Multi-Temperature and Ionization Effects on Bluff Body in Hypervelocity Flow

Ward W. Vuillemot

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

Master of Science

University of Washington

2001

Program Authorized to Offer Degree: Aeronautics and Astronautics Engineering

University of Washington

Graduate School

This is to certify that I have examined this copy of a master's project by

Ward W. Vuillemot

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.

Committee Members:

_____

Uri Shumlak

Date: _____

In presenting this project in partial fulfillment of the requirements for a Master's 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 project is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Any other reproduction for any purpose or by any means shall not be allowed without my written permission.

Signature_____

Date_____

University of Washington

Abstract

Multi-Temperature and Ionization Effects on Bluff Body in Hypervelocity Flow

by Ward W. Vuillemot

Project Advisor

Assistant Professor Uri Shumlak
Aeronautics and Astronautics

We numerically investigate the effect of ionization on hypersonic flows by using an approximate Riemann fully three-dimensional MHD solver, WARP3, which includes multiple temperature. The code calculates the ionization fraction as a function of temperature. We treat the flow as a single fluid with three (3) constituents, or namely ions, electrons, and neutrals.

It is believed that the inclusion of multi-temperatures effects may explain the experimentally measured increase of shock stand-off distance encountered when an ionized hypersonic flow stagnates bluff body.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

It goes without saying this section will never adequately acknowledge all those people who have affected the author in one fashion or another, and to whom he would like to express his deepest gratitude and thanks. First and foremost, the author is forever indebted to: *Uri Shumlak* whose generosity of spirit, experience, and insight has allowed this project to be written at all; *L. Phida Ung* who has supported the author these past two years in every conceivable way;*Bogdan Udrea* for his patience, support, and mentorship;*Timothy J. Curry* who helped the author to succeed as an engineer, and forever stands an the only necessary example of determination; *Yasuo Kitane* whose friendship is my great honor to share; *Ken Chadwick* who was first to provide the author an opportunity to grow as an engineer and researcher; both *William Rae* and *Joseph Mollendorf* whose mentorship gave confidence when most needed; and, all the *peoples of both departments at SUNY at Buffalo and University of Washington*.

Special thanks and acknowledgement goes to the author's family, both nuclear and extended, whose support, belief, and encouragement made this all possible...and all worth it.

Finally, the author would like to thank most his father, *Ward T. Vuillemot*, who taught the most important lesson. The world is not bettered through the momentous works of a few great people, but through the multitude of hands that is all of us, joyously laboring to improve ourselves and our surroundings little by little, day by day.

DEDICATION

To a thousand, million moments shared with L. Phida Ung.

Chapter 1

INTRODUCTION

The development of both near-term and far-term hypersonic (Mach $\gg$ 5) transportation will depend heavily upon a more thorough fundamental understanding of the relationship between vehicle and flow. At present, our understanding of this flight regime permits only modelling gross behaviors and interactions that are adequate for applications where issues of efficiency do not dominate design.

It is well understood that in order for us as a peoples to realize outer space with all its economic potential, the cost and practicality of high-speed transport such as the supersonic transport (SST) and United States Space Shuttle must be exponentially improved. Researchers in materials, aerodynamics, structures, propulsion, and many, many more fields are looking for a means to pull back the veil that surrounds hypersonic flight. Toward this end, the goal of this research is to provide a deeper understanding of the basic physics surrounding lowly ionized hypervelocity flow stagnating over bluff bodies. A motivating factor of this research is our interest to better manage the many losses incurred due to the presently unavoidable creation of a bow shock in the front of the vehicle.

It might interesting for the reader to learn that the shape of present high-speed vehicles such as the Space Shuttle are dominated more by poor material characteristics rather than aerodynamics. To point, skin friction at hypersonic speeds is so severe that the static temperature begins to ablate the vehicle surface [1, 12, 18, 3]. During the 1960s it was observed that sharp-pointed models in shock tunnels emerged with a consistent bluff nose due to ablation. It was later understand that the flow continued to ablate the stagnation region until a strong enough bow shock is generated in front

of the vehicle. To point, a bow shock has the affect of ”absorbing” much of the kinetic energy of the flow by diverting it around the vehicle. With the vehicle sees less of this kinetic energy, the static temperature decreases until ablation is halted. While this effect has the advantage of permitting the vehicle to continue its flight at high speeds, it results in an increase in the observable cross-section of the vehicle with respect to the flow. To point, the cost of hypersonic flight with bluff bodies is an increase in drag. This, of course, has the very deleterious effect of increasing the power required in order to plow the ”enlarged” vehicles through the atmosphere.

Furthermore, there are still other effects associated with the relatively large stagnation region and elevated temperatures. One such effect is called ”radio frequency (RF) black-out”, and it occurs during re-entry. As the return vehicles rams into the atmosphere, the temperatures around the vehicle become large enough to ionize the low density gas. As a result of the complex geometries and large physical gradients, the pressure gradients are equally severe. When this occurs, the flow around the vehicle can become ”cold”, or chemically non-reacting. That is to say, the energy necessary to de-ionize the flow is lost, and therefore the ions stream from the stagnation region around the vehicle creating an ion sheath. It is this sheath that effectively absorbs all incoming RF, resulting in a communications black-out for a major portion of the re-entry sequence.

Without going into any more depth, there are many issues surrounding the study of hypersonic flow as it pertains to making it far more technologically viable than it is presently. Some of the future rewards from this paper’s research may include, but are in no way limited to:

- reduced skin friction;
- reduced heat loads to vehicle;
- ability to exert a body force via Lorentz force, where multiple magnetic sources within the vehicle are used in parallel, producing roll, pitch, and yaw moments;

- boundary layer control whereby delaying the onset of boundary layer separation would permit a decrease in control effector (e.g. rudder, aileron) deflection angle from neutral plane resulting in decreased drag and skin friction;

- increase in scramjet combustion efficiency, where there is some evidence that increased number density of NO^+ produces increased flame temperature and combustion rate; and,

- bow shock control including shock stand-off distance and shock thickness.

1.1 Many Roads Travelled

This report will discuss the author's research efforts which can be best delineated between the development of computational tools, and the application thereof on the aforementioned problem. Without the former the later efforts would not be possible. Any research is never easily classified, and this paper is no different. The thrust of our research sits at the crossroads of lowly ionized plasmas, gas dynamics, high temperature gases, and computational fluid dynamics. The tools developed, or more precisely the functionality added to WARP3[1], along with all the work completed by others going before, have extended WARP3 capabilities to be arguably in a class all its own.

1.1.1 WARP3

In order for the reader to have a better appreciation of the author's efforts, a brief description of WARP3 [19] in its present state is presented. WARP3 is capable of solving magneto-hydrodynamic (MHD) equations in a fully coupled manner using a Roe-type approximate Riemann solver. The solver has been modified to include non-ideal effects that arise from temperature, radiation, and finite viscosity and resistivity. Furthermore, the author helped to relax the assumption that the fluid be fully ionized in order that local ionization fractions less than unity can be appropriately handled by

[1]University of **W**ashington **A**pproximate **R**iemann solver for **P**lasmas in three (**3**) dimensions; developed by Uri Shumlak, Ogden Jones, Bogdan Udrea, and the author.

the solver. In short, the fluid can range from completely neutral to completely ionized at a local level.

Present-day magnetohydrodynamic (MHD) solvers are typically explicit in nature, and therefore are computationally expensive to operate when considering anything but small time-scale phenomenon. However, as a result of the efforts of Bogdan Udrea [23], WARP3 is now both explicit and implicit, enabling a researcher to examine whatever time-scale is of interest[2].

Another innovation of WARP3 is the use of arbitrary finite volumes so it can model realistic three-dimensional geometries. Furthermore, these cells can be organized into blocks that can then be distributed to multiple processors using a method termed, aptly enough, *block domain decomposition*. Through the judicious use of industry standards that do not restrict the code to any one compiler or processor, WARP3 can be operated on a variety of platforms such as DEC Alphas, Pentiums running Windows NT v4.0, and supercomputer clusters.

1.1.2 Effects of Multiple Temperatures and Ionization

In the near-term, we desire to present a mechanism that can explain the observed shock stand-off distance and decrease of bow shock strength. This mechanism includes a generalization of the single-fluid MHD system of equations, where the ionization fraction is not assumed to be unity as is typical of most derivations [4, 13]. U. Shumlak believes that the presence of these aberrations to the bow shock can be explained by the inclusion of the three (3) constituents' temperatures of an ionized single-fluid flow, namely, neutrals, ions, and electrons.

[2]It is true that implicit codes are generally unconditionally stable for real-world conditions. It is also true that implicit codes can take large time steps, thereby allowing the examination of large time-scale phenomenon not as easily studied with explicit codes. However, as impressive as these advantages may seem, they come at a cost. The added computational cost to invert large (even sparse) matrices degrades the performance to such a degree that both implicit and explicit codes, especially in the case of WARP3, are about equal. This is in part due to the fact that ultimately both implicit and explicit forms work only on reducing local error per iteration, and therefore global error (residual) decreases at similar rates. However, the greatest advantage for implicit is increased spatial resolution. U. Shumlak *et al.* are presently working on the next generation numeric solver that will include fully two fluid flow along with multi-grid functionality, thereby addressing the problem of reducing global and local error in equal terms.

1.2 Organization of Paper

The paper is divided into four (4) main sections, each corresponding to a major chapter. Before we proceed, it must be noted that this paper, now submitted as a report detailing the author's two (2) years as a research assistant for U. Shumlak, was originally written as a thesis toward the author's Master of Science. Consequently, both Chapter 5 and Appendix F represent the author's efforts to provide direction and clarification for future research.

The first section (Chapter 2) presents the derivation of the multi-temperature single fluid MHD model, including all pertinent physics both plasma and otherwise. The assumptions[3] used in this derivation as well as the limitations of the model are also presented. The following section (Chapter 3) gives a detailed description of the algorithms developed for the solution of thermal diffusion and ionization. This section also describes some minor algorithms added to WARP3 including the calculation of pressure drag and new technique for obtaining higher Mach numbers. The next section (Chapter 4) will detail the various tests performed to verify that the above algorithms work as is expected by either analytical results or theory. Chapter 5 will introduce in detail the major thrust of application of WARP3 by this author, namely lowly ionized hypervelocity flow stagnating over a bluff body. Issues including the test matrix and choice of grids will be discussed. The last section will provide a summary, conclusion, and suggestions for future work.

Numerous appendices have been included that detail derivations and other relevant information in support of future research. Most important of all appendices is Appendix E. It contains the *User's Manual for WARP3*[4] for WARP3. It is hoped that this appendix, in particular, is maintained by future members of the laboratory.

---

[3]As will be shown, these assumptions are less restrictive than the neo-classical approach that assumes the flow is fully ionized.

[4]This manual represents an ongoing effort by the Computational Fluid Dynamics Lab at the University of Washington, having been originally included in Udrea's doctorate dissertation [23].

Chapter 2

PERTINENT PHYSICS

In this section we will introduce the physics necessary to understand what is known as magneto-hydrodynamics (MHD), along with its relevance to hypersonics and ionization.

2.1 Plasma Physics

Nearly the whole of our universe is composed of plasma. Even so, it is the least understood and appreciated of the four states of matter. Still, we encounter plasma in our daily lives all the time. Everyday[1] when we look at our Sun our eyes fall upon an enormous ball composed entirely of plasma. Whenever you drive an automobile, the chamber where the fuel is combusted utilizes plasma. Even the fluorescent lambs that are in nearly every office of the world use the electrical excitation of plasma to shed light on the matter[2] at hand.

Plasma, loosely defined, is as an electrically conducting fluid. In the simplest of terms, electrons through some process become detached from their hosts, or atoms. The atoms, originally neutral, become positively charged, or ions. The ions and electrons, along with the remaining neutrals are then the main constituents of the flow. The entire set of assumptions and models used to solve problems shifts.

Historically, the mind of an aerospace engineer has focused on more innocuous and inert matter such as air or water. However, recent trends for aircraft to go faster and higher and for longer durations has required the engineer to develop a new set of tools that can handle an environment where temperatures in the thousands of degrees

[1] For those of us in Seattle this might not be the case, though.

[2] All puns intended.

Kelvin is the norm, and not the exception. These high temperatures, which can at times be equal to the surface temperature of Sun[3], create a variety of issues that have yet to be completely resolved in the last fifty years since the inception of hypersonic research.

Whereas an aerospace engineer could once comfortably use the Euler set of equations for most flow regimes less than Mach 0.8, and resort to the full Navier-Stokes when necessary, hypersonic flow regime is a marriage between fluid dynamics and plasma physics. At a minimum, a hypersonics researcher must also have a complete appreciation of the single-fluid MHD model.

Plasma is often called the *fourth* state of matter after solids, liquids, and gases. The difference between these four states is determined mainly by temperature. At very high temperatures a gas becomes fully ionized, with the equilibrium degree of ionization described by the Saha equation,

$$\frac{n_i}{n_n} \approx 2.4 \times 10^{21} \frac{T^{\frac{3}{2}}}{n_i} e^{-U_i/kT} \tag{2.1}$$

where n_i and n_n is ion and neutral number density in m^{-3}, respectively; T is gas temperature in Kelvin, K; k is the Boltzmann constant; and, U_i is the ionization energy[4]. At room temperature, this number density fraction is on the order of 10^{-122}, but as a gas is heated the ratio will exponentially increase until it is fully ionized. For singly ionized gas, the ratio is equal to one.

Alternatively, a more general Saha equation provide by Zel'dovich [25] is,

$$\frac{n_{m+1} + n_e}{n_m} = 2\frac{u_{m+1}}{u_m} \left(\frac{2\pi m_e kT}{h^2} \right)^{1.5} e^{-U_i/kT} = K_{m+1}(T) \tag{2.2}$$

where m denotes the m^{th} ion involved in some said ionization process, and m_e is electron mass. By multiplying Eqn (2.2) by kT_e, a ratio of the partial pressures, $p_i = n_i kT$ is obtained. The electron partition function is denoted by u, representing simply the linear sum of product between a free electron's transitional partition function and its statistical weight.

---

[3] ~ 5000 K

[4] The energy required to remove the outermost electron from an atom

2.2 Plasma Models

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. Three dimensional applications of the kinetic model are limited to relatively low density plasmas.

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.

It is this later model that WARP3 embodies, and for which this project looks to extend. In the following sections the physics necessary to describe the fluid model of plasma is presented.

2.3 Single-fluid MHD Model

From the perspective of a physicist, the single-fluid MHD model is a model of plasma treated as a single hydrodynamic fluid acted upon by electric and magnetic forces [8, 13]. Conversely, from the perspective of a mathematician, it is a mixed set of hyperbolic and parabolic partial differential equations. 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.) This wave-like behavior advects various modes of information at varying speeds

from point to point, and parallels what occurs in the classical case of Euler flow with its positive and negative sonic wave. The parabolic part describes the diffusion of the variables due to resistivity, viscosity, and thermal diffusion. Diffusion works to act as a local averager of flow characteristic values, forcing local minimums and maximums to create in the best of situations a stable, steady state at the global level. Also, the parabolic fluxes are not part of the neo-classical single-fluid MHD model, but are necessary when attempting to predict a wide range of physical phenomenon pertinent to plasma physicists.

The hyperbolic PDEs are solved using an approximate Riemann solver that updates the variables inside cells. The cells discretize the domain of interest, where fluxes are calculated at the cell faces based on the solution of an approximate Riemann problem. The parabolic PDEs are solved in a similar manner that involves a locally aligned coordinate system aligned with the cell faces of the staggered grid. The staggered grid is a fictitious grid computed from the input grid, and where the faces of the fictitious grid correspond roughly to the input grid cell centers.

2.3.1 Summary of Assumptions

We begin our treatment by introducing our assumptions and notations.

- Plasma is not sufficiently hot, thereby singly charged ions, electrons, and neutrals are present, and the plasma is quasineutral $(n_e = Zn_i.)$[5]

- Ions and electrons are in equilibrium, and have Maxwellian distributions and temperatures are assumed to not be equal.

- The mass of the ion is significantly larger than the mass of electron, and therefore the ratio, $\frac{m_e}{m_i}$, is zero, and sum, $m_i + m_e$ is approximately m_i. This is extremely reasonable since for even hydrogen the ion (proton) is 1,800 times heaver than the electron.

[5]In the neoclassical MHD case, the plasma is assumed completely ionized, so that there are no neutral species.

- 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-relavistic.
- Electron inertia is neglected, which is valid for phenomenon that are sufficiently slow that electrons have time to reach dynamical equilibrium in regard to their motion along the magnetic field [8].
- 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 Debye length.

The following are assumptions that are typical of most single-MHD derivations, however they are **not** included in the following derivation in order that we treat lowly-ionized plasmas correctly.

- Ion and electron temperatures are assumed to be equal.
- Plasma is sufficiently hot that only singly positive ions and electrons are present and the plasma is neutral $(n_e = n_i.)$

2.4 Statistical Plasma Dynamics

The single-fluid model can be derived from statistical plasma dynamics. A more thorough review of the topic is found in Chapter 2 of Udrea's thesis [23]. Statistical plasma dynamics model the plasma 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 $f_\alpha(\vec{r}, \vec{c}, t)$. The distribution function f_α gives the statistical probability per unit volume of phase space $(\vec{r}, \vec{c})$ of finding particles of species α at position $\vec{r}$ and velocity $\vec{c}$ at time t.

2.4.1 Simple Relationships and Definitions

As with Dolan [5], we model the momentum transfer collision frequencies for ion $\rightarrow$ neutrals and electrons $\rightarrow$ neutrals as,

$$\nu_{in} \cong C n_n \tag{2.3}$$

$$\frac{\nu_{in}}{\nu_{en}} \cong \left(\frac{m_e}{m_i}\right)^{0.5} \tag{2.4}$$

where C is a constant with units of $L^3 t^{-1}$.

We assume that three constituent, single-fluid flow is partially singly ionized ($Z = 1, f_i \in (0, 1]$). We can express these assumptions with the ion, electron, and neutral number densities as,

$$\begin{aligned} Z = 1 \quad &\rightarrow \quad n_i = n_e \\ 0 \geq f_i \leq 1 \quad &\rightarrow \quad n_i = f_i n \\ &\qquad\quad n_n = (1 - f_i)\, n \end{aligned}$$

We can now express the collision frequencies Eqn (2.3, 2.4) in terms of the fluid number density, n, and ion fraction, f_i.

$$\nu_{en} = C n (1 - f_i) \tag{2.5}$$

$$\nu_{in} = C n (1 - f_i) \left(\frac{m_e}{m_i}\right)^{0.5} \tag{2.6}$$

Next, we further develop definitions for density, velocity, material pressure[6], charge

[6]Also referred to as *total pressure*

density, and current density by assuming that $n_i = n_e$ and that the plasma is a single fluid composed of multiple constituents.

The mass density is,

$$\rho = n_i m_i + n_e m_e \approx n\left(m_i + m_e\right) \approx n m_i \tag{2.7}$$

The mass velocity is,

$$\vec{u} = \frac{\left(n_i m_i \vec{u_i} + n_e m_e \vec{u_e}\right)}{\rho} \approx \frac{m_i \vec{u_i} + m_e \vec{u_e}}{m_i + m_e} = u_i + \left(\frac{m_e}{m_i}\right) \vec{u_e} \tag{2.8}$$

The material pressure is,

$$p = p_n + p_i + p_e = n_n k T_n + n_i k T_i + n_e k T_e \tag{2.9}$$

where k is the Boltzmann constant, n_n is the neutral number density, and T_n is the neutral temperature. The remaining variables are the ion and electron number densities and temperatures, respectively.

Additionally, the charge density is,

$$\sigma = \left(n_i - n_e\right) e \tag{2.10}$$

where e is the electron charge.

Finally, the current density is,

$$\vec{j} = n_i e \vec{u_i} - n_e e \vec{u_e} \tag{2.11}$$

2.4.2 Conservation Equations

Conservation equations describe how averaged quantities in a moving plasma change over 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 introduce each of these conservation equations, and where applicable its derivation. More complete set of derivations from the Boltzmann equation is found in many plasma textbooks [5, 23].

Conservation of Mass and Charge

The mass conservation equations are derived from the summation of the species conservation equations, where each is multiplied through by the mass of its species.

$$\frac{\partial n_i}{\partial t} + \vec{\nabla} \cdot (n_i \vec{u_i}) = 0 \tag{2.12}$$

$$\frac{\partial n_e}{\partial t} + \vec{\nabla} \cdot (n_e \vec{u_e}) = 0 \tag{2.13}$$

Multiplying Eqn (2.12, 2.13) by m_i and m_e, respectively we obtain with a little algebraic manipulation the *conservation of mass equation*, or

$$\frac{\partial \rho}{\partial t} + \vec{\nabla} \cdot \left(\rho \vec{\hat{u}} \right) = 0 \tag{2.14}$$

where $\vec{\hat{u}}$ denotes the fluid velocity versus the species' velocities.

Using Eqn (2.10) for the definition of the charge density, and subtracting Eqn (2.12) and Eqn (2.13) the *conservation of charge equation* is

$$\frac{\partial \sigma}{\partial t} + \vec{\nabla} \cdot \vec{j} = 0 \tag{2.15}$$

Conservation of Momentum

The single fluid momentum conservation equation is obtained by summing over all species for momentum conservation derived from statistical plasma dynamics' Boltzmann equation. The resulting equation is best written as,

$$\frac{\partial (\rho \vec{u})}{\partial t} + \vec{\nabla} \cdot \rho \vec{u} \vec{u} + \vec{\nabla} \cdot \overleftrightarrow{p} = \sum_{\alpha} n_\alpha e_\alpha \vec{E} + \sum_{\alpha} n_\alpha e_\alpha \vec{u_\alpha} \times \vec{B} \tag{2.16}$$

For three constituent species, this reduces to,

$$\frac{\partial (\rho \vec{u})}{\partial t} + \vec{\nabla} \cdot \rho \vec{u} \vec{u} + \vec{\nabla} \cdot \overleftrightarrow{p} = n_e e \vec{E} + n_e e \vec{u_e} \times \vec{B} + n_i e \vec{E} + n_i e \vec{u_i} \times \vec{B} \tag{2.17}$$

However, note that the neutral charge is zero therefore it can be *a priori* be dropped from Eqn (2.17). In addition, from relationships presented earlier for these three species, Eqn (2.17) is further simplified to,

$$\frac{\partial (\rho \vec{u})}{\partial t} + \vec{\nabla} \cdot \rho \vec{u} \vec{u} + \vec{\nabla} \cdot \overleftrightarrow{p} = n f_i e \vec{E} + n f_i e \vec{u_e} \times \vec{B} + n f_i e \vec{E} + n f_i e \vec{u_i} \times \vec{B} + n(1 - f_i) e \vec{E} \tag{2.18}$$

Another simplification of this equation is possible since the plasma has been assumed neutral, or $\vec{E}$ is zero. Plasma neutrality also implies that the total current consists only of conduction current as the total convection current is zero. Thus, the total current is

$$\vec{J} = \sum_{\alpha} \vec{J_\alpha} = n_i e \vec{u}_i - n_e e \vec{u}_e = \vec{j}.$$

or,

$$\vec{J} = \sum_{\alpha} \vec{J_\alpha} = n f_i e \vec{u}_i - n f_i e \vec{u}_e = \vec{j}.$$

In short, we write the conservation of momentum as,

$$\frac{\partial(\rho \vec{u})}{\partial t} + \vec{\nabla} \cdot \rho \vec{u}\vec{u} + \vec{\nabla} \cdot \overleftrightarrow{p} = \vec{j} \times \vec{B} \tag{2.19}$$

It should be noted that the pressure tensor, $\overleftrightarrow{p}$ as the sum of the stress tensor and an identity matrix multiplying the scalar pressure, which is itself the sum of the scalar pressures for the species. The more complete conservation of momentum is written as,

$$\frac{\partial(\rho \vec{u})}{\partial t} + \vec{\nabla} \cdot \rho \vec{u}\vec{u} + \vec{\nabla} p + \overleftrightarrow{\tau} = \vec{j} \times \vec{B} \tag{2.20}$$

Conservation of Energy

We obtain the conservation of energy equation by taking the momentum equation given by Eqn (2.20), multiplying through by the dot product of the fluid velocity, using the continuity of density, and the adiabatic relation $\frac{\partial p}{\partial \rho^\gamma} = 0$ to write the energy equation for an adiabatic fluid. This is equivalent to summing over the species in the conservation of energy derived from the Boltzmann equation.

$$\frac{\partial(\epsilon + \frac{1}{2}\rho u^2)}{\partial t} + \nabla \cdot \left(\epsilon + \frac{1}{2}\rho v^2) \vec{u} \right) + \vec{\nabla} \cdot \vec{q} + \nabla \cdot \left(\vec{u} \cdot \overleftrightarrow{\tau} \right) + \nabla \cdot (p \vec{u}) = \vec{E} \cdot \vec{j} \tag{2.21}$$

where the total internal energy is $\epsilon = \epsilon_i + \epsilon_e + \epsilon_n$, and the total heat flux vector is $\vec{q} = \vec{q}_i + \vec{q}_e + \vec{q}_n$. In addition, we have used the definition of the current density and pressure tensor presented in the previous section to help simplify our expression for the conservation of energy equation, Eqn (2.21).

2.4.3 Equation of Motion

Again, the equation of motion for a single fluid is derived from two fluid model provided with the Boltzmann equation. Let us begin by introducing the ion and electron momentum conservation equations.

$$n_i m_i \left(\frac{\partial \vec{u_i}}{\partial t} + \left(\vec{u_i} \cdot \vec{\nabla} \right) \vec{u_i} \right) = -\vec{\nabla} \cdot \vec{\bar{P}}_i + n_i e \left(\vec{E} + \vec{u_i} \times \vec{B} \right) + \vec{R_{ei}} - n_i m_i \vec{u_i} \vec{\nu_{in}} \tag{2.22}$$

$$n_e m_e \left(\frac{\partial \vec{u_e}}{\partial t} + \left(\vec{u_e} \cdot \vec{\nabla} \right) \vec{u_e} \right) = -\vec{\nabla} \cdot \vec{\bar{P}}_e + n_e e \left(\vec{E} + \vec{u_e} \times \vec{B} \right) - \vec{R_{ei}} - n_e m_e \vec{u_e} \vec{\nu_{en}} \tag{2.23}$$

where n_i, n_e, m_e, m_i, $\vec{u_i}$, $\vec{u_e}$ represent the particle densities, masses, and average velocities of ion and electrons, respectively. Also, e is the electron charge, ν_{in} and ν_{en} are the momentum transfer collision frequencies to neutral atoms, $\vec{E}$ and $\vec{B}$ are the electric and magnetic fields, $\vec{R_{ei}}$ is the rate of momentum transfer from electrons to ions, and $\vec{\nabla} \cdot \vec{\bar{P}}_i$ and $\vec{\nabla} \cdot \vec{\bar{P}}_e$ are the divergences of the ion and electron pressure tensors.

Next, we can now rewrite Eqn (2.22, 2.23) by changing the ion and electron number densities, and momentum transfer collision frequencies in terms of the ion fraction and fluid number density.

$$\begin{aligned} \left(\frac{\partial \vec{u_i}}{\partial t} + \left(\vec{u_i} \cdot \vec{\nabla} \right) \vec{u_i} \right) = & -\vec{\nabla} \cdot \vec{\bar{P}}_i + f_i n e \left(\vec{E} + \vec{u_i} \times \vec{B} \right) \\ & + \vec{R_{ei}} - n_i m_i \vec{u_i} C n \left(1 - f_i \right) \end{aligned} \tag{2.24}$$

$$\begin{aligned} f_i n m_e \left(\frac{\partial \vec{u_e}}{\partial t} + \left(\vec{u_e} \cdot \vec{\nabla} \right) \vec{u_e} \right) = & -\vec{\nabla} \cdot \vec{\bar{P}}_e + f_i n e \left(\vec{E} + \vec{u_e} \times \vec{B} \right) \\ & - \vec{R_{ei}} - n_i m_e \vec{u_e} C n \left(1 - f_i \right) \left(\frac{m_e}{m_i} \right)^{0.5} \end{aligned} \tag{2.25}$$

Recall that density can be written as the sum of its constituents, namely ions, electrons and neutrals.

$$\rho = n_i m_i + n_e m_e + n_n \left(m_i + Z m_e \right) \tag{2.26}$$

where Z is the degree of ionization. We can utilize the relationships between a constituent number density and the fluid density to write,

$$\rho = f_i n m_i + f_i n m_e + \left(1 - f_i \right) n \left(m_i + m_e \right)$$

and grouping like terms we have,

$$\begin{aligned}\rho &= f_i n\left(m_i + m_e\right) + \left(1 + f_i\right) n\left(m_i + m_e\right) \\ &= \left(m_i + m_e\right) n\left(f_i + 1 - f_i\right)\end{aligned}$$

with the final solution for the density of a singly ionized flow,

$$\rho = n\left(m_i + m_e\right) \cong n m_i, m_e \ll m_i \tag{2.27}$$

We can write the gross, single-fluid flow velocity as the sum contributions from all three constituents as,

$$\vec{\hat{u}} = \frac{\left(n_i m_i \vec{u_i} + n_e m_e \vec{u_e} + n_n\left(m_i + m_e\right)\vec{u_n}\right)}{\rho} \tag{2.28}$$

We can, as with density, rewrite to replace constituent densities and the fluid density, or

$$\vec{\hat{u}} = \frac{\left(f_i n m_i \vec{u_i} + f_i n m_e \vec{u_e} + \left(1 - f_i\right) n\left(m_i + m_e\right)\vec{u_n}\right)}{\rho}$$

Collecting terms and reducing we can write,

$$\vec{\hat{u}} = \frac{\left(f_i n m_i\left(\vec{u_i} - \vec{u_n}\right) + f_i n m_e\left(\vec{u_e} - \vec{u_n}\right) + n\left(m_i + m_e\right)\vec{u_n}\right)}{\rho}$$

Finally, multiplying through by $\frac{m_i}{m_i}$ we have the final form of the equation, or

$$\vec{\hat{u}} = f_i\left(\vec{u_i} - \vec{u_n}\right) + f_i\frac{m_e}{m_i}\left(\vec{u_e} - \vec{u_n}\right) + \left(1 + \frac{m_e}{m_i}\right)\vec{u_n} \tag{2.29}$$

We expect this generalized equation will reduce to the more notable single-fluid equation of motion for fully, singly ionized ($f_i = 1\, Z = 1$) flow. We can drop the neutral velocity, $\vec{u_n}$, as it goes by definition to zero which drops the third term completely, leaving us with the classical result,

$$\vec{\hat{u}} = \vec{u_i} + \frac{m_e}{m_i}\vec{u_e}$$

2.4.4 Generalized Ohm's Law

We multiply Eqn (2.22–2.23) by m_e and m_i, respectively and subtract the equations from each other.

$$\begin{aligned} f_i n m_i m_e \left(\frac{\partial \vec{u_i}}{\partial t} + \left(\vec{u_i} \cdot \vec{\nabla} \right) \vec{u_i} \right) = & - m_e \vec{\nabla} \cdot \vec{\vec{P}}_i + f_i n m_e e \left(\vec{E} + \vec{u_i} \times \vec{B} \right) \\ & + m_e \vec{R_{ei}} - f_i n^2 m_i m_e \vec{u_i} C \left(1 - f_i \right) \end{aligned} \tag{2.30}$$

$$\begin{aligned} f_i n m_e m_i \left(\frac{\partial \vec{u_e}}{\partial t} + \left(\vec{u_e} \cdot \vec{\nabla} \right) \vec{u_e} \right) = & - m_i \vec{\nabla} \cdot \vec{\vec{P}}_e + f_i n m_i e \left(vecE + \vec{u_e} \times \vec{B} \right) \\ & - m_i \vec{R_{ei}} - f_i n^2 m_i m_e \vec{u_e} C \left(1 - f_i \right) \left(\frac{m_e}{m_i} \right)^{0.5} \end{aligned} \tag{2.31}$$

