ \rho(v_x^2 + v_z^2) - B_x^2 - B_z^2 \\ \rho v_x v_y - B_x B_y \\ 2\rho v_x v_z - B_x B_z \\ 0 \\ v_x B_y - v_y B_x \\ 0 \\ (e + p + \frac{1}{2}B^2)v_x - \mathbf{B} \cdot \mathbf{v} B_x \end{bmatrix} \tag{E.3}$$

Note the inverse radial (x) dependency of the source term.

Appendix F

METRICS CALCULATION

In this appendix a rapid and accurate method for calculating the centroid, volume and face area vectors of a cell is presented. It is an adaptation of a more general method described by Messner and Taylor [47] for calculating mass properties of polyhedra.

A cell is a hexahedron (six-face polyhedron) as shown in Figure F.1. For each cell, the centroid position, volume and face area vectors are needed in the evaluation of fluxes. The formula giving the volume of a solid body is

$$V = \int_V d\tau \tag{F.1}$$

where $d\tau$ is an infinitesimally small volume element, and the formulae giving the position of the centroid are

$$x_C = \frac{\int_V x d\tau}{V},\ y_C = \frac{\int_V y d\tau}{V},\ z_C = \frac{\int_V z d\tau}{V}. \tag{F.2}$$

The face area vectors are vectors normal to the faces of the cell with the magnitude equal to the area of the face. The sign convention for these vectors is that their components point in the positive direction of the coordinate system. An efficient calculation of the volume and centroid position can be performed by lowering the order of the volume integrals in Eqn (F.1) and Eqn (F.2) and using a quadrature formula to calculate the value of the surface integrals.

Gauss' theorem $\int_V \nabla \cdot \mathbf{F} d\tau = \int_S \mathbf{F} \cdot d\boldsymbol{\sigma}$ is applied in order to lower the order of the volume integrals. S is the surface bounding volume V, $\mathbf{F}$ is a vector valued function and $d\boldsymbol{\sigma}$ is an infinitesimally small surface element.

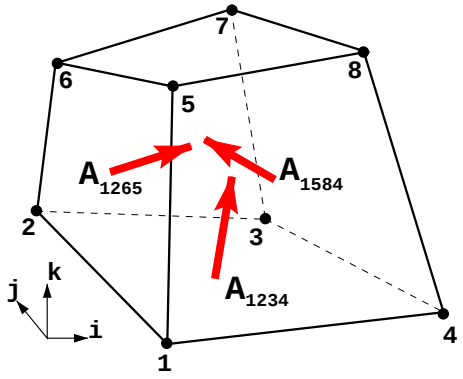

Figure F.1: Sketch of a cell showing vertex notation convention and face area vectors

For volume calculation a vector valued function is needed such that

$$\nabla \cdot \mathbf{F}_V = 1 \tag{F.3}$$

$\mathbf{F}_V$ can be easily obtained by noticing that $\nabla \cdot \mathbf{r} = 3$ where $\mathbf{r} = x\ \mathbf{i} + y\ \mathbf{j} + z\ \mathbf{k}$ is the position vector. By making $\mathbf{F}_V = \frac{1}{3}\mathbf{r}$ and replacing Eqn (F.3) in Eqn (F.1) the following surface integral is obtained

$$V = \frac{1}{3}\int_S \mathbf{r} \cdot d\boldsymbol{\sigma} = \frac{1}{3}\int_S (xn_x + yn_y + zn_z)d\sigma \tag{F.4}$$

where n_x, n_y and n_z are the components of the unit face area vector.

Similarly for the centroid calculation the vector valued function should give for the x coordinate, for example, $\nabla \cdot \mathbf{F}_{x_C} = x$. The function satisfying this relation is given by $\mathbf{F}_{x_C} = \frac{1}{2}x^2\mathbf{i}$. Applying Gauss' theorem yields

$$x_C = \frac{1}{2V}\int_S x^2\mathbf{i} \cdot d\boldsymbol{\sigma} = \frac{1}{2V}\int_S x^2 n_x d\sigma \tag{F.5}$$

and similarly, for the y and z coordinates of the centroid

$$y_C = \frac{1}{2V}\int_S y^2 n_y d\sigma \tag{F.6}$$

$$z_C = \frac{1}{2V}\int_S z^2 n_z d\sigma \tag{F.7}$$

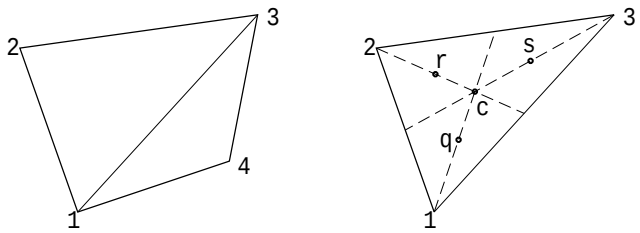

Figure F.2: A face is divided into two triangles. Points used in the quadrature rule are shown (c, q, r and s).

Integration in Eqn (F.4) to Eqn (F.7) is performed over the six faces of the cell. Each face is divided in two triangles (Figure F.2) so that a four-point quadrature rule can be applied. The quadrature rule in Eqn (F.8) described by Stroud [70] converts a continuous surface integral into a sum of four discrete values of the function under the integral, evaluated at specific points in the triangle. It is exact for polynomials up to the third order.

$$\int\limits_{S_{triangle}} f\,d\sigma = A_{triangle}[w_1 f(p_c) + w_2 f(p_q) + w_3 f(p_r) + w_4 f(p_s)] \tag{F.8}$$

Point p_c is the centroid of the triangle and p_q, p_r and p_s are located along lines from the centroid to the vertices such that the distance from the centroid to these points is 40 % of the distance from the centroid to the respective vertex. The weights are $w_1 = -\frac{27}{48}$ and $w_2 = w_3 = w_4 = \frac{25}{48}$.

Applied to the volume calculation the quadrature rule gives

$$\begin{aligned} V = &\sum_{i=1}^{12} A_i n_x [w_1 x_c + w_2 x_q + w_3 x_r + w_4 x_s] \\ &+ \sum_{i=1}^{12} A_i n_y [w_1 y_c + w_2 y_q + w_3 y_r + w_4 y_s] \\ &+ \sum_{i=1}^{12} A_i n_z [w_1 z_c + w_2 z_q + w_3 z_r + w_4 z_s] \end{aligned} \tag{F.9}$$

where A_i is the area of a triangle and the sum is taken over the triangles making the faces. It has to be noted that $A_i n_x$, $A_i n_y$ and $A_i n_z$ are the components of face area vectors for each triangle and are calculated using Eqn (F.10)

$$A_i n_x = \begin{vmatrix} y_1^i & z_1^i & 1 \\ y_2^i & z_2^i & 1 \\ y_3^i & z_3^i & 1 \end{vmatrix}, \; A_i n_y = \begin{vmatrix} z_1^i & x_1^i & 1 \\ z_2^i & x_2^i & 1 \\ z_3^i & x_3^i & 1 \end{vmatrix}, \; A_i n_z = \begin{vmatrix} x_1^i & y_1^i & 1 \\ x_2^i & y_2^i & 1 \\ x_3^i & y_3^i & 1 \end{vmatrix}. \tag{F.10}$$

Appendix G

TREATMENT OF NON-SIMPLY CONNECTED GRIDS

This section presents the capability of WARP3 to handle non-simply connected grids. In a non-simply connected grid some of the blocks are arranged so that three blocks, instead of four, share a common edge. Such types of grids were employed for the spheromak simulations and are generally used in simulations where cylindrical domains are discretized such that singularities at the axis are avoided. An example is given in Figure G.1 where only the boundaries of the block are outlined. It was mentioned in Section 4.1 that the exchange of data between two blocks that have a common face is implemented through passing of the values of the conserved variables from the two layers of cells next to the face into the two layers of internal of ghost cells next to the same face. It the blocks reside on the same processor than the data exchange is performed using array copy operations otherwise data is exchanged using derived data types and message passing.

