Development of the DRACO ES-PIC Code and Fully-Kinetic Simulation of Ion Beam Neutralization

Lubos Brieda

Thesis submitted to the faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Aerospace Engineering

Joseph J. Wang, Committee Chair
Douglas VanGilder, Committee Member
Wayne Scales, Committee Member

June 2nd, 2005
Blacksburg, Virginia

Keywords: PIC, Plasma Modeling, Neutralization

Development of the DRACO ES-PIC code and Fully-Kinetic Simulation of Ion Beam Neutralization

Lubos Brieda

(ABSTRACT)

This thesis describes development of the DRACO plasma simulation code. DRACO is an electro-static (ES) code which uses the particle-in-cell (PIC) formulation to track plasma particles through a computational domain, and operates within the Air Force COLISEUM framework. The particles are tracked on a non-standard mesh, which combines the benefits of a Cartesian mesh with the surface-resolving power of an unstructured mesh. DRACO contains its own mesher, called VOLCAR, which is also described in this work.

DRACO was applied to a fully kinetic simulation of an ion-beam neutralization. The thruster configuration and running parameters were based on the NASA's 40cm NEXT ion thruster. The neutralization process was divided into three steps. Electron dynamics was studied by assuming an initial beam neutralization, which was accomplished by injecting both electrons and ions from the optics. Performing the simulation on a full-sized domain with cell size much greater than the Debye length resulted in a formation of a virtual anode. Decrease of the cell size to match the Debye length was not feasible, since it would require a million-fold increase in the number of simulation nodes. Instead, a scaling scheme was devised. Simulations were performed on thruster scaled down by a factor of 100, but its operating parameters were also adjusted such that the produced plasma environment did not change.

Loss of electrons at the boundary of the finite simulation domain induced a numerical instability. The instability resulted in a strong axial electric field which sucked out electrons from the beam. It was removed by introducing an energy based particle boundary condition. Combination of surface scaling and energy boundary resulted in physically sound simulation results. Comparison were made between the Maxwellian and polytropic temperatures, as well as between simulation electron density and one predicted by the Boltzmann relationship.

The cathode was modeled individually from the beam by introducing a positively charged collector plate at a distance corresponding to the beam edge. The local Debye length at the cathode tip was too small to be resolved by the mesh, even if mesh-refinement was incorporated. Since the simulation was not concerned with the near-tip region, two modifications were performed. First, the a limiting value of charge density at the tip was imposed. Second, the cathode potential was allowed to float. These two modifications were necessary to prevent development of a strong potential gradient at the cathode tip.

The modified cathode model was combined with ion injection from the optics to model the actual beam neutralization. Three configurations were tested: a single thruster, a 2x2 cluster with individual cathodes and a similar cluster with a single large neutralizer. Neither of the cases achieved neutralization comparable to one in the base-line pre-neutralized case. The reason for the discrepancy is not known, but it does not seem to be due a loss of electrons at the walls. The difference could be due to limited extent of the modeled physics. An additional work is required to answer this question.

Acknowledgements

The research undertaken during my graduate study was financially supported by the Air Force Research Laboratory at Edwards Air Force Base. Additional support was provided by the Boeing Corporation and the Virginia Tech Department of Aerospace Engineering.

The results presented here were obtained thanks to many discussions with a large number of knowledgeable colleagues. First, most credit goes to my graduate advisor, Dr. Joseph Wang. Before beginning my research, I had only a vague concept of the term plasma (it's the fourth state of matter!), and the acronym PIC had no meaning to me. Dr. Wang's research topics allowed me to combine my interest in propulsion with my relatively good knowledge of programming. Although the beginning was fairly slow, I soon acquired a good knowledge of the modeling process, which allowed me to develop the DRACO code, and apply it to a number of physical problems. Dr. Wang's vast expertise in numerical modeling also turned out to be very helpful in dealing with numerically induced bugs.

Next, I owe a lot of gratitude to Doug VanGilder, Mike Fife, Matt Gibbons and the rest of the Air Force COLISEUM team. Their selection of Virginia Tech to develop a new module for the COLISEUM framework allowed me to work on this challenging, yet very interesting, project. An extra thanks goes to Doug for serving on my thesis committee and for allowing me to keep working on the COLISEUM project after my graduation.

I also want to thank Chris Hoffman and Preston Geren from Boeing. Their interest in DRACO, and the use of it to perform numerical neutralization studies introduced me to the world of fully-kinetic particle modeling. Frequent concerns and questions from Chris not only kept me busy for a while, they also forced me to question and further analyze the physics modeled by the code.

I am also grateful to Dr. Wayne Scales for joining my thesis committee. Dr. Scales' knowledge of wave interactions with plasma will sure become handy in future attempts to resolve the high potential in cathode neutralization results.

A great amount of thanks goes to Luke Scharf, the VT AOE system administrator. In just few years, Luke was able to bring the department's facilities back from the inner circle of computational hell. Most of the work presented here would not be possible without having access to the many powerful computers he acquired. Numerical implementation of the code also greatly benefited from many interesting points learned in Dr. Adrian Sandu's course on scientific computing.