In order to simplify our analysis of Eqn (2.30–2.31) we will examine the LHS and RHS separately. We begin with the LHS where we can write,

$$LHS = f_i n m_i m_e \left(\left(\frac{\partial \vec{u_i}}{\partial t} + \left(\vec{u_i} \cdot \vec{\nabla} \right) \vec{u_i} \right) - \left(\frac{\partial \vec{u_e}}{\partial t} + \left(\vec{u_e} \cdot \vec{\nabla} \right) \vec{u_e} \right) \right) \tag{2.32}$$

Next, we examine the RHS,

$$\begin{aligned} RHS = & 2 \vec{R_{ei}} \left(m_i + m_e \right) - m_e \vec{\nabla} \cdot \vec{\vec{P}}_i + m_i \vec{\nabla} \cdot \vec{\vec{P}}_e \\ & + f_i n e \left(\vec{E} \left(m_i + m_e \right) + \left(m_e \vec{u_i} - m_i \vec{u_e} \right) \times B \right) \\ & + f_i n^2 m_i m_e C \left(1 - f_i \right) \left(\vec{u_e} \left(\frac{m_e}{m_i} \right)^{0.5} - \vec{u_i} \right) \end{aligned} \tag{2.33}$$

The magnetic term (MT) can be further simplified by adding and subtracting from the velocity term of the magnetic curl the following terms: $m_i \vec{u_i}$; $m_e \vec{u_e}$; $m_i \vec{u_n}$; and, $m_e \vec{u_n}$.

$$\begin{aligned} MT = & f_i n e \left(m_e \vec{u_i} - m_i \vec{u_e} \right. \\ & + m_i \vec{u_i} + m_e \vec{u_e} + m_i \vec{u_n} + m_e \vec{u_n} \\ & \left. - m_i \vec{u_i} - m_e \vec{u_e} - m_i \vec{u_n} - m_e \vec{u_n} \right) \times \vec{B} \end{aligned}$$

We arrange the terms in such a manner that we introduce Eqn (2.29) and the definition of the current density, $\vec{J} = n e \left(\vec{u_i} - \vec{u_e} \right)$.

$$\begin{aligned} MT = & f_i n e \left(m_e \left(\vec{u_e} - \vec{u_n} \right) + m_i \left(\vec{u_i} - \vec{u_n} \right) + \left(m_i + m_e \right) \vec{u_n} \right. \\ & \left. - m_i \left(\vec{u_i} - \vec{u_e} \right) + m_e \left(\vec{u_i} - \vec{u_e} \right) \right) \times \vec{B} \end{aligned}$$

Due to like terms we can drop the term $m_e\left(\vec{u_i}-\vec{u_e}\right)$ as $m_e \ll m_i$. We multiply through by $\frac{m_i}{m_i}$ to write,

$$\begin{aligned} MT =& f_i n e m_i \left(\left(\vec{u_i}-\vec{u_n}\right) + \frac{m_e}{m_i}\left(\vec{u_e}-\vec{u_n}\right) + \left(1+\frac{m_e}{m_i}\right)\vec{u_n} \right. \\ & \left. -\left(\vec{u_i}-\vec{u_e}\right) \right) \times \vec{B} \end{aligned}$$

Taking into account the definition for $\vec{J}$ and $\vec{\hat{u}}$ the final equation for the magnetic term is,

$$MT = f_i \rho e \vec{\hat{u}} \times \vec{B} - f_i m_i \vec{J} \times \vec{B} \tag{2.34}$$

As a sanity check we can easily verify that for the classical case of a fully, singly ionized single-fluid flow the above equation reduces to,

$$MT = \rho e \vec{\hat{u}} \times \vec{B} - m_i \vec{J} \times \vec{B}$$

We note that since $m_e \ll m_i$ then $P = P_e + P_i$. And if we include Eqn (2.34) we can write a much simplified RHS as,

$$\begin{aligned} RHS =& 2\vec{R_{ei}}\left(m_i+m_e\right) - m_i \vec{\nabla}\cdot\vec{\vec{P}}_e \\ & + f_i \rho e \vec{E}\left(1+\frac{m_e}{m_i}\right) + f_i \rho e \vec{\hat{u}} \times \vec{B} - f_i m_i \vec{J} \times \vec{B} \\ & + f_i n^2 m_i m_e C \left(1-f_i\right)\left(\vec{u_e}\left(\frac{m_e}{m_i}\right)^{0.5} - \vec{u_i}\right) \end{aligned}$$

Dividing both the LHS and RHS through by $n m_i e$ we write,

$$LHS = \frac{f_i n m_e \left(\left(\frac{\partial \vec{u_i}}{\partial t} + \left(\vec{u_i}\cdot\vec{\nabla}\right)\vec{u_i}\right) - \left(\frac{\partial \vec{u_e}}{\partial t} + \left(\vec{u_e}\cdot\vec{\nabla}\right)\vec{u_e}\right)\right)}{ne}$$

$$\begin{aligned} RHS =& \frac{2\vec{R_{ei}}}{ne} - \frac{\vec{\nabla}\cdot\vec{\vec{P}}_e}{ne} + f_i\vec{E} + f_i\vec{\hat{u}}\times\vec{B} - f_i\frac{\vec{J}\times\vec{B}}{ne} \\ & + \frac{f_i n m_e C\left(1-f_i\right)}{e}\left(\vec{u_e}\left(\frac{m_e}{m_i}\right)^{0.5} - \vec{u_i}\right) \end{aligned}$$

Let us now recombine the LHS and RHS equations, writing the electric and magnetic parts in terms of everything else and dividing through by f_i, or

$$\vec{E}+\vec{\hat{u}}\times\vec{B}=\frac{1}{ne}\Bigg[m_e n\left(\frac{\delta\vec{u_i}}{\delta t}-\frac{\delta\vec{u_e}}{\delta t}\right)+\frac{\overbrace{\vec{J}\times\vec{B}}^{Hall}}{f_i}-\frac{\overbrace{\vec{\nabla}\cdot\vec{P_e}}^{diamagnetic}}{f_i}-\frac{\overbrace{2R_{ei}}^{resistive}}{m_i f_i}$$
$$-\underbrace{Cn^2 m_e\left(1-f_i\right)\left(\vec{u_e}\left(\frac{m_e}{m_i}\right)^{0.5}-\vec{u_i}\right)}_{ionization\leftrightarrow recombination}\Bigg] \tag{2.35}$$

As always we verify our results with the more established fully ionized, single-fluid flow where $f_i = 1$, and we can write the generalized Ohm's law as,

$$\vec{E}+\vec{\hat{u}}\times\vec{B}=\tfrac{1}{ne}\left[m_e n\left(\tfrac{\delta\vec{u_i}}{\delta t}-\tfrac{\delta\vec{u_e}}{\delta t}\right)+\tfrac{\vec{J}\times\vec{B}}{f_i}-\tfrac{\vec{\nabla}\cdot\vec{P_e}}{f_i}-\tfrac{2R_{ei}}{m_i f_i}\right] \tag{2.36}$$
$$\Vert$$
$$\textit{ideal conditions}$$
$$\Downarrow$$
$$\vec{E}+\vec{\hat{u}}\times\vec{B}\approx 0$$

2.4.5 Maxwell Equations

Maxwell's equations complete the single-fluid magnetohydrodynamics equations, providing a means to predict the electric and magnetic field evolutions, interactions, and propagations.

$$\nabla\cdot\mathbf{E}=\frac{1}{\epsilon_0}\rho^q \tag{2.37}$$

$$\nabla\cdot\mathbf{B}=0 \tag{2.38}$$

$$\nabla\times\mathbf{E}=-\frac{\partial\mathbf{B}}{\partial t} \tag{2.39}$$

$$\nabla\times\mathbf{B}=\mu_0\mathbf{j}+\mu_0\epsilon_0\frac{\partial\mathbf{E}}{\partial t} \tag{2.40}$$

where the later two are more commonly known as Faraday's Law and Ampere's Law, respectively[7].

Plasma is assumed to be made of electrons, ions, and neutrals. More importantly, we assume 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 (2.37) 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 (2.40) can be neglected. With these approximations Maxwell's equations become

$$\nabla \cdot \mathbf{E} = 0 \tag{2.41}$$

$$\nabla \cdot \mathbf{B} = 0 \tag{2.42}$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \tag{2.43}$$

$$\nabla \times \mathbf{B} = \mu_0 \mathbf{j} \tag{2.44}$$

2.4.6 Equation of State

An important ingredient in combining all the equations presented above is some *equation of state*, which relates the bulk properties of a flow, namely pressure, density and temperature. Typically the *perfect gas law* is applied,

$$p = R\rho T$$

[7]It is misleading to state the above set of equations are the complete set of Maxwell's equations since only the later two of the above equations are necessary to define a electro-magnetic system. Eqn (2.37) is obtained by taking the divergence of Ampere's Law and using the relation $\frac{\partial \rho}{\partial t} = -\vec{\nabla} \cdot \vec{J}$. Eqn (2.38) is obtained by taking the divergence of Faraday's law, and has the interesting interpretation that Mother Nature does not tolerate magnetic monopoles. Nonetheless, these two equations, while superfluous, are extremely useful.

where p is pressure, $R = \Re/MW$ is the specific gas constant where $\Re$ is the universal gas constant and MW is the molecular weight, ρ is density, and T is temperature. Firstly, we can also use *adiabatic law* that states,

$$\frac{d}{dt}\left(\frac{p}{\rho^{\gamma}}\right) = 0$$

where $\gamma = c_p/c_v$ is the ratio of specific heats. Alternatively, *isothermal law* might be more applicable, in which case

$$\tfrac{d}{dt}\left(\tfrac{p}{\rho}\right) = 0 \tag{2.45}$$

or

$$p = n\,(T_i + T_e) \tag{2.46}$$

where T_e, T_i are constant.

2.4.7 Complete MHD Set of Equations

The equations of motion (Eqn (2.24, 2.25)), generalized Ohm's Law (Eqn (2.36)), conservation of momentum equation (Eqn (2.20)), and conservation of energy equation (Eqn (2.21) together with Maxwell's equations (Eqn (2.41–2.44)) describe the single fluid model used in this thesis. They can be grouped in a system of partial differential equations that is expressed as

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

$$\nabla\cdot\begin{bmatrix} 0 \\ \mu\overleftrightarrow{\tau} \\ \overleftrightarrow{E}_{res} \\ \mu\vec{v}\cdot\overleftrightarrow{\tau} - \frac{1}{\mu_0^2}[\overleftrightarrow{\eta}\cdot(\vec{\nabla}\times\vec{B})]\times\vec{B} + \overleftrightarrow{\kappa}\cdot\vec{\nabla}T \end{bmatrix} \tag{2.47}$$

Eqn (2.47) has eight partial differential equations and nine unknowns. The unknowns are density (ρ), three components of velocity vector (v), three components of the magnetic field (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 a fluid. 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 (2.47).

2.5 Thermal Diffusion

The thermal diffusion only affects the energy term of the parabolic system of PDEs. Let us begin with a vigorous derivation of the thermal diffusion in cartesian coordinates. It should be noted that while WARP3 is globally curvilinear it is sufficient to consider it cartesian at the local level of individual cells. Therefore, the following derivation is directly applicable to WARP3.

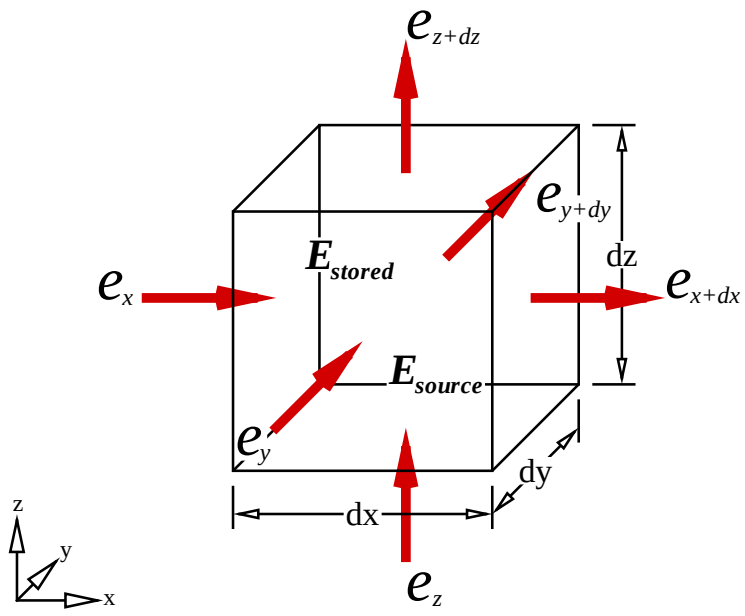


Figure 2.1: Infinitesimally (differential) control volume

Consider a homogenous medium with no bulk motion (advection). We will apply the conservation of energy to an infinitesimally small control volume, $dxdydz$, shown in Figure 2.1. Knowing the conduction heat rates at surfaces x, y, and z, and that the heat conduction moves in a direction perpendicular to each surface, then the heat conduction rate at the other surfaces, namely $x + dx$, $y + dy$, and $z + dz$ are given by Taylor Series expansion about the point (x, y, z), or

$$e_{x+dx} = e_x + \frac{\partial e_x}{\partial x}dx + \frac{\partial^2 e_x}{\partial x^2}\frac{dx^2}{2!} + H.O.T. \tag{2.48}$$

$$e_{y+dy} = e_y + \frac{\partial e_y}{\partial y}dy + \frac{\partial^2 e_y}{\partial y^2}\frac{dy^2}{2!} + H.O.T. \tag{2.49}$$

$$e_{z+dz} = e_z + \frac{\partial e_z}{\partial z}dz + \frac{\partial^2 e_z}{\partial z^2}\frac{dz^2}{2!} + H.O.T. \tag{2.50}$$

We simplify the above equations by approximating the Taylor Series to first-order, writing

$$e_{x+dx} \approx e_x + \frac{\partial e_x}{\partial x}dx \tag{2.51}$$

$$e_{y+dy} \approx e_y + \frac{\partial e_y}{\partial y}dy \tag{2.52}$$

$$e_{z+dz} \approx e_z + \frac{\partial e_z}{\partial z}dz \tag{2.53}$$

As shown in Figure 2.1, there is energy within the control volume related to the energy sink/sources and stored energy. The former is expressed as,

$$E_{source} = \frac{\partial e}{\partial t}dxdydz \tag{2.54}$$

where this simply states the time rate of change of energy generated/dissipated within the said control volume. The later energy is expressed as,

$$E_{stored} = \rho c_p \frac{\partial T}{\partial t}dxdydz \tag{2.55}$$

where it indicates the time rate of change of the thermal energy per unit volume. Applying the conservation of energy, all the energy terms must balance. Quite simply, this is

$$E_{in} + E_{source} - E_{out} = E_{stored} \tag{2.56}$$

Expanding out with the definition of these said energies, we have more explicitly

$$e_x + e_y + e_z + \frac{\partial e}{\partial t} dx\,dy\,dz - e_{x+dx} - e_{y+dy} - e_{z+dz} = \rho c_p \frac{\partial T}{\partial t} dx\,dy\,dz \tag{2.57}$$

Next, the appropriate terms are replaced by the relationships indicated by Eqn (2.51–2.53).

$$-\frac{\partial e_x}{\partial x} dx - \frac{\partial e_y}{\partial y} dy - \frac{\partial e_z}{\partial z} dz + \frac{\partial e}{\partial t} dx\,dy\,dz = \rho c_p \frac{\partial T}{\partial t} dx\,dy\,dz \tag{2.58}$$

From Fourier's Law, the heat conduction time rate of changes are

$$e_x = -\kappa dy dz \frac{\partial T}{\partial x} \tag{2.59}$$

$$e_y = -\kappa dx dz \frac{\partial T}{\partial y} \tag{2.60}$$

$$e_z = -\kappa dx dy \frac{\partial T}{\partial z} \tag{2.61}$$

$$\tag{2.62}$$

The final, most general form of the thermal diffusion equation is

$$\frac{\partial}{\partial x}\left(\kappa \frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(\kappa \frac{\partial T}{\partial y}\right) + \frac{\partial}{\partial z}\left(\kappa \frac{\partial T}{\partial z}\right) + \frac{\partial e}{\partial t} = \rho c_p \frac{\partial T}{\partial t} \tag{2.63}$$

2.5.1 Thermal Conductivity

We use as an approximation for thermal conductivity the Spitzer model[8] [20]

$$\kappa = 1.96 \times 10^{-9} \epsilon_T(Z) \delta_T(Z) \frac{T^{5/2}}{Z \ln \Lambda} \tag{2.64}$$

where T is temperature in Kelvin, Z is the mean ion charge[9], $\ln \Lambda$ is the Coulomb logarithm[10]. The coefficient $\epsilon_T(Z)$ and $\delta_T(Z)$ are correction factors, where for $Z = 1$, then $\epsilon_T = 0.419$ and $\delta_T = 0.225$.

However, when the plasma is magnetized the thermal conductivity is no longer scaler, but becomes very anisotropic, i.e. thermal conductivity occurs easily along magnetic field lines, and with difficulty transverse to these field lines. While the Spitzer

[8]Note that this model assumes fully ionized, non-magnetized plasma.

[9]In the case of WARP3, the ion charge is always unity

[10]For most, if not all, plasmas of interest to the researcher are on the order of 10^1, where a mean value of 14 is typical.

model is still applicable[11], we relate these parallel and transverse modes where we assume in WARP3 that this thermal conductivity mirrors the electric resistivity tensor, $\overleftrightarrow{\sigma}$, such that it lays dominantly along the magnetic field lines and weakly perpendicular to such said magnetic field lines. This is to say, the charged species (electrons and ions) gyrate around the magnetic field lines by a radius term called, aptly enough, the gyro radius. This motion along with the drift velocity along the magnetic field lines comes to dominant all charged species motion. However, according to Goldston *et al* [8] in regard to thermal conductivity along field lines, it is mainly attributable to electrons; whereas transverse transport is dominated by ions only when sufficiently large amounts of energy, such as collisions with another ion, perturbs a particle to "jump" to a nearby field line a distance on the order of an ion Larmor radius.

The parallel and perpendicular thermal conductivities are then related to each other by

$$\frac{\kappa_{||}}{\kappa_{\perp}} = 1 + \frac{\tau_{ee}}{\tau_{pe}} \tag{2.65}$$

where τ_{ee} is the electron-electron reaction rate, and τ_{pe} is the proton-electron reaction rate which is also known as the electron plasma period, or $\tau_{pe} = 2\pi/\omega_{pe}$. Typical values for these two rates are on the order of 10^{-5} and 10^{-12}, respectively. Therefore, the ratio of the parallel thermal conductivity to the perpendicular conductivity is then on the order of 10^{7}; a value used directly in WARP3.

In vector form thermal conductivity is expressed as,

$$\overleftrightarrow{\kappa} = \kappa_{||}\vec{b}\vec{b} + \kappa_{\perp}(\overleftrightarrow{I} - \vec{b}\vec{b}) \tag{2.66}$$

where $\vec{b}$ is the unit vector oriented along the magnetic field, $\kappa_{||}$ is the thermal conductivity parallel to the magnetic field, and $\kappa_{\perp}$ is the thermal conductivity transverse to the magnetic field.

[11]It is important to emphasize that Spitzer conductivity is valid along magnetic field lines, but requires a correction factor for perpendicular conductivity.

Also, Eqn (2.66) can be written in full tensor form as

$$\overleftrightarrow{\kappa} = \begin{bmatrix} \kappa_x & 0 & 0 \\ 0 & \kappa_y & 0 \\ 0 & 0 & \kappa_z \end{bmatrix}. \tag{2.67}$$

It is illustrative to mention a simple case when the magnetic field lies parallel to the z-direction, $\vec{B} = \|B\|\hat{k}$, where the tensor yields

$$\overleftrightarrow{\kappa} = \begin{bmatrix} \kappa_\perp & 0 & 0 \\ 0 & \kappa_\perp & 0 \\ 0 & 0 & \kappa_{\|} \end{bmatrix}. \tag{2.68}$$

2.6 Peclet Number

In this section, we briefly examine the dimensionless independent heat transfer parameter, or *Peclet Number*. It is defined as,

$$Pe = \frac{uL}{\alpha} \tag{2.69}$$

where u is velocity, L is some characteristic length, α is the thermal diffusivity. Alternatively, the Peclet number can be written as the product between the Reynolds number, $Re = uL/\nu$, and Prandtl number, $Pr = c_p\mu/\kappa = \nu/\alpha$; where ν is the kinematic viscosity, c_p is the specific heat at constant pressure, μ is viscosity, and κ is the thermal conductivity. Finally, recalling the definition for the thermal diffusivity as $\alpha = \kappa/\rho c_p$, then Eqn (2.69) as

$$Pe = \frac{uL\rho c_p}{\kappa} \tag{2.70}$$

The Peclet number provides useful insight the flow pertaining to its thermal characteristics. Both Eqn (2.69) and Eqn (2.70) indicate that for large values transverse motion (advection) dominates. Conversely, for small numbers the flow is dominated by random motion (conduction).

2.7 Multiple Temperatures

Now that we have defined a set of PDEs in Eqn (2.47), we wish to determine how the parabolic fluxes given by the RHS of the equation affect the species' temperature evolution. In particular, we are interested in predicting the species (neutrals, ions, and electrons) temperature evolution in order to develop a more accurate ion fraction evolution. Toward this end, we are strictly interested in the evolution of the electron temperature, as this is the driving source of energy that characterizes the ion fraction evolution. Nevertheless, due to the relative simplicity of tracking all three (3) constituents, WARP3 reports all of them.

The classical single-fluid MHD model assumes that the electron and ion temperatures are the same. However, due to a variety of factors that include differences in

mass and mechanisms for transport, this assumption is only valid for a relatively small region of physical phenomenon. We have introduced independent time evolution of the species, allowing for each species' growth to be affected by the appropriate mechanisms and interspecies transmissions.

These temperature evolutions mirror the energy evolution of the material flow, and are proportional to the density fraction of each species. We define the density fraction as,

$$fi = \frac{n_i}{n_n + ni} \tag{2.71}$$

$$f_n = \frac{n_n}{n_n + ni} = 1 - f_i \tag{2.72}$$

where we assume that the degree of ionization, Z, equals 1. Because we are only interested in singly ionized gas, we can also introduce the following relationship,

$$f_e = Z f_i \rightarrow f_i \tag{2.73}$$

As will be shown later, it is important to realize that the bulk pressure calculated in pervious versions of WARP3 can be expressed as the sum of the partial, or constituent, pressures.

$$p_{bulk} = p_n + p_i + p_e \tag{2.74}$$

2.7.1 Mechanisms for Transmission

We will now introduce the mechanisms for transmission of energy to each specie. There are three (3) main processes that involve all the parabolic fluxes computed with WARP3; namely, viscous, resistive, and thermal diffusion.

Viscosity

Viscous interaction occurs between only neutrals and ions when the relative velocity between the two species is not zero.

$$\left.\frac{\partial p_n}{\partial t}\right|_{visc} = (\gamma - 1) f_n P_{visc} \tag{2.75}$$

$$\left.\frac{\partial p_i}{\partial t}\right|_{visc} = (\gamma - 1) f_i P_{visc} \tag{2.76}$$

Resistivity

Resistivity for the single-fluid MHD model is more precisely the electrical resistivity. As such, only the electrons are affected by this mechanism.

$$\left.\frac{\partial p_e}{\partial t}\right|_{visc} = (\gamma - 1) P_{res} \tag{2.77}$$

Thermal Diffusion

Each species conducts thermal energy, and so each species is affected.

$$\left.\frac{\partial p_n}{\partial t}\right|_{cond} = (\gamma - 1) P_{cond_n} \tag{2.78}$$

$$\left.\frac{\partial p_i}{\partial t}\right|_{cond} = (\gamma - 1) P_{cond_i} \tag{2.79}$$

$$\left.\frac{\partial p_e}{\partial t}\right|_{cond} = (\gamma - 1) P_{cond_e} \tag{2.80}$$

Bremßtrahlung Radiation

The Coulomb interaction results in the acceleration of an electron toward an ion. Furthermore, we know from electromagnetic theory that whenever an electron is accelerated it will emit radiation in the form of photons. This loss mechanism is termed, "Bremßtrahlung radiation". It should be obvious that only electrons will be effected by this loss mechanism.

$$\left.\frac{\partial p_e}{\partial t}\right|_{rad} = (\gamma - 1) P_{rad} \tag{2.81}$$

Adiabatic Compression

Having examined the various energy transfer mechanisms, let us characterize the material energy of the flow. Recall that the total energy is,

$$e = \frac{p}{\gamma - 1} + \rho\frac{\vec{v}\cdot\vec{v}}{2} + \frac{\vec{B}\cdot\vec{B}}{2} \tag{2.82}$$

where the first term is the contribution from pressure, the second term is the kinetic contribution, and the third term is the magnetic contribution. The last two terms do not contribute to the species evolution; however, the first term holds an important mechanism to correctly calculate the species evolution.

The above mechanisms are strictly parabolic in nature, whereas the ideal single-fluid MHD equations are strictly hyperbolic in nature. It should be expected that the hyperbolic component of the system will also contribute to the species evolution. In short, there will be a difference between the total energy of the species when compared to the total energy of the flow that includes both hyperbolic and parabolic flux contributions. And this difference can be expressed as the first term of Eqn (2.82).

Any adiabatic process is defined as a process that precludes any two systems from interacting thermally. Adiabatic compression is the volumetric compression of the system without an increase or decrease in the thermal properties of said system. Therefore, each species in the flow will be affected equally by this compression.

This is solved for in the algorithm by computing the total energy due to parabolic and radiative mechanisms, and then subtracted from the total energy computed for the hyperbolic process. The remainder is energy due to this said adiabatic compression, and is added to each species equally.

Summary of Mechanisms

The following summarizes the various mechanisms for energy transmission for each species, thereby proving a complete of the system.

For electrons the mechanisms are,

$$\left.\frac{\partial p_e}{\partial t}\right|_{total} = (\gamma - 1)\left(P_{res} + P_{cond_e} + P_{rad}\right) \tag{2.83}$$

For ions the mechanisms are,

$$\left.\frac{\partial p_i}{\partial t}\right|_{total} = (\gamma - 1) f_i P_{visc} + (\gamma - 1) P_{cond_i} \tag{2.84}$$

Similarly, neutrals' mechanisms are,

$$\left.\frac{\partial p_n}{\partial t}\right|_{total} = (\gamma - 1) f_n P_{visc} + (\gamma - 1) P_{cond_n} \tag{2.85}$$

Finally, the generation of power associated with each mechanism needs to be expressed. Udrea's dissertation [23] provides us the source of viscosity and resistivity as,

$$P_{visc} = \frac{1}{ReAl} \vec{\nabla} v \cdot \overleftrightarrow{\tau} \tag{2.86}$$

$$P_{res} = -\frac{1}{RmAl} \vec{\nabla} \cdot \overleftrightarrow{\eta} \cdot (\vec{\nabla} \times \vec{B}) \times \vec{B} \tag{2.87}$$

where Rm is the magnetic Reynolds number, Al is the Alfvén number, $\overleftrightarrow{\tau}$ is the viscosity tensor, and $\overleftrightarrow{\eta}$ is the electrical resistivity. Section 2.4.7 reveals that thermal conductivity is

$$P_{cond} = \frac{m_{ion}}{2PeAl} \vec{\nabla} \cdot \overleftrightarrow{\kappa} \cdot (\vec{\nabla} T_n + \vec{\nabla} T_i + \vec{\nabla} T_e) \tag{2.88}$$

where m_{ion} is the mass of the ion, Pe is the Péclet number, and $\overleftrightarrow{\kappa}$ is the thermal conductivity tensor. Finally, the power associated with Bremßtrahlung radiation is

$$P_{rad} = -C_{rad} Z_{eff} \left(\rho f_i\right)^2 T_e^{1/2} \tag{2.89}$$

where C_{rad} is the Bremßtrahlung radiation constant, Z_{eff} is the effective level of ionization, and T_e is the electron temperature.

Each of the non-dimensional terms are defined as

$$Al \equiv \frac{V_a}{V} \tag{2.90}$$

$$Re \equiv \frac{VL}{\nu} \tag{2.91}$$

$$Rm \equiv \frac{\mu_o L V}{\eta} \tag{2.92}$$

$$Pe \equiv \frac{LV}{k} \tag{2.93}$$

where the characteristic variables are length L, velocity V, Alfén velocity $(V_a = B/\sqrt{\mu_o \rho})$, kinematic viscosity ν, electrical resistivity η, and thermal diffusivity k $(k = \kappa/\rho c_p)$, and where μ_o is the permeability of free space $(4\pi \times 10^{-7})$.

2.8 Time Dependent Ionization

In this section we will introduce the process termed *ionization*, where electrons are stripped from their atoms via some mechanism. This ionization results in the three said species that we are presently investigating.

Excitation of electrons into higher states of atoms and ionization occurs by identical mechanisms, but differing situations. Ionization is a limiting case of excitation, where a bound electron acquires enough energy so that it sufficiently leaves the atom.

There are three processes that we must distinguish from; namely, electron impact; heavy particle impact; and, photon impact. We write these three (3) processes as

$$A + e = A^+ + e + e \tag{2.94}$$

$$A + B = A^+ + B + e \tag{2.95}$$

$$A + h\nu = A^+ + e \tag{2.96}$$

where A and B denote heavy particles, e denotes electrons, and $h\nu$ denotes photons. Through the reversal of any of these three processes, recombination is achieved.

Presently, WARP3 considers only the first of these three, or in the single ionization process consisting of a gas composed of identical atoms and bombarded by electrons. We will assume that all the atoms are ionized from the ground state, and also that the process of recombination captures the electron back into the ground state.

The ion fraction rate equation is a simple stoichiometric rate equation based upon the processes outlined by Eqn (2.94).

$$\frac{dn_e}{dt} = \alpha_e n_n n_e - \beta_e n_i n_e^2 \tag{2.97}$$

where the first term of the RHS corresponds to the RHS of Eqn (2.94), the second term relates to the LHS of the same equation. An astute reader might note that the

second term of the above equation has one more n_e than is typically found in textbooks. However, this above equation and the textbooks are identical; it is only the semantics that are misleading. The difference is found in the recombination rate constant where the number of recombinations per unit volume per unit time is sometimes expressed as $Z_{rec} = b_e n_i n_e$. This relates to our equation as $b_e = beta_e n_e$. However, do note that the units for α_e and β_e are not the same, whereas b_e and α_e are the same; namely, unit volume per unit time.

2.9 Hypervelocity Flows

WARP3 was originally developed to solve problems of electrically conducting fluids found in fusion reactors such as spheromaks, tokamaks, Z-pinches, et cetera. As the ideal MHD equations are derived from the assumption of a fully ionized flow this naturally excluded flows that also were composed of neutrals along with electrons and ions. These excluded flows are typified by hypervelocity flows that occur around hypersonic vehicles, re-entry vehicles, and rockets.

However, with the addition of the author's contributions: thermal diffusion; time rate of change on ionization; and, multiple temperature effects, WARP3 is capable of solving a far greater range of flow conditions. Further to point, in order to understand the author's application of WARP3 both plasma physics and hypervelocity flow must be understood. This section will provide the necessary information so that the entirety of this project is placed in its appropriate context.

2.9.1 Definition

From a generalist's perspective, hypervelocity flows, also known as hypersonics, is not clearly delineated by anything as distinctive as a sonic boom that marks the shift from subsonic (Mach $\ll 1$) to supersonic (Mach $\gg 1$). That is to say, hypersonics does not designate a change in the fundamental nature of the transmission of information within the flow; however, it nonetheless signifies a fundamental change in the nature of the flow composition. While there is not an exact moment when the flow changes

from supersonic to hypersonic, most researchers do settle upon the value of Mach 5 as a reasonable marker. Of course, for some researchers it might be more reasonable to say the change occurs at Mach 4 or Mach 7. It all depends on the researcher's perspective and particular sensitivity to the physics at hand. Ultimately it is important to understand that the study of hypersonics involves a large range of velocities ($\approx$ Mach 5 and greater), and as Anderson [2] aptly writes, "hypersonic flow is best defined as that regime where certain physical flow phenomena become progressively more important as the Mach number is increased to higher values."