As illustrated in Figure G.1 only if the blocks have a common face at opposite cardinal points the layers of cells that get passed from one block to the other are oriented properly. For example blocks **1** and **2** have a common face that corresponds to East in block **1** and West in block **2**. Data exchanged between blocks **1** and **2** has the correct orientation in the Ojk plane. However blocks **2** and **5** have a common face that corresponds to North in block **1** and North in block **5**. Data exchanged between blocks **2** and **5** needs special treatment because the order in which the cells are stored in block **2** is opposite to the direction in block **5**.

The treatment is implemented in WARP3 in a single step performed after each data exchange between blocks using array copy operations. After all data has been

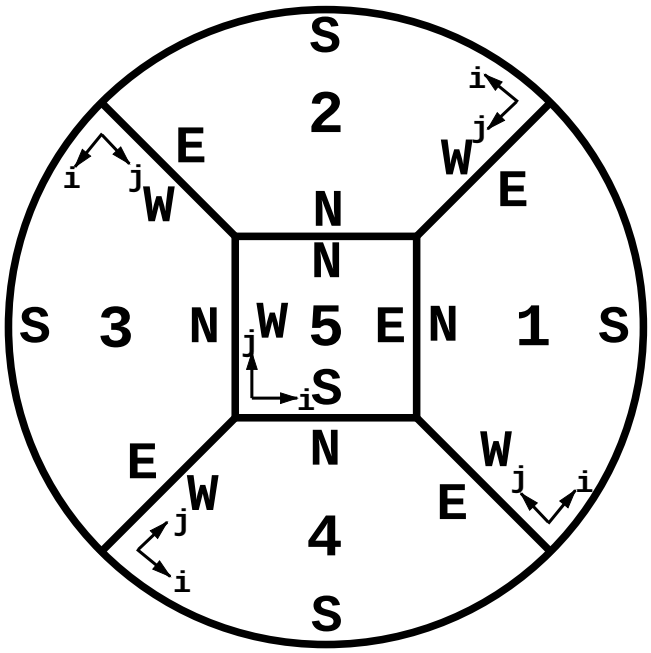

Figure G.1: Top view of a non-simply connected grid used in discretization of domains of circular cross section. Block numbers, grid orientation and cardinal points are shown. Note that any three blocks have a common edge.

Table G.1: Table showing pairs of faces where the exchanged data has to be rearranged to match the ij ordering.

Sender face	Receiver face			
	North (1)	East (2)	South (3)	West (4)
North (1)	TRUE	FALSE	FALSE	TRUE
East (2)	FALSE	TRUE	TRUE	FALSE
South (3)	FALSE	TRUE	TRUE	FALSE
West (4)	TRUE	FALSE	FALSE	TRUE

received from the neighbors a procedure reverses the order of the data in the internal ghost cells according to a truth table shown in G.1. The procedure that implements reordering is simpler that what it seems to be at first inspection. In Table G.1 next to the face labels the ID numbers of the faces are given. It can be noticed that the ordering of the ghost cell faces has to be changed only in the cases when data is passed to a like face ($ID_{recv} = ID_{send}$) and when the sum of the IDs is equal to five ($ID_{recv} + ID_{send} = 5$). In WARP3 the faces are labeled by their IDs and in the procedure that implements the reordering another procedure finds the ID of the sending face and checks the above relationships.

Appendix H

WARP3 USER'S GUIDE

This appendix intends to be a user guide for WARP3. It assumed that the users have a basic familiarity with the UNIX operating system and with the Fortran programming language. The appendix starts with the description of the input file for a quick reference and continues with a pseudo code description of WARP3. The last section presents a table with all the main variables used by WARP3.

H.1 Code Organization

This section gives a description of the program flow and presents the main procedures used by WARP3. Pseuducode rather than source code listings are given in order to make abstraction of the programming language and hopefully make the code easy to understand.

It was found that passing the input file name as an argument to the code is necessary when multiple "production" runs are started on a system that uses queuing, such as the IBM SP2. Since command line arguments are not a standard Fortran90 feature the main program is a standard ANSI C function called `main.c`. The main program calls `warp3.f` with a single argument, the name of the input file. The Fortran procedure `warp3.f` calls the main procedure of WARP3 such as the procedures that read in the grid file and the procedures that evaluate the hyperbolic and parabolic fluxes and the source terms.

All the procedures and functions of WARP3 have a fair amount of comments and their argument lists were written using the Fortran90 attribute `INTENT` that specifies

which of the arguments is an input argument and which is an output argument. The `INTENT` attribute not only makes the code clearer to read but it enhances the robustness of the code since the compiler can identify an erroneous line of code that attempts to assign a value to a variable that has been declared an input variable (with `INTENT(IN)` only. More that that in some cases (depending on the compiler) using this attribute improves code performance by optimizing the alignment of the variables in memory. Following is pseudo code listing of `warp3.f` that describes the main sequence of calls and gives some details about each main procedure.