Special thanks goes to the other students at the capLAB: Julien Pierru, Binh Tran, Raed Kafafy, Randy Spicer, Bob Kikolski and Hyunju Jeong. Julien proved to be a good friend, and his constant frustration with coding showed me that my own problems are perhaps not so bad. I appreciate Julien and Binh's use of DRACO in their own graduate research. Raed deserves

credit for the creation of IFE, even though I was often frustrated with the communication difficulties between C and F90. Randy and Bob, my undergraduate helpers were very helpful in debugging of the code and will be taking over future DRACO development at Virginia Tech. Hyunju also used DRACO for some of her initial work, and her simulations led to introduction of many new features in DRACO, such as the thin-plate particle sinks.

Then there were many other students and friends who helped me with my research, by both answering my questions, and providing good distraction from the constant workload. First, Mark Santi and Shannon Cheng from MIT helped a lot with my initial frustration with the COLISEUM framework. Visiting this odd pair in Cambridge was a great excuse to go on a road trip. I had a great time living with Shannon during my summer internship at AFRL, sharing our single mini fridge, even though we were both constantly busy writing papers. Going on backpacking trips with my friends from the VT Outdoor Club also helped me remain as stress-free as possible while being a grad student. Of course, thanks also goes to my parents. Sarah Johnson also deserves a lot of credit for putting up with my constant lack of time, especially during her visits to Blacksburg. To everyone who helped out, whether mentioned here or not, thank you.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Basics of Electric Propulsion

1.1.1 Overview

Electric thrusters generate propulsive force by accelerating charged particles using electromagnetic fields. The velocity of the emitted beam is computed directly from the conservation of energy

$$v_{eq} = \sqrt{\frac{2q}{m}\Delta\phi} \tag{1.1}$$

where q is the charge of the particles and m is their mass. Higher exit velocity can be achieved by simply applying a larger potential gradient, $\Delta\phi$. Since the thrust, T, of an ideal rocket is given by

$$T = \dot{m}v_{eq} \tag{1.2}$$

a higher exit velocity leads to an increased thrust. The potential drop is generated by the power supply and hence the thrust of an electric thruster is limited only by the amount of available electric power.

Unfortunately, the thrust achievable by current state-of-the-art electric thrusters is very small (less than 1N) due to a low propellant mass flow rate, $\dot{m}$. Electric rockets are thus not capable of producing sufficient lift-off forces to overcome the gravitational pull of the Earth. However, the rocket equation states that

$$e^{-\frac{\Delta v}{v_{eq}}} = \frac{M_b}{M_0} \tag{1.3}$$

where Δv = mission specific velocity change
M_0 = initial mass of the rocket
M_b = burn out mass, $M_0 - M_{fuel}$

A rocket with a higher v_{eq} requires less fuel to achieve same Δv than a rocket with a lower v_{eq}. This property of electric propulsion (EP) is very attractive to mission designers, since it means that a smaller portion of the total system mass needs to be devoted to fuel. An

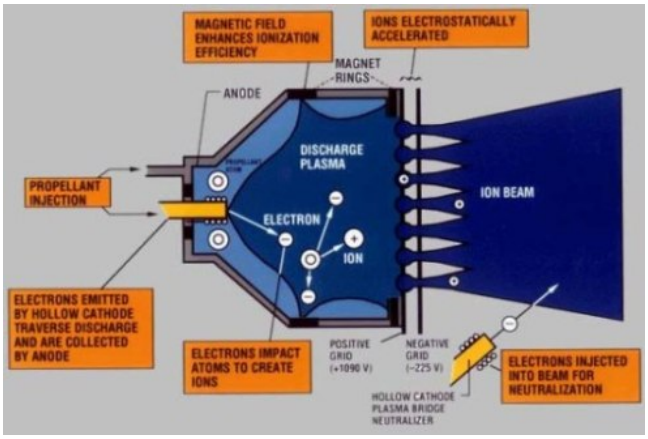


Figure 1.1: Schematic drawing of a typical ion thruster[1]. Neutral propellant is introduced into the ionization chamber. Electrons, introduced from a cathode are trapped within the chamber by an externally applied magnetic field. Collisions with the neutral gas produce ions which are then extracted through a set of closely spaced grids.

interplanetary probe can thus carry a larger payload, or the initial mass can be reduced, allowing the probe to be launched on a smaller (and thus cheaper) launch vehicle.

1.1.2 Ion Thrusters

Electric thrusters can be classified according to the methods used to create and accelerate the charged particles. Two types of EP thrusters most applicable to satellite propulsion are *Hall* and *ion* thrusters.

Hall thrusters consist of an inner central magnet surrounded by an outer magnetic ring. Neutral gas is introduced into the annulus through a supply line located in the back part of the thruster. An anode plate is also located near the back of the thruster. A single external cathode provides electrons which flow into the channel. The electrons get trapped by the external magnetic field and start drifting around the annulus. The magnetic field strength is generally highest near the thruster exit, resulting in a higher electron density near this region. As the electrons orbit around the thruster centerline, they collide with the neutral gas, generating ions and secondary electrons. These ios are then accelerated out of the annular channel by a potential gradient due to the greater electron concentration near the exit.