However, we still have yet to satisfactorily explain what Anderson means by "certain physical flow phenomena", or how hypersonics deviates from the study of supersonics. The answer lies, at least indirectly, within the definition of Mach number, where it is given by,

$$M = \frac{v_{flow}}{v_{sonic}} \tag{2.98}$$

which is the simple ratio of the magnitude of the flow velocity to the sonic velocity. Furthermore, the sonic velocity is

$$v_{sonic} = \sqrt{\frac{\gamma p}{\rho}} \tag{2.99}$$

where $\gamma = c_p/c_v$ is the ratio of specific heats. Therefore, by substituting Eqn (2.99) into Eqn (2.98) the Mach number is more appropriately written as

$$M = \frac{v_{flow}\sqrt{\rho}}{\sqrt{\gamma p}} \tag{2.100}$$

The final ingredient needed is a definition of pressure, p. We recall from the energy equation with the absence of magnetic energy,

$$p = (\gamma - 1)\left(e - \frac{1}{2}\rho v^2\right) \tag{2.101}$$

which simply states that pressure is proportional to the difference between kinetic energy to internal energy. Again, substituting Eqn (2.101) into Eqn (2.100), the Mach number yields,

$$M = \sqrt{\frac{1}{\gamma(\gamma-1)}}\left(\frac{\rho^{-1/2}v_{flow}}{\sqrt{e - \frac{1}{2}\rho v^2}}\right) \tag{2.102}$$

Squaring both sides, we have

$$M^2 = \frac{1}{\gamma(\gamma-1)}\left(v_{flow}^2 \frac{\rho}{e-\frac{1}{2}\rho v^2}\right) \tag{2.103}$$

$$M^2 = \frac{1}{\gamma(\gamma-1)}\left(\frac{\rho v_{flow}^2}{e} - 2\right) \tag{2.104}$$

Ultimately, the mach number is proportional to the ratio of kinetic energy to internal (thermal) energy. Another way of looking at this is to say that when the flow velocity approaches zero, all the kinetic energy is converted to thermal energy. Conversely, when the flow velocity is accelerated faster and faster it is in essence converting more and more of the thermal energy into kinetic energy.[12]

This relationship yields a very intuitive means of understanding why the temperature at the stagnation region[13] of hypersonic vehicles can reach truly astronomical temperatures, matching the temperature at the surface of Sun with values in the range of 5000 Kelvin.

As mentioned above, through the definition of the Mach number we would indirectly understand how the flow regime labelled "hypersonics" is in turn defined. The fact that flow has a large amount of energy that manifests as translational (kinetic) energy or random (thermal) energy results in a change in the chemical composition of the flow, especially dissociation and to some extent ionization.

2.9.2 Characteristics

Following from Anderson [2], let us now introduce some of the significant physics that occur within the hypersonic regime.

---

[12]The loss of thermal energy for high Mach numbers also explains how hypersonic shock tunnel (HST) operational ranges are, in part, determined. HST have fixed enthalpy as a function of the power plants. With the right conditions, too high Mach number flows result in condensation of oxygen within the test chamber that can have a variety of adverse effects.

[13]Flow velocity is zero. When the angle of attack α is zero this is typically the nose of the craft. As this angle changes, so does the stagnation point on the vehicle.

Viscous Interaction

The high amount of kinetic energy within a hypersonic flow is converted, in part, into internal energy as it is slowed through the boundary layer; a process called viscous dissipation. In turn, the temperature within the boundary region also accordingly increases. Recall that the viscosity coefficient is proportional to temperature, and so this increased temperature yields a larger boundary layer. Also, pressure normal to the surface is constant, and so the equation of state, $\rho = p/RT$, where R is the specific gas constant tells us that density becomes more rarefied in this region. Consequently, for the same mass flow rate the boundary layer must become larger to accommodate this reduced density. Both these increases in the viscous boundary layer results in quicker growth than at lower speeds.

Indeed, for compressible, laminar flow over a plate the boundary layer grows like,

$$\delta \propto \frac{M_\infty^2}{\sqrt{Re_x}}$$

Therefore, for very large Mach numbers the boundary layer grows exponentially to extremely large values. The size of this boundary layer can have very adverse effects on how a body interacts with the flow, effectively making the body cross-section appear larger to the flow than its actual physical dimensions. For high reynolds numbers where the flow is effectively inviscid, the outer inviscid flow interacts with the inner, viscid flow. This interaction, termed viscous interaction, causes a further burgeoning of the viscous boundary layer leading to a shift in the pressure distribution over the body. Ultimately, the aggregate effects are changes in lift, drag, and stability of the vehicle. Also, heat transfer and friction within the boundary layer are enhanced by viscous interaction.

High-Temperature Flows

While we only examined the translational and random energies associated with Mach number, the random energies generated near the vehicle surface due to friction can lead to excitation of oscillation energies within the molecules themselves. The oscillation of these molecules leads to the separation of complex gases such as air into

its constituent molecules, a process termed *dissociation*. Furthermore, bombardment of free electrons upon the dissociated molecules leads to the stripping of an electron from the ground state of the molecules, or ionization. For the purposes of the author's research it is assumed that the flow is fully dissociated and composed entirely of monatomic gas. However, there are other chemical reactions that can be considered such as the addition of ablative materials from the hypersonic vehicle surface mixing with the boundary layer giving rise to truly chemically reacting boundary layer.

One aspect of this high-temperature flow that is not accounted for in WARP3 is local changes in the ratio of specific heats, γ. WARP3 instead assumes a global γ that does not vary with temperature. However, at the high temperatures associated with hypersonic flow, the effect of this ratio cannot be overstated. Per Figure 1.18 of Reference [2], while the temperature at the stagnation point for increasing Mach number grows exponentially for a calorically perfect gas it is more or less linear when the gas is allowed to chemically equilibrate. In other words, where the real-world temperature of the Apollo return-capsule at Mach 36 was in the range of 11,000 Kelvin, if the flow did not chemically react and had a constant γ then the temperature would have been significantly higher[14].

[14]So much so, that without this chemical process mankind would not be have been able to go much faster than Mach 3 with present-day material properties; we would have never gotten to the Moon let alone low Earth orbit.

Chapter 3

NUMERICAL ALGORITHM

In this chapter we develop a set of numerical equations to express the physics described in the previous chapter. It is important that the reader recognize that numerical equations are algebraic in nature, and are at best second-order approximations of their real-world differential cousins. Even more subtle is that the mathematics describing the physics is itself an approximation of the bulk properties that has been found to sufficiently model the phenomenon of concern. Therefore, it is indeed dangerous to to place too much faith in our math, algebraic or otherwise, without an intuitive appreciation of Mother Nature in her own element.

3.1 Multiple Temperatures

As detailed in Section 2.7, we take note of the time rate of change in pressure of each species. We can then multiple through the time step to arrive at the change in pressure for the given iteration of all the species. Finally, the Perfect Gas Law,

$$p = R\rho T \tag{3.1}$$

where p is pressure, R is the universal gas constant divided by the molecular weight, ρ is density, and T is the material temperature. It can written more appropriately for plasmas as

$$p = RmnT \tag{3.2}$$

where ρ has been replaced by the product the mass, m, and number density, n.

Next, we are interested in determining how the LHS expands out when taking the time rate of change of pressure, or

$$\frac{\partial p}{\partial t} = Rm\left(n\frac{\partial T}{\partial t} + T\frac{\partial n}{\partial t}\right) \tag{3.3}$$

Let us simplify by writing Rm as K, and solving for ∂T, we have

$$\partial T = \frac{\frac{\partial p}{K} - T\partial n}{\rho} \tag{3.4}$$

where we have multiplied through by ∂t.

Eqn 3.4 can immediately transcribed into numerics notation as

$$T_j^{n+1} = T_j^n + \frac{\frac{p_j^n - p_j^{n-1}}{K} - T_j^n\left(n_j^n - n_j^{n-1}\right)}{\rho} \tag{3.5}$$

where the superscript denotes temporal changes, and subscript denote spatial changes, and where we have solved for the temperature at the j_{th} cell for the next $(n+1)$ time step.

3.1.1 Adiabatic Compression

As mentioned previously, the total change in energy represented by the sum of the species' change in energy will differ from the change in energy due to hyperbolic fluxes. These said hyperbolic fluxes occur regardless of the properties of the species, and is purely a volumetric compression of the entire volume. Consequently, this adiabatic compression will change the change in energy of each species equally.

The total pressure of the system is given by the energy equation,

$$p_{hyper} = (\gamma - 1)e - \frac{1}{2}(\rho v^2 + B^2) \tag{3.6}$$

And the total pressure is the sum of the partial pressures,

$$p_{para} = p_i + p_e + p_n \tag{3.7}$$

$$p_{para} = n_i T_i + n_e T_e + n_n T_n \tag{3.8}$$

The adiabatic pressure is then given by the difference of these two pressures, or

$$p_{adb} = p_{hyper} - p_{para} \tag{3.9}$$

Pressure is straightforwardly converted to temperature by dividing through by the bulk number density, which is the sum the species number densities.

$$p_{adb} = \frac{p_{hyper} - p_{para}}{n_i + n_e + n_n} \tag{3.10}$$

Finally, we update each specie's temperature, or

$$T_i = T_i + p_{adb} \tag{3.11}$$

$$T_e = T_e + p_{adb} \tag{3.12}$$

$$T_n = T_n + p_{adb} \tag{3.13}$$

3.2 Thermal Diffusion Fluxes

Thermal diffusion is very similar in nature to viscosity or electrical resistivity from the standpoint of numerics. The following section utilizes work completed for Udrea's dissertation [23]. As mentioned by Udrea, it is important that the parabolic fluxes provide a spatial accuracy at least equal to the hyperbolic fluxes. Since WARP3 is overall a second order algorithm, the parabolic fluxes need to reflect this.

As introduced in Section 2.4.7, the thermal conductivity tensor $\overleftrightarrow{\kappa}$ is given by,

$$\overleftrightarrow{\kappa} = \begin{bmatrix} \kappa_x & 0 & 0 \\ 0 & \kappa_y & 0 \\ 0 & 0 & \kappa_z \end{bmatrix}.$$

We begin by writing out the time rate of change of energy due to thermal diffusion fluxes as,

$$\frac{\partial e}{\partial t} = \frac{1}{PeAl}\vec{\nabla} \cdot (\overleftrightarrow{\kappa} \cdot \vec{\nabla} T) \tag{3.14}$$

Note that we have dropped notation for each species to simplify the overall notation, however thermal diffusion due to each species is computed according to the procedures outlined here. Next, we integrate over the volume of the cell,

$$\int_{V_C} \frac{\partial e}{\partial t} dT = \frac{1}{PeAl} \int_{V_C} \vec{\nabla} \cdot (\overleftrightarrow{\kappa} \cdot \vec{\nabla} T) dT \tag{3.15}$$

The order of the integral is reduced from volume to surface by applying the Divergence Theorem; hence the volume integral becomes a surface integral of the nature,

$$\left.\frac{\partial e}{\partial t}\right|_{ther} = \frac{1}{ReAlV_C} \int_{\sum_C} (\overleftrightarrow{\kappa} \cdot \vec{\nabla} T) d\sigma \tag{3.16}$$

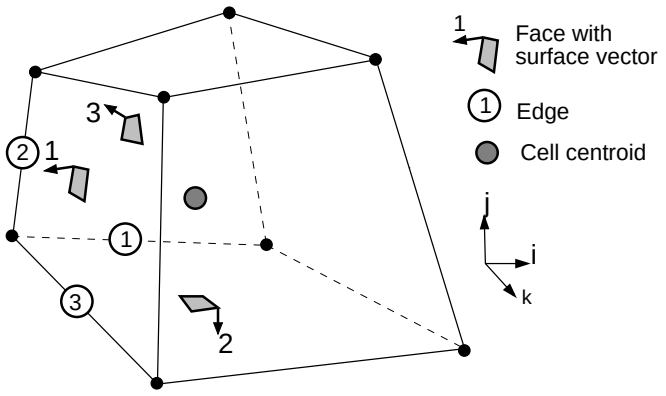


Figure 3.1: Face and vertex notation convention of a real cell used for the calculation of the parabolic fluxes. Face 1 area vector points in the negative i direction, face 2 area vector points in the negative j direction and face 3 area vector points in the negative k direction. [Taken directly from Udrea [23]]

where σ is the surface area of cell $\mathcal{C}$. The LHS of Eqn (3.16) can be approximated by a first-order derivative, or $\frac{\partial e}{\partial t} \approx (e^{n+1} - e^n)/\Delta t$. The integrals are approximated as the sum of the faces for each cell. All terms within the integral are calculated explicitly using values for each constituent temperature (ion, electron, and neutral) at time level n.

$$e^{n+1} = e^n + \frac{\Delta t}{ReAlV_{\mathcal{C}}} \sum_{i=1}^{6} (\overleftrightarrow{\kappa} \cdot \vec{\nabla} T) \vec{n}_i A_i \tag{3.17}$$

The second term of the RHS is called the thermal diffusion fluxes. A staggered grid approach is used to compute these fluxes, where the values of $\overleftrightarrow{\kappa}$ must be calculated at the cell faces of the real cells. The present methodology is compute these values on the fly as they are needed using simple averages. The face and edge notations, first introduced by Udrea [23] are re-introduced in Figures 3.1, 3.2.

The species temperatures are calculated based on the ionization fraction. These species temperatures are for the cell centers, where we first calculate the temperatures at the cell faces, or

$$T^a_{face} \approx \frac{1}{2} \left(T_{ijk} + T_{i-1jk} \right) \tag{3.18}$$

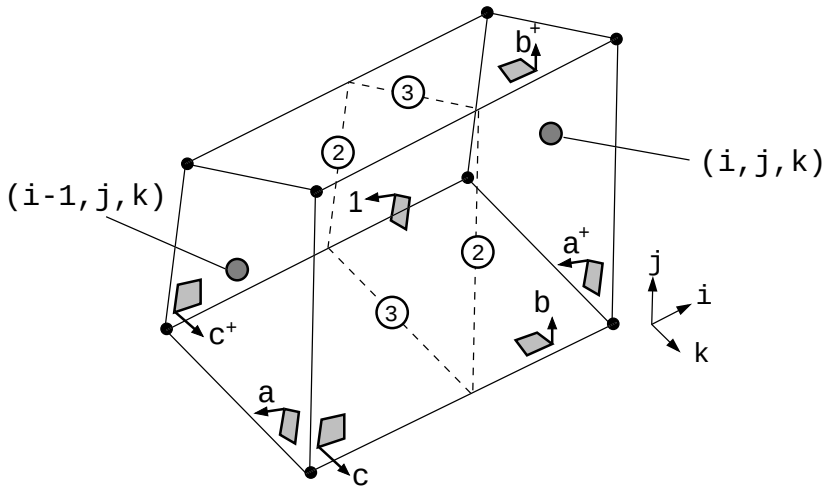


Figure 3.2: Auxiliary cell for face 1. Faces a and a^+ correspond to the i direction and pass through the centroids of cells ($i-1$,j,k) and (i,j,k) respectively. Faces b and b^+ correspond to the j direction and faces c and c^+ correspond to the k direction. [Taken directly from Udrea [23]]

Next the gradient of the temperature is calculated.

$$\vec{\nabla} T_x^a \approx \frac{A_x^a}{V_\mathcal{C}} T_{ijk}^a\Big|_{face} \tag{3.19}$$

$$\vec{\nabla} T_y^a \approx \frac{A_y^a}{V_\mathcal{C}} T_{ijk}^a\Big|_{face} \tag{3.20}$$

$$\vec{\nabla} T_z^a \approx \frac{A_z^a}{V_\mathcal{C}} T_{ijk}^a\Big|_{face} \tag{3.21}$$

$$\vec{\nabla} T_x^b \approx \frac{A_x^b}{V_\mathcal{C}} \frac{1}{2} \left(T_{ijk}^b\Big|_{face} + T_{i-1jk}^b\Big|_{face} \right) \tag{3.22}$$

$$\vec{\nabla} T_y^b \approx \frac{A_y^b}{V_\mathcal{C}} \frac{1}{2} \left(T_{ijk}^b\Big|_{face} + T_{i-1jk}^b\Big|_{face} \right) \tag{3.23}$$

$$\vec{\nabla} T_z^b \approx \frac{A_z^b}{V_\mathcal{C}} \frac{1}{2} \left(T_{ijk}^b\Big|_{face} + T_{i-1jk}^b\Big|_{face} \right) \tag{3.24}$$

$$\vec{\nabla} T_x^c \approx \frac{A_x^c}{V_\mathcal{C}} \frac{1}{2} \left(T_{ijk}^c\Big|_{face} + T_{i-1jk}^c\Big|_{face} \right) \tag{3.25}$$

$$\vec{\nabla} T_y^c \approx \frac{A_y^c}{V_\mathcal{C}} \frac{1}{2} \left(T_{ijk}^c\Big|_{face} + T_{i-1jk}^c\Big|_{face} \right) \tag{3.26}$$

$$\vec{\nabla} T_z^c \approx \frac{A_z^c}{V_\mathcal{C}} \frac{1}{2} \left(T_{ijk}^c\Big|_{face} + T_{i-1jk}^c\Big|_{face} \right) \tag{3.27}$$

Finally, the divergence of the thermal conductivity with the temperature gradient is

obtained by the product of the two multiplied by the area, or

$$\kappa \cdot \vec{\nabla} T \approx \kappa_x|_{face} \vec{\nabla} T_x A_x + \kappa_y|_{face} \vec{\nabla} T_y A_y + \kappa_z|_{face} \vec{\nabla} T_z A_z \tag{3.28}$$

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

$$V^a = \frac{1}{2}(V_{ijk} + V_{i-1jk}), \tag{3.29}$$

$$\mathbf{A}^a = \frac{1}{2}(\mathbf{A}^1_{ijk} + \mathbf{A}^1_{i-1jk}). \tag{3.30}$$

The volumes corresponding to the other five faces are

$$V^{a+} = \frac{1}{2}(V_{i+1jk} + V_{ijk}), \tag{3.31}$$

$$V^b = \frac{1}{2}(V_{ijk} + V_{ij-1k}), \tag{3.32}$$

$$V^{b+} = \frac{1}{2}(V_{ij+1k} + V_{ijk}), \tag{3.33}$$

$$V^c = \frac{1}{2}(V_{ijk} + V_{ijk-1}), \tag{3.34}$$

$$V^{c+} = \frac{1}{2}(V_{ijk+1} + V_{ijk}) \tag{3.35}$$

Similar formulas are employed for calculation of the face area vectors for each of the faces of the auxiliary cell.

The value of the thermal conductivity 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 3.2. The same is true for thermal conductivity at face a^+. However, faces b, b^+ and c, $c+$ do not pass through cell centers so that the thermal conductivity at these faces is computed using a simple average between the values at cell centers in

the corresponding directions. The thermal conductivity tensor is calculated as follows

$$\kappa^{a} = \kappa_{i-1jk} \tag{3.36}$$

$$\kappa^{a^{+}} = \kappa_{ijk} \tag{3.37}$$

$$\kappa^{b} = \frac{1}{2}(\kappa_{ij-1k} + \kappa_{ijk}) \tag{3.38}$$

$$\kappa^{b^{+}} = \frac{1}{2}(\kappa_{ijk} + \kappa_{ij+1k}) \tag{3.39}$$

$$\kappa^{c} = \frac{1}{2}(\kappa_{ijk-1} + \kappa_{ijk}) \tag{3.40}$$

$$\kappa^{c^{+}} = \frac{1}{2}(\kappa_{ijk} + \kappa_{ijk+1}) \tag{3.41}$$

A better approximation of the thermal conductivity and volumes would be to use a weighted average. However, as it already necessary that surrounding cells be approximately the same size for boundary conditions, use of simple averages would appear to be sufficient. Nonetheless, it is straightforward enough to change the simple averages to weighted, where the distance from cell face to cell center is employed.

Thus far the calculation of the temperatures and thermal conductivity tensor at face 1 has been introduced. The same methodology is used for face 2 and face 3, except that extensions are made in the j and k directions, respectively. Faces 4–6 do not need to be computed for each cell as it is done so for its neighboring cells. In summary, this process computes the thermal diffusion fluxes at faces 1–3, each in the three cartesian directions (x, y, z) and for each of the three auxiliary faces (a, b, c) is repeated for each of the three species (ions, electrons, and neutrals).

3.3 Time Dependent Ionization

As outlined in Section 2.7.1, WARP3 models the bombardment of electrons on a heavy particle given by the stoichiometric equation,

$$A + e = A^{+} + e + e \tag{3.42}$$

The number of ionization events per unit volume per unit time is given by Zel'dovich [25] as

$$Z_e^{ion} = n_a n_e \int_v^\infty \sigma_e(v) v f_e(v) dv \tag{3.43}$$

where $f_e(v)dv$ is the Maxwell velocity distribution function which corresponds to the electron temperature, T_e, and $\sigma_e(v)$ is the ionization cross section for electron impact. We define a quantity termed the *ionization rate constant*, α_e, and is given by $\int_v^\infty \sigma_e(v) v f_e(v) dv$. Therefore, Eqn (3.43) is written as

$$Z_e^{ion} = n_a n_e \alpha_e \tag{3.44}$$

Combining this equation with Eqn (3.43), we find that the constant for ionization from the ground level of the atoms is given by

$$\alpha_e = \int_{v_k}^\infty \sigma_e v f_e(v) dv \tag{3.45}$$

$$= \sigma_e \vec{v} \left(\frac{I}{kT_e} + 2 \right) e^{-\frac{I}{kT_e}} \tag{3.46}$$

where $\vec{v}$ is the mean thermal speed of the electrons,

$$\vec{v_e} = \left(\frac{8kT_e}{\pi m_e} \right)^{1/2}$$

and σ_e is the average of the cross section, $\sigma_e(v)$. It has been shown both theoretically and experimentally that the cross section near the threshold depends linearly on the electron energy, ϵ_e, with

$$\sigma_e(v) \approx C(\epsilon_e - I) \tag{3.47}$$

Where C is a constant determined from experiments, and with units of area per unit energy. And, more pointedly, σ_e corresponds precisely to the electron energy given by $\epsilon_e = I + kT$, thereby yielding $\sigma_e = CkT_e$.

Finally, we write the complete ionization rate constant by combining Eqn (3.45) along with the thermal speed and reaction cross section,

$$\alpha_e = CkT_e \left(\frac{8kT_e}{\pi m_e}\right)^{1/2} \left(\frac{I}{kT_e} + 2\right) e^{-\frac{I}{kT_e}} \tag{3.48}$$

3.3.1 Recombination Rate Constant

Before writing down the rate equation for the first ionization process, we need to examine the *recombination rate constant*, which we denote by β_e. From Zel'dovich [25], the rate constants α_e and β_e are related by the principle of detailed balancing,

$$K(T_e) = \frac{\alpha_e}{\beta_e} \tag{3.49}$$

where the equilibrium is determined by the Saha equation, Eqn (2.2), reintroduced here as,

$$K(T_e) = \frac{n_e + n_i}{n_n} = \frac{g_i}{g_n} \frac{2\left(2\pi m_e kT_e\right)^{1.5}}{h^3} e^{-U_i/kT_e} \tag{3.50}$$

where we introduce g, the statistical weight of the electron partition function.

We can now write the recombination rate constant using Eqn (3.49) to combine Eqn (3.48) and Eqn (3.50)

$$\beta_e = \frac{CkT_e \left(\frac{8kT_e}{\pi m_e}\right)^{1/2} \left(\frac{I}{kT_e} + 2\right) e^{-\frac{I}{kT_e}}}{\frac{g_i}{g_n} \frac{2(2\pi m_e kT_e)^{1.5}}{h^3} e^{-U_i/kT_e}} \tag{3.51}$$

The final form of the equation after straightforward algebraic manipulation is

$$\beta_e = \frac{g_n}{g_i} \left(\frac{I}{kT_e} + 2\right) \frac{h_3 \sigma_e}{2\pi^2 m_e^2 kT_e} \tag{3.52}$$

3.3.2 Statistical Weight, g

One obstacle to developing a numerical model of the ionization rate equation is developing a reasonable ratio of electron statistical weights, or g. Fortunately, Gryzwski [9]

provides an analytical solution to the dilemma.

$$g_i(x) = \frac{1}{x}\left(\frac{x-1}{x+1}\right)^{\frac{3}{2}}\left[1+\frac{2}{3}\left(1-\frac{1}{2x}\right)\ln\left(2.7+(x-1)^{\frac{1}{2}}\right)\right] \tag{3.53}$$

where $x = \frac{E_e}{U_i}$, and U_i is the binding energy of the orbital electron, E_e is the kinetic energy of the bombarding electron. The logarithmic terms results from momentum distribution of the of form $f(\nu_e) \rightarrow \frac{1}{\nu^3}$ for $\nu_e \gg v_e$. Figure 3.3.2 compares Eqn (3.53) with and without the natural logarithmic term .

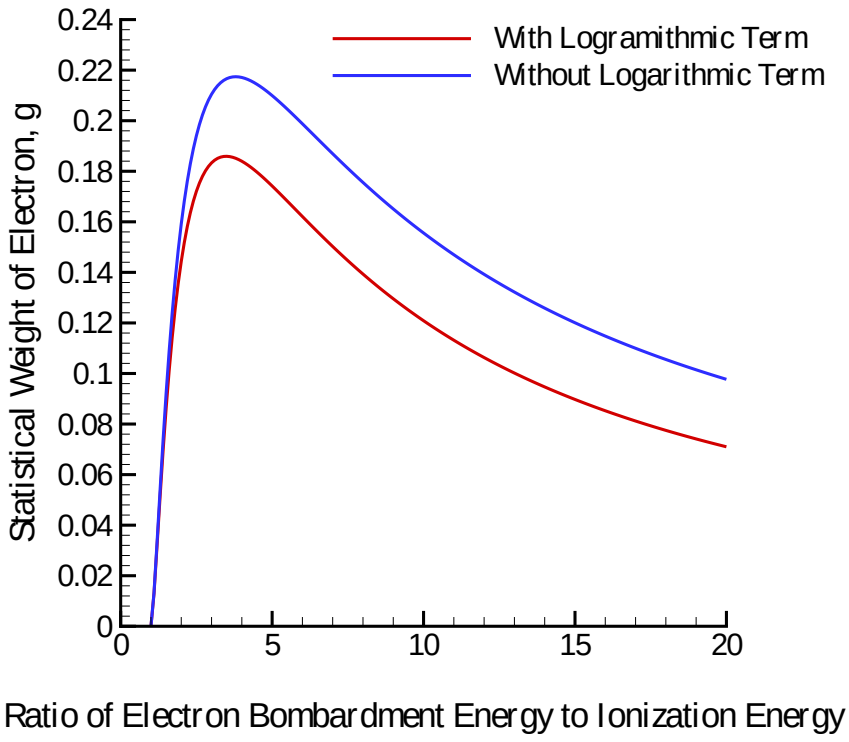


Figure 3.3: Ionization function, g, for electrons as a function the ratio between the kinetic energy of the electron, E_e, to ionization potential, U_i. Both ionization function with and without the natural logarithmic term is plotted. Note that the ratio for A both is maximum when the ratio, x, is approximately 3.

3.4 Transparent Wall Boundary Condition

In order to obtain higher Mach numbers without subjecting the flow to gross discontinuities it has been suggested that any obstructions (e.g. walls) can be gradually made opaque thereby introducing discontinuities that are sufficiently small enough not to create negative (non-realistic) pressures within cells. This technique is analogous to having the walls covered with infinitesimally small suction holes, whereby flow passes through. Initially, the flow would pass through the walls unobstructed, in essence the flow acting as if the walls are non-existent. The holes are gradually closed until the flow "sees" the entirety of the wall.

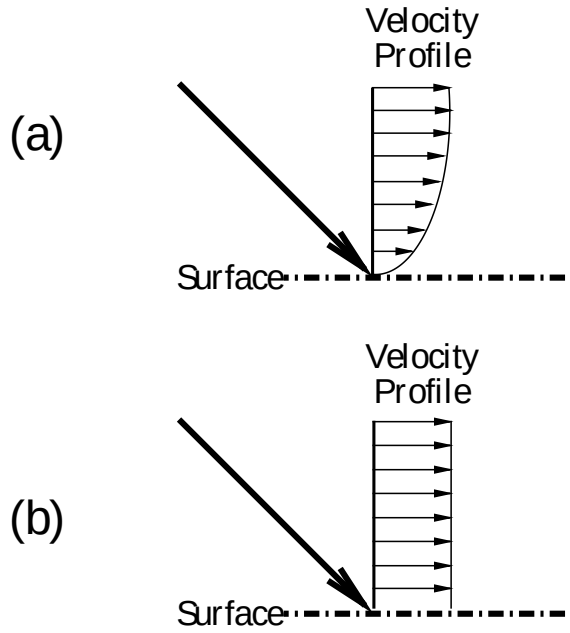


Figure 3.4: Two wall conditions, where *(a)* is no slip wall (viscid) conditions; and, *(b)* is slip wall (inviscid) conditions. For *(a)*, the velocity profile goes to zero at the wall, while for *(b)* v_x is constant along the direction normal to the wall.

In order to accomplish this numerically, we have modified the wall boundary condition so that we linearly increase the magnitude of momentum transferred from the wall cells to the flow cells. Furthermore, we had to differentiate between slip wall and no slip wall conditions. Figure 3.4 shows these conditions. The slip wall condition is used for Eulerian (inviscid) flow, where the velocity component parallel to the wall remains the same, or

$$\frac{\partial \vec{v_x}}{\partial y} = 0$$

Conversely, for viscid flow, the velocity component normal to the wall goes to zero.

Initially, all boundary conditions that are set to the value of "wall" are made transparent to the flow by not reflecting momentum. In order to satisfy the slip wall condition, only the components normal to the cell surface are modified. Figure 3.5 shows the initial, some time after, and final momentum vectors used to linearly make the wall opaque for both slip wall and no slip wall conditions.

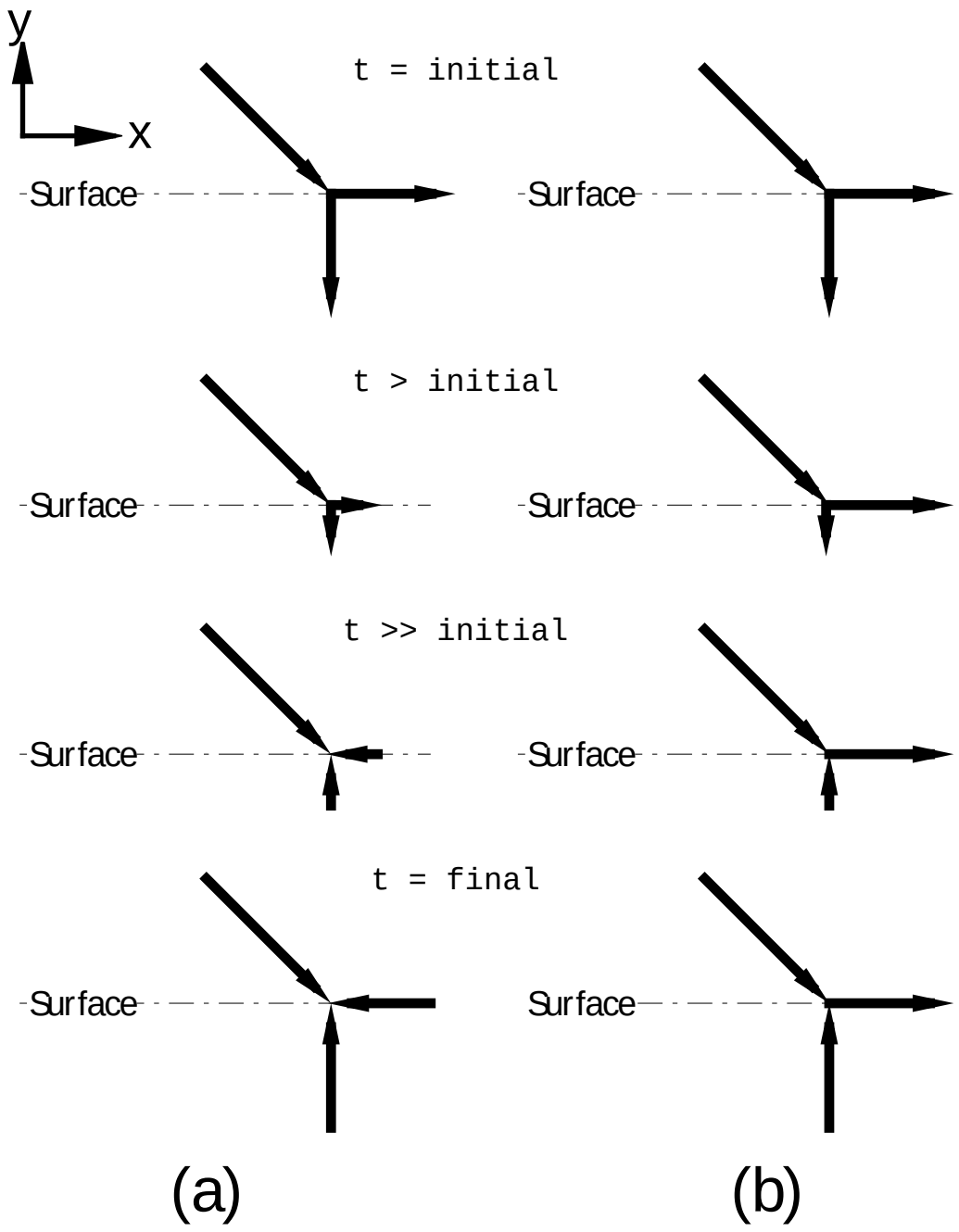


Figure 3.5: Two wall conditions, where *(a)* is no slip wall (viscid) conditions; and, *(b)* is slip wall (inviscid) conditions. Both *(a)* and *(b)* show the time evolution of the momentum vectors for no slip and slip wall conditions.