Since the procedure that performs the implicit advance of the solution is rather large and complex its pseudo code description is also given.

```
** warp3.f starts here **
JobInit
   - calls MPI initializations and enrolls the job as an MPI job
   - it returns the task (or processor) ID and total number
     of processors

InputLoad
   - task 0 reads the input file from disk and uses MPI to send the
     data to the co-workers
   - NOTE: if new variables are added to the input file this
     procedure has to be modified accordingly

FindHosts
   - finds out the name of the hosts and writes them to a file
     specified in the input

JobCheck
```

```
   - checks the number processors assigned to job against the
     number of processors desired by user. Job stops if they are
     not equal

BlockInfo
   - finds the local to global mapping and determines the maximum
     array sizes that are used for the dynamic memory allocation

FlipFind
   - finds the truth table for non-simply connected grids

AllocSp
   - allocates memory for the run time sized arrays
   - if there is not sufficient memory the code stops and
     the procedure returns with an error message that shows
     where the memory allocation stopped

select(gridfile_style)
"tecplot" : TecPlotGridLoad
   - loads a grid file that uses Tecplot ASCII format
"fbin"   : FBinGridLaod
   - loads a grid file that uses Fortran binary format
       end

Geometry
   - calculates cell volumes, centroid positions, face
     area vectors and cell centroid to face center distances
```

```
Init
   - initializes the conserved variables and the viscosity
     and resistivity arrays. The plain vanilla initialization
     procedure only assigns the values that were input by the
     user to the conserved variables in each block. For application
     specific initializations this procedure has to be modified

if (restart) then

   ReadRst
      - reads in the restart file

else

   while (t<=tfin) and (icount < ncycfin)

      if (icount=0) and implicit then
         courant = 0.85
      else
         courant = courant_phys
      end
            - if this is the first iteration and the
              implicit mode is selected use a Courant
              number of 0.85 since the first time step
              is first-order explicit

      if (icount=0) or (not implicit) then
         - first step of the implicit mode is taken with the
```

```
  first order in time accurate explicit algorithm

BC
   - calls the boundary conditions subroutines

FaceSend
FaceRecv
BdryWait
BdryWait
   - performs the first set of data exchange between
     blocks

if (resistive or viscous) then
   EdgeSend
   EdgeRecv
   BdryWait
   BdryWait
end
   - if the parabolic terms are on perform a second
     set of message exchanges

FlipFaces
   - change the ordering of the internal ghost cells
     according to the truth table previously calculated
     (only on non-simply connected grids)

if (icount=0)
   select(dump_style)
```

```
   "tecplot" : TecPlotDump
      - writes plot files to disk using Tecplot ASCII format
   "fbin" : PrimBinDump
      - writes plot files to disk using Fortran binary format
   end
end

TimeStep
   - computes the time step

RiemannSolver
RiemannSolver
   - computes the hyperbolic fluxes for the x and y directions

if (axisymmetric) then
   h = ZERO
else
   RiemannSolver
end
   - if the simulation is three dimensional then it computes
     the fluxes in the z direction

DivBSource
   - computes source term proportional to the divergence of
     the magnetic field

ResistiveFlux
   - computes the parabolic flux due to resistivity
```

```
   ViscousFlux
      - computes the parabolic flux due to viscosity

   AxiSymmetricSource
      - computes the axisymmetric source terms

   dqCalc
      - computes the increment of primitive variables

   UpdateSol
      - updates the solution (by adding dQ to Q)

   Converge
      - calculates the two-norm of the residual

else
   - implicit scheme starts here

   qnm1 = qn
   qn = q
      - advances the time levels (qn was initialized in Init)

   PTimeLoop
      - performs m pseudo time loops. See the pseudo code description
        of this procedure below

   TimeStep
```

```
            - calculates the time step

         t = t + dt
         icount = icount + 1
            -increments time and iteration counter

         if (mod(t,tplt_dump)<=dt) or (mod(icount,ncycplt_dump)=0) then

            select(dump_style)
            "tecplot" : TecPlotDump
            "fbin" : PrimBinDump
               - dumps plot files every tplt_dump time units or
                 every ncycplt_dump iterations
         end

   end   - main loop ends here

end      - restart if statement ends here

* warp3.f ends here **
```

Following is a pseudo code description of the implicit time stepping procedure called `PTimeLoop` for Pseudo Time Loop. Most of the procedures called inside `PTimeLoop` are the same with those called in the main procedure `warp3` and for the sake of brevity some of the obvious calls are not commented.

```
** PTimeLoop starts here **

- allocates memory for the local dynamic arrays
```

```
while (m<=ncyc_pseudo)

   BC

   FaceSend
   FaceRecv
   BndryWait
   BndryWait

   EdgeSend
   EdgeRecv
   BndryWait
   BndryWait
      - since corner points are needed to calculate Jacobians the
        edges are exchanged even if the parabolic terms are off

   FlipFaces

   RiemannSolver
   RiemannSolver

   if (not axisymmetric) then
      RiemannSolver
   endif

   ComplexFluxJacobian
   ComplexFluxJacobian
```

```
   - computes flux Jacobians in the x and y directions

if (not axisymmetric then
   ComplexFluxJacobian
end
   - if the simulation is three dimensional then it
     computes the flux Jacobians in the z direction
   - NOTE: the procedures that compute the flux Jacobians using
           a limit formulation are also available

DivBSource

if (resistive) then

   ResistiveFlux

   LimitResistFluxJacobian

end
   - if the parabolic resistive terms are on then compute the resistive fluxes
     and their Jacobian (only a limit resistive flux Jacobian procedure
     is available)

if (viscous) then

   ViscousFlux

   LimitViscFluxJacobian
```

```
   end
      - if the parabolic resistive terms are on then compute the resistive fluxes
        and their Jacobian (only a limit viscous flux Jacobian procedure is
        available)

   TimeStep
      - computes the pseudo time step

   dtstar_inv = ONE / dt_star

   SymmetricGaussSeidel
      - calls the procedure that finds the solution to the large sparse
        (block heptadiagonal) system of linear equations that results from
        the implicit discretization

        UpdateSol

   m = m + 1

end      - pseudo time loop ends here

Converge

- deallocates memory used for dynamics arrays

** PTimeLoop ends here **
```

H.2 Input File Structure

In this section the structure of the input file is discussed using a spheromak simulation input file as an example. (Sometimes input files are called input decks by the people who used punch cards in their youth.)

Inputs to WARP3 are read into the code from Fortran `namelists` stored in an ASCII file. The input file is divided into seven main namelists. They are required to be present in each input file and are discussed in detail below. The user can add namelists to the input file if required by the application.

1. `sizes` - contains the sizes of the blocks in the grid in number of real cells in the i, j and k directions of each block.

2. `controls` - contains the inputs necessary to control WARP3 such as application name, diagnostic procedure switch, grid file format, grid file name and directory, etc.

3. `blocks` - contains the total number of blocks, the topology description of the grid and the two sets of boundary conditions: magnetic and hydrodynamic (fluid)

4. `algorithm` - contains the inputs needed to control the flow of the algorithm and constants used in computations

5. `physics` - contains physics related inputs such as the ratio of heat capacities, the (inverses) of the reference Reynolds and Lundquist numbers and the switches that toggle the calls of the parabolic flux calculations

6. `globaldata` - contains the values of the primitive variables that the domain is initialized with

7. `localdata` - contains the values of the primitive variables that are used to initialize individual blocks

Each of the namelists mentioned above are described below in detail together with an application specific namelist.