The acceleration in ion thrusters, shown in figure 1.1, is instead induced by a potential gradient applied across a set of closely spaced grids, or *ion optics*. The optics are metal plates, usually manufactured from molybdenum, containing a large number of fine holes. The first grid, the screen grid, contains larger holes than the second, accelerator grid, and helps in guiding of the ion beamlets through the grids. The advantage of the gridded approach is that a highly focused beam is created, with exit velocities exceeding those achievable in Hall thrusters.

Neutral gas is ionized in an ionization chamber, which is separated from the external

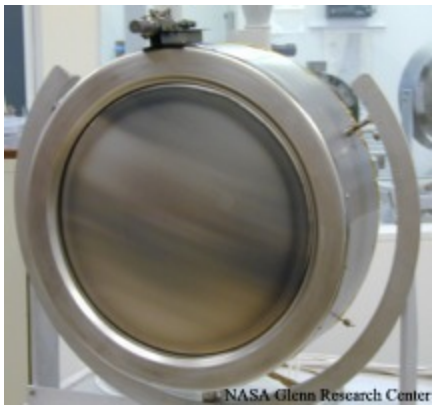


Figure 1.2: 40cm NASA NEXT ion thruster [2].

medium by the grids. The ionization electrons are introduced by a hollow cathode located within the chamber. Flow of electrons to the walls is minimized by trapping the electrons in magnetic field lines, which are generated by several ring-shaped magnets. Thruster charge balance is maintained by ejecting wall-collected electrons through an external neutralizer cathode.

The feasibility of electric propulsion has been successfully demonstrated by NASA's Deep Space 1 mission, which used a 30cm NSTAR ion thruster as its primary propulsion system. Currently, several new ion thruster models are in development. One of these, the NASA Evolutionary Xenon Thruster (NEXT) 40cm ion thruster, is shown in figure 1.2. The increase of beam diameter from 30 to 40 cm resulted in doubling of the beam area. A higher thrust was thus achieved without increasing the plasma density near the thruster exit. The thruster was tested with power input ranging from 1.1 to 6.9kW, which is a significant increase from the 2.5kW input to the NSTAR. The thruster achieved up to 4060 seconds of ISP, and produced 0.238N of thrust at 69% efficiency[2].

1.2 Numerical Plasma Modeling

1.2.1 Thruster-Spacecraft Interactions

Ionization chambers operate at limited efficiency levels, and thus only a fraction of the supplied propellant will become ionized. The neutral gas can exit the chamber by a random walk across the accelerator grid. Although the rate of neutral gas leakage is much lower than the rate of the ion emission, the low exit velocity is responsible for increasing the neutral density near the thruster exit. Some neutrals will undergo a charge-exchange (CEX) collision with the ions, in which the charge of the two particles is switched without affecting their velocities. The result is a presence of slow moving ions near the thruster exit region.

Generally, the potential on the spacecraft will be several volts negative in respect to the ion beam. This potential gradient is sufficiently strong to attract the CEX ions towards the spacecraft. Some backflowing ions will collide with the spacecraft surface, and if their incident

energy is sufficiently large, will sputter off spacecraft native material. Sputtering of the ion thruster optics is responsible for the limited lifetime of ion thrusters. Interaction with solar arrays can also reduce the power-gathering capabilities of the spacecraft. Some sputtered material will undergo additional CEX collisions. Molybdenum, the material typically used to manufacture the ion thruster optcs, is highly contaminating. Interactions with critical scientific instruments can lead to premature mission end due to a layer of molybdenum depositing on a surface of the sensor.

The overall ion dynamics are governed by the potential difference between the beam and the spacecraft. Ionization of the neutral propellant results in creation of secondary electrons. These electrons must be ejected, otherwise a highly negative potential would develop on the spacecraft. If the potential difference between the beam and the spacecraft is larger than the initial potential drop used to accelerate the ions, the ions will reverse their velocities and will flow back to the spacecraft, thus negating the initial thrust.

The actual potential distribution in the beam is governed by variations in charge density. High beam potential will cause the ion beam to diverge. This is a natural response of the plasma, since the collective force of a large number of positive particles will yield a highly repulsive force. From the standpoint of thrust maximization, small divergence angles are desired. However, not much is known about the actual process of ion beam neutralization. Good understanding of ion beam neutralization will become even more significant once large clusters of ion thrusters neutralized using a single cathode are deployed. However, study of the neutralization process in a laboratory is a difficult task. Electrons are highly mobile, and respond rapidly to any potential gradient. The plasma environment in a vacuum chamber is influenced by the presence of background gas and potential sheath near the walls of the tank. Presence of measurement probes modifies the potential distribution in the tank. All of these effects can strongly influence the motion of the electrons. Furthermore, erosion damage due to sputtering often requires several thousand of hours of continuous thruster operation. Performing such studies in a vacuum chamber is a lengthy and a costly process.

1.2.2 Lorentz Force and the Maxwell's Equations

Instead, plasma dynamics can be studied using numerical modeling. Numerical modeling uses computers to propagate the plasma based on physical models describing individual and collective motions of charged particles. Using a fluid model to accurately describe plasma requires at least some prior knowledge of the solution, since the fluid approximation is dependant on a number of coefficients. Values of these coefficients may not be known prior to starting of the simulation. Hence, a particle-based kinetic model of plasmas must often be used.