3.5 Pressure Drag

Drag, excluding body forces and viscosity, is simply the net flow of momentum through the surface of some volume. Recall that the mass flow across the elemental area dS is $\rho\vec{u}\cdot dS$; therefore, the flow of momentum per second through dS is

$$(\rho\vec{u}\cdot dS)\,\vec{u}$$

The net flow of momentum out of the volume is then the summation of the elements (cells), or

$$D = \oint_S (\rho\vec{u}\cdot dS)\,\vec{u} \tag{3.54}$$

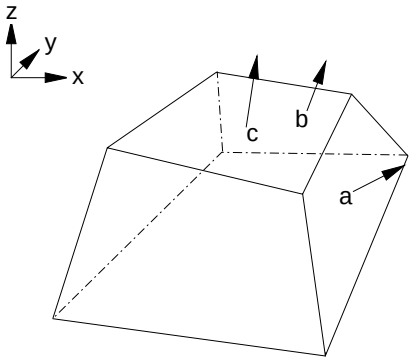


Figure 3.6: A cell in curvilinear space that is not necessarily aligned with the constituent directions x, y, or z. Only three of the six surfaces is used since the adjacent cells simply negate their contributions. Three vectors normal to the surfaces a, b, and c are used to determine how much each cell face contributes to the drag in the above-mentioned constituent directions.

In order to implement this, a few things need to be considered. Figure 3.5 shows an element in curvilinear space. First, for a grid in curvilinear space where the local faces of a cell do not necessarily align with the constituent coordinates x, y, and z, then the summation of each face's contribution to each component of drag in these

said directions must be accounted for. Second, the average of the interior and exterior (ghost) cells is taken to obtain a better approximation of the momentum at the volume surface. Finally, assurances needed to made so that the results are consistent with aerospace notation where area pointing vectors always point outward from the volume, differing with WARP3 notation where the area pointing vectors always point in the positive direction. The final solution looks similar to the following,

$$D_n = \sum_{surfacecells} C\left(\rho \hat{u}_n\right) \times \left(dS_{a,n} + dS_{b,n} + dS_{c,n}\right) \tag{3.55}$$

where C is a correction factor to correlate WARP3 convention to standard convention and is either 1 or -1, $\rho \hat{u}_n$ is the average momentum in the n direction, and $dS_{\alpha,\beta}$ is the surface area of side α in the β direction. Finally, n is one of the three constituent directions x, y, or z.

3.6 Inclusion of Magnetic Dipole in Flow Field

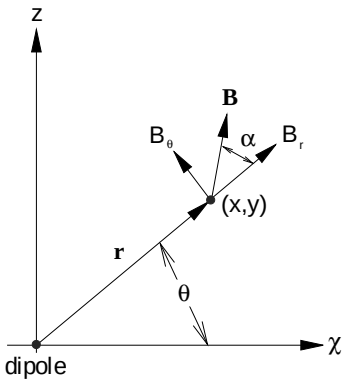


Figure 3.7: Magnetic dipole with magnet field at position (x, y).

WARP3 "wedge" application has been extended to include a magnetic dipole located somewhere within the xy plane; in short, useful for axisymmetric situations. The magnet can be oriented in any direction in this said plane. It is assumed that the magnet

is sufficiently far enough away from the region of interest so that we can represent the dipole as a point source.

Figure 3.6 shows a magnetic dipole with some magnetic field at position (x, y), where the pertinent vectors have been labeled. The magnetic field at points distant from its axis are given by,

$$B_r = \frac{\mu_o \mu \cos(\theta)}{4\pi r^3} \tag{3.56}$$

Also, the magnetic field at points perpendicular are given by,

$$B_\theta = \frac{\mu_o \mu \sin(\theta)}{2\pi r^3} \tag{3.57}$$

where $\mu_o = 4\pi \times 10^{-7}$ is the permeability of space, μ is the magnetic moment[1], and r is the distance from the dipole to the point of interest.

We begin by computing the magnitude of the magnetic field,

$$B = \sqrt{B_r^2 + B_\theta^2}$$

This value is used to convert from cylindrical to cartesian coordinates, namely

$$B_x^{'} = B \times \sin(\theta + \alpha) \tag{3.58}$$

$$B_y^{'} = B \times \cos(\theta + \alpha) \tag{3.59}$$

where α is the angle between B_r and B_θ. Again, the magnetic field strength is calculated,

$$B = \sqrt{B_x^{'2} + B_y^{'2}}$$

Finally, the system is rotated to the orientation as indicated by the input deck.

$$B_x = B \times \sin(\beta + \alpha) \tag{3.60}$$

$$B_y = B \times \cos(\beta + \alpha) \tag{3.61}$$

where α is the angle between $B_x^{'}$ and $B_y^{'}$, and β is the orientation angle of the dipole

[1]For a solenoid, this is simply $I \times N \times A$, where I is current, N is the number of turns, and A is the cross-sectional area

3.7 Algebraic Re-derivation of Fast/Slow Alfvén Speed

In order for WARP3 to properly resolve the fast and slow Alfvén speeds for use in the eigen system, it became necessary to rederive the algebraic equations for these waves in cases where the magnetic field is zero. In particular the motivation for this derivation stems from the fact that since the original derivation included components of the magnetic field in the denominator of terms, resulting in segmentation fault errors. A more robust set of equations are necessary to handle the flow systems investigated by the author and presented later in this thesis. The final set of equations are presented in Eqn (3.62), while the complete derivation is left to Appendix 6.2.

$$\alpha_{f,s}^2 = \pm\frac{1}{2}\left(\frac{1-\frac{B^2}{\rho a^2}}{\sqrt{\left(1+\frac{B^2}{\rho a^2}\right)^2-\frac{4\gamma p B_x^2}{\rho^2 a^2}}}\pm 1\right) \tag{3.62}$$

Chapter 4

BENCHMARKS

This chapter will introduce various benchmark runs used to prove the validity of various subsolvers within WARP3. These solvers include: thermal conductivity; ion fraction; three constituent temperature; pressure drag; and, time evolution wall boundary condition opacity. It cannot be overstressed the importance of these benchmarks and procedures. Too often engineers forget the nature of our discipline; that is, one of approximating real-world phenomenon. In the area of numerics we work in a direction opposite, but nonetheless complimentary to, researchers in ”test-tube” laboratories. Whereas other types of research must strive to eliminate all but the salient physics, computational fluid dynamicists instead seek to develop mathematical models that sufficiently predict only the physics necessary (or so we believe) to understand the system. In other words, whereas others start with everything and work down, we start with nothing and work up.

As such, CFD codes are only as good as the physics they model. These said models must be built upon sound physical principles that invariably come from experiments conducted in real-world environments. The field of numerics will for the very long future remain a subset to other forms of research. This is not to say that this field is secondary to others, but to instead illuminate a very important aspect of this type of research; we work in the role of support for other research by providing very sanitized physical models to further study indepth phenomenon observed back in the real-world.

In summary, the following benchmarks are *sanity checks* of WARP3's robustness. We must first show that a subsolver, when operating independent of the rest of the code, acts in a manner we can analytically predict. Without doing so it is impossible to determine if the results from when all the sub-solvers work together is either gospel or garbage, especially when we consider all the non-linear relationships that

Table 4.1: One-dimensional thermal bar test conditions for examining thermal diffusion solver.

Initial Conditions
$T(x,0) = 1$
$\rho(x,0) = 1$
Boundary Conditions
$T(0,t) = 0$
$\rho(l,t) = 1$
Block Conditions
$Pe = 1000$
$\kappa = 1 \times 10^{-3}$

exist between these solvers.

4.1 Thermal Diffusion

A major component added to WARP3 by the author is the addition of thermal fluxes to the RHS of the MHD equations. In order to validate this solver, we examine a one-dimensional bar. The bar is initially heated to a single temperature throughout for $t \leq t_o^-$. At $t = t_0$, both of the ends of the bar are set to some other temperature. For some time thereafter, the time evolution for $t \geq t_o^+$ is then examined to see how it compares with the analytical solution. Moreover, as $t \to \infty$ the solution reduces to a linear solution. Both these analytical solutions are compared to the thermal diffusion solver.

For the case of one-dimensional thermal diffusion, the neo-classical parabolic equation is

$$\frac{\partial^2 u}{\partial x^2} = a^2 \frac{\partial u}{\partial t} \tag{4.1}$$

Using the conditions outlined in Table 4.1 and the method of separation of variables,

the obtained analytical solution is

$$u(x,t) = \sum_{n=1,3,\ldots}^{\infty} \frac{4}{n\pi} \sin(n\pi x) \exp^{-(n a \pi)^2 t} \tag{4.2}$$

where x and t are the independent variables distance and time.

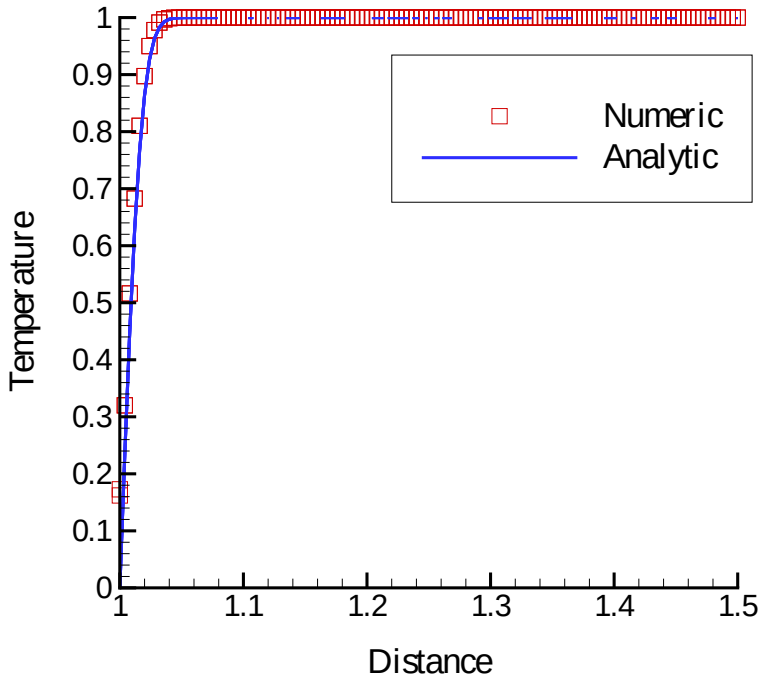


Figure 4.1: One-dimensional bar temperature distribution compared to analytical solution at $t = 100$.

It also important to determine that the solver is not biased; that is, the solver works equally well no matter the orientation of the bar. Therefore, while not shown in this section, WARP3 was tested with grids oriented in all three orthogonal directions (x, y, and z) and also in some arbitrary orientation of all three planes.

4.2 Ionization

In this section we examine the ion fraction solver. We are interested determining if the ion fraction, f_i is indeed bounded between 0 and 1. Furthermore, we are interested to observe how the solver approaches steady state when the initial ion fraction is something other than the steady state value.

Figure 4.2 shows the ion fraction temperature range for Argon (Ar) gas. For very low temperatures, the ion fraction is effectively 0. As the temperature (eV) nears unity, ion fraction rises rapidly; and, the gas is completely ionized near a temperature of 10 eV. Also, Figure 4.2 shows that the ion fraction is indeed bounded by the limits mentioned above. It furthermore indicates that the solver should work reasonably for temperatures from 0 to 100 eV. While not shown, the solver returned identical results when pressure or density were varied while density or pressure were held constant, respectively. This is an obvious result as the ion fraction is a strong function of temperature and number densities, only.

Figure 4.2 shows how the solver handles two initial conditions set higher and lower than the steady state value. We note a discrepancy between the cases, though. When the initial condition is some very small value it takes the solver much longer to approach the steady state value. That is, because the time rate of change of ion fraction is proportional to the product with the ion number density, n_i; if the initial ion number density is 0, then it is impossible for the ion fraction to be anything but 0. Furthermore, when the initial ion fraction is very small, numerical round-off error occurs; the result being that it is difficult for the ion fraction to increase very rapidly. This is ultimately the root of difference between the two cases presented in Figure 4.2.

4.3 Pressure Drag

Due to the nature and direction of the author's work, namely exploring the effects of elevated electron temperatures on bow shock formation and drag reduction for hypervelocity flow stagnating over bluff bodies, a pressure drag solver was written by the author. This section examines the performance of this solver.

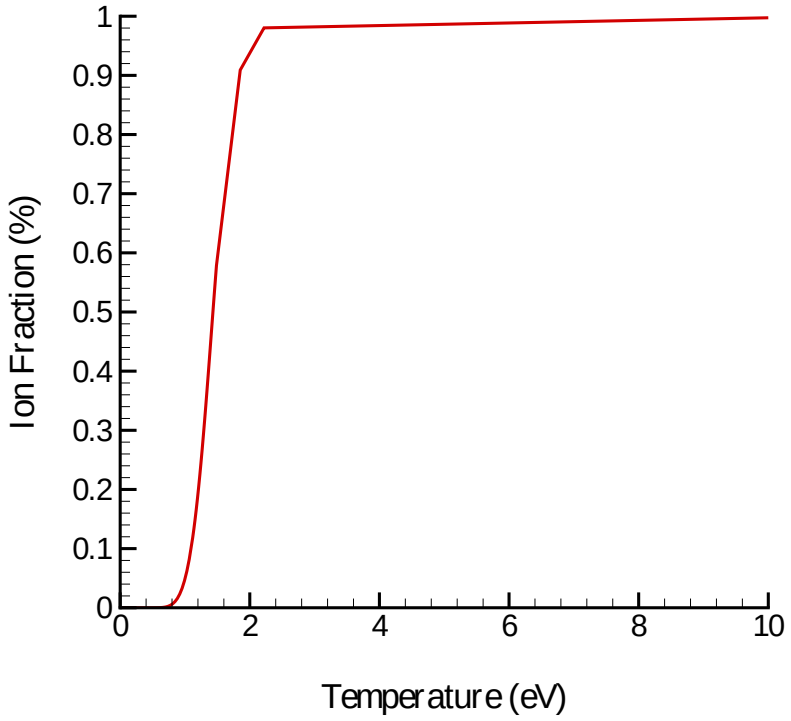


Figure 4.2: Validation of ion fraction solver for WARP3. Ion fraction is solved for density ($\rho \in [1.1 \times 10^{-2}, 1.1 \times 10^{5}]$) where pressure ($p = 3.05 \times 10^{5}$) is held constant. Ion fraction shown as a dependent variable of temperature (eV), where the perfect gas law provides the relationship, $p = R\rho T$, and where is the specific gas constant of argon gas.

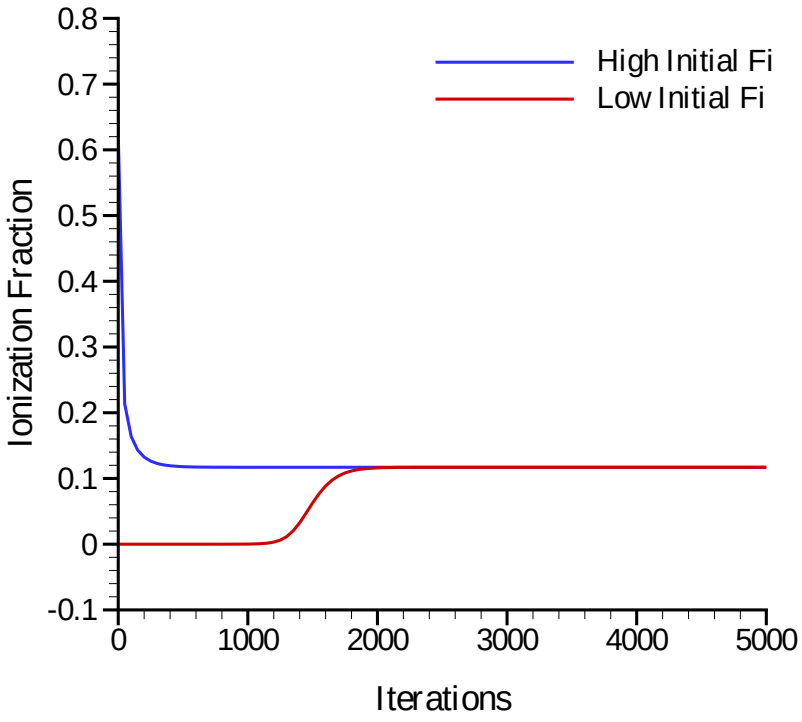


Figure 4.3: Validation of ion fraction solver for WARP3. For two cases the ion fraction is set to some value other than the steady state value and permitted to evolve. In both cases the solver eventually returns the steady state ion fraction value for Argon gas at 0.5 eV and standard atmosphere.

We expect that pressure drag will increase during the subsonic regime. When the flow reaches sonic speed, a detached bow shock is generated starting at the stagnation point and extending out and backward from this point. The creation of the bow shock indicates that the flow is no longer able to compensate for the increased kinetic energy of the flow solely through simple compression. Because the bow shock, as with shocks in general, is a discontinuity generated to alleviate problems associated with compression we might expect the pressure drag to also jump at sonic velocity. As the flow speed passes into the supersonic regime, the pressure drag should once again increase. Table 4.3 and Figure 4.3 show this trend.

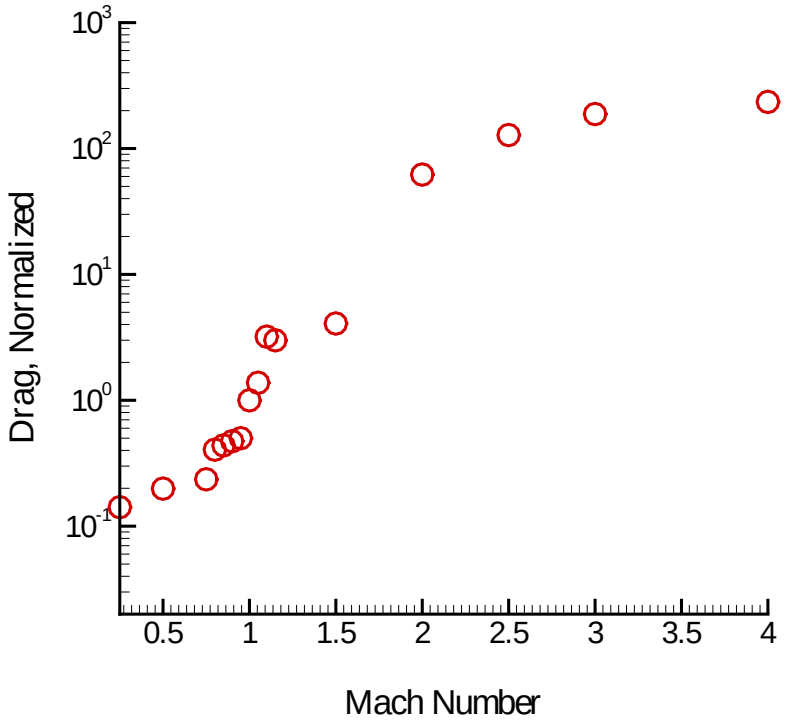


Figure 4.4: Validation of pressure drag solver. Values for drag normalized to sonic velocity (Mach = 1) drag, and shown on semi-log chart. Note the extreme jump around Mach 1; where the bow shock is initiated. The lack of a definitive jump is due to oscillations in the solution as evidenced by the oscillation of the residual. Furthermore, this oscillation of the solution is a real-world phenomenon, and not numeric artifact.

Table 4.2: Pressure drag for hemispherical cylinder for subsonic, sonic, and supersonic flow velocity. Pressure, density calculated for 50,000 feet standard atmosphere. Diatomic nitrogen is used as an approximate to air (approximately $80\%N_2, 20\%O_2$.)

Mach Number	**Drag**	**Density**	**Pressure**	**Temperature**
		$[kg \times m^{-3}]$	$[Pa]$	$[K]$
0.25	20.50	0.1948	12,108	217
0.50	28.78	0.1948	12,108	217
0.75	34.13	0.1948	12,108	217
0.80	58.58	0.1948	12,108	217
0.85	63.44	0.1948	12,108	217
0.90	68.49	0.1948	12,108	217
0.95	72.39	0.1948	12,108	217
1.00	144.8	0.1948	12,108	217
1.05	199.7	0.1948	12,108	217
1.10	463.8	0.1948	12,108	217
1.15	434.1	0.1948	12,108	217
1.50	590.0	0.1948	12,108	217
2.00	8,996	0.1948	12,108	217
2.50	18,530	0.1948	12,108	217
3.00	27,200	0.1948	12,108	217
4.00	33,990	0.1948	12,108	217

Table 4.3: Analytic and numeric (WARP3) results for the gas dynamics supersonic flow problem with $M_1 = 3$ and $\theta = 25^o$ shown in Figure 4.4.

	Analytic	**Numeric**
M_2	1.7	1.704
β	44	43.97

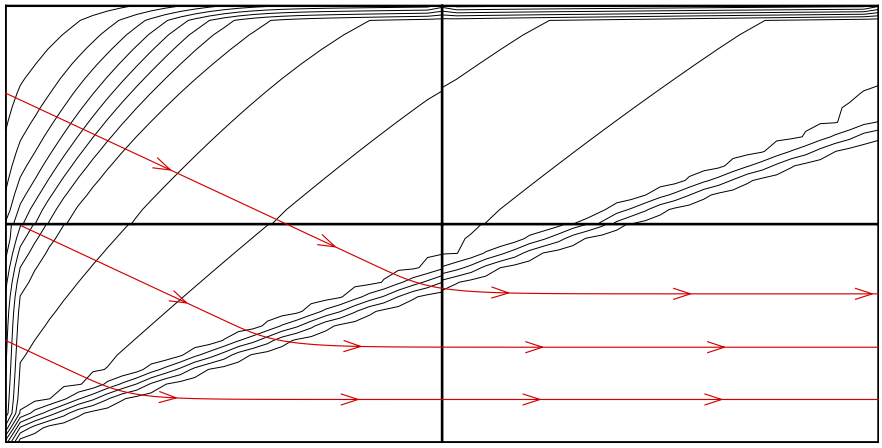
4.4 Supersonic Wedge

This section is both a qualitative and quantitative study of the interaction with ionizing flow impinging upon a rectangular wedge in a hypervelocity flow. For those familiar with Udrea's thesis [23] will notice that the author has duplicated Udrea's own benchmark tests. For this reason, readers are encouraged to reference this source for further details.

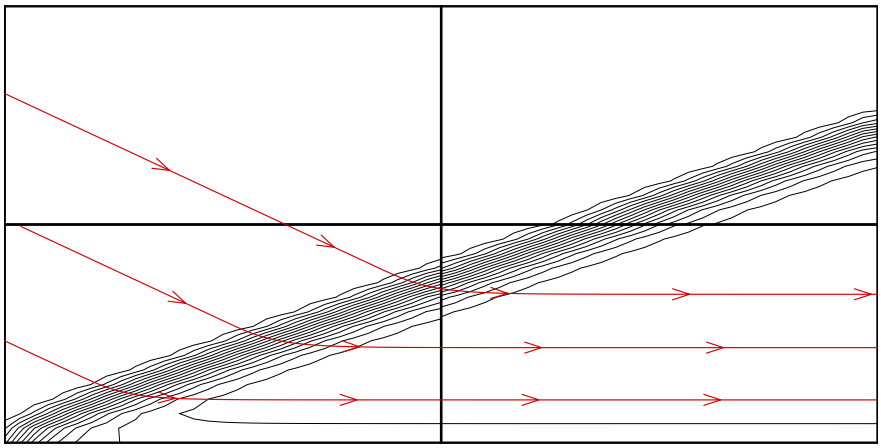
We are interested in seeing how the ion fraction solver handles this type of flow. As the flow crosses through the shock, translation (kinetic) energy is converted into random-motion (thermal) energy; thereby we expect an elevated ion fraction level "behind" the shock. To ensure we see this trend, the freestream temperature is set high enough that numeric noise (round-off error) does not impede the development of the expected trend. (See Section 4.1 and Figure 4.2 for further details on numeric round-off.) Figure 4.4 shows the results.

4.5 Transparent Wall Boundary Condition

This section briefly outlines results obtained from modifications made to the wall boundary conditions as outlined in Section 3.3.2. In particular, we are interested in determining if ramping up the opacity of wall boundary conditions permits higher Mach number flow rather than similarly ramping up the inflow boundary condition Mach number. In the following case both the ramping up opacity and inflow Mach number have been conducted exclusive of each other, even though it is possible to utilize both



(a)



(b)

Figure 4.5: Hypersonic flow over a rectangular wedge at 25^{o} angle of attack to the freestream flow. The flow's temperature is elevated. Ion fraction contours **(a)** and density contours **(b)** along with velocity streamlines are shown. Solved using 4 blocks of 15×30 cells each.

Table 4.4: Comparison of two methods for obtaining high Mach number flow. This table shows the maximum freestream Mach number obtained for Euler flow. A rectangular wedge at 25^{o} is employed.

Method	**Mach Number**	**Ramp Up Time**
Wall	14.625	10.0
Inflow	25.0[1]	10.0

simultaneously with WARP3.

In this study, we examine the highest possible Mach number possible for each method given up a ramp-up time of 10 characteristic seconds, and shown in Table 4.5.

At present it appears that ramping the inflow mach number provides better stability versus ramping up the wall opacity. The author suspects that the solver might be applying first-order and second-order solutions to adjacent cells that might lead to the segmentation fault that the author encountered with wall (opacity) ramping. In conclusion, the author believes the evidence is presently inconclusive as to whether one method is superior to another. So much so, that each method have preferred applications, and even hybridization may be in order, *i.e.* employing first mach ramping and then wall ramping to obtain the best results.

4.6 Blast Theory

This section provides both a verification that the fluid conservation equations are properly defined, but also that they can handle high kinetic energy flows such as the one's examined by the author. Furthermore, it is shown that the generation of the bow shock in the front of the hemispherical cylinder is not an artifact of the grid. In other words, regardless of the geometry of the grid the results are the same.

Before we introduce the results, let us first examine blast-wave theory. We begin by assuming that time scale for release of energy into the flow is much, much smaller

than the time scale for shock formation. In short, the flow instantaneously realizes the presence of energy. Coupled with this we realize that we can equate the instantaneous point blast to a bluff body in a hypervelocity flow. When a point blast is stationary then in two-dimensions it results in an ever-expanding circle of discontinuity. However, when the same point blast is moving the resultant shape of the blast discontinuity is a parabola. If we accept the validity of the assumption that energy is instantaneously realized by the flow, then the behavior of a point blast is exactly the same for a bluff body in ideal conditions (*ergo* Euler.)

According to Anderson and others, blast wave theory [2, 16, 17] provides a very reasonable approximation of the bow shock for bluff bodies. However, the accuracy of the model degrades when viscous (low Reynolds) effects come to dominate the interaction between flow and body. Moreover, discrepancies are to be expected given the nature of our simplifications. Neither is the bluff body a point source, nor does the flow instantaneously "absorb" energy; nevertheless the results provide a good metric of the code.

Anderson [2] gives the cylindrical blast wave as,

$$p = k\rho_{infty}\left(\frac{E}{\rho_{infty}}\right)^{1/2} t^{-1} \tag{4.3}$$

where p is pressure, rhp_{infty} is freestream density, E is energy impinged upon the bluff body, and t is time. Also, k is given by

$$k = \frac{\gamma^{2(\gamma-1)/(2-\gamma)}}{2^{(4-\gamma)/(2-\gamma)}} \tag{4.4}$$

Eqn (4.3) gives the pressure near the stagnation point as a function of time. However, the vertical component can obtained by the noting that,

$$r = \left(\frac{E}{\rho_{infty}}\right)^{1/4} t^{-1} \tag{4.5}$$

We further advance the case by momentarily returning to our original assumption, mainly that the flow immediately realizes energy. Therefore, the spatial change of energy of flow must equal to the opposing force (drag) generated by the bluff body, or

$$\frac{dE}{dx} = D \tag{4.6}$$

In non-dimensional terms then for some unit length the energy is equal to the drag, or $E = D$. We define the drag in relation to the freestream kinetic energy, $1/2\rho_\infty V_\infty^2$, and the base area of the cylinder, $\pi d^2/4$, along with some coefficient of drag, C_d. Therefore, we have

$$E = D = C_d \frac{1}{2}\rho_\infty V_\infty^2 \frac{\pi d^2}{4} \tag{4.7}$$

If we also note from equivalence principle[2] [2] that $t = x/V_\infty$ and use perfect gas law to relate density to pressure then Eqn (4.3) becomes

$$p = k\frac{\gamma p_\infty}{\gamma R T_\infty}\sqrt{\frac{\pi}{8}}V_\infty^2\sqrt{C_d}\left(\frac{x}{d}\right)^{-1} \tag{4.8}$$

For the ratio of specific heats, $\gamma = 1.4$, the equation becomes

$$\frac{p}{p_\infty} = 0.0681 M_\infty^2\sqrt{C_d}\left(\frac{x}{d}\right)^{-1} \tag{4.9}$$

More directly, the shape of the shock can be obtained by instead writing

$$\frac{r}{d} = 0.792 C_d^{1/4}\sqrt{\frac{x}{d}} \tag{4.10}$$

where x is the distance measured from the nose in the direction of the flow, r is the distance in the direction perpendicular to x, C_d is the coefficient of drag, and d is the cylinder diameter.

However, Eqn (4.10) does not take into account the freestream pressure in front of the shock. Consequently, another equation derived from work done by Sakurai [16, 17] takes this factor into account, and is identified by the inclusion of the freestream Mach number in the following equation.

$$\frac{r}{d} = M_\infty C_d^{1/2} 0.795\sqrt{\frac{x/d}{M_\infty C_d^{1/2}}\left(1 + 3.15\left(\frac{x/d}{M_\infty C_d^{1/2}}\right)\right)} \tag{4.11}$$

where M_∞ is the freestream Mach number, C_d the coefficient of drag, and x/d is the ratio of distance from cylinder to cylinder diameter.

[2]Simply stated, the equivalence principle states the non-steady state is identical to the steady state less one spatial dimension. In other words, a steady state system in three dimensions is equivalent to its related non-steady state in two dimensions.

Table 4.5: Test case conditions for Figure 4.6. Pressure and density are for at 15,000 m using data from the U.S. Standard Atmosphere, 1976.

Mach Number	**Cylinder Diameter**	**Drag Coefficient**
M_∞	d	C_d
5	2	2

For the purposes of this thesis, and in keeping with Anderson's notation, Eqn (4.10) is referred to as the *first approximation*, while Eqn (4.11) is referred to as the *second approximation*. It is important to note that these two approximations on based on real-world experiments, where the coefficients have been adjusted to better match experimental data.