• `sizes` **namelist** This namelist contains three integer one-dimensional arrays that store the number of cells in each block. The arrays are statically dimensioned with MAX_BLOCKS. In the example shown here, which is a low resolution test case, the total number of blocks is twelve and each block has 4 cells in the i and j directions and 10 cells in the k direction.

```
&sizes
 icels =  4,  4,  4,  4,  4,  4,  4,  4,  4,  4,  4,  4,
 jcels =  4,  4,  4,  4,  4,  4,  4,  4,  4,  4,  4,  4,
 kcels = 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
/
```

• `controls` **namelist** This namelist contains all the main controls that tell WARP3 what and from where to read data and where to write data, when to write data and restart files and where. The variable `application_name` contains the identifier of a specific application, in this case the application is a spheromak simulation. `run_diagnostic` is a logical variable that tell WARP3 to call or not a diagnostic procedure. Some applications require a diagnostic procedure and the user might want to turn off the diagnostic procedure to speedup the simulation. `gridfile_style` tells WARP3 what type of grid file it should read from the disk. WARP3 can only read two types of files. First type of grid file format is a Tecplot ASCII format. This format is preferred since the grid file can be directly read into Tecplot and checked. The second format is Fortran binary which allows generation of smaller grid files than Tecplot ASCII. The same formats (or styles) can be specified for the WARP3 output files. The Tecplot ASCII files can be concatenated and post-processed into Tecplot

binary files using a c-shell script. A similar script can be used for Fortran binary output files only that it involves a couple of intermediate steps. `ncycplt_dump` is the number of outer loop (physical time) iterations after which a data dump is performed. The alternative is to specify the dimensionless time intervals when data dumps are performed with `tplt_dump`. Both `ncycplt_dump` and `tplt_dump` are initialized by default with very large numbers so if one of them is not specified in the input file the variable specified will be the one used for data dumps. Similarly the restart data dumps are performed either after each `ncycrst_dump` iterations or `trst_dump` dimensionless time units. `grid_dir` and `grid_file` are strings that store the name of the directory where the grid file resides and the name of the grid file respectively. `out_dir` and `rst_dir` are used to pass the name of the directory where the output and restart data files will be written. Obviously these directories have to exist on the UNIX file system and the user has to take care to create them (if they do not exist) before running the code. `hosts_file` passes to WARP3 the name of the file where the name of the hosts that the job runs on are written. The hosts file is needed by a script that kills WARP3 jobs running on these hosts automatically.

```
&controls
 application_name = "spheromak"
 run_diagnostic = .true.
 gridfile_style = "tecplot"
 dump_style = "tecplot"
 ncycplt_dump = 25,
 ncycfin = 250,
 trst_dump = 5.0d1,
 grid_dir = "/mhd4/udrea/grids/smak"
 grid_file = "smak.lores.tec"
 out_dir = "/mhd4/udrea/results/smak/cylinder/ar3/tests"
```

```
 rst_dir = "/mhd4/udrea/results/smak/cylinder/ar3/tests"
 hosts_file = "/mhd4/udrea/results/smak/cylinder/ar3/tests/smak.hosts"
/
```

• `blocks` **namelist** This namelist is used to pass the information needed by WARP3 to map the block positions in the grid and also the information about the boundary conditions that should be applied at the boundaries of the domain. The total number of blocks in the grid is specified in the variable `nblock_tot`. Variable `block_topo` stores the topology of the grid, i.e. the relationship of one block to its neighbors. `block_topo` is a two dimensional integer array declared as `block_topo(7,MAX_BLOCKS)`. The block IDs of the neighbors are stored in the first six positions of each row and the seventh position stores the ID of the processor or task to which the respective block is placed. The order in which the neighbor block IDs are specified is $North(j+)$, $East(i+)$, $South(j-)$, $West(i-)$, $Top(k+)$ and $Bottom(k-)$, where in parentheses the index directions have been indicated. If there are no neighbors in a certain direction a -1 is specified instead. This tells WARP3 that a boundary condition (BC) should be applied at that face. Using the example shown here block 1 has block 2 at the $North$, a BC at $East$, block 8 at $South$, block 9 at $West$ and BCs at the Top and $Bottom$. Block 1 is assigned to processor 0. (Following the MPI notation the processor (task) IDs start at 0 and end at $p-1$ where p is the number of processors assigned to the parallel job.)

The boundary conditions are passed to WARP3 following a similar convention. Only that now the two-dimensional arrays that are used to specify the BCs are strings and are declared with `character(LEN=MAX_LEN) :: bc_mag(6,MAX_BLOCKS)` for the magnetic BCs and similarly for the hydrodynamic (fluid) BCs (`bc_hydro`). The order in which the BCs are specified is the same with the order of the neighbors IDs described above. If there at a face there exists a neighbor instead of a boundary of the domain the keyword `"none"` is specified instead of the type of BC. In the example

given here block 1 has a neighbor at the *North*, a perfectly conducting wall BC at *East*, a neighbor at *South* and another neighbor at *West* and perfectly conducting walls at *Top* and *Bottom*.

```
&blocks
 nblock_tot = 12,
 block_topo = 2, -1,  8,  9, -1, -1, 0,
              3, -1,  1,  9, -1, -1, 0,
              4, -1,  2, 10, -1, -1, 1,
              5, -1,  3, 10, -1, -1, 1,
              6, -1,  4, 11, -1, -1, 2,
              7, -1,  5, 11, -1, -1, 2,
              8, -1,  6, 12, -1, -1, 3,
              1, -1,  7, 12, -1, -1, 3,
              2,  1, 12, 10, -1, -1, 4,
              4,  3,  9, 11, -1, -1, 4,
              6,  5, 10, 12, -1, -1, 5,
              8,  7, 11,  9, -1, -1, 5,

  bc_mag = "none","conductr","none","none","conductr","conductr",
           "none","conductr","none","none","conductr","conductr",
           "none","conductr","none","none","conductr","conductr",
           "none","conductr","none","none","conductr","conductr",
           "none","conductr","none","none","conductr","conductr",
           "none","conductr","none","none","conductr","conductr",
           "none","conductr","none","none","conductr","conductr",
           "none","conductr","none","none","conductr","conductr",
           "none","none","none","none","conductr","conductr",
```

```
            "none","none","none","none","conductr","conductr",
            "none","none","none","none","conductr","conductr",
            "none","none","none","none","conductr","conductr",

   bc_hydro = "none","wall","none","none","wall","wall",
              "none","wall","none","none","wall","wall",
              "none","wall","none","none","wall","wall",
              "none","wall","none","none","wall","wall",
              "none","wall","none","none","wall","wall",
              "none","wall","none","none","wall","wall",
              "none","wall","none","none","wall","wall",
              "none","wall","none","none","wall","wall",
              "none","none","none","none","wall","wall",
              "none","none","none","none","wall","wall",
              "none","none","none","none","wall","wall",
              "none","none","none","none","wall","wall",
 /
```

• `algorithm` **namelist** This namelist contains the information needed to control the algorithm. The two Courant numbers used are specified by `courant_phys` for the physical Courant number and `courant_pseudo` for the pseudo time Courant number. The number of pseudo time iterations (internal loop) is specified by `ncyc_pseudo`. The controls for the symmetric Gauss-Seidel (SGS) solver are redundant. The user can specify either the number of sweeps the solver should take, through `ncyc_sgs` or a convergence criterion for through `eps_sgs`. For the applications described in this work it has been noted that between 5 or 6 sweeps of the SGS solver are needed for the error between two consecutive solutions to be brought to less than 1×10^{-10}. The small increment that is used to calculate the flux Jacobians is specified by `eps_jacobian`

and the eigenvalue "rounding" coefficient is specified by `eps_sonic`. The `implicit` flag tell WARP3 to run either in implicit or explicit mode. For debugging purposes WARP3 was run in first order in space resolution mode and the `first_order_space` flag has been left in place for future work. If axisymmetric simulations are performed the `axisymmetric` flag has to be set to TRUE. To stop WARP3 when the two norm of the residual dropped a certain order of magnitude the `converge_ord` variable is specified. (In this case the simulation is time dependent so that the `converge_ord` has a large value to avoid any interference with the simulation.)

```
&algorithm
 courant_phys = 5d0,
 courant_pseudo = 8.5d-1,
 ncyc_pseudo = 10,
 ncyc_sgs = 100000,
 eps_sgs = 1d-10,
 eps_jacobian = 1.0d-7,
 eps_sonic = 1.0d-4,
 implicit = .true.
 first_order_space = .false.,
 axisymmetric = .false.,
 converge_ord = 19
/
```