The force acting on a charged particle moving in an electromagnetic field is given by the Lorentz formula:

$$\vec{F} = q\left(\vec{E} + \vec{v} \times \vec{B}\right) \tag{1.4}$$

where $\vec{F}$ = force term, N
q = charge on the particle, C
$\vec{E}$ = electric field, V/m
$\vec{v}$ = velocity of the particle, m/s
$\vec{B}$ = magnetic field, T

The electro-static (ES) formulation, which is presented in this work, assumes that the plasma induced magnetic field is negligible, $\partial\vec{B}/\partial t = 0$. The ES version of eq. 1.4 is then

$$\vec{F} = q\left[\left(\vec{E}_a + \vec{E}_{ind}\right) + \vec{v} \times \vec{B}_a\right] \tag{1.5}$$

The electric field can be seen to consist of two components. The a subscript denotes an externally applied field, which, in the case of the electric field, is due to the presence of charged objects. The *ind* subscript refers to the plasma self-induced field. Since the self-induced magnetic field is ignored in the ES formulation, it is ignored in 1.4.

Any meaningful plasma code must be able to compute the self-induced electric field. Since plasma can be thought of as a fluidic conductor, it must satisfy the Maxwell's equations. Only equations 1 and 2,

$$\nabla \cdot \vec{E} = \frac{\rho}{\varepsilon_0} \tag{1.6}$$

$$\nabla \times \vec{E} = -\frac{\partial \vec{B}}{\partial t} \tag{1.7}$$

$$\tag{1.8}$$

are of importance in the ES formulaiton. Since the magnetic field is time invariant, $\nabla \times \vec{E} = 0$ and the field is conservative. A scalar potential, $\vec{E} = -\nabla\phi$, thus exists. Substitution of eq. 1.6 leads to the Possion's equation

$$\nabla^2\phi = -\frac{\rho}{\varepsilon_0} \tag{1.9}$$

1.2.3 Particle-In-Cell method

The Particle-In-Cell (PIC) method was described in detail by Birdsall [6]. The method couples the kinetic description of plasma with a grid-based representation of the electric field. The continuous domain is discretized into a volumetric mesh by defining a collection of nodes, and their associated connectivity.

The volume of the particle is assumed to follow the volume of the cell in which the particle resides. Hence, each particle can be though to be carrying a representative charge density, ρ_p, equal to $\rho_p = q/V_{cell}$. The global charge density term, needed to solve the Poisson equation, is then obtained by *weighing* (or *scattering*) each particle onto surrounding grid nodes.

Since the number of real plasma particles is extremely large (a m^3 volume of an ion-thruster plasma contains O(15) ions and just as many electrons), the PIC method introduces the concept of a *macro-particle*. Each macro-particle deposits charge density of *sw* (specific weight) real particles, but its dynamics are governed by the single particle parameters. Hence, $\rho_{mp} = sw \cdot \rho_p$. From now on, the term "particle" will be used to describe the computational "macro-particle."

Once the global charge density is known, a new instance of the electric field can be calculated. The process here is to use an elliptic Poisson solver to solve the Poisson's equation, followed by a numerical differentiation to obtain the electric field. Having new values of the electric field allows to recompute the Lorentz force acting on each particle, by performing a *gather* operation of the field terms onto each particle position. The force term is then used to update the velocity of each particle, followed by moving of the particles by a finite

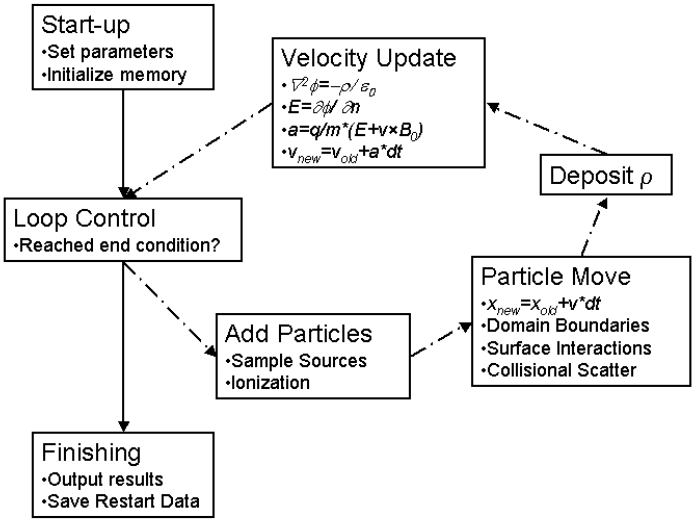


Figure 1.3: An overview of the PIC process. The plasma density is computed from position of macro-particles. The density is used to update the electric field. The macro-particles are then moved to new positions and the whole process is repeated. Collision modeling can be introduced to account for scatter and ionization.

distance. A collisional modeling can be used next to scatter the particle or to introduce new particles due to ionization. New particles are also introduced (to simulate a thruster firing, for instance) by sampling particle sources. The new spatial distribution of the particles leads to a new charge density, and the entire process is repeated. This iterative process terminates after some prescribed condition is met. In the code presented here, called DRACO, the end condition can be exceeding a prescribed number of time steps, reaching a steady-state, or reaching a desired *plasma* time. Figure 1.3 shows a schematic of this process.