Figure 4.6 presents a comparison of Eqn (4.10,4.11) with numerical results obtained with WARP3. Table 4.6 provides the test case conditions.

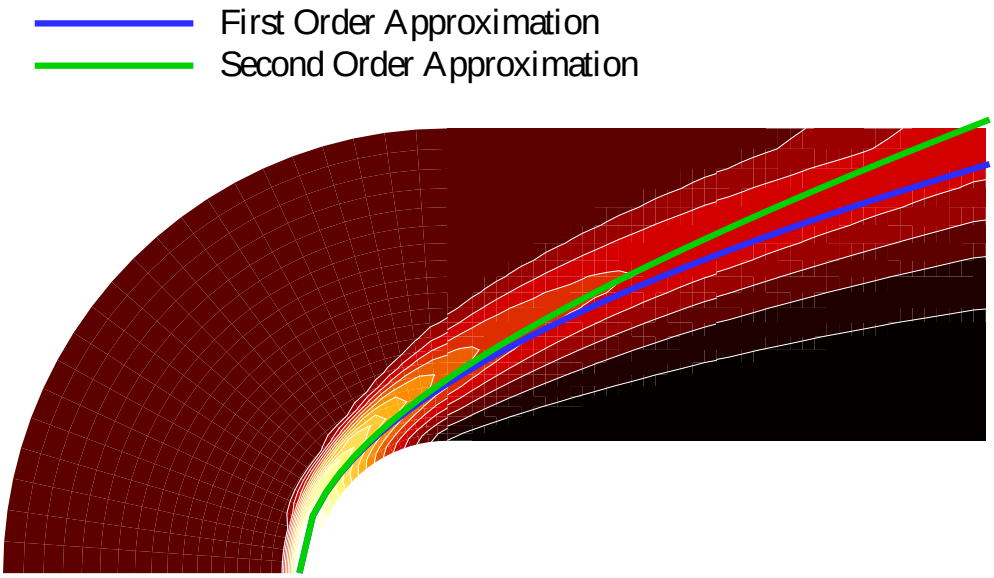


Figure 4.6: Examination of WARP3 robustness to handle severe bow shock formation as a hypervelocity flow stagnates over a hemispherical cylinder. Blast wave theory and numerical results provide validation of the code, where both first and second approximations are presented as solid blue and green lines, respectively. There is an excellent correlation between the blast wave theory approximations and the contours indicating that WARP3 is congruent with experimental data. Contours represent local density; flow is Euler (inviscid), with slip wall conditions at the walls.

Chapter 5

HEMISPHERICAL CYLINDER, SUGGESTIONS FOR FUTURE RESEARCH ON

In this chapter we introduce the author's suggestions for future application of WARP3 to the study of ionized flow over bluff bodies in hypervelocity flows. The author had originally intended to complete the proceeding test matrix; however, due to unforeseen problems coupled with other obligations the research had to be unfortunately abandoned. Nevertheless, the present status of this uncompleted research is presented as is in the hopes that in the future another researcher will complete the following line of inquiry.

The extent of this research is to determine what if any effects change and/or occur when the thermal conductivity and ion fraction solvers are included in solving for hypervelocity flow over a bluff body. In particular, it is anticipated that this set of physics to the magnetohydrodynamic equations will sufficiently explain previously observed alteration of bow shock shape, strength, and stand-off distance.

5.1 Confidence in Code, Grid

In order to establish the relevancy of certain physics for the above described system, a variety of tests including code benchmarks (see previous chapter) must be conducted to provide confidence in our results.

For the most part, code benchmarks look to isolate a single solver in order to show the validity of that one said solver irrespective of the remaining solvers. However, in order for us to move forward with this research, WARP3 as a whole needed to be thoroughly tested; especially considering the scope of this research is a significant departure from previous work done by U. Shumlak *et al.*

We first constructed a sparse grid (20×60) that was not optimized for the physics. To point, the grid did not *a priori* assume the shape of the bow shock. Instead, the shape of the interested bluff body (hemispherical cylinder) was used as a starting point for the entire system. In so doing, coupled with blast wave theory (see Section 4.5), we could definitively determine if the bow shock was a result of the physics of WARP3 solvers, or an artifact of the grid. In other words, Udrea's [23] inclusion of generalized grid with ability for arbitrary local orientation of cells is further verified by the lack of bias to fluxes normal to the cell faces.

Next, we need to determine an optimal size for the grid in order to ensure the relevant physics are captured. In this initial study we are concerned with the gross changes to the flow structure. While the boundary layer growth is an issue with high-temperature flows at hypervelocity, the scope of this research excludes these physics. A factor for this is due to the computational cost imposed by generating enough cells (10 to 15 cells) within the boundary layer region. Nevertheless, this study determined that for a 3 block system, each block composed of 20×20 cells (see Figure 5.1), is optimal. We reached this conclusion after running WARP3 with a 3 block system with twice this resolution, or 40×120 (see Figure 5.1). Comparison of the two results for identical conditions showed that there was not sufficient differences between the two resolutions to warrant using the higher resolution system. It should be mentioned that the author also generated a six-block grid; however, there is no savings in computational cost by increasing the number of blocks at this relatively low resolution. Further reduction in the number of cells per block would result in an increase in network traffic, saturating the computational time with ghost-cell swapping over actual computation.

In addition to using the three block grid shown in Figure 5.1, it was later realized that a second grid was necessary. Magnetic field boundary conditions, namely no monopoles, made it necessary to reconfigure the leading grid edge (left-hand side in previous figures). It should be obvious in Figure5.1 that the leading edge lays over the hemispherical cylinder. If a magnetic field is included to the system, a monopole is created at the hemispherical cylinder's perfectly conducting surface. See Section 5.3 for further discussion on the issue of magnetic boundary conditions.

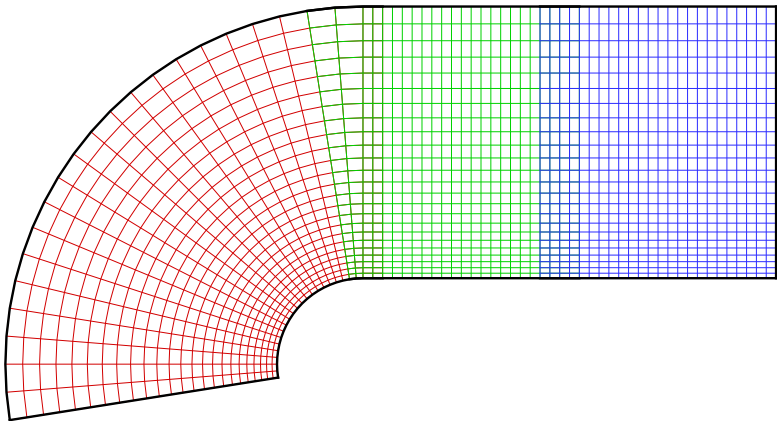


Figure 5.1: Three (3) block system, each block is composed of 20×20 cells, or a total of 1,200 cells.

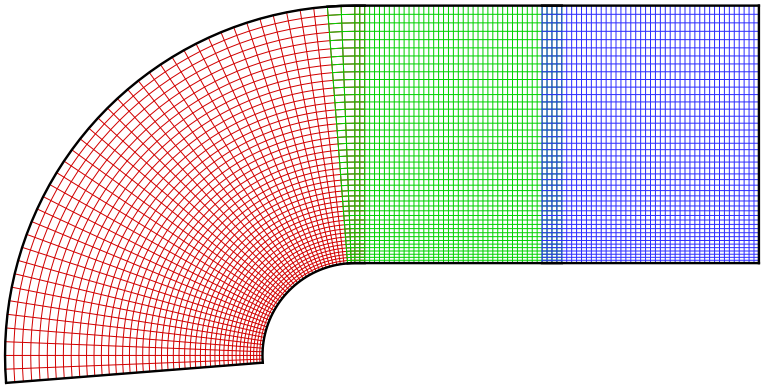


Figure 5.2: Three (3) block system, each block is composed of 40×40 cells, or a total of 4,800 cells.

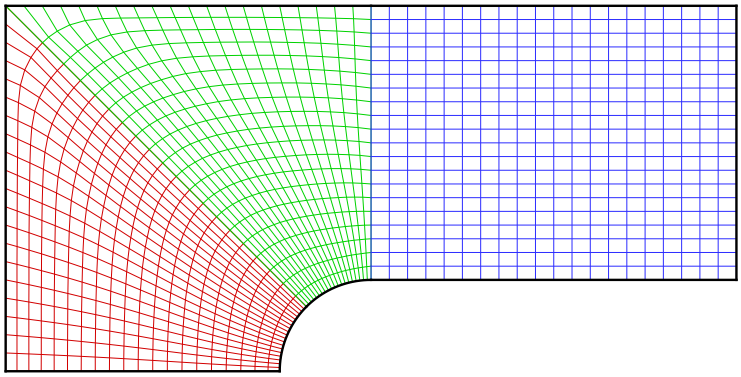


Figure 5.3: Three (3) block system, each block is composed of 20×20 cells, with a revised leading edge in comparison to the grid system shown in Figure 5.1.

The following grid shown in Figure 5.1 permits a magnetic field perpendicular to the flow to be introduced without causing errors.

5.2 Test Matrix

Table 5.2 outlines the number of runs conducted with WARP3 including flow conditions, magnetic field strength and orientation, and grid. This test matrix guided our research approach, and provides a summary of what physics we believe are pertinent to current and future research in this field. We are extremely confident of the code's robustness at low Mach numbers, therefore runs 1–3 provide a reliable metric to use against runs 4–17.

In this research we vary three parameters; namely: electron temperature; magnetic strength (relative to stagnation pressure); and, magnet orientation. In the first variable, we are interested to see if elevated electron temperature[1] affects the bow shock

[1]We are not interested in the exact mechanism to generate elevated electron temperatures, though

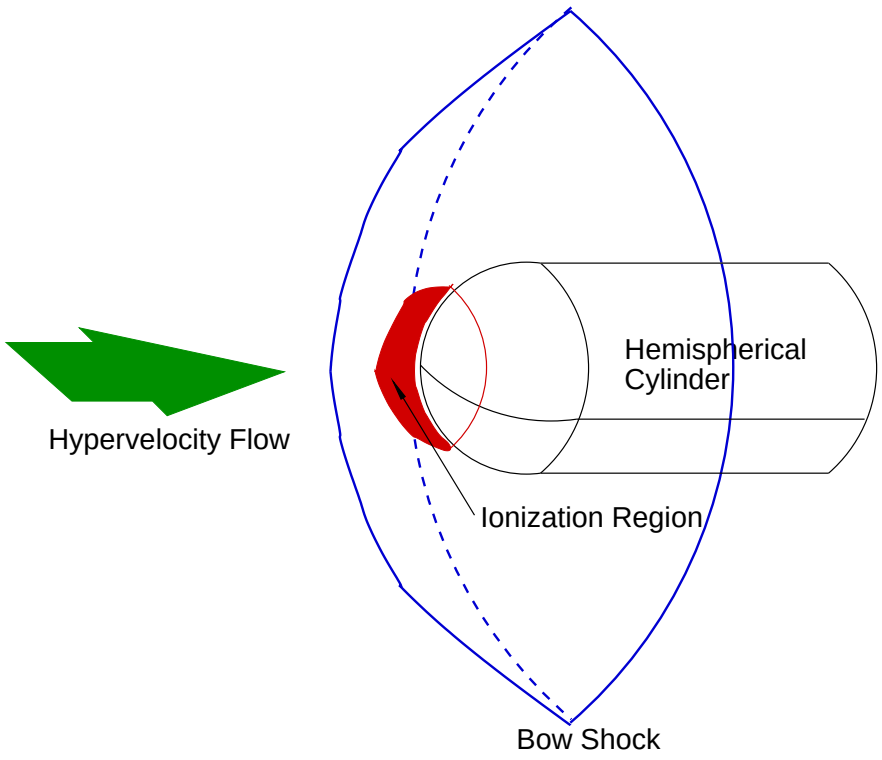


Figure 5.4: Schematic of hemispherical cylinder showing location of bow shock, ionization region, and stagnation point.

characteristics through the resultant increase in ion fraction density in and around the stagnation point. In both the second and third variable, we are interested determining what type of affect a magnetic field again has on the bow shock characteristics; both with and without elevated electron temperatures. In the case of the magnetic field oriented parallel to the freestream direction, we will refute or dispute whether the magnetic field lines bend ”inward” as they pass through the shock front.

---

something similar to a tailored-frequency microwave waveguide as demonstrated by K. Chadwick *et al.* is applicable.

Table 5.1: Test Matrix. The magnetic field, if any, is oriented in one of three ways; parallel to the flow direction; perpendicular to the flow direction; and, dipole situated within the hemispherical cylinder. For further information see Table 5.2 for flow conditions and Figure 5.2. Note: **Thm** means thermal, and **Vsc** means viscous.

Run No.	**Mach No.**	**Stagnation Pressure**	**Magnetic Pressure**	**Magnet Type**	**Non-Ideal Fluxes**	**Electron Temp.**
1	3	1.31×10^5	N/A	None	None	0.0187
2	3	1.31×10^5	N/A	None	Thm	0.0187
3	3	1.31×10^5	N/A	None	Thm, Vsc	0.0187
4	8	8.72×10^5	N/A	None	None	0.0187
5	8	2.56×10^6	N/A	None	None	1.0
6	8	8.72×10^5	N/A	None	Thm, Vsc	0.0187
7	8	2.56×10^6	N/A	None	Thm, Vsc	1.0
8	8	8.72×10^5	0.523	Dipole	Thm, Vsc	0.0187
9	8	2.56×10^6	0.897	Dipole	Thm, Vsc	1.0
10	8	8.72×10^5	2.09	Dipole	Thm, Vsc	0.0187
11	8	2.56×10^6	3.59	Dipole	Thm, Vsc	1.0
12	8	8.72×10^5	0.523	$B_{\|\|}$	Thm, Vsc	0.0187
13	8	2.56×10^6	0.897	$B_{\|\|}$	Thm, Vsc	1.0
14	8	8.72×10^5	2.09	$B_{\|\|}$	Thm, Vsc	0.0187
15	8	2.56×10^6	3.59	$B_{\|\|}$	Thm, Vsc	1.0
16	8	8.72×10^5	0.523	$B_{\perp}$	Thm, Vsc	0.0187
17	8	2.56×10^6	0.897	$B_{\perp}$	Thm, Vsc	1.0
18	8	8.72×10^5	2.09	$B_{\perp}$	Thm, Vsc	0.0187
19	8	2.56×10^6	3.59	$B_{\perp}$	Thm, Vsc	1.0

Table 5.2: Reference conditions for runs shown in Table 5.2. These value enable calculation of real-world values from non-dimensional WARP3 output.dat

Magnetic Field, B_{ref} [T]	1.121×10^{-3}
Density, ρ_{ref} [$\frac{kg}{m^3}$]	1.0
Length, a [m]	1.121×10^{-3}
Ratio of Specific Heats, γ	1.4
Kinematic Viscosity, η_{ref} [$\frac{m^2}{K}$]	7.34×10^{-5}
Thermal Conductivity, κ_{ref} [$\frac{W}{mK}$]	0.05
Resistivity, ν_{ref}	1.0

Table 5.3: Flow conditions for runs shown in Table 5.2 utilizing reference values provided in Table 5.2

Mach Number	**3**	**8**	**8**
Electron Temperature, [eV]	0.019	0.019	1.0
Pressure, [Pa]	12,050	12,050	35,385.6
Density, [$\frac{kg}{m^3}$]	0.1948	0.1948	0.1948
Sonic Speed, [$\frac{m}{s}$]	295	295	506
Ion Fraction	7.88×10^{-31}	7.88×10^{-31}	0.036
Peclet Number	5.22×10^{6}	1.39×10^{7}	4.09×10^{8}
Reynolds Number	3.56×10^{9}	9.49×10^{9}	2.79×10^{10}
Lundquist Number	0.328	0.875	2.57

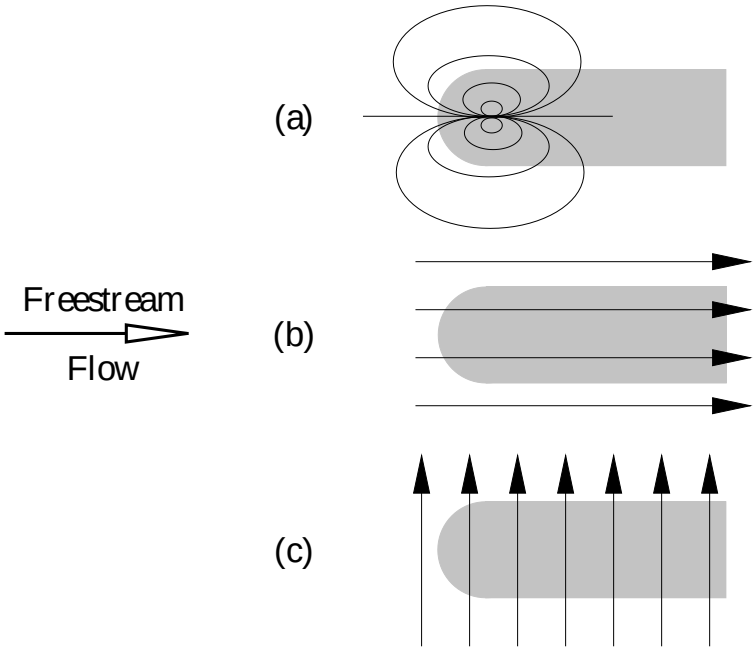


Figure 5.5: Three (3) magnetic orientations used for Mach 8 flows where: **(a)** is a dipole located within the hemispherical cylinder; **(b)** is for a magnetic field parallel with the freestream flow direction; and, **(c)** is for a magnetic field perpendicular with the freestream flow direction. For further information see Table 5.2.

Table 5.4: Comparison of drag in direction along freestream flow to electron temperature. (See Table 5.2 for comparison with magnetic field strength and orientation. Values are non-dimensional. Note that the drag in the other two directions, namely transverse to the flow, are irrelevant for a two-dimensional, axisymmetric flow where there are expected to be zero.

Run No.	**Mach No.**	**Stagnation Pressure**	**Electron Temp.**	**Drag**
1	3	1.31×10^5	0.0187	??
2	3	1.31×10^5	0.0187	??
3	3	1.31×10^5	0.0187	??
4	8	8.72×10^5	0.0187	??
5	8	2.56×10^6	1.0	??
6	8	8.72×10^5	0.0187	??
7	8	2.56×10^6	1.0	??
8	8	8.72×10^5	0.0187	??
9	8	2.56×10^6	1.0	??
10	8	8.72×10^5	0.0187	??
11	8	2.56×10^6	1.0	??
12	8	8.72×10^5	0.0187	??
13	8	2.56×10^6	1.0	??
14	8	8.72×10^5	0.0187	??
15	8	2.56×10^6	1.0	??
16	8	8.72×10^5	0.0187	??
17	8	2.56×10^6	1.0	??
18	8	8.72×10^5	0.0187	??
19	8	2.56×10^6	1.0	??

Table 5.5: Comparison of drag in direction along freestream flow to magnetic field strength and orientation. (See Table 5.2 for comparison with electron temperature. Values are non-dimensional. Note that the drag in the other two directions, namely transverse to the flow, are irrelevant for a two-dimensional, axisymmetric flow where there are zero.

Run No.	**Mach No.**	**Stagnation Pressure**	**Magnetic Pressure**	**Magnet Type**	**Drag**
1	3	1.31×10^5	N/A	None	??
2	3	1.31×10^5	N/A	None	??
3	3	1.31×10^5	N/A	None	??
4	8	8.72×10^5	N/A	None	??
5	8	2.56×10^6	N/A	None	??
6	8	8.72×10^5	N/A	None	??
7	8	2.56×10^6	N/A	None	??
8	8	8.72×10^5	0.523	Dipole	??
9	8	2.56×10^6	0.897	Dipole	??
10	8	8.72×10^5	2.09	Dipole	??
11	8	2.56×10^6	3.59	Dipole	??
12	8	8.72×10^5	0.523	$B_{\|\|}$	??
13	8	2.56×10^6	0.897	$B_{\|\|}$	??
14	8	8.72×10^5	2.09	$B_{\|\|}$	??
15	8	2.56×10^6	3.59	$B_{\|\|}$	??
16	8	8.72×10^5	0.523	$B_{\perp}$	??
17	8	2.56×10^6	0.897	$B_{\perp}$	??
18	8	8.72×10^5	2.09	$B_{\perp}$	??
19	8	2.56×10^6	3.59	$B_{\perp}$	??

5.3 Magnetic Boundary Conditions

As mentioned in Section 5.1, it was realized after initiating the test matrix that the original grid system was not sufficient for inclusion of magnetic fields. Namely, there were two overriding issues that needed to be resolved. Firstly, the hemispherical cylinder is perfectly conducting. In other words, all magnetic field lines will never touch the cylinder, but instead "lay down" over it. Secondly, monopoles do not exist.

Regarding the first issue, it should be obvious that the grid system shown in Figure 5.1 makes it impossible to initialize a perpendicular or parallel magnetic field without creating a discontinuity at the cylinder surface. Furthermore, per the second issue, we can no longer make the simplification of axi-symmetry.[2] Note that if we simplify the flow to axi-symmetric then monopoles are implicit at the center line when perpendicular magnetic field is included.

For perpendicular magnetic field, the easiest solution is to modify the leading edge shown in Figure 5.1 so that it, too, is perpendicular with the flow. The magnetic boundary condition is switched on at this edge, and the field is allowed to naturally "march" into the system. Up to this point, the author has had problems getting WARP3 to converge, however initial results have shown good, qualitative agreement with theory. Namely, we have seen the magnetic field lines become "bent" as they pass through the bow shock as is expected.

[2]This does not prove a major change in our system, though. Namely, we move from studying a hemispherical cylinder to studying a semi-hemispherical block.

Chapter 6

CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK

In chapter we present a summary of our research and project efforts, along with suggestions for future research using WARP3. It is hoped that by noting certain weakness of the code and limitations imposed by the present set of assumptions that we will help facilitate the nature and attitude that has gone into developed WARP3. That is to say, constantly pushing the envelope in three primary directions: finite volume and numerical techniques; understanding of plasma physics and other related fields; and, distributed, parallel computational techniques.

6.1 Thermal Diffusion, Ionization and Drag

It has been shown in the preceding sections have introduced both the analytical and numerical mathematics required to successfully add thermal diffusion (thermal parabolic flux), time rate of change of ionization, and pressure (momentum) drag. Numerical results obtained using WARP3 that substantiate these said derivations. The author does not anticipate further modification to these sub-solvers for the foreseeable future.

6.2 Suggestions for Future Work

As with any research, there are as many roads traversed as there are new roads discovered to go down and explore. This research is no different. While not as many inroads have been made toward answering our initial set of questions and postulates as had been hoped, there are still even more questions that need answering.

In particular, a present limitation of WARP3 is the assumption of a global ratio of specific heats, or γ. A much better set of predictions is possible by creating a local value for this parameter that could be a function of bulk temperature. It is even possible to

couple this ratio with the species temperature if other research shows this to be true. An excellent example of the limitations of a global ratio of specific heats occurs when calculating the re-entry temperatures of a vehicle. Per Figure 1.18 of Anderson [2], the temperature for a calorically perfect gas is on the order of $14,000K$ at a velocity of $6km/s$. However, taking into effect the changes to the ratio of specific heats as a function of temperature, a much better value is $6000K$ for the same speed.

Another area for future consideration is refinement of the grid used in this study. There are many weak points to the grid; including computational waste on the windward side where much of the grid is outside of the bow shock. Furthermore, it has been suggested that the grid needs to be elongated to obtain better results with real-world experiments. Also, it would be interesting to see what types of boundary layer growth occurs; however this would be computationally expensive as the small grid size near the surface would drive the time step size down.

Finally, the author has prepared a small explanation of the relevant physics regarding phase-change (ablation), found in Appendix F. In addition, radiative transfer along with catalytic boundary conditions are also considered. It is hoped that WARP3 may one day include solvers for these physics, thereby providing further tools for the research of hypersonics.

BIBLIOGRAPHY

[1] John D. Anderson. *Modern Compressible Flow with Historical Perspective*. McGraw-Hill Book Company, 1982.

[2] John D. Anderson. *Hypersonic and High Temperature Gas Dynamics*. McGraw-Hill Book Company, 1989.

[3] John J. Bertin. *Hypersonic Aerothermodynamics*. American Institute of Aeronautics and Astronautics, 1994.

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

[5] James Thomas Dolan. *Fusion research : principles, experiments and technology*. Pergamon Press, 1982.

[6] J.P. Galambos. *Measurement of the Internal Torodial Magnetic Field on the Helicity Injected Tokamak using the Transient Internal Probe*. PhD thesis, University of Washington, 1996.

[7] Galambos, J.P., Bohnet, M.A., Jarboe, T.R., Mattick, A.T. Magnetic field measurements using the transient internal probe (tip). *RSI*, pages 469–473, 1996.

[8] Goldston, Robert J., Rutherford, Paul H. *Introduction to Plasma Physics*. Institute of Physics Publishing, 1995.

[9] Michael Gryzwski. Classical theory of atomic collisions, i. theory of inelastic collisions. *Physics Review A*, 138:336, 1988.

[10] Hunter, L.W., Kuttler, J.R. Enthalpy method for ablation-type moving boundary problems. *Journal of Thermophysics and Heat Transfer*, 5, number 2:240 – 241, 1991.

[11] W.H.T. Loh. *Dynamics and Thermodynamics of Planetary Entry*. Prentice-Hall International Press, 1963.

[12] Mikhailov, G.K., Parton, V.Z. *Super- And Hypersonic Aerodynamics and Heat Transfer*. CRC Press, 1993.

[13] Dwight R. Nicholson. *Introduction to Plasma Theory*. John Wiley and Sons, 1983.

[14] M.N. Ozisik. *Radiation Transfer and Interactions with Conduction and Convection*. Wiley-Interscience, ??

[15] M.N. Ozisik. *Heat Conduction*. Wiley-Interscience, 2 edition, 1993.

[16] A. Sakurai. On the propagation and structure of the blast wave i. *Journal of the Physical Society of Japan*, 8:662, September-October 1953.

[17] A. Sakurai. On the propagation and structure of the blast wave ii. *Journal of the Physical Society of Japan*, 9:256, March-April 1954.

[18] George Schmidt. *Physics of High Temperature Plasmas*. Academic Press, 2 edition, 1979.

[19] Uri Shumlak. An implicit, conservative multi-temperature mhd algorithm. describes inclusion of implicit solver to WARP3, 1998.

[20] V. Spitzer. *Physics of Fully Ionized Gases*. Interscience Publishers New York, 1956.

[21] M. Storti. Numerical modeling of ablation phenomena as two-phase stefan problems. *International Journal of Heat Mass Transfer*, 38(15):2843 – 2854, 1995.

[22] Karl-Hermann Tacke. Discretization of the implicit enthalpy method for planar phase change. *International Journal for Numerical Methods in Engineering*, 21(3):543 – 554, 1985.

[23] Bogdan Udrea. *An Advanced Implicit Solver for MHD*. PhD thesis, University of Washington, 1999.

[24] Wong, S.K., Walton A. Numerical solution of single-phase stefan problem using a fictitious material. *Numerical Heat Transfer B*, 35:211 – 223, 1999.

[25] Zel'dovich, Ya. B., Raizer, Yu. P. *Physics of Shock Waves and High-Temperature Hydrodynamic Phenomena*, volume 1. Academic Press, 1966.

Appendix A

ALGEBRAIC REDERIVATION OF FAST/SLOW ALFVÉN WAVES

We begin by defining both the fast and slow Alfvén speeds as

$$\alpha_f^2 = \frac{a^2 - c_s^2}{c_f^2 - c_s^2} \tag{A.1}$$

$$\alpha_s^2 = \frac{c_f^2 - a^2}{c_f^2 - c_s^2} \tag{A.2}$$

where $a^2 = \gamma p/\rho$ is the square of the sonic speed, c_f and c_s are the fast and slow magnetosonic speeds, respectively. The fast and slow magnetosonic speeds are given by

$$c_f^2 = \frac{1}{2}\left(\frac{\gamma p}{\rho} + \frac{B^2}{\rho} + \sqrt{\left(\frac{\gamma p + B^2}{\rho}\right)^2 - \frac{4\gamma p B_x^2}{\rho}}\right) \tag{A.3}$$

$$c_s^2 = \frac{1}{2}\left(\frac{\gamma p}{\rho} + \frac{B^2}{\rho} - \sqrt{\left(\frac{\gamma p + B^2}{\rho}\right)^2 - \frac{4\gamma p B_x^2}{\rho^2}}\right) \tag{A.4}$$

where $gamma = c_p/c_v$ is the ratio of specific heats, p is pressure, ρ is density, B is the magnetic field, and B_x is the magnetic field in the x direction.

Limiting Cases

There are two limiting cases that we need to consider for the magnetosonic fast and slow speeds given by Eqn (A.3, A.4); namely when either the magnetic field or pressure equal zero.

$$B = O, \begin{cases} c_f^2 = \frac{\gamma p}{\rho} = a^2 & \rightarrow \alpha_f^2 = 1 \\ c_s^2 = 0 & \rightarrow \alpha_s^2 = 1 \end{cases} \tag{A.5}$$

$$P = O, \begin{cases} c_f^2 = \frac{B^2}{\rho} & \rightarrow \alpha_f^2 = \frac{a^2}{c_f^2} \\ c_s^2 = 0 & \rightarrow \alpha_s^2 = 1 - \frac{a^2}{c_f^2} \end{cases} \tag{A.6}$$

Eqn (A.1) is expanded out using definitions given by Eqn (A.3,A.4),

$$\alpha_f^2 = \frac{a^2 - \frac{1}{2}\left(\frac{\gamma p}{\rho} + \frac{B^2}{\rho} - \sqrt{\left(\frac{\gamma p + B^2}{\rho}\right)^2 - \frac{4\gamma p B_x^2}{\rho^2}}\right)}{\sqrt{\left(\frac{\gamma p + B^2}{\rho}\right)^2 - \frac{4\gamma p B_x^2}{\rho^2}}} \tag{A.7}$$

Combining like terms we write,

$$\alpha_f^2 = \frac{1}{2}\left(\frac{a^2 - \frac{B^2}{\rho}}{\sqrt{\left(\frac{\gamma p + B^2}{\rho}\right)^2 - \frac{4\gamma p B_x^2}{\rho^2}}} + 1\right) \tag{A.8}$$

Equally, Eqn (A.2) is expanded out using the same definitions for the magnetosonic speeds,

$$\alpha_s^2 = \frac{\frac{1}{2}\left(\frac{\gamma p}{\rho} + \frac{B^2}{\rho} - \sqrt{\left(\frac{\gamma p + B^2}{\rho}\right)^2 - \frac{4\gamma p B_x^2}{\rho^2}}\right) - a^2}{\sqrt{\left(\frac{\gamma p + B^2}{\rho}\right)^2 - \frac{4\gamma p B_x^2}{\rho^2}}} \tag{A.9}$$

Again, combining like terms we write,

$$\alpha_s^2 = \frac{1}{2}\left(\frac{\frac{B^2}{\rho} - a^2}{\sqrt{\left(\frac{\gamma p + B^2}{\rho}\right)^2 - \frac{4\gamma p B_x^2}{\rho^2}}} - 1\right) \tag{A.10}$$

Finally, we note that Eqn (A.8, A.10) are of the similar form, and can be expressed as

$$\alpha_{f,s}^2 = \pm\frac{1}{2}\left(\frac{1 - \frac{B^2}{\rho a^2}}{\sqrt{\left(1 + \frac{B^2}{\rho a^2}\right)^2 - \frac{4\gamma p B_x^2}{\rho^2 a^2}}} \pm 1\right) \tag{A.11}$$

Appendix B

NORMALIZING TEMPERATURE AND NUMBER DENSITY

This appendix briefly outlines the normalization of species' temperatures and number densities.