• `physics` **namelist** This namelist contains the variables that store information related to the physics model used by WARP3. The ratio of heat capacities (c_p/c_v) is passed to WARP3 in `gam`. The mass of the ion `massi` is not currently used but will be in future version of the code. Logical variables `viscous` and `resistive` tell WARP3 to include (if TRUE) viscous and respectively resistive effects by calling the corresponding parabolic flux procedures. The inverses of the Lundquist number and

of the Reynolds number are input in `lund1` and `reyn1` respectively. It is important to note here that in the normalized system of Eqns (2.62) the multiple of the Lundquist and Reynolds number with the Alfvèn number appear. The inputs to WARP3 are the inverses of these multiples `lund1`= $(LuAl)^{-1}$ and `reyn1`= $(ReAl)^{-1}$. Since the reference fluid velocity V cancels from these products it is inferred that the inverse of the Lundquist number from the input file is `lund1`= η_{ref}/ac_a and the inverse of the Reynolds number is `reyn1`= ν_{ref}/ac_a, where η_{ref} is the reference resistivity, ν_{ref} is the reference kinematic viscosity, a is a reference length and c_a is the Alfvèn speed calculated at the reference magnetic field and density. The type of resistivity model is input through `resitivity_model` which is a string. Logical `slip_wall` specifies if the velocity at the wall is null (FALSE) or if it can have a finite (non-null) value (TRUE).

```
&physics
 gam = 1.6d0,
 massi = 1.0d0,
 viscous = .false.,
 resistive = .false.,
 lund1 = 1.0d-6,
 reyn1 = 1.0d-6,
 resistivity_model = "none"
 slip_wall = .true.
/
```

• `globaldata`**namelist** The variables in this namelist are used to specify the initial values of the primitive variables in the entire domain. The variables are declared scalars and they are `rhoig` for density, `vxig,vyig,vzig` for velocity components, `bxig,byig,bzig` for magnetic field components, and `betaig` for pressure. In this example only the values of the density and and pressure are specified since the velocity

and magnetic field components have special initializations.

```
&globaldata
 rhoig = 1.0d0,
 betaig = 4.0d-2,
/
```

- `localdata` **namelist** Variables in this namelist specify the initial values of primitive variables per block. The variables are one-dimensional arrays of real declared as `real(KIND=R_SZE) :: rhoi(MAX_BLOCKS)`. The namelist is empty in this application since the per block initial primitive variables are not needed. However the namelist has to be present in the input file because the procedure that reads the input file assumes that the namelist is always present in the input file.

```
&localdata
/
```

- `smak` **namelist**

This is an additional namelist used to input data needed for the spheromak stability simulations and is declared in a module stored in a different file than the main WARP3 module file, called APPMODULES.f.

The variable used for this application are `smak_type` which is a string containing the description of the spheromak geometry (either cylindrical or CTX like). `R_ent` and `L_ent` are the radius and length of the entry region and `R_fc` and `L_fc` are the radius and length of the flux conserver. The number of points in the grid that is used to determine an initial perturbation in terms of a velocity field is specified by `nrt_vel` and `nzt_vel` for the radial and axial directions respectively. The number of points in the grid used to define an equilibrium magnetic field are specified by `nrt_bfield` and `nzt_bfield`. The initial perturbation is applied if the logical variable `apply_pert` is set to TRUE. The maximum mode number is specified by `nMax`. The initial perturbation is scaled by a factor specified in `vel_fak`. Two strings `vfield_type` and

`bfield_type` specify if the initial fields should be interpolated from data read from files or if they are calculated analytically. The files that contain the initial velocity perturbation data and equilibrium magnetic field are passed through `vel_file` and `bfield_file`.

```
&smak
 smak_type = "cylinder"
 R_ent = 1.00d0,
 L_ent = 1.70d0,
 R_fc = 1.00d0,
 L_fc = 3.00d0,
 nrt_vel = 75,
 nzt_vel = 225,
 nrt_bfield = 41,
 nzt_bfield = 41,
 apply_pert = .true.,
 nMax = 1,
 vel_fak = 1.00d-2,
 vfield_type = "interpolated"
 bfield_type = "analytic"
 vel_file = "/mhd4/udrea/init_smak/veloc/vel.hires.ar3.dat"
 bfield_file = "/mhd4/udrea/init_smak/magfield/eq.ar3.const.dat"
/
```

H.3 List of Variables

This section presents a list of the main variables in the current version of WARP3 together with a brief description. The variables are declared in modules and the module names are also listed in the table. F90 modules have a role equivalent to the

common blocks of F77 but in WARP3 very little use is made of common blocks to pass global variables to procedures. Instead variables are passed through argument lists in the procedure calls. Besides code robustness it is hoped that this feature makes the code easier to read. The modules are stored in file called WARP3MODULES.f

All real variables have been declared so that they are the equivalent of double precision F77 variables. A typical real variable declaration is `real(KIND=R_SZE) :: var` where `R_SZE` is the number of bytes necessary to represent a double precision number and was obtained using an F90 intrinsic function `R_SZE = selected_real_kind(p=8)`. All integer variables are declared with the standard `integer` type and all the strings are declared with `character(LEN=MAX_LEN) :: string` where `MAX_LEN=128`. Some of the real variables such as those used to store geometry data or the flux variables are run-time declared arrays for which memory is allocated after the input file is read and maximum array sizes are calculated. In the table the allocatable arrays are shown followed by a series of comma separated colons enclosed in parentheses similar to their F90 declarations. For example the variable that stores the x coordinate of cell vertices is listed as `x(:,:,:,:)`. The first three colons stand for the i, j, k position and the last colon stands for the block number. Another example is the conserved variables vector `q(:,:,:,:,:)` which has a similar structure to the x coordinate array only that the fourth position is the index of the conserved variables (from 1 to `C_VARS`) and the fifth position is the block number.