1.3 Heritage Code

1.3.1 PLUME PIC Algorithm

DRACO is based on a 3D ES-PIC code, developed by Wang [7], called PLUME. The PLUME code was a stand-alone program written in Fortran77 to simulate the backflow of CEX ions on the NASA's DS1 mission. The code's major features were:

- The fast beam ions were not tracked, instead they were represented by an analytical beam profile.
- The production of CEX was modeled by introducing new CEX particles in the plume according to a pre-computed CEX production rate [8]. The neutral plume was also loaded according to an analytical description.

- The CEX ions were tracked according to the previously described PIC scheme. The flow of CEX was assumed to be collisionless, since the mean free path of the ions was larger than the typical satellite dimension.
- The code operated on an uniform Cartesian grid.
- The satellite was modeled as a solid cylinder with a planar solar array. The surface shape was described analytically, but the dimensions were loaded from an input file. The grid nodes located within the solid objects were marked as *internal*, and potential on them was fixed.
- The Poisson's equation was solved using the DADI scheme. Dirichlet, Neumann zero ($\partial\phi/\partial n = 0$), and periodic boundary conditions were supported.
- Particle surface interaction was performed by checking the particle's final position against the analytical surface definition. Since the code only tracked CEX ions, the collided particles were removed from the simulation.
- The code ran for a prescribed number of time steps. An automatic check for steady state was not available.
- Only a quarter of the domain was simulated due to the problem symmetry.
- All memory was allocated using static arrays.
- The source production rate was specified by the desired number of particles to be injected each time step, and their representative number density. The code then computed the necessary specific weight of each particle.
- Simulation results were outputted in an unknown 3D format. The list of simulation variables which were outputted was hard-coded, and it included the potential, charge density, components of the electric field, and the current density vectors.
- Diagnostic properties, such as the total simulation energy, and the combined particle momentum were also saved at each time step.
- The input parameters were specified in normalized units. The simulation results were also saved in normalized quantities. Relying on a normalized input required the user to have a good understanding of the normalization process before the code could be used to model different operating conditions.

1.3.2 IFE Field Solver

Lin, et al, [9] proposed a new method for solving of elliptic differential equations based on an interface finite element (IFE) formulation. This method combined the efficiency of a Cartesian grid with the object resolving power of an unstructured, body-fitted mesh. Each Cartesian cell was subdivided into several elements (triangles or tetrahedrons). Planar cuts of *interface* elements were used to describe the object boundaries. The cuts were computed by intersecting the element edges with an analytical description of the surface.

A three-dimensional IFE potential solver for the PIC algorithm was developed by Kafafy. Each Cartesian cell is divided into five tetrahedrons. The solver did not directly distinguish between solid objects and the free space. Instead, each region was assigned a relative local permittivity (ε_0). This formulation allowed to model the presence of semi-conductors. Objects at a fixed potential were given a very high value of the local ε_0. This feature was used extensively by Kafafy to model dielectrics in ion thruster optics [10].

1.3.3 COLISEUM frame-work

Around the same time as the work on the IFE solver was being performed, Virginia Tech was tasked by the Air Force Research Laboratory to develop a Cartesian-based plasma simulation module for its COLISEUM framework [11]. The idea behind COLISEUM was to develop a flexible and user-friendly software package, which would be able to tackle a wide variety of plasma problems. The framework consisted of a core library providing basic I/O support, and several plasma simulation modules. Simulations progressed according to commands in the `coliseum.in` script file.

The flexibility of COLISEUM is largely due to the large fidelity range of its available plume models. A simple sputtering analysis could be performed by instantaneous ray tracing. Since ray-tracing is not capable of resolving plasma dynamics, more accurate analysis required the use of a PIC algorithm. A PIC module based on a body-fitted, unstructured tetrahedral mesh (AQUILA) was developed at MIT [12], and was subsequently inherited by AFRL. However, the particle push operation can be computationally expensive on an unstructured mesh. The DRACO module, based on the interface model of Lin, was thus developed to simplify the particle push (and thus to allow a higher number of simulation particles), while providing support for detailed surface definition.

Common to all simulation modules was the presence of a triangular surface mesh. The triangles were grouped together into various *components*, and any desired component properties were specified in a secondary input file. Material definition along with coefficients for material interactions were also loaded from a separate file.

The core library also contained several types of particle sources. Any of these sources could be attached to a component, and particle would then be sampled from the actual surface triangles.

An attractive feature of the framework was that it imposed only minimum requirements on the development of the new module. For instance, the parameters needed to define each component were left at the discretion of the actual module. Yet, at the same, it provided the developer with an existing set of tools which, although not directly tied to plasma physics, would normally take a significant time to develop.

1.4 Thesis Overview

Chapters 2 and 3 of this thesis describe development of the DRACO plasma simulation module, and its supporting volumetric mesher, VOLCAR. DRACO was then used to model ion beam neutralization using a fully kinetic Particle-In-Cell approach. However, several problems were encountered on the way. Chapter 4 describes the modeling approach. A new ion source model has been developed, based on experimentally measured values of beam current

density. A geometry scaling method was developed to allow the simulation to resolve the Debye length while retaining a numerically sound number of simulation nodes. A numerical instability was also observed, and was traced to the removal of electrons at the external domain boundaries. A new particle boundary condition based on conservation of energy was developed. Chapter 4 chapter also introduces modifications made to a neutralizer cathode. The simulation was not able to properly resolve the electric field at the cathode tip due to a large electron density. The electric field was thus fixed by introducing a limiting value of electron charge density near the tip.