B.1 Temperature

In order to non-dimensionalize temperature, we begin with the neo-classical perfect gas law (equation of state),

$$\frac{p}{\rho} = \frac{MW}{R}T \tag{B.1}$$

where MW is the molecular weight and R is the universal gas constant. Next, we define the non-dimensional temperature as

$$\bar{T} = \frac{T}{T_{ref}} = \frac{\bar{p}}{\bar{\rho}} \tag{B.2}$$

where both non-dimensional pressure and density[1] are,

$$\bar{p} = \frac{p}{B_{ref}^2 \mu_o} \tag{B.3}$$

$$\bar{\rho} = \frac{\rho}{\rho_{ref}} \tag{B.4}$$

Setting both Eqn (B.1) and Eqn (B.2) to T, we solve for T_{ref}. After some very straightforward algebraic manipulation, we obtain,

$$T_{ref} = \frac{B_{ref}^2}{\rho_{ref}\mu_o}\frac{MW}{R} \tag{B.5}$$

or,

$$\bar{T} = \frac{T}{\frac{B_{ref}^2}{\rho_{ref}\mu_o}\frac{MW}{R}} \tag{B.6}$$

---

[1]See Udrea [23] on page 28 for a complete set of non-dimensional variables.

B.2 Number Density

Similar to the above section we begin with the dimensional form of the bulk pressure that is equal to the sum of the bulk pressures, or

$$p = p_i + p_e + p_n \tag{B.7}$$

where the subscripts i, e, and n denote ions, electrons, and neutrals, respectively. Pressure can be related to the random motion of the particulars in a given volume exerting a force on the surface of the said volume. We write this idea as

$$p = n_i T_i + n_e T_e + n_n T_n \tag{B.8}$$

The non-dimensional for of Eqn (B.8) is simply

$$\bar{p} = \bar{n}_i \bar{T}_i + \bar{n}_e \bar{T}_e + \bar{n}_n \bar{T}_n \tag{B.9}$$

Again, we use the definitions from Eqn (B.3,B.5) to expand both Eqn (B.8) and Eqn (B.9). Setting both equations equation to each other, and after some algebraic manipulation we obtain

$$\bar{n} = n \frac{MW}{R \rho_{ref}} \tag{B.10}$$

Using Eqn (B.4) along with the definition of the dimensional number density, $n = \rho / m_i$, to write

$$\begin{aligned} \bar{n} &= \frac{\rho}{\rho_{ref}} \frac{MW}{R m_i} \\ \bar{n} &= \rho_{ref} \frac{1}{AMU \times R} \end{aligned} \tag{B.11}$$

where m_i is the mass of the ion and AMU is the conversation from kilograms to atomic mass units.

Appendix C

NORMALIZATION OF IONIZATION RATE EQUATION

This appendix outlines the author's efforts to normalize the ionization rate for use with WARP3. This ionization rate, n_i, will be used to obtain the ion fraction, f_i, which will affect the hyperbolic terms of our MHD set of equations. The following provides a heuristic overview in order to better place in context the rationale for the final method.

C.1 Rationale

It might be argued that it would be more straightforward to rewrite the ionization rate in terms of the ion fraction directly. However, recall the relation between these two terms is,

$$N_i = f_i \frac{\rho_i}{m_i} \tag{C.1}$$

where ion density and mass are ρ_i and m_i, respectively.

It can easily be shown that we need to calculate a coupled equation of density and ion fraction if we were to solve for the ion fraction directly. Instead, the author believes it will be computationally less expensive to normalize the ionization rate with a similarity parameter and then solve for the ion fraction. This report will outline this effort.

C.2 Normalization

Let us begin by considering Zel'dovich's [25] equation for the ionization rate,

$$\frac{dN_e}{dt} = \alpha_e N_a N_e + \beta_e N_i {N_e}^2 \tag{C.2}$$

where the subscripts i, e, and a represent ion, electron, and atom quantities.

However, as we are only interested in singly ionized regimes we can simplify Eqn (C.2) and write,

$$\frac{dN_i}{dt} = \alpha_e N_a N_i + \beta_e {N_i}^3 \tag{C.3}$$

Next, following in the spirit of work completed by Bogdan Udrea's work to normalize the MHD equations we, too, can do the same with Eqn (C.3). Let us introduce the convention of placing a bar over normalized quantities, writing

$$\begin{aligned} \bar{N} &= \frac{N_i}{N_{ref}}, \\ \bar{t} &= \frac{t}{\frac{L}{V}} \end{aligned}$$

We can now write the normalized equation including a similarity parameter, Sh ,

$$\frac{d\bar{N}_i}{d\bar{t}} = Sh\left(\alpha_e N_a N_i + \beta_e {N_i}^3\right) \tag{C.4}$$

where the similarity parameter is defined as,

$$Sh \equiv \frac{\frac{L}{V}}{N_{ref}}$$

As we have not looked at neither α_e nor β_e one might be inclined to rewrite the similarity parameter with reference quantities used with Udrea's [23] own normalization. Or, in short we can write,

$$Sh \equiv \frac{\frac{a}{C_a}}{\frac{\rho}{m_i}}$$

where, a , C_a , ρ , and m_i are: plasma length; alfvén speed; density; and ion mass, respectively. Unfortunately, this is but the beginning. We must now look at the terms α_e and β_e .

C.2.1 Ionization Rate Parameter, α_e

Let us begin by writing out the ionization rate parameter,

$$\alpha_e = \sigma_e \bar{v}_e \left(\frac{I}{kT_e} + 2\right) e^{-\frac{I}{kT_e}} \tag{C.5}$$

where σ_e , $\bar{v_e}$, I , T_e are: cross-sectional reaction rate; thermal velocity; ion potential; and, electron temperature, respectively. More succinctly, we write:

$$\begin{aligned} \sigma_e &= CkT \\ v_e &= \left(\frac{8kt}{\pi m_e}\right)^{\frac{1}{2}} \end{aligned}$$

where we can now rewrite Eqn (C.5) as,

$$\alpha_e = CkT\left(\frac{8kt}{\pi m_e}\right)^{\frac{1}{2}}\left(\frac{I}{kT_e}+2\right)e^{-\frac{I}{kT_e}} \tag{C.6}$$

We can rearrange Eqn (C.6), and replace kT with $\bar{T}$, the normalized temperature and m_e with $\bar{m}$ where $\bar{m} \equiv \frac{m_e}{m_i}$.

$$\bar{\alpha_e} = \left[\phi^{-1}\bar{C}\right]\left[{m_i}^{-\frac{1}{2}}\left(\frac{8\bar{T}}{\pi\bar{m}}\right)^{\frac{1}{2}}\right]\left[\phi\left(\bar{I}+\frac{2\bar{T}}{\phi}\right)\right]\left[\left(e^{-\frac{\bar{I}}{\bar{T}}}\right)^{\phi}\right] \tag{C.7}$$

where we define $\phi \equiv \rho_{ref}{C_a}^2$ for the sake of simplicity.

In order to complete the normalization we simplify Eqn (C.7) with ${m_i}^{-\frac{1}{2}}$ is pulled out front to become a part of the Shumlak number. However, both the third and fourth terms of the above equation pose a problem. The second part of the third term needs to be non-dimensionalized, possibly by multiply through by $\frac{{B_{ref}}^2}{\mu_o}$. The fourth term is of concern with its power of the ϕ^{th} degree. Nevertheless, we can write a partially normalized ionization rate parameter as,

$$\bar{\alpha_e} = {m_i}^{-\frac{1}{2}}\bar{C}\left(\frac{8\bar{T}}{\pi\bar{m}}\right)^{\frac{1}{2}}\left(\bar{I}+\frac{2\bar{T}}{\phi}\right)\left(e^{-\frac{\bar{I}}{\bar{T}}}\right)^{\phi} \tag{C.8}$$

C.2.2 Recombination Rate Parameter, β_e

Now let us introduce the recombination rate parameter,

$$\beta_e = \frac{g_a}{g_+}\left(\frac{I}{kT_e}+2\right)\frac{h^3\sigma_e}{2\pi^2{m_e}^2kT} \tag{C.9}$$

where we note that g are the probabilities of ionization for atoms and ions, respectively– and according to Zel'dovich we can assume that they are first-order driven, or

$$\frac{g_a}{g_i} \simeq 1$$

In short order let us introduce the normalized version of Eqn (C.9) by writing,

$$\bar{\beta}_e = \left[\left(\bar{I} + \frac{2\bar{T}}{\phi}\right)\right]\left[\phi^{-1}\bar{C}\right]\left[\frac{h^3}{2\pi^2 m_e \bar{T}}\right] \tag{C.10}$$

and after simplifying we write succinctly,

$$\bar{\beta}_e = \frac{h^3}{2\pi^2 m_e}\left(\bar{I} + \frac{2\bar{T}}{\phi}\right)\frac{\bar{C}}{\bar{T}} \tag{C.11}$$

C.3 Results

We can now rewrite our results thus far by inserting them back in Eqn (C.4),

$$\frac{d\bar{N}_i}{d\bar{t}} = Sh\left(m_i{}^{-\frac{1}{2}}\bar{C}\left(\frac{8\bar{T}}{\pi\bar{m}}\right)^{\frac{1}{2}}\left(\bar{I} + \frac{2\bar{T}}{\phi}\right)\left(e^{-\frac{\bar{I}}{\bar{T}}}\right)^{\phi} N_a N_i + \frac{h^3}{2\pi^2 m_e}\left(\bar{I} + \frac{2\bar{T}}{\phi}\right)\frac{\bar{C}}{\bar{T}} N_i{}^3\right) \tag{C.12}$$

Unfortunately, as noted before, the above solution is not adequate. The author was not satisfied that the ϕ^{th} ordered term can be sufficiently reduced. Further, the issue of the non-dimensional $\frac{2\bar{T}}{\phi}$ term would need to be mathematically satisfied, too.

C.4 Partial Normalization

At this point, a less rigorous approach that does not attempt to normalize the entire equation is necessary. Let us begin by considering Zel'dovich's equation for the ionization rate of a singly ionized species, or

$$\frac{d\overline{N_i}}{d\bar{t}} = Sh\left(\alpha_e N_a N_i + \beta_e N_i^3\right) \tag{C.13}$$

where the similarity parameter is,

$$Sh \equiv \frac{\frac{L}{V}}{N_{ref}}$$

and we apply the single-fluid model assumptions to the flow, or,

$$\begin{aligned} T_i &= T_e \\ N_i &= N_e \\ \frac{m_e}{m_i} &\ll 1 \end{aligned}$$

C.4.1 Rate Parameters

The ionization and recombination rate parameters are, respectively,

$$\alpha_e = \sigma_e \overline{v_e} \left(\frac{I}{kT_e} + 2 \right) e^{-\frac{I}{kT_e}} \tag{C.14}$$

$$\beta_e = \frac{g_a}{g_i} \left(\frac{I}{kT_e} + 2 \right) \frac{h^3 \sigma_e}{2\pi^2 m_e^2 k T_e} \tag{C.15}$$

where,

$$\begin{aligned} \sigma_e &= CkT_i \\ \\ \bar{v_e} &= \left(\frac{8kT_i}{\pi m_e} \right)^{1/2} \\ &\Downarrow \\ \overline{v_e} &= m_i^{-1/2} \left(\frac{8kT_i}{\pi \overline{m}} \right)^{1/2}, \overline{m} = \frac{m_e}{m_i} \\ \\ \frac{g_a}{g_i} &\approx 1 \end{aligned}$$

We can expand and simplify Eqn (C.14, C.15) to write,

$$\alpha_e = CkT_i \left(m_i^{-1/2} \left(\frac{8kT_i}{\pi \overline{m}} \right)^{1/2} \right) \left(\frac{I}{kT_e} + 2 \right) e^{-\frac{I}{kT_e}} \tag{C.16}$$

$$\beta_e = C \left(\frac{I}{kT_e} + 2 \right) \frac{h^3}{2\pi^2 m_e^2} \tag{C.17}$$

C.4.2 Dimensionalizing Temperature

As shown in the previous report, it is too difficult to permit temperature, T, to remain in non-dimensional form. Consequently, the author has decided to dimensionalize the temperature in an attempt to simplify the overall normalization of the equation.

We begin by noting the non-dimensional and dimensional forms of temperature are written respectively as,

$$\begin{aligned} \overline{T} &= \frac{\overline{P}}{\overline{\rho}} \\ T &= \frac{P}{\overline{R}\rho}, \overline{R} = \frac{\Re}{MW} \end{aligned}$$

where $\Re$ is the universal gas constant and MW is the molecular mass.

We substitute for the dimensional forms of pressure, P, and density, ρ, with their non-dimensional forms to write the final form of the dimensional temperature as,

$$T = \left(\frac{A_v m_i}{\Re}\right)\left(\frac{\overline{P}}{\overline{\rho}}\right)\left(\frac{B_{ref}^2}{\mu_o \rho_{ref}}\right) \tag{C.18}$$

where A_v is the Avogadro constant.

C.5 Normalized Ionization Rate Parameter

The final form of our equation in its entirety becomes,

$$\frac{d\overline{N_i}}{d\overline{t}} = Sh \left(\begin{array}{c} CkT\left(m_i^{-1/2}\left(\frac{8kT}{\pi \overline{m}}\right)^{1/2}\right)\left(\frac{I}{kT}+2\right)e^{-\frac{I}{kT}} N_a N_i \\ + \\ C\left(\frac{I}{kT}+2\right)\frac{h^3}{2\pi^2}\frac{\sqrt{m_i m_e}}{m_e} N_i^3 \end{array} \right)$$

where the similarity parameter is,

$$Sh \equiv \frac{\frac{a}{C_a}}{\frac{\rho_{ref}}{m_i^2}}$$

We can further simplify this similarity parameter by noting that the Alfvén speed, C_a, is equal to $\frac{B_{ref}}{\sqrt{\mu_o \rho_{ref}}}$. Or,

$$Sh \equiv \frac{a \mu_o^{1/2} m_i^{1/2}}{B_{ref}^2 \rho_{ref}^2}$$

C.6 Required Variables for Input Deck

This method requires the addition of four (4) variables to the input deck. Notably these variables are,

- B_{ref}, reference magnetic field
- ρ_{ref}, reference density
- a, plasma length

- m_i, mass of the ion

However, the first three of these can be for the most part set to default values and forgotten. To point, when set to 1.121×10^{-3}, 1.0, and 3.04803×10^{-1}, respectively then the similarity parameter is equal to unity, or

$$Sh = L_{plasma} * \sqrt{\frac{\mu_o \times \rho_{ref}}{B_{ref} \times \rho_{ref}}} \tag{C.19}$$

Appendix D

DIATOMIC GAS TABLE

The following table is taken directly from Zel'dovich [25] found on page 389, Chapter 6, Section 11 and titled "Ionization by Electron Impact." It provides a list of the available diatomic gases for use with WARP3 ionization calculator. Additionally, one column has been added that includes the atomic mass of each molecule. The researcher has the option of configuring any one or all of the gas parameters, thereby providing great versatility when making runs without having to necessarily recompile between runs.

Ionization energy, I, refers to the amount of energy necessary to remove an electron from ground state and sufficiently separate it from the atom. Ionization constant, C, refers to a constant relating reaction cross section to ionization energy ionization energy. Region of applicability refers the energy region where the tabled values are appropriate, though in WARP3 we use these values on a much wider range than shown. Electron energy refers to the amount of kinetic energy an electron has when bombarding an atom. Finally, the cross section refers to the area perpendicular to the atom trajectory where interaction between itself and electrons occurs.

Table D.1: Ionization by Electron Impact

Atom	**Atomic Mass**	**Ion. Energy**	**Ion. Constant**	**Region of Application**	**Electron Energy**	**Cross Section**
	m	I_e	$C \times 10^{17}$		U_e	$\sigma \times 10^{16}$
molecule	*AMU*	eV	$cm^2 \times eV^{-1}$	eV	eV	cm^2
H_2	2.01588	15.4	0.59	16-25	70	1.1
He	4.002602	24.5	0.13	24.5-35	100	0.34
N	14.00674	14.6	0.59	15-30	100	2.1
N_2	28.01348	15.6	0.85	16-30	110	3.1
O	15.9994	13.6	0.60	14-25	80	1.5
O_2	31.9988	12.1	0.68	13-40	110	—
NO	30.00614	9.3	0.82	10-20	100	3.25
Ar	39.948	15.8	2.0	15-25	100	3.7
			1.7	15-18		
Ne	20.1797	21.5	0.16	21.5-40	160	0.85
Hg	200.59	10.4	7.9	10.5-13	42	5.4
			2.7	10.5-28		

Appendix E

WARP3 USER'S GUIDE, SECOND EDITION

The original **WARP3 User's Guide** was produced in Udrea's doctoral dissertation [[23]]. The work of both Udrea and the author occurred simultaneously but separately, and as a consequence the first guide did not include the author's contributions as they were still a work in progress. Much thanks goes to Bogdan Udrea for the considerable amount of effort that went into the first edition.

This appendix is intended to be a user guide for WARP3. It assumes that the users have a basic familiarity with the UNIX operating system and with FORTRAN90 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 next section presents in tabular form all the main variables used by WARP3. The next sections describe how to operate WARP3 on both Alpha clusters and IBM SP2, including listings of useful scripts. Finally, a brief description of how to expand WARP3 capabilities is provided.

E.1 Code Organization

This section gives a description of the program flow and presents the main procedures used by WARP3. Pseudocode 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 procedures that read in the grid file and 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 than that, in some cases (depending on the compiler) using this attribute improves code performance by optimizing the alignment of the variables in memory.

E.1.1 WARP3.f

The 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.

`Job Init`

–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 coworkers processors

–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`

–TecPlotDump; writes plot files to disk using Tecplot ASCII format

`fbin`

–PrimBinDump; writes plot files to disk using 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

–in the case of including ionization, a second procedure is called in order to correctly set normalization factors along with the diatomic gas mass, ionization energy, et cetera

–*See init_ionfraction.f for details.*

```
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 or thermal) then
                    EdgeSend
                    EdgeRecv
                    BdryWait
                    BdryWait
               end
               if (parabolic)
                    FlipFaces
```

```
     –change the ordering of the internal ghost cells accord-
     ing to the truth table previously calculated (only on non-
     simply connected grids)
     if (icount=0)
               select (gridfile_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
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
     –if the simulation is three dimensional then it computes
     the fluxes in the z direction
end
DivBSource
–computes source term proportional to the divergence of the
magnetic field
ResistiveFlux
```

```
        –computes the parabolic flux due to resistivity
        –computes contribution to change in electron temperature
        ViscousFlux
        –computes the parabolic flux due to viscosity
        –computes contribution to change in neutral and ion tempera-
        tures
        ThermalFlux
        –computes the parabolic flux due to thermal diffusion
        –computes contribution to change in neutral, ion, and electron
        temperatures
        AxiSymmetricSource
        –computes the axisymmetric source terms
        dqCalc
        –computes the increment of primitive variables
        UpdateSol
        –updates the solution (by adding dQ to Q)
        UpdateTemperature
        –updates the neutral, ion, and electron temperatures
        IonRate
        –updates the local ionization fraction as a function of electron
        temperature
        Converge
        –calculates the two-norm of the residual else
        –implicit scheme starts here
    else
        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 + d
                icount = icount + 1
                –increments time and iteration counter
                if (mod(t,tplt_dump)<=dt) or
                 (mod(icount,ncycplt_dump)=0) then
                    select (gridfile_style)
                         tecplot
                         –TecPlotDump; writes plot files to disk using Tecplot
                         ASCII format
                         fbin
                         –PrimBinDump; writes plot files to disk using FOR-
                         TRAN binary format
                    end
                end
            end
        end
    end
```

E.1.2 PTIMELOOP.f

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.

```
    –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
end
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 avail-
     able)
     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 avail-
     able)
     if (thermal) then
          ThermalFlux
          LimitViscFluxJacobian
     end
     –if the parabolic thermal terms are on then compute the thermal 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 hep-
     tadiagonal) system of linear equations that results from the implicit dis-
     cretization
     UpdateSol
     UpdateTemperaure
     IonRate
     m = m + 1
end
–pseudo time loop ends here
Converge
```

–deallocates memory used for dynamics arrays

E.2 Input File Structure

In this section the structure of the input file is discussed using a spheromak simulation input file[1] as an example. Note that the following example does not cover all variables, though. Please consult E.2 for a complete listing along with default values.

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, Lundquist and Peclet numbers and the switches that toggle the calls of the hyperbolic and parabolic flux calculations

6. `globaldata` - contains the values of the primitive variables that the domain is initialized with

[1] Sometimes input files are called input decks by the people who used punch cards in their youth.

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/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 (See Udrea [23] non-dimensional set of equations) 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`$= \eta\_ref/ac\_a$ and the inverse of the Reynolds number is `reyn1`$= \nu\_ref/ac\_a$, 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"
/
```

E.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 broken into three categories; namely, primitive; input; and, application. The first and second category variables are declared in `WARP3MODULES.f`, while the third category variables are declared in `APPMODULES.f`. The ”primitive” category variables can only be modified from editing `WARP3MODULES.f` and then re-compiling the code. The ”input” category variables are accessible through the input file, controlling the basic functionality of WARP3. Finally, the ”applications” category variables are specific to applications created by the researcher, and are accessible through the input file. While this last category can never be an exhaustive list of available applications, nonetheless the most commonly used applications are included. Note that some variables listed in the last two categories may not be available to the input file. In these instances the variable name is proceeded by ♣.

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.

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.

E.3.1 Primitive

Name	**Type**	**Comments**	**Module**
R_SZE = 8	INTE	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	INTE	maximum length of strings - used to size strings such as file names	basic_const
MAX_BLOCKS = 200	INTE	maximum number of blocks per processor	basic_const

continued on next page

Primitives *continued from previous page*			
Name	**Type**	**Comments**	**Module**
TAG_FAK = 1000	INTE	factor used to uniquely identify MPI tags	basic_const
C_VARS = 8	INTE	number of conserved variables	basic_const
startTime	REAL	initializes timer (MPI) - used to calculate a job wall clock time	basic_const
endTime	REAL	finishes timer(MPI) - used to calculate a job wall clock time	basic_const
unit = 10	INTE	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 F90 intrinsic function	universal_const
BIGG	REAL	largest real value - obtained with huge F90 intrinsic function	universal_const
PIE	REAL	number π	universal_const
MU0	REAL	μ_0 - magnetic permitivity of free space, $4\pi \times 10^{-7}$	universal_const
AMU	REAL	Atomic Mass Units, $1.66053873 \times 10^{-27}$ [AMU $\times kg^{-1}$]	universal_const

continued on next page

Primitives *continued from previous page*			
Name	**Type**	**Comments**	**Module**
UniGasCnt	REAL	Universal Gas Constant, R, $8.3135d3$ [$J \times (kg - molK)^{-1}$]	universal_const
BoltzCnt	REAL	Boltzmann Constant, k, $8.617342d-5$ [$eV \times K^{-1}$]	universal_const
PlankCnt	REAL	Plank Constant, h, $6.62606876 \times 10^{-34}$ [$J \times s$]	universal_const
eV2J	REAL	convert electron volts, eV, to joules, J, $1.602176462 \times 10^{-19}$ [$J \times eV^{-1}$]	universal_const
AvogadroNbr	REAL	Avogadro Number, $6.02214199 \times 10^{23}$	universal_const
masse	REAL	electron rest mass, 9.1094×10^{-31} [kg]	universal_const
IonFracLowerLimit	REAL	ion fraction lower limit, 1.0×10^{-50} (must never be ZERO)	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

continued on next page

Primitives *continued from previous page*			
Name	**Type**	**Comments**	**Module**
`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 in one time step	`consrvd_arrays`
`etax( :, :, :, :)`	REAL	electric resistivity in x direction	`diffus_arrays`
`etay( :, :, :, :)`	REAL	electric resistivity in y direction	`diffus_arrays`

continued on next page

Primitives *continued from previous page*

Name	Type	Comments	Module
`etaz( :, :, :, :)`	REAL	electric resistivity in z direction	`diffus_arrays`
`visc( :, :, :, :)`	REAL	dynamics viscosity of	`diffus_arrays`
`kappax( :, :, :, :)`	REAL	thermal conductivity in x direction	`diffus_arrays`
`kappay( :, :, :, :)`	REAL	thermal conductivity in y direction	`diffus_arrays`
`kappaz( :, :, :, :)`	REAL	thermal conductivity in z direction	`diffus_arrays`
`fi( :, :, :, :)`	REAL	ion fraction	`diffus_arrays`
`dTi( :, :, :, :)`	REAL	change in temperature of ions	`tempera-ture_arrays`
`dTe( :, :, :, :)`	REAL	change in temperature of electrons	`tempera-ture_arrays`
`dTn( :, :, :, :)`	REAL	change in temperature of neutrals	`tempera-ture_arrays`
`Ti( :, :, :, :)`	REAL	temperature of ions	`tempera-ture_arrays`
`Te( :, :, :, :)`	REAL	temperature of electrons	`tempera-ture_arrays`
`Tn( :, :, :, :)`	REAL	temperature of neutrals	`tempera-ture_arrays`
`Temperature( :, :, :, :, :)`	REAL	all three temperatures in one for message passing purposes	`tempera-ture_arrays`

continued on next page

Primitives *continued from previous page*			
Name	**Type**	**Comments**	**Module**
icels (MAX_BLOCKS)	INTE	number of cells in i direction	sze_arrays
jcels (MAX_BLOCKS)	INTE	number of cells in j direction	sze_arrays
kcels (MAX_BLOCKS)	INTE	number of cells in k direction	sze_arrays
ierr	INTE	message error ID number (MPI)	misc_vars
taskID	INTE	task or processor ID	misc_vars
iMax	INTE	maximum array size on a processor	misc_vars
jMax	INTE	maximum array size on a processor	misc_vars
kMax	INTE	maximum array size on a processor	misc_vars
flip_ijOrd	LOGI	truth table used for non-simply connected grids	misc_vars
t	REAL	dimensionless time	misc_vars
dt	REAL	dimensionless time step	misc_vars
icount	INTE	iterations counter	misc_vars
faceSendReq	INTE	request handles for the MPI_Wait	misc_vars
faceRecvReq		operations for double face message passing	
edgeSendReq	INTE	request handles for the MPI_Wait	misc_vars

continued on next page

Primitives *continued from previous page*			
Name	**Type**	**Comments**	**Module**
edgeRecvReq		operations for frame message passing	
idir	INTE	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

E.3.2 Input

Name	**Type**	**Comments**	**Module**
application_name	CHAR	application identifier - needed for application specific I/O	main_ctrl
run_diagnostic	LOGI	switch that toggles diagnostics procedure calls	main_ctrl
ncycplt_dump	INTE	number of cycles between plot file dumps	main_ctrl
tplt_dump	REAL	dimensionless time units between plot file dumps	main_ctrl
restart	LOGI	if TRUE then WARP3 reads a restart file dumped at iteration ncyc rst_begin	main_ctrl

continued on next page

Inputs *continued from previous page*			
Name	**Type**	**Comments**	**Module**
ncycrst_begin	INTE	number of cycles from where WARP3 restarts	main_ctrl
ncycrst_dump	INTE	number of cycles between restart file dumps	main_ctrl
trst_dump	REAL	dimensionless time units between restart file dumps	main_ctrl
ncycfin	INTE	number of iterations after which WARP3 stops	main_ctrl
tfin	REAL	dimensionless time units after which WARP3 stops	main_ctrl
gridfile_style	CHAR	takes values tecplot for Tecplot ASCII or fbin for FORTRAN binary	main_ctrl
dump_style	CHAR	takes values tecplot for Tecplot ASCII or fbin for FORTRAN binary	main_ctrl
grid_file	CHAR	name of the grid file	main_ctrl
grid_dir	CHAR	directory where the grid file resides	main_ctrl
out_dir	CHAR	directory where the data files are dumped	main_ctrl

continued on next page

Inputs *continued from previous page*			
Name	**Type**	**Comments**	**Module**
rst_dir	CHAR	directory where the restart files are dumped	main_ctrl
hosts_file	CHAR	name of the file where the processor names are dumped (full path)	main_ctrl
out_int=1	INTE	every out_int iteration, WARP3 dumps to STDOUT status of run; useful for long runs where the status report is piped to output file that can become excessively large	main_ctrl
nblock_tot	INTE	total number of blocks	block_info
nblock	INTE	number of blocks per processor	block_info
block_topo	INTE	stores the IDs of neighbors and the process ID to which the block is distributed	block_info
blockID	INTE	local to global mapping	block_info
bc_mag	CHAR	magnetic BCs for each block	block_info
bc_hydro	CHAR	hydrodynamic (fluid) BCs for each block	block_info
courant_phys	REAL	physical Courant number	alg_ctrl

continued on next page

Inputs *continued from previous page*			
Name	**Type**	**Comments**	**Module**
courant_pseudo	REAL	pseudo time Courant number	alg_ctrl
ncyc_pseudo	INTE	maximum number of pseudo time iterations	alg_ctrl
ncyc_sgs	INTE	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	INTE	WARP3 stops if the residual drops these many orders of magnitude	alg_ctrl
implicit	LOGI	if FALSE run explicit mode only	alg_ctrl
first_order_space	LOGI	if TRUE WARP3 runs in first order accurate in space mode	alg_ctrl
axisymmetric	LOGI	if TRUE WARP3 runs in the axisymmetric mode	alg_ctrl
parabolic	LOGI	if TRUE WARP3 calculates hyperbolic fluxes	alg_ctrl
continued on next page			

Inputs *continued from previous page*			
Name	**Type**	**Comments**	**Module**
hyperbolic	LOGI	if TRUE WARP3 calculates parabolic fluxes Note: It is possible to turn off hyperbolic fluxes, leaving on the parabolic fluxes and use WARP3 to determine thermal balance between two metals, using for designing containers to hold plasma.	alg_ctrl
gam	REAL	specific heats ratio (c_p/c_v)	phys_ctrl
massi	REAL	ion mass [AMU]	phys_ctrl
viscous	LOGI	if TRUE turn on viscous parabolic fluxes	phys_ctrl
resistive	LOGI	if TRUE turn on resistive parabolic fluxes	phys_ctrl
thermal	LOGI	if TRUE turn on thermal parabolic fluxes	phys_ctrl
magnetic	LOGI	TRUE if $\|\|B\|\| = 0$ turn on, else false	phys_ctrl
ionization	LOGI	if TRUE calculate ionization fraction	phys_ctrl
reyn1	REAL	inverse of Reynolds number	phys_ctrl
			continued on next page

Inputs *continued from previous page*			
Name	**Type**	**Comments**	**Module**
lund1	REAL	inverse of Lundquist number	phys_ctrl
peclet1	REAL	inverse of Peclet number	phys_ctrl
resistivity_model	CHAR	resistivity model selector	phys_ctrl
conductivity_model	CHAR	thermal conductivity model selector, currently only "pgl" is available	phys_ctrl
slip_wall	LOGI	if TRUE then the tangent velocity at the wall has a finite value	phys_ctrl
Brems = 1.7×10^{-38}	REAL	Bremsstrahlung Radiation Constant	phys_ctrl
cond_ratio1 = 1×10^{-7}	REAL	thermal conductivity ratio $\frac{\kappa\_\parallel}{\kappa\_\perp}$ = cond_ratio	phys_ctrl
coulomb_log = 14	REAL	plasma parameter, Λ	phys_ctrl
gas	CHAR	atomic molecule of flow see D for list of gases	phys_ctrl
Bref	REAL	reference magnetic field Λ, default is 1, [Tesla]	phys_ctrl
RHOref	REAL	reference density, default is 1, [$kg \times m^{-3}$]	phys_ctrl
PlasmaLength	REAL	reference plasma length, default is 1, [m]	phys_ctrl
IonSimilarParmtr ♣	REAL	ion similarity parameter	phys_ctrl

continued on next page

Inputs *continued from previous page*			
Name	**Type**	**Comments**	**Module**
IonTempCnt	REAL	ion temperature conversion, default is 0	phys_ctrl
IonCSCnt	REAL	ion cross-section, default is 0, [$cm^2 \times eV^{-1}$]	phys_ctrl
IonPotent	REAL	ionization potential energy, default is 0, [eV]	phys_ctrl
IonBombard	REAL	ionization bombardment energy, default is 0, [eV]	phys_ctrl
dragx♣	REAL	pressure drag in x direction	phys_ctrl
dragy♣	REAL	pressure drag in y direction	phys_ctrl
dragz♣	REAL	pressure drag in z direction	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
continued on next page			

Inputs *continued from previous page*			
Name	**Type**	**Comments**	**Module**
`bzig`	REAL	overall initial magnetic field in z direction	`global_init_ctrl`
`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`
`kappaig`	REAL	overall initial thermal conductivity	`global_init_ctrl`
`fiig`	REAL	overall initial ionization fraction; if less than 0 then initial value is steady state, otherwise if $\in [0,1]$ then initial f_i set to `fiig`	`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`
continued on next page			

Inputs *continued from previous page*			
Name	**Type**	**Comments**	**Module**
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
kappai(:)	REAL	overall initial thermal conductivity in each block	loca_init_ctrl
fii(:)	REAL	overall initial ionization fraction in each block	local_init_ctrl

E.3.3 Applications

Name	**Type**	**Comments**	**Module**
nrt_vel	INTE	number of cells in r direction of linear stability code	smak_data
nzt_vel	INTE	number of cells in z direction of linear stability code	smak_data

continued on next page

Applications *continued from previous page*			
Name	**Type**	**Comments**	**Module**
nrt_bfield	INTE	number of cells in r direction of equilibrium code	smak_data
nzt_bfield	INTE	number of cells in z direction of equilibrium code	smak_data
smak_type	CHAR	spheromak type of "tunacan" or "cylinder"	smak_data
R_ent=-1	REAL	radius of the entry region	smak_data
L_ent=-1	REAL	length of the entry region	smak_data
R_fc	REAL	radius of the flux conserver	smak_data
L_fc	REAL	length of the flux conserver	smak_data
bfield_file	CHAR	full path of the bfield data files	smak_data
vel_file	CHAR	full path of the velocity data files	smak_data
disp_file_n1	CHAR	full path of the displacement data files	smak_data
disp_file_n2	CHAR	full path of the displacement data files	smak_data
bfield_type	CHAR	type of initialization for the magnetic fields "interpolated" or "analytic"	smak_data

continued on next page

Applications *continued from previous page*			
Name	**Type**	**Comments**	**Module**
vfield_type	CHAR	type of initialization for the velocity fields "interpolated" or "analytic"	smak_data
vel_fak	REAL	velocity factor after normalization the initial perturbation velocity field is multiplied with this number	smak_data
n1_fak	REAL	velocity factor after normalization the initial perturbation velocity field is multiplied with this number	smak_data
n2_fak	REAL	velocity factor after normalization the initial perturbation velocity field is multiplied with this number	smak_data
nMax	INTE	maximum mode number	smak_data
apply_pert	LOGI		smak_data
displacements	LOGI		smak_data
beta_inf=1	REAL	freestream pressure	wedge_data
rho_inf=1	REAL	freestream density	wedge_data
alpha_inf=1	REAL	freestream angle of attack	wedge_data
continued on next page			

Applications *continued from previous page*			
Name	**Type**	**Comments**	**Module**
Bx_inf=0	REAL	freestream magnetic field in x direction	wedge_data
By_inf=0	REAL	freestream magnetic field in y direction	wedge_data
Bz_inf=0	REAL	freestream magnetic field in z direction	wedge_data
machRamp = .false.	LOGI	if TRUE linearly increase Mach number starting when time = t_lo_speed, continue till time = t_lo_speed when Mach number = Mach_hi_speed Permits higher steady state Mach number	wedge_data
Mach_inf_lo = 1	REAL	Mach number until time equal t_lo_speed	wedge_data
Mach_inf_fi = 1	REAL	Mach number after time equal t_lo_speed t_raise_speed	wedge_data
t_lo_speed = 0	REAL	time when to begin ramping up Mach number	wedge_data
t_raise_speed = 1	REAL	how long to ramp up Mach number	wedge_data

continued on next page

Applications *continued from previous page*			
Name	**Type**	**Comments**	**Module**
wallRamp = .false.	LOGI	if TRUE linearly increase opacity of wall b.c. to flow Permits higher steady state Mach number	wedge_data
Tran_lo = 1	REAL	transmissivity of wall b.c.. if 1 then completely transparent. If -1 then completely opaque.	wedge_data
t_lo_tran = 0	REAL	time when to begin wall b.c. opacity	wedge_data
t_raise_tran = 1	REAL	how long to ramp up wall b.c. opacity	wedge_data
x0 = 0	REAL	position of magnetic dipole in x direction	dipole_data
y0 = 0	REAL	position of magnetic dipole in y direction	dipole_data
mag_moment = 0	REAL	magnetic moment for simple geometry this is current times area	dipole_data
dipole_orientation = 0	REAL	orientation of dipole [radians]	dipole_data
dipole_exists = 0	LOGI	turns dipole ON/OFF	dipole_data
rho_ref	REAL	reference density [$kg \times m^{-3}$]	zap_data
continued on next page			

Applications *continued from previous page*			
Name	**Type**	**Comments**	**Module**
VA_ref	REAL	reference Alfven speed [$m \times s^{-1}$]	zap_data
mi	REAL	proton (ion) mass [kg]	zap_data
Rgas	REAL	gas constant of plasma [$J \times kg^{-1} \times K^{-1}$]	zap_data
T_quart	REAL	quarter period	zap_data
nxt	INTE	cells in i direction	ripple_data
nyt	INTE	cells in j direction	ripple_data
eigenfile	CHAR	file with perturbation data	ripple_data
aa	REAL	reference length	ripple_data

E.4 Applications Explained

This section is intended to provide the researcher with an introduction to a number of applications written for WARP3 that extend its functionality. Whenever a researcher creates a particularly useful application, it is hoped that they will add to this section thereby benefiting the community of WARP3 users.

Please consult Section E.3.3 for an explanation of each application's set of variables. This section will explain usage and technique associated with a particular application.

E.4.1 Wedge

The wedge application permits two means of controlling how the flow velocity is adjusted. When large flow velocities are desired it is necessary to slowly ramp up the flow velocity as to not introduce gross discontinuities within the flow region. The overriding reason is that these discontinuities create regions of negative pressure, resulting

in a segmentation fault when WARP3 attempts to calculate the sonic velocity where it takes the square root of pressure.

Freestream Conditions

There are six variables that control the freestream conditions. They are `beta_inf`, `rho_inf`, `alpha_inf`, `bx_inf`, `by_inf`, and `bz_inf`. These six will set the pressure, density, angle of attack, and magnetic field in x, y, and z direction, respectively.

Wall Ramping

If `wallRamp` equals `.true.` then wherever a hydrodynamic boundary condition is set to `wall` its opacity can be slowly ramped up. When `Tran_lo` equal 1 then all walls are invisible (transparent) to the flow. In short, the boundary condition is effectively set to `copy`. While it not expected that this variable needs to be changed from its default of 1, for the sake of completeness is can be adjusted within the input deck. Conversely, if this variable is set to -1 then all walls are completely opaque; which with foresight is quite fruitless. When the computational time equals the value set by `t_lo_tran` then WARP3 linearly increases the wall opacity. When the computational time is greater than or equal to the sum of `t_lo_tran` and `t_lo_tran` then the wall is completely opaque.

Inflow Ramping

If `machRamp` equals `.true.` then wherever a hydrodynamic boundary condition is set to `inflow` the flow velocity measured as Mach number can be slowly ramped up. When the computational time equals the value set by `t_lo_mach` then WARP3 linearly increases the inflow Mach number with an initial value set by `Mach_inf_lo`. When the computational time is greater than or equal to the sum of `t_lo_mach` and `t_lo_mach` then the inflow Mach number is set by `Mach_inf_hi`.

Various Techniques

It should be pointed out that both the wall and inflow boundary conditions can be ramped in parallel, in series, or separately as so desired. For example, the global flow velocity is set to the sonic velocity. The flow is permitted to stabilize for some amount of computational time while all the walls, initially made transparent, are ramped up to opacity. At this point, the inflow is slowly ramped from Mach 1 to some higher Mach number.

A word of caution accompanies using the wall ramping technique. In cases where severe regions of stagnation exist such as with bluff bodies, it has been observed that wall ramping can produce unstable regions of negative pressure. Furthermore, it might be hard to interpret the results until the flow reaches steady state. Using this technique is analogous to having the walls covered in infinitesimally small holes. As the value of opacity is increased these holes become smaller and smaller until they disappear.

E.4.2 Dipole

This application is an extension of wedge, having all the same features mentioned above plus one. This application allows the placement of a two-dimensional magnetic dipole oriented in the $x - y$ plane anywhere in this said plane. The dipole does not necessarily need to be within the computational region, but it should be obvious that is should be sufficiently strong enough for its effects to reach into the computational domain.

E.5 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

Alpha station 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.

E.5.1 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/w3

 atlas.aa.washington.edu        0   /mhd4/udrea/mhd/warp3/w3
 coeus.aa.washington.edu        1   /mhd4/udrea/mhd/warp3/w3
 crius.aa.washington.edu        1   /mhd4/udrea/mhd/warp3/w3
 epimetheus.aa.washington.edu   1   /mhd4/udrea/mhd/warp3/w3
 gaea.aa.washington.edu         1   /mhd4/udrea/mhd/warp3/w3
 hyperion.aa.washington.edu     1   /mhd4/udrea/mhd/warp3/w3
 iapetus.aa.washington.edu      1   /mhd4/udrea/mhd/warp3/w3
 metis.aa.washington.edu        1   /mhd4/udrea/mhd/warp3/w3
#mnemosyne.aa.washington.edu    1   /mhd4/udrea/mhd/warp3/w3
#oceanus.aa.washington.edu      1   /mhd4/udrea/mhd/warp3/w3
#phoebe.aa.washington.edu       1   /mhd4/udrea/mhd/warp3/w3
#prometheus.aa.washington.edu   1   /mhd4/udrea/mhd/warp3/w3
#rhea.aa.washington.edu         1   /mhd4/udrea/mhd/warp3/w3
#tethys.aa.washington.edu       1   /mhd4/udrea/mhd/warp3/w3
#thea.aa.washington.edu         1   /mhd4/udrea/mhd/warp3/w3
#themis.aa.washington.edu       1   /mhd4/udrea/mhd/warp3/w3
```