Name	Type	Purpose and Comments	Module
`R_SZE=8`	INTEGER	size of real variables - declared to be equivalent to F77 double precision reals with `R_SZE = selected_real_kind(p=8)`	`basic_const`
`MAX_LEN=128`	INTEGER	maximum length of strings - used to size strings such as file names	`basic_const`
`MAX_BLOCKS=200`	INTEGER	maximum number of blocks per processor	`basic_const`
`TAG_FAK=1000`	INTEGER	factor used to uniquely identify MPI tags	`basic_const`
`C_VARS=8`	INTEGER	number of conserved variables	`basic_const`
`startTime`	REAL	initializes timer (MPI) - used to calculate	`basic_const`

continued on next page

continued from previous page

Name	Type	Purpose and Comments	Module
		a job wall clock time	
`endTime`	REAL	finishes timer(MPI) - used to calculate	`basic_const`
		a job wall clock time	
`unit=10`	INTEGER	disk I/O unit number	`basic_const`
`ZERO=0.0`	REAL	real null value	`universal_const`
`HALF=0.5`	REAL	0.5 value	`universal_const`
`HALF=1.0`	REAL	1.0 value	`universal_const`
`TINIE`	REAL	smallest real value - obtained with `tiny`	`universal_const`
		F90 intrinsic function	
`BIGG`	REAL	largest real value - obtained with `huge`	`universal_const`
		F90 intrinsic function	
`PIE`	REAL	number π	`universal_const`
`MU0`=$4\pi \times 10^{-7}$	REAL	μ_0 - magnetic permitivity of free space	`universal_const`
`x(:,:,:,:)`	REAL	x position of cell vertices	`geom_arrays`
`y(:,:,:,:)`	REAL	y position of cell vertices	`geom_arrays`
`z(:,:,:,:)`	REAL	z position of cell vertices	`geom_arrays`
`xc(:,:,:,:)`	REAL	x position of cell centroids	`geom_arrays`
`yc(:,:,:,:)`	REAL	y position of cell centroids	`geom_arrays`
`zc(:,:,:,:)`	REAL	z position of cell centroids	`geom_arrays`
`ccfcd(:,:,:,:,:,:)`	REAL	distance from cell centroid to face center	`geom_arrays`
`F(:,:,:,:,:)`	REAL	hyperbolic fluxes in x direction	`flux_arrays`
`G(:,:,:,:,:)`	REAL	hyperbolic fluxes in y direction	`flux_arrays`
`H(:,:,:,:,:)`	REAL	hyperbolic fluxes in z direction	`flux_arrays`
`rhs(:,:,:,:,:)`	REAL	parabolic fluxes and source term fluxes	`flux_arrays`
`q(:,:,:,:,:)`	REAL	conserved variables at current time step	`consrvd_arrays`
`qn(:,:,:,:,:)`	REAL	conserved variables at time step n	`consrvd_arrays`
`qnm1(:,:,:,:,:)`	REAL	conserved variables at time step $n-1$	`consrvd_arrays`
`dq(:,:,:,:,:)`	REAL	residual or variations of conserved variables	`consrvd_arrays`
		in one time step	
`etax(:,:,:,:)`	REAL	electric resistivity in x direction	`diffus_arrays`
`etay(:,:,:,:)`	REAL	electric resistivity in y direction	`diffus_arrays`
`etaz(:,:,:,:)`	REAL	electric resistivity in z direction	`diffus_arrays`
`visc(:,:,:,:)`	REAL	dynamics viscosity of	`diffus_arrays`
`icels(MAX_BLOCKS)`	INTEGER	number of cells in i direction	`sze_arrays`
`jcels(MAX_BLOCKS)`	INTEGER	number of cells in j direction	`sze_arrays`
`kcels(MAX_BLOCKS)`	INTEGER	number of cells in k direction	`sze_arrays`

continued on next page

continued from previous page

Name	Type	Purpose and Comments	Module
`application_name`	CHARACTER	application identifier - needed for application specific I/O	`main_ctrl`
`run_diagnostic`	LOGICAL	switch that toggles diagnostics procedure calls	`main_ctrl`
`ncycplt_dump`	INTEGER	number of cycles between plot file dumps	`main_ctrl`
`tplt_dump`	REAL	dimensionless time units between plot file dumps	`main_ctrl`
`restart`	LOGICAL	if TRUE then WARP3 reads a restart file dumped at iteration `ncycrst_begin`	`main_ctrl`
`ncycrst_begin`	INTEGER	number of cycles from where WARP3 restarts	`main_ctrl`
`ncycrst_dump`	INTEGER	number of cycles between restart file dumps	`main_ctrl`
`trst_dump`	REAL	dimensionless time units between restart file dumps	`main_ctrl`
`ncycfin`	INTEGER	number of iterations after which WARP3 stops	`main_ctrl`
`tfin`	REAL	dimensionless time units after which WARP3 stops	`main_ctrl`
`gridfile_style`	CHARACTER	takes values `tecplot` for Tecplot ASCII or `fbin` for Fortran binary	`main_ctrl`
`dump_style`	CHARACTER	takes values `tecplot` for Tecplot ASCII or `fbin` for Fortran binary	`main_ctrl`
`grid_file`	CHARACTER	name of the grid file	`main_ctrl`
`grid_dir`	CHARACTER	directory where the grid file resides	`main_ctrl`
`out_dir`	CHARACTER	directory where the data files are dumped	`main_ctrl`
`rst_dir`	CHARACTER	directory where the restart files are dumped	`main_ctrl`
`hosts_file`	CHARACTER	name of the file where the processor names are dumped (full path)	`main_ctrl`
`nblock_tot`	INTEGER	total number of blocks	`block_info`
`nblock`	INTEGER	number of blocks per processor	`block_info`
`block_topo`	INTEGER	stores the IDs of neighbors and the process ID to which the block is distributed	`block_info`
`blockID`	INTEGER	local to global mapping	`block_info`
`bc_mag`	CHARACTER	magnetic BCs for each block	`block_info`
`bc_hydro`	CHARACTER	hydrodynamic (fluid) BCs for each block	`block_info`

continued on next page

continued from previous page

Name	Type	Purpose and Comments	Module
courant_phys	REAL	physical Courant number	alg_ctrl
courant_pseudo	REAL	pseudo time Courant number	alg_ctrl
ncyc_pseudo	INTEGER	maximum number of pseudo time iterations	alg_ctrl
ncyc_sgs	INTEGER	number of symmetric Gauss-Seidel sweeps	alg_ctrl
eps_sgs	REAL	convergence criterion for the symmetric Gauss-Seidel solver	alg_ctrl
eps_jacobian	REAL	increment used to calculate the flux Jacobians	alg_ctrl
eps_sonic	REAL	eigenvalue smoothing factor	alg_ctrl
converge_ord	INTEGER	WARP3 stops if the residual drops these many orders of magnitude	alg_ctrl
implicit	LOGICAL	if FALSE run explicit mode only	alg_ctrl
first_order_space	LOGICAL	if TRUE WARP3 runs in first order accurate in space mode	alg_ctrl
axisymmetric	LOGICAL	if TRUE WARP3 runs in the axisymmetric mode	alg_ctrl
gam	REAL	specific heats ratio (c_p/c_v)	phys_ctrl
massi	REAL	ion mass	phys_ctrl
viscous	LOGICAL	if TRUE turn on viscous parabolic fluxes	phys_ctrl
resistive	LOGICAL	if TRUE turn on resistive parabolic fluxes	phys_ctrl
reyn1	REAL	inverse of Reynolds number	phys_ctrl
lund1	REAL	inverse of Lundquist number	phys_ctrl
resistivity_model	CHARACTER	resistivity model selector	phys_ctrl
slip_wall	LOGICAL	if TRUE then the tangent velocity at the wall has a finite value	phys_ctrl
rhoig	REAL	overall initial density	global_init_ctrl
vxig	REAL	overall initial velocity in x direction	global_init_ctrl
vyig	REAL	overall initial velocity in y direction	global_init_ctrl
vzig	REAL	overall initial velocity in z direction	global_init_ctrl
bxig	REAL	overall initial magnetic field in x direction	global_init_ctrl
byig	REAL	overall initial magnetic field in y direction	global_init_ctrl
bzig	REAL	overall initial magnetic field in z	global_init_ctrl