Modeling results obtained with this newly developed method of fully kinetic electron modeling are presented in chapter 5. Three configurations were studied: a single thruster, a 2×2 cluster with individual cathodes, and a 2×2 cluster with a single central neutralizer. These results were compared with simplified model, in which pre-neutralization was assumed by injection of both electrons and ions from the optics. Discrepancies were observed, and they seem to be due to instabilities present in the beam. Interestingly, replacing the four individual cathodes in a cluster configuration with a single central cathode resulted in a better beam neutralization.

Chapter 2

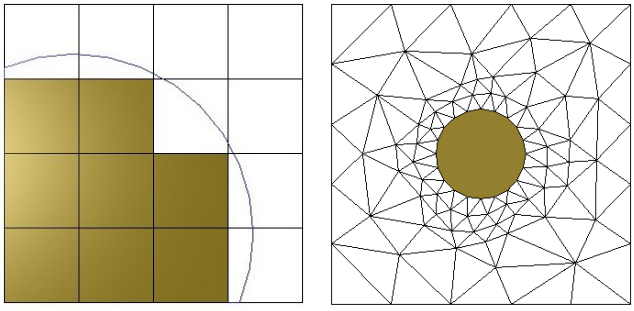
Structured Tetrahedral Grid

2.1 Overview of gridding techniques

Performing a numerical simulation of a 3D domain requires development of a scheme to represent the continuous spatial volume in a computer accessible format. Common method is to discretize the domain through a computational mesh. Mesh is a collection of nodes and an associated cell connectivity. The cells act as the most elementary building blocks of the domain. Physical parameters are stored at the node locations (node-centered scheme), from where they are interpolated to any point within the cell boundary using a weighing scheme.

The use of meshes in kinetic simulations, such as PIC codes, imposes another requirement on the structure of the mesh. Although the particles are moved in the continuous physical domain, their position in respect to the mesh nodes is needed to calculate the charge density and to apply new acceleration. Since a simulation can contain millions of particles, the mesh structure should allow for a fast tracking of particles using both the physical and the cell coordinates.

Figure 2.1 shows two methods of representing the space around a sphere, using a two-



(a) structured Cartesian (b) unstructured body-fitted

Figure 2.1: Two methods of representing the domain around a “sphere”.

dimensional approximation, in which the sphere is shown as a circle, Cartesian blocks are squares, and finite-element tetrahedrons are shown as triangles.

Division of the domain into a set of equally sized bricks results in an uniform structured Cartesian mesh. Each node is referred to using a triple $i-j-k$ index, with $i \in [0:nx-1]$, $j \in [0:ny-1]$, $k \in [0:nz-1]$, where nx, ny and nz is the number of nodes in the respective direction. The first node is numbered 0 to follow the C language memory indexing. Connectivity is implied from $x = x_0 + i \cdot dx$, where x_0 is the origin and dx is the cell spacing. Matching a particle with a cell is simple, since the $i-j-k$ index of the parent cell can be computed by simply inverting the connectivity relationship.

A big disadvantage of the uniform Cartesian mesh is its inability to precisely resolve complicated surface geometry. Objects are resolved on a mesh by flagging contained nodes as internal. Since the information can only be stored at the node locations, the objects degenerate into a "stair-case" representation. This degeneration affects both the field solution as well as the particle boundary check. The latter can be corrected by performing the particle check against an analytical model of the surface. However, this technique is practical only for simple geometries.

The surface boundary can be resolved accurately if a body-fitted unstructured mesh is used. The unstructured mesh, also shown in figure 2.1, utilizes a number of arbitrary-shaped volumetric cells to describe the domain. Drawback of unstructured meshes is the increased computational overhead needed to track particles. The index of the owning cell cannot be obtained by an inversion of connectivity, since an analytical expression for connectivity does not exist. Hence, matching a particle with a cell requires a search through all cells for the one tetrahedron containing the particle. Such a search is computationally wasteful. A speed-up can be obtained by allowing each cell to keep a list of owned particles. The move operation is performed on a cell-by-cell basis, and particles leaving the cell are shipped to the appropriate neighbor. The neighbor is determined from the face through which the particle left.

The fast particle push makes the Cartesian mesh superior for simulations involving a large number of particles, and low-fidelity surface detail. Such is the case in a study of plume expansion and backflow. However, if the simulation is dominated by near-surface effects, as may be the case in detailed sputtering or contamination modeling, surface resolution becomes crucial, and an unstructured mesh should be used.

Several modified methods exist which tend to mitigate the disadvantages of the two primary meshing schemes. One such a technique uses a stretched Cartesian mesh. The mesh uses uniformly ordered nodes, however, the spacing between the nodes varies through the domain. Matching a particle to a cell can still be performed analytically, but the operation may be more computationally expensive. The stretched Cartesian mesh will still reduce smooth surfaces into a stair-case, however the step size can be decreased.