The program group file has three (3) 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`.

Maintenance Automation

If operating from the University of Washington Computational Fluid Dynamics Laboratory Alpha cluster, the proceeding PERL5 scripts can be invoked from a line terminal by adding a line to your `$HOME.cshrc` file. The following line needs to be added, `set path = ($path ~wwv/bin)` , where this tells your session to look for executables in the author's `\bin` directory. This command will become active the next time you log in. Otherwise, you can immediately re-instate the `.cshrc` shell resource file by typing, `source .cshrc`, from the directory where the file is located.

In order to make the following scripts executable, you must ensure that the first line of the file points correctly to PERL5. Consult your network administrator for this information. Also, you will need to verify that the file has executable permission. If you are the only running the script, then go to the directory where the script is located, and type `chmod 700 scriptname`, where scriptname is the name that you save the file as. Please consult the `man` pages for `chmod` in order to allow read, write, and execute permissions for group or world. These scripts will run on any standard UNIX/LINUX operating system. In order to make them run properly in a Windows or

Macintosh environment, the modifications required is the removal of file locking, and directory delimitation. On Windows, directories are separated by a backslash, "\", and on Macintosh by a colon, ":".

Also, both scripts include a help file that can be viewed by typing, `scriptname -h`.

The following is `.titanscfg`, the configuration file used by both commands `titans` and `killtitans`.

```
# This configuration file is used by
# scripts written by Ward W. Vuillemot
# You should only need to configure this file
# to effect changes in the scripts -- in short,
# leave the scripts alone unless you know what
# you are doing.  You have been warned.  :)
#
# contact Ward at wwv@u.washington.edu with
# questions, comments, or bugs

# the Titans
# list of all machines available on DEC cluster
# other non-DEC machines can be added if a
# mixed MPI environment is permissible
@hosts = (
  'atlas.aa.washington.edu',
  'crius.aa.washington.edu',
  'cronus.aa.washington.edu',
  'epimetheus.aa.washington.edu',
  'gaea.aa.washington.edu',
  'hyperion.aa.washington.edu',
  'iapetus.aa.washington.edu',
  'metis.aa.washington.edu',
```

```
  'mnemosyne.aa.washington.edu',
  'phoebe.aa.washington.edu',
  'oceanus.aa.washington.edu',
  'prometheus.aa.washington.edu',
  'rhea.aa.washington.edu',
  'tethys.aa.washington.edu',
  'thea.aa.washington.edu',
  'themis.aa.washington.edu',
);

# Titans to exclude from the usual work load
# add/remove machines that are to be used for
# things other than CPU farming
@excluded_hosts = ( 'cronus.aa.washington.edu', );

# the users presently accessing the Titans
# this needs to be uptodate, otherwise the
# calculation of percent load will be incorrect
@users=(
  'udrea',
  'brian',
  'wwv',
  'shumlak',
  'ralph',
  'ewig',
  'wjones',
  'schelle',
  'jloveric',
  'aberle',
);
```

```
# used for file locking
# DO NOT CHANGE
$EXCLUSIVE_LOCK = 2;
$UNLOCK = 8;

1;
```

There are three (3) arrays that can be configured. `@hosts` lists the name of the available hosts. `excluded_hosts` lists what hosts of `@hosts` is unavailable for inclusion with `w3.pg`. This enables certain hosts to be used for individual use for short periods of time, then returned to the pool all without modifying the scripts. Finally, `@users` lists those people accessing the cluster. If a user is not listed in this array but is accessing the cluster, the CPU load reported by the other scripts will be unable to report their usage.

Titans: Generating w3.pg

In order to minimize the impact of initiating a run on a set of machines presently engaged in another run, it is necessary to be able to determine the load of each CPU. With a limited number of people accessing a cluster, it possible to organically keep track of all runs. Furthermore, with a large number of CPUs connected to a cluster, it can be time-consuming to log into each machine and run `TOP` to determine the top fifteen (15) CPU intensive tasks.

The following set of scripts will automate the above process, allowing the researcher to quickly determine what CPUs have low CPU loads, effortlessly creating a `w3.pg` from this information. Note that due to formatting constraints, " ... ' are used to denote line continuation.

The following is `titans`, a PERL5 script that automatically generates the `w3.pg` for use with WARP3 and parallel jobs.

```
#!/sw4/bin/alpha-osf/perl
```

```
use Getopt::Std;
require "/mhd1/usrs/wwv/bin/.titanscfg";

# temporary file that is used to store results
$file = 'checktop.dat';
$output_file = 'w3.pg';

# when the script was invoked
@now = localtime(time);
$now[4];
$now[5] = 1900;
$now[1] = "0" . $now[1] if ($now[1] < 10);

# global variables that can be changed via normal UNIX inputs
# check if there was any inputted variables
@output_hosts;
$opt_h = 0;
$opt_i = 0;
$opt_n = $#hosts;
$opt_p = 50;

$opt_v =  0;

$cpu_avail = 0;

getopts('hin:o:p:vu:');

if ($opt_u) \{
  @users = ();
```

```
  my @temps = split /\,/,$opt_u;
  foreach $temp (@temps) {
    push(@users,$temp);
 }
}

&display_help if $opt_h;

if (($opt_p > 100) or ($opt_p < 0)){
  print "-p flag too high, range of [0-100]\n" ;
  exit;
}

print "There are not enough processors (-n flag) ...
than there the $#hosts hosts.\n" and exit if ($opt_n > $#hosts);

@users = sort @users;

foreach $_ (@users) {
 $user .= $_ . '|' unless $_ =~ @users[$#users];
 $user .= $_ if $_ =~ @users[$#users];
}

print qq~
Waking the Titans. . .you ARE brave!
This will take a few moments. . .
Only the first $opt_n Titans who are less
than $opt_p% busy will be mentioned.
~;
```

```
foreach $host (sort @hosts){
  my $not_this_host = 0;

  # do not look for work for those Titans that are excluded
  # from the reindeer games
  foreach $excluded_host (@excluded_hosts){
    $not_this_host = 1 if ($excluded_host =~ /$host/);
 }
  next if $not_this_host;

  # okay, we have a Titan willing to play ball. . .
  # use this variable to keep track of how many are
  # already on our team stop when the team is equal
  # to $opt_n my $total =
  0;

  # flock the file, then run a system command that pipes
  # the results tothe flocked file.  Then read the file for
  # later processing.  Erase the file and also also un’flock
  # the file for the next iteration.

  flock($file,$exclusive_lock);
  system("rsh $host top | egrep \"$user\" > $file");
  open(FILE,"$file") if (-e $file);
  @_ = <FILE>;
  close(FILE);
  unlink($file);
  flock($file,$unlock);
  @sorted_processes = @_;
```

```
  # take apart the lines to get the percent CPU used and
  # add this to the total
  foreach $_ (@_){
    $line = shift(@sorted_processes);
    @cols = split (" "),$line;
    foreach $col (@cols){
      if ($col =~ /\%$/){
        $col =~ s/\%//;
        $total = $total} $col;
     }
   }
 }

  # if the % CPU used is less than some value then output,
  # otherwise the CPU is too busy to be of any worth to anyone.
  if ($total <= $opt_p) {
    print qq~
......$host
......Total CPU Use (%):  $total
    ~;
    $cpu_avail;
    print @_ if $opt_u;
    push(@output_hosts,$host);
    last if $cpu_avail >= $opt_n;
 }
}


if ($opt_o){
  if ($cpu_avail < $opt_n) {
```

```
    print "\nSorry, there are only $cpu_avail Titans available, ...
           not the $opt_n you requested.  Try back later.\n";
    exit;
 }
  my $i = 0;
  print STDOUT "\n" . 'Enter location of w3 program ...
               (default: /mhd1/usrs/wwv/warp3/w3): ';
  my $w3_location = "\/mhd1\/usrs\/wwv\/warp3\/w3\n";
  my $input = <STDIN>;
  $w3_location = $input if (length($input) gt 1);
  if ((-e $opt_o) && !$opt_i) {
    print STDOUT "$opt_o exists!  Do you want to overwrite [y/n]? ";
    my $continue = <STDIN>;
 }
  print 'Aborting output.' and exit if ($continue =~ /y/i);

  flock($opt_o,$exclusive_lock);
  unlink $opt_o if -e $opt_o;
  open(W3OUT,"> $opt_o") || die "Unable to open $opt_o";
  format W3OUT =
@<<<<<<<<<<<<<<<<<<<<<<<<<<<     @<<<<...
     @<<<<<<<<<<<<<<<<<<<<<<<<<<<<
$host_name,$i,$w3_location
.

  print  W3OUT qq~#
# This file, $opt_o, was generated automatically with
# the checktitans script on $now[4]/$now[3]/$now[5]
# at $now[2]:$now[1].
#
```

```
# If you have any problems, please contact its author
# and curator, Ward W. Vuillemot (wwv\@aa.washington.edu).
# Thank You.
#
~;
  foreach $host_name (@output_hosts) {
    write W3OUT;
    \_$i unless $i >= 1;
 }
  close(W3OUT);
  flock($opt_o,$unlock);
}

exit;

sub display_help {

print qq~
This program will check all those CPUs on the CFD Lab
Alpha farm.  There exists some adjustable variables
that can be added to the command line to help streamline
the output.

-h          this help screen

-i          interactive
              default: true
              Only useful if -o is invoked.  It will, by default,
              ask if you want to overwrite w3.pg if it already exists.
```

```
-n [value]  number of processors
              default: $#hosts
              range: between 0 and $#hosts
              This flag must accompany the -o flag.
              At present this program will assign the first
              available CPU as the master (ie, a value of 0
              in the w3.pg file) and all other subsequent
              CPUs as slaves (ie, a value of 1 in the w3.pg file).

-o [file]   w3.pg output file.
              This will generate a file for use with w3.
              Only those CPUs whose % CPU used is less than or
              equal to the value provided with the -p flag.
              This will prompt you for the directory location
              of your w3 program (the third column of the w3.pg
              file).  No output will be generated if the number
              of available CPUs is less than the number of
              processors (-n flag) needed.


-p [value]  percent inactive
              default: $opt_p
              range: between 0 and 100
              This can be used to look for all CPUs whose % CPU
              used is less than or equal to the value provided
              by the -p flag.  Otherwise the default value is used.

-u [name]   usernames
              default: all registered users of the Alpha farm
              This is useful if you want to see what processes
```

```
                are being run by a particular user.  The name must
                be their UNIX username.  It is a comma-delimited
                list.  An example would be:
                  -u wwv,bogdan

-v          verbose mode
                default: false
                 This flag will show all thos processes on each CPU
                 being run by the users.  The users can be modified
                 with the -u flag.
~;
exit;
}
```

Let us provide one example to help illustrate the usefulness of the script. If we wanted to run on four(4) CPUs that had no more than 20% CPU load, then the command is `titans -n 4 -p 20 -o w3.pg`, where `n` is the number of processors, `p` is the percentage of CPU load, and `o` is the name of the output file. The script will then log onto each processor in the order it is listed in the `@hosts` array of the configuration file listed above. Once it finds four (4) processors that match the CPU load, the script will prompt you to enter the location of the executable, `w3`. If `w3.pg` already exists, it will also ask you if you wish to overwrite the existing file; answer accordingly.

Killtitans: Killing Daughter Tasks of Run

On the occasion when WARP3 encounters a segmentation fault error during a parallel run, the daughter sessions may still remain active. A simple check of this is accomplished in UNIX by logging into the mother[2] task, and typing `ps -u username` where "username" is the researcher's. This will display all tasks presently active on that ma-

[2]Mother denotes the CPU designated with 0 in the second column of the w3.pg file, and daughters denote all other CPUs marked with 1.

chine that were activated by said user. It is then possible to kill the desired task(s) by typing, `kill -KILL taskID`

The script is sufficiently robust enough not to kill any processes on the CPU where the command is issued. It should be noted that if the user's main session in on processor A, then this script should be issued from processor A even if the runs were executed on processors A, B, and C. Issuing this script from any of the other two processors will kill **all processes** on A, effectively logging the user out prematurely. In short, this script kills with extreme prejudice.

The following is `titans`, a PERL5 script that automatically generates the `w3.pg` for use with WARP3 and parallel jobs. It includes a help file that can be viewed by typing, `titans -h`.

```
#!/sw4/bin/alpha-osf/perl

use Getopt::Std;
require "/mhd1/usrs/wwv/bin/.titanscfg" or ...
        "The config file is missing!" and die;

# check if there was any inputted variables
getopts('hs:t:');

&display_help if ($opt_h);

# figure out what titans to kill
if (! defined $opt_t){
  $opt_t = 'mpi.hosts';
  print "No MPI host file specified.  Will try using $opt_t.\n";
  unless (-e $opt_t) {
    print "Darn! $opt_t does not exist!\n";
    $opt_t = 'mpi.hosts';
```

```
 }
}

if (-e $opt_t) {
  print "Found $opt_t!\n";
  flock($opt_t,$EXCLUSIVE_LOCK);
  open(MPIHOSTS,"$opt_t") ;
  print "Opened $opt_t\n";
  @mpihosts=<MPIHOSTS>;
  close(MPIHOSTS);
  flock($opt_t,$UNLOCK);
} else {
  # if no flag then try to kill them all
  @mpihosts = @hosts;
  # check whether the bonehead is really a bonehead or not
  print "Warning! Since you did not specify a host file\n";
  print "nor could I found one I've decided to kill 'em all.  \n";
  print "Do you want this? yes/[no] ";
  $continue = <STDIN>;
  unless ($continue =~ /y/i){
    exit;
 }
}

if (! defined $opt_s) {
  $opt_s = $ENV{'HOST'};}

# this goes through and checks stuff interactively
# (the safest way to do it)
my $count = 0;
```

```
my $total_hosts = 0;
my @host_temp;
foreach $host (sort @hosts){
  if (! defined $opt_s){
    $count = $#hosts;
    @host_temp = @hosts;
} else {
    if ($host =~ /^$opt_s/) {
      $host_temp[$count] = $host;
      $count;}}}

until ($count == -1){
  if ($count > 1) {
    print "What one of these servers are you running
    script from?\n";

    $count = 0;
    foreach $_ (@host_temp)
      { $count;
      print "($count) $_\n" ;
   }
    print STDOUT "Enter a number (1-$count): ";
    $what_titan = <STDIN>;
    chop($what_titan);
    unless (($what_titan < 1) || ($what_titan > $count)) {
      $count = -1;
   }
 } else {
    $what_titan = $count -1;
    $count = -1;
```

```
 }
  $opt_s = $host_temp[$what_titan-1];
}

print "Gotcha.  You are on the Titan named $opt_s.\n";

print qq~

Gonna qwil da Titans!  Qwil da Titans!
He! He! That swilly wabbit is gonna get it now!
~;

foreach $host (sort @mpihosts){
# don't kill ourself
  next if ($host =~ /^$opt_s/);
  chop($host) if ($host =~ /\n$/);
  print "\nTrying to kill $host\n";
# send a remote shell command to each host and kill all
# processes on it
  system ("rsh $host /usr/bin/kill -KILL -1") ;

# tell them that a Titan is down for the count
  print "......$host is dead.\n";
}

print "\n";

exit;

sub display_help {
```

```
print qq~
This script executes a remote shell command on all Titans
except the one you are on to kill them.  Cool, huh?

-h          this help screen

-s          system you are executing the script from
              The name of the Titan you are executing this
              script from.  Only part of the whole name is
              necessary, though.  You can foreshorten the
              name to a few letters.

              This is no longer necessary as the script
              will try to determine the host on its own;
              but, if you want to override, then please feel
              free to do so.

              Example:  If you are running the script from
                        cronus.aa.washington.edu then the
                        following command is sufficient
                          % checktitans -s cron
                        If you type
                          % killtitans -s cr
                        then it will ask if you are on
                          crius.aa.washington.edu or
                          cronus.aa.washington.edu

                        If the letters are not unique nor corre-
                        spond to 2 or more machines then
                        it will ask you to choose from all of them.
```

```
-t          mpi host file used to determine what Titans
             to kill You can use the mpi host file outputted by
             MPI to kill those titans after a job -- they some-
             times they sometimes do not exit nicely.
             The file can be referenced absolutely or relatively.

              Example: If you are in /dir1/dir2/
                       and the mpi.host file is at /dir1/dir3/
                       then the following two commands are
                       identical
                         % killtitans -s cron -t ../mpi.host
                         % killtitans -s cron -t /dir1/dir3/mpi.host
~;
exit;
}
```

To operate, go to the directory where the file denoted by the `hosts_file` in the `Control` module of the input deck file is located. Assuming that system filepath is set correctly, typing `killtitans -t host_file` will invoke the script. The script will look for the `host_file` file, read in the list of daughter tasks, and then kill them one by one reporting its progress CPU by CPU. If the user does not specify a host file, the script will look for a file named `mpi.hosts`. If this file is not found, the script will determine from which host the user invoked the script, and then ask if the user wants all other tasks on all other hosts to be terminated. The user must then type `y` and hit return to continue. Otherwise, hitting return by itself will quit the script.