continued on next page

continued from previous page

Name	Type	Purpose and Comments	Module
		direction	
betaig	REAL	overall initial normalized pressure	global_init_ctrl
etaig	REAL	overall initial resistivity	global_init_ctrl
viscig	REAL	overall initial viscosity	global_init_ctrl
rhoi	REAL	initial density in each block	local_init_ctrl
vxi	REAL	initial velocity in x direction in each block	local_init_ctrl
vyi	REAL	initial velocity in y direction in each block	local_init_ctrl
vzi	REAL	initial velocity in z direction in each block	local_init_ctrl
bxi	REAL	initial magnetic field in x direction in each block	local_init_ctrl
byi	REAL	initial magnetic field in y direction in each block	local_init_ctrl
bzi	REAL	initial magnetic field in z direction in each block	local_init_ctrl
betai	REAL	initial normalized pressure in each block	local_init_ctrl
etai	REAL	initial resistivity in each block	local_init_ctrl
visci	REAL	initial viscosity in each block	local_init_ctrl
ierr	INTEGER	message error ID number (MPI)	misc_vars
taskID	INTEGER	task or processor ID	misc_vars
iMax, jMax, kMax	INTEGER	maximum arrays sizes on a processor	misc_vars
flip_ijOrd	LOGICAL	truth table used for non-simply connected grids	misc_vars
t	REAL	dimensionless time	misc_vars
dt	REAL	dimensionless time step	misc_vars
icount	INTEGER	iterations counter	misc_vars
faceSendReq, faceRecvReq	INTEGER	request handles for the MPI_Wait operations for double face message passing	misc_vars
edgeSendReq, edgeRecvReq	INTEGER	request handles for the MPI_Wait operations for frame message passing	misc_vars
idir	INTEGER	direction index (1 for i, 2 for j and 3 for k)	misc_vars
tnorm	REAL	two (Euler) norm of the residual	misc_vars
tnorm0	REAL	two (Euler) norm of the residual at t=0	misc_vars

H.4 Running WARP3

This section explains how to run WARP3 on the Aeronautics and Astronautics Alpha cluster and on the MHPCC IBM SP2. The Alpha cluster is normally used for testing, debugging and running relatively smaller resolution cases for verifying the simulation setup. Access to the Alpha cluster is open to the CFD lab students and there is no scheduling mechanism is used. The Alpha cluster consists of sixteen DEC Personal Alphastation 433 workstation connected by a 100 Mbps Ethernet connection. The IBM SP2 is normally used for high resolution runs and it requires the user to have an account and a SecureID encryption card. The IBM SP2 runs are submitted to the system through a specialized queueing system called `loadleveler` that insures that the nodes allocated to a job are use in exclusivity by that job. The MHPCC IBM SP2 system consists of two major subsystems, the older tsunami and typhoon with a total of about three hundred processors available. Details on the IBM SP2 system (both hardware and software) can be found on the MHPCC web page.

Instructions on how to run WARP3 on the departmental Alpha cluster are presented first to familiarize the user with the basic concepts and then details about using the IBM SP2 queueing system are presented last. It is assumed that the user has a copy of the most recent and stable version of WARP3 and that the code has been compiled and an executable named `w3` is present in the working directory. The latest version of WARP3 is available from the CVS repository on the departmental network.

H.4.1 Running WARP3 on the Alpha Cluster

Besides the executable an input file and a program group file are needed. A few input files and the program group file are located in the repository and are normally automatically uploaded with the source code. The input file structure is declared in a previous section of this appendix. The program group file specifies the processors

on which WARP3 will run and is names `w3.pg` to be consistent with the name of the executable (`w3`). A listing of the program group file is given below and explanations follow.

```
# program group file for the parallel implicit MHD code
# B Udrea, Apr 22, 1998
# modify the path to correspond to your working copy

# cronus is a graphics workstation - not normally
# used for runs, sometimes used for debugging
#cronus.aa.washington.edu      0   /mhd4/udrea/mhd/warp3/implicit/w3

 atlas.aa.washington.edu       0   /mhd4/udrea/mhd/warp3/implicit/w3
 coeus.aa.washington.edu       1   /mhd4/udrea/mhd/warp3/implicit/w3
 crius.aa.washington.edu       1   /mhd4/udrea/mhd/warp3/implicit/w3
 epimetheus.aa.washington.edu  1   /mhd4/udrea/mhd/warp3/implicit/w3
 gaea.aa.washington.edu        1   /mhd4/udrea/mhd/warp3/implicit/w3
 hyperion.aa.washington.edu    1   /mhd4/udrea/mhd/warp3/implicit/w3
 iapetus.aa.washington.edu     1   /mhd4/udrea/mhd/warp3/implicit/w3
 metis.aa.washington.edu       1   /mhd4/udrea/mhd/warp3/implicit/w3
#mnemosyne.aa.washington.edu   1   /mhd4/udrea/mhd/warp3/implicit/w3
#oceanus.aa.washington.edu     1   /mhd4/udrea/mhd/warp3/implicit/w3
#phoebe.aa.washington.edu      1   /mhd4/udrea/mhd/warp3/implicit/w3
#prometheus.aa.washington.edu  1   /mhd4/udrea/mhd/warp3/implicit/w3
#rhea.aa.washington.edu        1   /mhd4/udrea/mhd/warp3/implicit/w3
#tethys.aa.washington.edu      1   /mhd4/udrea/mhd/warp3/implicit/w3
#thea.aa.washington.edu        1   /mhd4/udrea/mhd/warp3/implicit/w3
#themis.aa.washington.edu      1   /mhd4/udrea/mhd/warp3/implicit/w3
```

The program group file has three columns. The first column contains the name (or the IP address) of the workstations, the second column contains the number of processes on each workstation and the third column contains the full path to the executable. The # signs in front of a line represent are comments and the columns in the program group files can be space or tab separated. In the listing shown above the sixteen workstations of the parallel cluster are shown with a seventeen workstation that is not normally used for runs (`cronus`) since it is intended to be used as a high end graphics station.

In the example given here the program group file is setup such that the parallel job runs on the eight workstations that don't have comments in front of their names. The workstation from where the job is started should have a `0` in the second column and its processor (task) ID is zero. All the other members of the pool should have a `1` in the second column. Details on program group files can be found in MPI User's Guide available online at `www-unix.mcs.anl.gov/mpi/`.

Assuming that the input file, the grid and the directories where output files will be written are ready the user has only to login into the workstation identified as `0` (in this case `atlas`) change the directory to the location where the executable and the program group file are and at the system prompt issue the command `%w3 input.myinput` where `%` is the system prompt. Note that if no input file name is given after `w3` then WARP3 looks for a file called `input` by default.