Creating a stretched mesh to resolve a large number of surfaces would require a high-order expression for the cell size. For this reason, the stretched mesh is ideal only for simulations containing just a single high-detail region. Mesh-refinement is another meshing technique in which embedded child meshes are used to increase local resolution. However, although mesh refinement has its advantages, it still does not overcome the fundamental limitation of Cartesian-mesh surface representation.

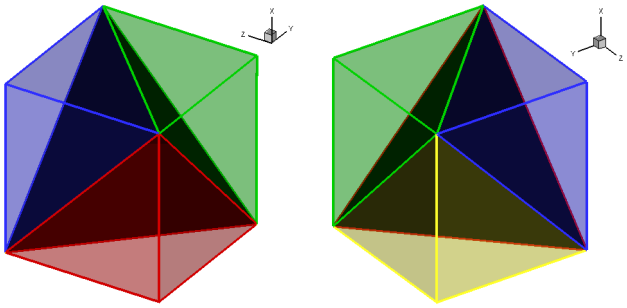
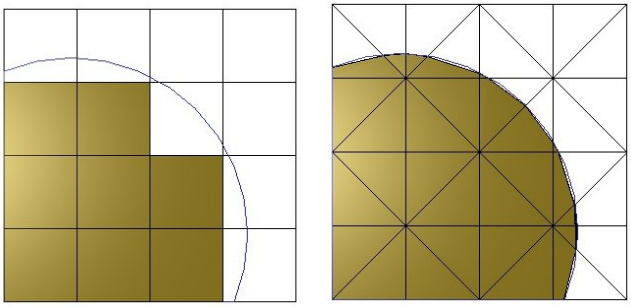


Figure 2.2: Tetrahedron represenation of a cube



(a) structured Cartesian (b) overlaid tetrahedral

Figure 2.3: Representation of a circle on a Cartesian and on an interface tetrahedral mesh

2.2 Structured tetrahedral interface model

The mesh used by DRACO is a combination of a primary Cartesian and a secondary tetrahedral mesh. This approach combines the simplicity of a Cartesian mesh with the surface resolving power of an unstructured mesh. The simulation volume is first covered by an uniform Cartesian mesh. However, each Cartesian cell is subsequently divided into five tetrahedrons, as shown in figure 2.2.

The surface is described using planar cuts of the *interface* tetrahedral elements. Figure 2.3 shows, using a 2D approximation, the power of this simple model. Even though only linear cuts are used, the interface mesh shown on the right manages to represent the circle with a great degree of accuracy.

The tetrahedrons share their vertices with the Cartesian cells, greatly reducing the computational overhead needed to move particles. In fact, the particle scatter/gather operation is identical to one performed on a standard Cartesian mesh. The particle boundary check is somewhat more involved, but it is still simpler and less computationally expensive compared to a body-fitted mesh. The actual particle boundary check is described in a great detail in chapter 3.

The mesh is based on the concept of an interface. The interface creates a barrier between

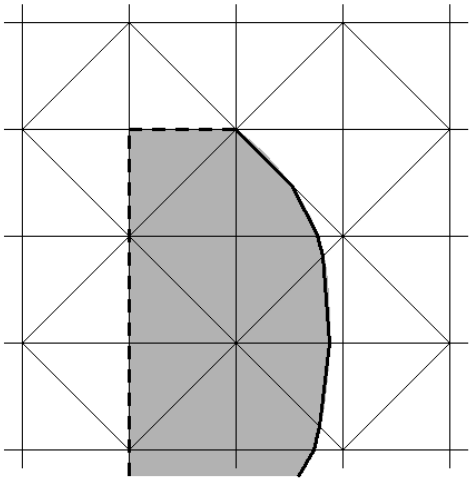


Figure 2.4: If a surface is flushed with Cartesian blocks, a direct transition from external to internal elements will exist, without an intermediate interface. The flushed faces are shown by a dashed line. The solid line indicates the interface cuts.

the nodes and elements which are internal (the objects) and those that are external (the "vacuum"). The boundary is described by a collection of interface tetrahedrons, each being divided into two halves by the planar intersection with the surface. It should be pointed out, however, that not all configurations will generate an actual interface between the two half-domains. Surfaces which are flushed with the boundaries of the Cartesian cells do not cut through any tetrahedrons and no transition zone exists between the internal and external elements. This concept is illustrated in figure 2.4.

However, although the interface mesh is attractive to PIC simulations, its non-standard gridding technique required the development of an internal mesh generator. The meshing module was named VOLCAR, an abbreviation for VOLumetric CARtesian mesher. Generation of the mesh requires three tasks:

- Initialization of data structures, and creation of the tetrahedron connectivity
- Creation of the interface domain by intersection of tetrahedral cells with the surface mesh
- Classification of all nodes and non-interface elements as internal or external

The second and third items are described in full detail in the following sections. Besides these elementary tasks, VOLCAR also contains support for mesh refinement, data output and additional post-processing tasks. These are also described in this chapter.