E.5.2 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 " "
```

E.5.3 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 ($\rho, v_x, v_y, v_z, B_x, B_y, B_z, p$) along with the constituent temperatures and ion fraction (f_i, T_i, T_n, T_e) at the cell vertices are written. The second type of output file is a restart file where the: eight conserved variables ($\rho, \rho v_x, \rho v_y, \rho v_z, B_x, B_y, B_z, e$); the components of the resistivity (η_x, η_y, η_z); kinematic viscosity ν; the components of the thermal diffusion ($\kappa_x, \kappa_y, \kappa_z$); and, the constituent temperatures and ion fraction (f_i, T_i, T_n, T_e) 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 is possible to write FORTRAN or C procedures callable from Tecplot that would load the desired files into the Tecplot without concatenation. Please consult Tecplot documentation for details.

E.6 Expanding WARP3

This section will briefly outline what sections of WARP3 can be easily extended by the researcher, including new applications, boundary conditions, I/O formats, solvers, et cetera.

E.6.1 Code Style and Philosophy

WARP3 code has been diligently maintained to follow a certain set of prescribed standards to ensure easy maintenance and operation for any researcher willing to learn the standards. As mentioned previously, both `warp3.f` and `ptime.f` maintain the integrity of the code flow, and are not meant to be used for intensive calculations or logics. Consequently, an overview of procedures is readily apparent to anyone who looks at these two (2) files.

While FORTRAN90 does include `COMMON BLOCK` variable passing invoked with the `USE ModuleName` command, it was decided that this results in code that is too difficult to understand, let alone maintain or develop. Consequently, all variables declared in modules are passed from `warp3.f` using subroutine argument list passing. Not only does make it easier to develop and maintain the code, coupled with the `INTENT` command, both the researcher and compiler can better anticipate the use of every variable used in said subroutine.

Regarding commenting code, as a minimum each subroutine includes a header comment section to briefly describes what the subroutine scope is. Further comments are encouraged, especially when programming techniques are used that might obfuscate

the physics being duplicated numerically. More to point, whenever possible the code of WARP3 follows the logic implicit in the physics.

Appendix F

ABLATION AND RADIATIVE TRANSFER

In this appendix we will investigate two physical phenomenon of interest to plasma physics and hypersonic researchers. Namely, these are: moving boundary problems, better known as phase-change problems [15]; and, radiative transfer [11]. In particular, we are interested in learning what set of equations and relevant physics are necessary to describe these phenomenon as they pertain to high-temperature flow. In addition, we are interested in positing a set of equations in a manner that will readily lend itself to WARP3[1] [23], an approximate Riemann fully three-dimensional magneto-hydrodynamics (MHD) solver.

WARP3, with the inclusion of the aforementioned physical phenomenon such as moving boundaries (ablation), will be better positioned to handle topics of interest to researchers of plasma physics and hypersonics. A prime example of the utility of these solvers would be to help better predict the ablative effects of terbium-doped borosilicate used in Transient Internal Probe (TIP) [7, 6] being conducted at the University of Washington Department of Aeronautics and Astronautics.

F.1 Introduction

In this report we interested in examining how ablation and radiant heat transfer can be numerically solved using WARP3. We must remain vigilant of the fact that in both cases these physical processes are in many cases better suited to particle methods such as particle-in-cell (PIC) or Monte Carlo. In contrast, WARP3 solves the fluid as a continuum using the conservation laws derived from the Boltzmann equation. It would seem therefore that they are mutually exclusive; however, we will discuss

[1] University of Washington Approximate Riemann Plasma solver in 3 dimensions

how accommodations can be made so that WARP3 can, by and large, include these physical phenomenon. Nonetheless, it is must be stressed that secondary effects such as chemical composition are best left to more suitably adapted numerical solvers.

Ablation is of particular interest to hypersonic study as it has considerable effect upon the flow chemistry around the vehicle. The problem of ablation for aerospace engineers emerged as a consequence of studying vehicle re-entry into an atmosphere. From the standpoint of physics, re-entry vehicles have a singular objective. Quite simply, the vehicle must rid itself of its kinetic energy, expressed as some orbital velocity, by the time of it reaches the planet surface.

A vehicle begins descent into an atmosphere with extremely high velocity. Because the vehicle is well in excess of the sonic velocity, in other words supersonic, the flow around the vehicle does not have enough time to move out of the way; namely, information cannot propagate upfield to ”inform” the yet to be impacted gas what is coming and that it should “get the heck out of the way,” as it were. Consequently, the flow slams into the vehicle generating severe pressure gradients to the front of the vehicle. Furthermore, around the stagnation point[2] the region heats up as translation (kinetic) energy is turned into random (thermal) energy. However, Mother Nature has devised a means of normalizing, at least in part, the disturbance initiated by this high kinetic source. This normalizing force is most often referred to as a shock wave, or in cases of extreme bluff body shape evolves into a detached shock, or bow shock.

As just alluded to, the shape of the object traveling at hypervelocity will generate a shock that is a function of its body shape. More importantly, the shape also determines the strength of the shock. Pointed objects, such as a cone generally create a weak shock at the stagnation point, or tip of the vehicle. Conversely, bluff bodies, such as the United States Space Shuttle nose, create a strong shock at the stagnation point. When the body shape is severe enough, back pressure generated from the flow behind the shock forces the shock to move upfield away from the stagnation point until equilibrium is reached; this distance is termed the shock stand-off distance. In these

[2]Point in flow where velocity adiabatically goes to zero

instances, the shock looks similar to a stretched bow, and therefore is aptly called a "bow shock". We will explain shortly why this later shape is preferred for hypersonic vehicles, and also why it may induce ablation of the vehicle surface.

As the flow passes through the shock, it is decelerated and turned away from the object. A large portion of the flow near the surface of the object will collide with it, generating heat as it imparts some its kinetic energy into thermal energy, *i.e.* random energy. A shock generated at these speeds will extend some distance from the object. Gas passing through the shock is heated, where only the flow, and not the vehicle, sees this fraction of the total heat load. Furthermore, the fraction of the heat load absorbed by the shock is directly proportional to the strength of the shock. For this reason, bluff body is preferred over more contoured shapes. That is to say, bluff body enhances the amount of heat load imparted into the flow, thereby diminishing the total heat load subjected to the object.

There are three major methods of dissipating the remaining fraction of heat load on the vehicle. In cases where the re-entry is steep-angle or ballistic, deceleration is accomplished in a very short period of time and, while the heating rate is very high, the total heat load integrated over the total time for deceleration may mean that the heat load is not exceedingly large. In these instances, a blunt-nose composed of thick-skin is utilized as a heat-sink. However, the low heat capacity of metals means that an extremely thick nose must be used. The weight restrictions can be considerable.

The second method is to use a shallow re-entry angle so that the majority of deceleration occurs in the upper atmosphere that is less dense. Therefore, the volumetric heat rate will be small. In this instance, if the heat flow into the vehicle can be equilibrated to the heat radiated by the body then a thin-shelled, non-heat-sink, radiation-cooled shield can be employed.

Finally, the third method available is to use a shield composed of insulating material. When the shield reaches melting or sublimation temperature, the material is ejected into the flow, thereby reducing the total amount of heat in the vehicle. This method of ablation has the added effect of enveloping the vehicle with this insulation material so that the heat flow is retarded. It is this third method that is most often

utilized in the development of re-entry vehicles.

However, ablation is not merely limited to hypersonic vehicles. Indeed, the Plasma Dynamics Group of the Aeronautics and Aerospace Department at the University of Washington is conducting research to develop a probe capable of returning the magnetic field profile within a confined plasma. An unclad crystal probe is shot through the plasma chamber at $2\frac{km}{s}$ while a neutral beam impinges upon the front. A major point of degradation in the quality of results occurs at the onset of ablation to the probe due to the hostile environment. Presently, the group would like to better model its probe in order to optimize its design. Inclusion of ablation along with appropriate boundary conditions including the "belief time[3]" to WARP3 would provide extremely useful for this research.

We will begin with an overview of the phase-change problem in general, mathematical terms. This will include examining two perspectives – one, heat conduction, and the other, enthalpy – to better appreciate the complexities of this set of problems. Next, we will examine the radiant transport equations using Fick's Law. Finally, we will discuss how these two phenomenon can be included in WARP3.

F.2 Phase Change Problems

The question of phase-change, better known as ablation to aerospace engineers, is one that is still considered by many to still be at the forefront of research. At the outset, it would seem in its infancy, where one-dimensional problems, both numeric and analytic, are the norm and not the exception. However, this conclusion misses the true nature of this type of problem – complexity. As a set of equations, it is easy enough to understand by advanced undergraduates. However, as a set of equations applied to problems other than half-space, ergo extremely simplified systems, the equations prove exceedingly contentious for even the best applied mathematician. So much so that these problems are still proving onerous to researchers some 170 years after phase-change problems were initially tackled.

[3]Time required for material to be heated before ablation begins.

Phase-change problems (PCP)[4] were first solved analytically by Lamé and Clapeyron in 1831, and Stefan in 1891. It is Stefan who popularized this type of problem studying ice formation. Accordingly, PCP are often also referred to Stefan problems.

F.2.1 Mathematical Formulation

The development of the system of equations necessary to sufficiently describe PCP is obtained by considering the heat flux at the interface between solid to liquid, or liquid to gas, or even solid to gas. In cases with gas, we must satisfy the implicit assumption tied to Fourier's Law that governs heat conduction. Namely, we must be able to consider the system is a continuum. More specifically, in the case of a gas, the number density must be sufficiently high that conservation laws suitably describe kinetic effects.

From Fourier's Law for heat flux, the heat conduction equation for both the liquid and solid is,

$$\frac{\partial^2 T_s(x,t)}{\partial x^2} = \frac{1}{\alpha_s}\frac{\partial T_s(x,t)}{\partial t}, \quad -\infty < x < s(t), t > 0 \tag{F.1}$$

$$\frac{\partial^2 T_l(x,t)}{\partial x^2} = \frac{1}{\alpha_l}\frac{\partial T_l(x,t)}{\partial t}, \quad s(t) < x < \infty, t > 0 \tag{F.2}$$

where we assume that each is defined as a semi-infinite space going to infinity in opposite directions from the interface, $s(t)$. However, Eqn (F.1) and Eqn (F.2) have three (3) unknowns, or namely,

$$T_s(x,t)$$
$$T_l(x,t)$$
$$s(t)$$

where T_l and T_s are liquid and solid temperature as a function of position and time, respectively, and $s(t)$ is the interface deep within the interior and is a function of time.

Next, we need to develop a third equation that closes the system. We start by considering the heat flux at the interface, $s(t)$. Similar to the development of the heat

[4]The reader should not confuse this acronym with the more often used chemical phencyclidine. However, according to the literature both can be a mind-altering experience

conduction equation from Fourier's law, we sum the heat fluxes, knowing that they are equal to the rate of heat liberated during, say, melting. In other words,

$$\begin{pmatrix} \text{heat flux in} \\ \text{x direction} \\ \text{at solid} \end{pmatrix} - \begin{pmatrix} \text{heat flux in} \\ \text{x direction} \\ \text{at liquid} \end{pmatrix} = \begin{pmatrix} \text{latent heat} \\ \text{of fusion} \\ \text{per unit} \\ \text{interface area} \end{pmatrix}$$

The mathematical representation for this is written as,

$$\kappa_s \frac{\partial T_s}{\partial x} - \kappa_l \frac{\partial T_l}{\partial x} = \rho L \frac{\partial s(t)}{\partial t}, x = s(t) \tag{F.3}$$

It is the time rate of change of position of the interface, $\frac{\partial s(t)}{\partial t}$, that both frames our real interest in PCP, while also proving to be the most contentious part to solving them. We should immediately realize that this term is simply the velocity of the interface, whereby liquid is solidified, or conversely, a solid is liquefied. We rewrite Eqn (F.3) to reflect this realization, or

$$\kappa_s \frac{\partial T_s}{\partial x} - \kappa_l \frac{\partial T_l}{\partial x} = \rho L u_x(t), \frac{ds(t)}{dt} = u_x(t) \tag{F.4}$$

Therefore, to summarize, our final set of equations is presented as a complete system.

$$\frac{\partial^2 T_s(x,t)}{\partial x^2} = \frac{1}{\alpha_s} \frac{\partial T_s(x,t)}{\partial t} \tag{F.5}$$

$$\frac{\partial^2 T_l(x,t)}{\partial x^2} = \frac{1}{\alpha_l} \frac{\partial T_l(x,t)}{\partial t} \tag{F.6}$$

$$\kappa_s \frac{\partial T_s}{\partial x} - \kappa_l \frac{\partial T_l}{\partial x} = \rho L u_x(t) \tag{F.7}$$

Effect of Density Differences

In the above set of equations, Eqn (F.5, F.6, and F.7) assume that there is no difference in density between liquid and solid. When, however, density is not constant at the interface it results in liquid motion across the interface. When we take into account diffusive and convective terms, Eqn (F.4) becomes,

$$\kappa_s \frac{\partial T_s}{\partial x} - \kappa_l \frac{\partial T_l}{\partial x} = (\rho_l h_l - \rho_s h_s) u_x(t) - \rho_l h_l u_l \tag{F.8}$$

where h_s and h_l are the enthalpies per unit mass for solid and liquid, respectively, and u_l is the velocity of the liquid. However, application of mass conservation yields $u_l = (1-\rho_s/\rho_l)u_x$. If we recognize that the latent heat, L, is the difference of enthalpies, h, then we can reduce Eqn (F.8) such that it is only in terms of our original three (3) unknowns, or more precisely

$$\kappa_s \frac{\partial T_s}{\partial x} - \kappa_l \frac{\partial T_l}{\partial x} = \rho L u_x \tag{F.9}$$

Effect of Convection

In cases when we can neglect diffusion from liquid to solid, and where heat transfer is dominated by convection[5], the heat flux at the interface is written as,

$$\kappa_s \frac{\partial T_s}{\partial x} h(T_\infty - T_m) = \rho L u_x \tag{F.10}$$

where T_∞ and T_m are freestream and melting temperatures, respectively.

F.2.2 Enthalpy Formulation

It would appear that the simplest method numerically for a moving interface is to a moving, adaptive mesh. However, this can cause a variety of problems for finite difference and finite volume solvers. There is also the issue of *tracking* the interface. This added level of complexity steers the solver away from its principal purpose – in pursuit of scientific inquiry.

An alternative method is the *enthalpy formulation* that can be solved on fixed grids, such as is the case for WARP3. The phase-change is a singularity at the interface in the enthalpy-temperature relationship, whereby the interface is *captured* naturally by the solver [10]. Both S.K. Wong *et al* [24] and M. Storti [21] have advanced the state of enthalpy formulation techniques using a novel technique of fictitious material that has the same properties as the ablating material with the caveat that its specific heat is vanishingly small. This has the effect of instanteously transporting the heat applied

[5]Corollary to this, the "enthalpy method" utilized by many to solve PCP make the interface, $s(t)$, static. In so doing, they need to compensate by including a convective term to the energy term.

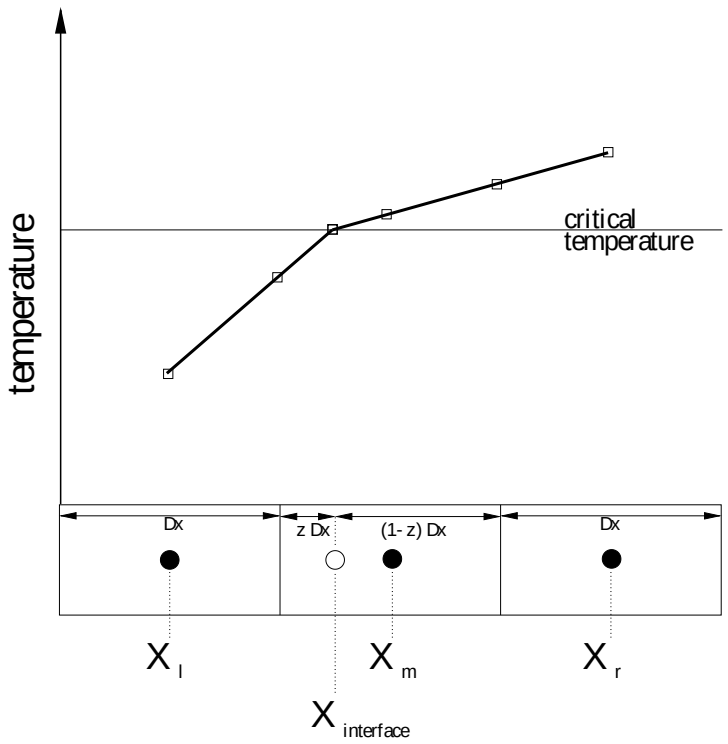


Figure F.1: Three cells, X_m, X_l, and X_r, with temperature profile shown to top of cells. Horizontal line denotes critical temperature, T_c. Cell centers are denoted by black dot, and phase interface by hollow dot.

at the boundary onto the ablation surface. Unfortunately, this technique fails to add any other relevant physics. Furthermore, application of the heat load at the boundary, and not the ablation surface, has the effect of spreading the heat load over the entire ablation surface in a global manner, regardless of localized effects at the boundary. S.K. Wong *et al* have proposed using a highly anisotropic thermal conductivity that is dominate in the direction toward the ablation surface, but vanishingly small in directions perpendicular to the incident heat flow. However, as far is this author is aware, they have not published any results for this idea.

Following Tacke [22], we recall that a solid holds sensible heat per unit mass, or

$$c_p(T - T_c)$$

where c_p is the specific heat, T is the temperature of the substance, and T_c is the critical temperature where phase-change occurs. In addition to sensible heat, in the case of a liquid, there is also latent heat per unit mass, L. For a pure substance, the sum of sensible and latent heat is the enthalpy, h. The temperature and enthalpy are related as follows,

$$h = \begin{cases} c_p(T - T_c) & T \leq T_c \\ c_p(T - T_c) + L & T > T_c \end{cases} \tag{F.11}$$

Using this formulation, we are able to determine the temperature of the substance with the following relations,

$$T = \begin{cases} T_c + h/c_p & h < 0 \\ T_c & 0 \geq h \leq L \\ T_c + (h - L)/c_p & h > L \end{cases} \tag{F.12}$$

Finally, we need to relate temperature to enthalpy where both density, ρ, and thermal conductivity, κ, are independent of temperature. We can write this as,

$$\rho \frac{\partial h}{\partial t} = \kappa \frac{\partial^2 T}{\partial x^2} \tag{F.13}$$

where we apply this to the entire domain. Note that we do not need to track the interface position, $s(t)$, as was the case with the previous formulation. Instead, we can obtain the position of the ablation interface by examining the temperature profile where temperature is the critical temperature , T_c.

Numerically, it is quite trivial to solve Eqn (F.13). We write the forward time, centered space (FTCS) formulation as,

$$h_i^{n+1} = h_i^n + \frac{\kappa}{\rho} \frac{\Delta t}{\Delta x^2} \left(T_{i+1}^n - 2T_i^n + T_{i-1}^n \right) \tag{F.14}$$

where subscripts denote spatial index and superscripts are temporal index. Initially, the reader might be tempted to argue that either the temporal or spatial differences,

or both, would need to be adjusted near the interface. However, the enthalpy formulation is meant to abdicate one from having to explicitly "track" the interface position. However, as we will discuss shortly, in order to reduce temperature oscillations it becomes necessary to track the relative position of the interface within a control element. Regardless, though, in principle the interface position is captured by the physics.

A solver begins with some initial condition and boundary conditions. Next, the solver is advanced some value of Δt where a new value for enthalpy, h_i^{n+1} is solved from the old values for enthalpy and temperature, h_i^n and T_i^n. Once computed, the new temperatures are computed using the relations provided in Eqn (F.12). We can further simply Eqn (F.14) by rewriting the temperatures in the form of the heat flux given by Fourier's Law, or

$$h_i^{n+1} = h_i^n + \frac{1}{\rho}\frac{\Delta t}{\Delta x}\left(q_r^n - q_l^n\right) \tag{F.15}$$

where,

$$q_l = \kappa\frac{T_m - T_l}{\Delta x} \tag{F.16}$$

$$q_r = \kappa\frac{T_r - T_m}{\Delta x} \tag{F.17}$$

is the heat flux to the left and right of the cell center, respectively, and where the heat flux is moving from the right to the left (higher temperature to lower temperature). Note that there is no difference in formulation for cases when the solid and liquid thermal conductivities do not coincide, $\kappa_s \neq \kappa_l$.

Eliminating Temperature Oscillations

However, a major problem with the enthalpy formulation per se is oscillation of temperature within the domain. The reason for this is explained by Tacke [22] who provides a near oscillation-free scheme. According to Tacke, the oscillations are of a period corresponding to the time the phase front must pass through a control element. In short, the above scheme assumes that values at points (nodes) is an average of the finite volume, which is reasonable for areas not undergoing phase-change. However, this is not the case of a finite volume where the ablation interface is presently moving through.

The concept of *fraction solid* is used to better determine where the phase front is located within a control volume, as shown in Figure F.1. When an element is at the critical temperature, the phase front within the element is linearly interpolated as,

$$x_{\text{interface}} = x_{\text{solid}} + f\Delta x \tag{F.18}$$

where x_{solid} is the position of the element face about some cell center in the direction where the adjacent cell has yet to ablate, Δx is the width of the cell, and f is defined as the fraction solid. A reasonable value for this is,

$$f = 1 - \frac{h}{L} \tag{F.19}$$

From this, Tacke has developed an improved discretization scheme whereby the contributions of the ablating cell are better distributed in the energy balance formulation. Namely, the element containing the phase interface is,

$$\begin{pmatrix} \text{total} \\ \text{heat} \end{pmatrix} = \begin{pmatrix} \text{latent} \\ \text{heat,} L \end{pmatrix} + \begin{pmatrix} \text{sensible} \\ \text{heat of} \\ \text{liquid} \end{pmatrix} - \begin{pmatrix} \text{sensible} \\ \text{heat of} \\ \text{solid} \end{pmatrix}$$

In terms of formal mathematics, this is written more succinctly as,

$$h\Delta x = L(1-\zeta)\Delta x + c_p(T_r - T_m)(1-\zeta)\frac{\Delta x}{2} - c_p(T_m - T_l)\zeta\frac{\Delta x}{2} \tag{F.20}$$

where ζ is defined in Figure F.1.

From Eqn (F.20) Tacke provides a third-order equation for ζ in terms of T_l and T_r, or

$$\zeta^3(-4-2L-2R) + \zeta^2(4+3L+3R+4f) + \zeta(3-4f) - R - 3f = 0 \tag{F.21}$$

where,

$$L = \frac{c_p}{L}(T_m - T_l) \tag{F.22}$$

$$R = \frac{c_p}{L}(T_r - T_m) \tag{F.23}$$

and where f is given by Eqn (F.19).

Now that a better estimation of the interface position can be computed, the heat fluxes can be better computed within the ablating cell as,

$$q_l = \kappa \frac{T_m - T_l}{x_{\text{interface}} - x_l} \tag{F.24}$$

$$q_r = \kappa \frac{T_r - T_m}{x_r - x_{\text{interface}}} \tag{F.25}$$

where

$$x_{\text{interface}} = x_{\text{solid}} + \zeta \Delta x \tag{F.26}$$

Finally, the new algorithm developed by Tacke and based upon the enthalpy formulation is as follows. The temperature, T, enthalpy, h and interface position, $x_{\text{interface}}$ are known at time, t. The enthalpies for the domain are updated using Eqn (F.15). Next, heat fluxes for the cell being ablated along with cells to the left and right are computed using Eqn (F.24). For all other cells, the heat fluxes are computed using Eqn (F.16). Next, the temperature is computed as before using Eqn (F.12) with the exception of the ablating cell. The solid fraction, ζ, is computed using Eqn (F.21), and the new interface is computed using Eqn (F.26). In the case of the ablating cell, its temperature, T_m, is computed using similar triangles to obtain the following linear relation,

$$T_m = \frac{\zeta}{3/2 - \zeta} \Delta x \, (T_c - T_r) + T_c \tag{F.27}$$

Special care must be taken in cases when the ablation interface passes from one cell to another during some time-step, n. Again, Tacke provides the following solution. In cases when $\zeta > 1$ then the enthalpy in the i cell is adjusted by

$$\delta h_i = \frac{\Delta t}{\rho \Delta x} (1 - r) \, (q_r^* - q_r) \tag{F.28}$$

where r is

$$r = \frac{1 - \zeta^n}{\zeta^{n+1} - \zeta^n}$$

and the adjusted heat flux for the right face is better approximated by,

$$q_r^* = \kappa \frac{T_m - T_l}{x_{\text{interface}} - x_l}$$

Also, cell $i+1$ is correspondingly corrected by subtracting δh.

Finally, the stability for enthalpy formulation of this form is given by,

$$\Delta t \leq \frac{1}{2}\Delta x^2 \frac{\rho c_p}{\kappa}$$

In cases where the improved discretization is not used, then the $1/2$ should be replaced by $1/3$.

F.3 Radiant Heat Transfer

The question of radiant heat transfer in relation to hypersonics is extremely arguable. Depending on the exact physics, or to be precise, depending on the temperature of the flow, the answer to whether we should consider radiant heat transfer varies. Initial research into manned re-entry vehicles during the NASA Mercury and Apollo programs precipitated research that ultimately determined that conductive heat transfer was significantly larger than radiant heat transfer for temperature flow under 10,000 Kelvin. In other words, when the flow speed is Mach 15 or below, contribution from radiant heat can be neglected [2, 12].

Further complications include whether the flow is viscous and non-adiabatic. However, it is well known that for a Reynold's number[6] in excess of 10,000 then the flow can be assumed inviscid. Furthermore, both viscosity and nonadiabatic conditions occur in the thin boundary-layer about the vehicle at hypervelocity [12]. From this, we may safely assume that a hypervelocity flow is overall both inviscid and adiabatic.

With this simplification, the energy equation for Euler flow is simply,

$$\frac{\partial e}{\partial t} + \vec{\nabla}\left((e+p)\,\vec{u}\right) = -\vec{\nabla} q_r \tag{F.29}$$

where e is energy per unit volume, p is pressure, u is velocity, and q_r is the radiant heat flux. The reader is encouraged to reread Chapter 2 of to better understand how this fully integrates into the magneto-hydrodynamic (MHD) equations. However, it should be obvious upon inspection that the RHS of Eqn (F.29) needs to be included in the sum of the parabolic fluxes (resistivity, viscosity, and thermal conduction).

[6] Ratio of momentum to viscosity

F.3.1 General Case for Radiant Flux

As straightforward as the above equation is, it does not properly define the radiant flux, q_{rad}. We turn to Anderson [2] to develop a better sense of this term. We begin with a geometric argument with line radiation incident on a differential volume. *A priori* we can state that the change in intensity is simply the difference between the energy emitted to the energy absorbed, or

$$dI_\nu = J_\nu ds - k_\nu I_\nu ds \tag{F.30}$$

where I_ν is the intensity for some said frequency, J_ν and k_ν are the emission and absorption coefficient, respectively, and ds is the change in distance along a line drawn through the volume and parallel to the line radiation. Dividing through by ds, as it were, we have the *radiative transfer equation* along some specified direction.

$$\frac{dI_\nu}{ds} = J_\nu - k_\nu I_\nu \tag{F.31}$$

We obtain the gradient of the radiant heat transfer, $\vec{\nabla} q_{rad}$, by integrating over all space, $(x, y, z) \in [0, 4\pi]$ and frequencies, $\nu \in [0, \infty]$, or

$$\vec{\nabla} q_{rad} = \int_0^\infty \int_{4\pi} J_\nu d\omega d\nu - \int_0^\infty \int_{4\pi} k_\nu I_\nu d\omega d\nu \tag{F.32}$$

However, we can simplify Eqn (F.32) by assuming that the material is isotropic. Therefore, we should expect that it emits radiation in all directions, or

$$\vec{\nabla} q_{rad} = 4\pi J - \int_0^\infty \int_{4\pi} k_\nu I_\nu d\omega d\nu \tag{F.33}$$

F.3.2 Simplification of Radiant Flux

From the perspective of a numericalist, Eqn (F.33) poses a considerable about of problems to any code. In particular, we take note that we must integrate over all space for the second term in the LHS. The number of calculations per iteration is on the order of N^2, where N is the number of cells in the domain of interest.

We can further simplify radiant heat transfer by assuming our gas is blackbody emissive. In this instance, taking note from Ozisik [14] the total black-body emissive

flux is,

$$q_{rad}(T) = n^2 \bar{\sigma} T^4 \tag{F.34}$$

where n is the index of refraction and $\bar{\sigma}$ is the Stefan-Boltzmann constant. From this the gradient is merely,

$$\vec{\nabla} q_{rad}(T) = n^2 \bar{\sigma} \vec{\nabla} \left(T^4 \right) \tag{F.35}$$

where we assume that the index of refraction, n, does not vary spatially. As we will shortly show, Eqn (F.34) will prove extremely useful and congruent with radiation transport theory.

F.3.3 Diffusion Approximation

A difficulty with radiation, as mentioned previously, is that it is of integral form. Ergo, it does not lend itself to incorporation within numerical solvers *easily*. However, if we assume that the radiation is of a single frequency – monochromatic – and that there is not a large dependence on angularity, we find that the radiation transport can be shown in differentiable form. One approach is the P_N method, also known as the Legendre-polynomial expansion method, that looks to expand the angular dependence of radiation into a set of Legendre polynomials. However, the method is extremely cumbersome, and not well-suited for any thing other than pedagogical pursuits.

However, the diffusion approximation looks to exploit the P_1 approximation that involves only the dominant eigenmodes. The approximation is valid for mediums without a strong dependence of angularity, that is to say weak anisotropically. Also, the medium must not be highly transparent, either.

In the case of diffuse radiation, the exact form of the energy continuity equation is merely,

$$\begin{pmatrix} \text{time rate} \\ \text{of change} \\ \text{of energy} \end{pmatrix} + \begin{pmatrix} \text{energy} \\ \text{leakage} \end{pmatrix} + \begin{pmatrix} \text{energy} \\ \text{absorption} \end{pmatrix} = \text{source}$$

We write this more succinctly as,

$$\frac{1}{u}\frac{\partial \phi}{\partial t} + \vec{\nabla} \cdot \vec{J} + K_t \phi = S \tag{F.36}$$

where u is the velocity of the energy carrier, ϕ is the radiation intensity, J is simular to ϕ with the same units of intensity per unit area, K_t is the attenuation coefficient[7] and has units of inverse unit length, and S is the source term that has units of energy per unit volume. Note that the velocity of the energy carrier is *not* the velocity of the flow. Radiation, for cases involving hypersonics, is photons, or $u = c_0/n$ where n is the index of refraction, and c_0 is the speed of photons in a vacuum. However, the carrier may be neutrons as is the case for nuclear reactors which may be more relevant for research such as TIP.

In the case of Eqn (F.36), we need to find a relationship between ϕ and J. Fick's Rule allows us to find an approximate form, written as,

$$\frac{1}{u}\frac{\partial J}{\partial t} + \sigma J = -\frac{1}{3}\vec{\nabla}\phi \tag{F.37}$$

where σ is the transport cross section. We can ignore the first term since u is on the order of 10^5 or above for most energy carriers of interest. So doing, we can write the relationship between J and ϕ as,

$$J = -D\vec{\nabla}\phi \tag{F.38}$$

where

$$D = \frac{1}{3\sigma}$$

We return to Eqn (F.36), replacing all instances of J with its definition provided in Eqn (F.38) to write the diffusion equation as,

$$\frac{1}{u}\frac{\partial \phi}{\partial t} - D\vec{\nabla}^2\phi + K_t\phi = n^2\bar{\sigma}\vec{\nabla}\left(T^4\right) \tag{F.39}$$

where the source term, S, has been replaced with the black-body emission shown previously, and the diffusion coefficient, D is assumed to be constant scalar throughout

[7] It is the sum of the absorption and scattering coefficients.

the domain. In the case of using black-body emission, this is acceptable within terms of the diffusion equation since both assume monochromatic emission, or a lack of dependence on the frequency of the radiation carrier. What is important to note is the similarity between Eqn (F.39) and the thermal conductivity equation,

$$\frac{\partial T}{\partial t} + \kappa \vec{\nabla}^2 T = S$$

where the energy density, ϕ, acts like temperature, T, and the diffusion coefficient, D, is similar to the thermal conductivity coefficient, κ. Therefore, the mode of information transportation is strikingly similar between diffusive radiation and thermal conductivity.

Finally, we now can see that gradient of the radiant heat flux can be written into two forms, either Eqn (F.35) or Eqn (F.39) as shown below, respectively.

$$\vec{\nabla} q_{rad}(T) = n^2 \bar{\sigma} \vec{\nabla} \left(T^4\right)$$

or,

$$\vec{\nabla} q_{rad}(T) = \frac{1}{u} \frac{\partial \phi}{\partial t} - D \vec{\nabla}^2 \phi + K_t \phi$$

F.4 Summary, a Perspective on WARP3

In this section we will discuss briefly how both ablation and radiant heat transfer presented in the previous sections apply to WARP3. In both instances we have clearly shown that either phenomena reduces numerically to relatively straightforward differentiable set of equations. In the case of ablation, the enthalpy formulation appears to provide a much easier means of integration, especially since it is ideally suited to fixed-mesh solvers.

The author foresees no problems with including an ablation solver, per se. However, such a solver will require that information be included in the grid arrays to track whether a cell is solid, liquid, or ablative. Furthermore, each block should be able to be globally set in a similar manner. For example, a block set to "liquid" will also calculate all the relevant flow physics including hyperbolic and parabolic fluxes. If a

block is set to “solid” then only the relevant parabolic fluxes will be calculated. When a block is “ablative” then the solver will need to determine whether each of its cells are solid, liquid or ablative, where the relevant physics are duly linked to each state. Furthermore, in the case of a cell being “ablative” and Tacke’s discretization presented in Section F.2 is implemented, another array can be used to track the phase-front within the cell itself.

In the case of radiation transport, the ability to write the diffusion equation in differentiable form (versus integral form) is an enormous advantage. Furthermore, the gradient of the radiant heat flux can written for cases of “one-wave” phenomenon where we assume there is only frequency (monochromatic), or where specific frequency are not issue or concern. In instances where the radiation is purely black-body emission, the contribution of the radiant flux to the energy equation is extremely straightforward. In instances where the radiation intensity is more complicated, it is still relatively simple to include these effects within the already established framework of WARP3.

Of course, it cannot be overly stressed that the diffusion approximation is only applicable to optically opaque mediums. Further to point, most plasmas are considered to be optically transparent, so that the inclusion of the diffusion approximation will only not always be applicable. Nevertheless, in instances where ablation is occurring, it may be reasonable to relax assumptions about transparency.

The only significant physical phenomena excluded from our discussion is catalytic boundary conditions. The only reasonable instances when these can be included would be when we can assume that effects of ablated material(s) have an effect upon the bulk properties of the fluid flow. In these instances, inclusion of catalytic boundary conditions should be coupled to the enthalpy solver for ablation. Thereby, an array can be used to track the local ejection of ablative material’s mass density into the flow per unit of time, whereby this information is used to create a weighting matrix that can be used to adjust the local thermal conductivity, or diffusion coefficient, or other scalar quantity.

The author believes that addition of these above-mentioned solvers would signif-

icantly enhance the capabilities of WARP3 in the realm of hypersonic research. In addition, these solvers have a wide variety of application to topics pertinent to plasma physics such as TIP. Also, even the simple ability to denote cells and blocks of cells as liquid, solid, or ablative would provide an important tool in developing laboratory equipment. For example, the design of ZAP included the calculation of the wall-coating thickness, which could, given the above additions, be handled by WARP3 in a very straight-forward manner.

VITA

The author was born in Rochester, New York on December 18, 1973. When he was twelve, his family moved to Skaneateles, New York where he graduated from high school in June of 1992. He would venture to Japan as a Rotary International Exchange Student in 1992. He returned one year later after living with five wonderful host-families: Demura; Kochihara; Kitagawa; Fumuro; and, Fuwa. Afterwards, he entered Purdue University in the Fall of 1993, where he would remain for one year as a freshman engineering hopeful. However, the taste of Japan was too strong in his memory, and he would never again be satisfied unless he was exploring his two passions: engineering and Japanese. And so, in September of 1994 he would transfer to SUNY at Buffalo located, aptly enough, in Buffalo, New York. After one year at Buffalo, Ward returned to his second home, Kanazawa, Japan to study at Kanazawa University. Upon the return to the United States he befriended Timothy J. Curry; without his friendship, encouragement, and assistance Ward would have never have succeeded as he did. He would also befriend L. Phida Ung, with whom he has shared as many splendid moments as any one person can hope for. In June of 1998, the author graduated from SUNY at Buffalo with a B.S. in aerospace engineering. In the Fall of 1998, Ward moved from the East coast to the West coast, settling in Seattle, Washington where he entered the University of Washington graduate program as a research assistant with Uri Shumlak of the Plasma Dynamics Group of the Department of Aeronautics and Astronautics. Since then he has been able pursue concurrently two Master of Science degrees in aerospace engineering and technical Japanese. Ward again departed for Japan for a third time in September of 2000, where he worked as an intern at ZEXEL Corporation. He returned in early of 2001, where he and L. Phida continue to live in wonderful Seattle, Washington.