If the user desires WARP3 to perform a sequential run, then only one processor name should be left uncommented and the second column corresponding to that processor should contain a `0`.

H.4.2 Running WARP3 on the MHPCC IBM SP2

The IBM SP2 queueing system requires that the parallel job be submitted using a script file called a `command` file. There are other various system requirements not given here but described in detail on the MHPCC web page. As previously it is assumed that

the user has a version of WARP3 compiled and ready to run and the environment variables are set as specified in MHPCC documentation. Below a command file is listed and explanation for each line are given as comments in the file. The command that which the user has to issue at the system prompt is `%llsubmit mycommandfile` where `mycommandfile` is the name of the command file.

The command file was used for a parallel run with twenty-four processors. Similarly to a UNIX shell script the `#` signs are used to comment out whatever is at the right of them. Peculiar to this type of script is that `loadleveler` options and commands have a `#@` prefix as it can be seen below. The reason for that is that the options and commands can be easily commented out by just removing the `@` from the prefix. The comments given are sufficient for a user familiar to the system. If some of the options and commands are unclear the user is referred to the excellent online documentation available on the MHPCC web pages at `www.mhpcc.edu`.

```
#!/bin/csh
#####################################################################
# Maui High Performance Computing Center
# FILE: smak.ar1.cmd
# DESCRIPTION:  LoadLeveler command file for the parallel
#               implicit Riemann  solver for MHD (WARP3) used
#               for the spheromak runs
# AUTHOR: 11/7/98 Bogdan Udrea - from an example by Blaise Barney
#####################################################################

# My account number - needed for accounting CPU time usage
#@ account_no  = AFOSR-0330-000

# Parallel job name - needed by loadleveler
```

```
#@ job_name = w3

# This is the shell script that takes care of starting the
# parallel job, distributes the parallel code to the processors
# and manages the environment and the communications
# (poe stands for parallel operating environment)
#@ executable = /usr/bin/poe

# The arguments to poe are the name of the parallel code executable
# followed by the name input file
#@ arguments = w3 input.smak.cyl.16.ar1.24p

### file and directory specifications
# initialdir specifies the directory where the parallel
# code and the input file are located
#@ initialdir      = /u/udrea/mhd/work

# output specifies the name of the file (full path) where
# any output from the parallel code is written
#@ output          = /u/udrea/mhd/work/w.$(Cluster).out

# error specifies the name of the file (full path) where
# any error messages from the parallel code and operating
# environment are written
#@ error           = /u/udrea/mhd/work/w.$(Cluster).err

### time limit for job (in hh:mm:ss)
#@ wall_clock_limit  =  5:59:59
```

```
### parallel environment specs
#@ job_type        = parallel
#@ requirements    = (Adapter == "hps_user") && (Feature == "tsunami")
#@ environment     = MP_EUILIB=us;MP_INFOLEVEL=0;MP_LABELIO=yes
#@ min_processors  =  24
#@ max_processors  =  24

# specify to send me mail when done
#@ notification         = always
#@ notify_user          = udrea@aa.washington.edu

### commit it
#@ queue

echo "Starting the job..."
echo " "
```

H.4.3 Output files

A brief description of the output files is given in this section to familiarize the user with the way WARP3 produces output files and how they can be post-processed so that the results can be visualized with Tecplot.

There are two types of output files. The first type is a plot only file where the cell vertex coordinates and the eight primitive variables $(\rho, v_x, v_y, v_z, B_x, B_y, B_z, p)$ at the cell vertices are written. The second type of output file is a restart file where the eight conserved variables $(\rho, \rho v_x, \rho v_y, \rho v_z, B_x, B_y, B_z, e)$ and and the components of the resistivity (η_x, η_y, η_z) and kinematic viscosity ν at cell centers are written.

The format of this files can be either Tecplot ASCII or Fortran binary. Fortran

binary files are smaller than Tecplot ASCII files but they are not portable between systems so that they require local post-processing.

Each processor writes the data from the blocks assigned to it to a file that has a name made of four parts separated by period (.) sign. Each of the first three parts serves to identify the type of output file (plot or restart), the processor that wrote it and the iteration where the file was written. The forth part is just an extension traditionally used for Tecplot ASCII files (`tec`). For example, lets say that a parallel job run on two processors writes plot files every 100 iterations and restart files every 5000 iterations. The the first output file (iteration counter is null) for processor with ID=0 would be `w3.000.000000.tec` and the second output file would be `w3.000.000100.tec`. Similarly the second processor (ID=1) writes the output files with names `w3.001.000000.tec` and `w3.001.000100.tec` respectively. The names of the first restart files (written at iteration 5000) are `r3.000.005000.tec` and `r3.001.005000.tec`. WARP3 can write data at specified non dimensional time intervals also and then it is possible to have file names such as `w3.000.002533.tec` and `w3.001.002533.tec` for example.

For post-processing all the output files should reside in the same directory. A UNIX script file called `mycat` has been written that concatenates the output files and converts the plot files to Tecplot binary by calling the `preplot` executable. The command line arguments for `mycat` are the type of the output file that should be processed, `-p` for plot files and `-r` for restart files, and the full path to the directory where the output files are. For example to postprocess the plot files in a directory called `/mhd4/udrea/results/new` the command line at the system prompt would be `%mycat -p /mhd4/udrea/results/new`. The UNIX shell script was designed with some degree of robustness but it can be improved by rewriting the command line argument parser. The post-processed plot files for the example given above are called `w000000.plt` and `w000100.plt` and the post-processed plot file is called `r005000.tec`. Extension `plt` for the post-processed plot files is standard for Tecplot

binary files. Since WARP3 has only a Tecplot ASCII read capability for restart files the restart files are concatenated only so that they keep the `tec` extension.

It is possible to use Tecplot binary output formats for plot files but then the file concatenation would be a problem. However with the new capabilities of Tecplot v7.5 it might be possible to write Fortran or C procedures callable from Tecplot that would load the desired files into the Tecplot without concatenation.

VITA

The author was born in Bucharest, Romania on June 27, 1966. He graduated from the N. Balcescu (now Sf. Sava) High School in 1984 and passed the entrance exams for the Aeronautical Engineering Department of the Bucharest Polytechnic Institute (now the Polytechnic University). After the drudgery of one year of mandatory military service he proceeded with his education and in summer of 1990 he obtained his engineering degree from the Aeronautical Department with a specialization in rocket propulsion systems. He then worked for four years in the aerospace industry in the propulsion area in Romania (between 1990 and 1992) and in Canada (between 1992 and 1994) where he immigrated at the end of 1992. In June 1994 he got married with his lovely wife Roxana in Vancouver, British Columbia and in September he went back to school to better himself and learn computational fluid dynamics and plasma physics at the University of Washington Aeronautics and Astronautics Department. In August 1997 Roxana gave birth to a smart and beautiful baby girl named Anna Matilda who is the joy of our lives. The author enjoys playing basketball, or used to before the baby got born, and is a fan of computer combat flight simulators. He dearly loves his wife and daughter.