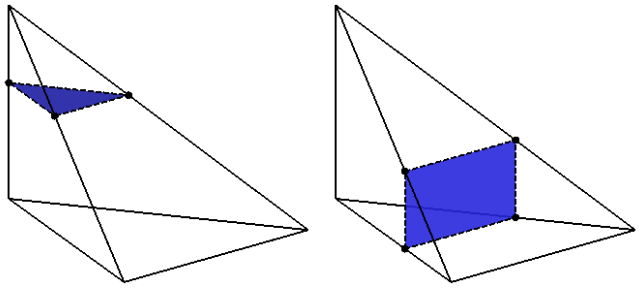


Figure 2.5: Valid 3 and 4 point interface cuts

2.3 Surface intersection

2.3.1 Computation of Intersection Points

VOLCAR only supports simple planar cuts of the interface elements. A plane can cut a tetrahedron in two ways, as shown in figure 2.5. Since the plane cannot originate inside the tetrahedron, the interface surface can be determined by intersecting the tetrahedron edges with the surface definition. The surface is represented using a triangular mesh, and thus the intersection points are computed using a line-triangle intersection (LTI) algorithm.

Two conditions must be satisfied for an intersection between a line and a triangle to exist. First, the two endpoints of the line must not lie on the same side of the plane to which the triangle belongs. This condition guarantees that the line segment intersects the plane. Second, the intersection point must be located inside the triangle.

The check for a line-plane intersection is simple. The simulation volume Ω is cut by the plane Γ into two halves: Ω_+ and Ω_-, such that:

$$\hat{n} \cdot (\vec{x} - \vec{x_0}) \begin{cases} = 0, & \forall \vec{x} \in \Gamma \\ > 0, & \forall \vec{x} \in \Omega_+ \\ < 0, & \forall \vec{x} \in \Omega_- \end{cases} \tag{2.1}$$

where $\vec{x}$ is some arbitrary point, and Γ is defined by its normal vector $\hat{n}$ and a point $\vec{x_0}$ located on the plane. If a substitution of both endpoints into eq. 2.1 results in a RHS with the same sign, both points lie in the same half-domain, and an intersection is not possible.

If the line cuts through the plane, the intersection point is found from

$$\vec{p} = \vec{x_1} + t\,(\vec{x_2} - \vec{x_1}) \tag{2.2}$$

where $\vec{x_1}$ and $\vec{x_2}$ are the two endpoints of the edge and t is given by

$$t = \frac{\hat{n} \cdot (\vec{x_0} - \vec{x_1})}{\hat{n} \cdot (\vec{x_2} - \vec{x_1})} \tag{2.3}$$

Lastly, a check needs to be made to verify that the intersection point is internal to the triangle. Several methods were explored. The three FE linear basis functions Ψ_1 to Ψ_3 can be evaluated on the triangle. The range of each Ψ_i will be in $[0:1]$ as long as the point is

internal to the triangle. Another approach compared the two secondary angles α_1 and α_2 formed at each node by connecting the vertex to p to the actual angle at the vertex. The point is internal as long as $\alpha_1 + \alpha_2 = \alpha$ is true for all three nodes.

However, a simpler method is based on the area of the triangle. Three sub-triangles can be formed by connecting two original vertices to the intersection point. The area of each triangle is given by:

$$\mathbb{A}x_1x_2x_3 = \frac{|\,(\vec{x_2} - \vec{x_1}) \times (\vec{x_3} - \vec{x_1})\,|}{2} \tag{2.4}$$

If the point is internal, the three subtriangle will together cover the entire area of the parent triangle, or:

$$\frac{\mathbb{A}x_2x_3p + \mathbb{A}x_3x_1p + \mathbb{A}x_1x_2p}{\mathbb{A}x_1x_2x_3} \begin{cases} = 1, & p \in \triangle x_1x_2x_3 \\ \neq 1, & p \notin \triangle x_1x_2x_3 \end{cases} \tag{2.5}$$

2.3.2 Modification for non-smooth surfaces

The intersection algorithm described in the previous section works well for smooth surface, but problems arise if sharp edges or corners terminate inside the tetrahedron. Such configurations will result in tetrahedral edges being cut more than once. However, the existing intersection model supports only a single cut per edge, so the additional cuts will be discarded.

The problem of bad intersections due to edges can be partly eliminated by snapping all corners and edges to the grid cells. A surface that is flushed with the faces of a Cartesian cell does not cut through the a tetrahedron and does not create interface elements.

COLISEUM organizes surface triangles into *zones*. A zone is a collection of triangles of a single component type which does not contain any edges with angle sharper than a user prescribed value. The default limiting angle is 44°. A surface mesh of a cylinder, for example, will be separated into three zones: the two flat "caps" and the round body.

VOLCAR uses the zone information to limit the intersection to only those surface triangles which have a potential to cut through Cartesian cells. The intersection code loops through all the zones, and if it determines that the zone is flushed with the Cartesian cells, it skips the zone. Otherwise, all zone triangles are intersected with the mesh.

The check consists of three parts. First, due to the Cartesian structure, a flushed zone will contain triangles with normal vectors along the $\hat{i}$, $\hat{j}$ or $\hat{k}$ direction. VOLCAR picks the first surface triangle of the zone in question, and checks the orientation of its normal vector. If it is oriented properly, it next determines whether the triangle is flushed with the Cartesian cells. This involves computing the cell fraction for the triangle centroid along the direction of the normal vector. For instance, for a triangle with a normal vector in the $\hat{i}$ direction, the cell fraction, cf is

$$cf = Dec\,[(x - x_0)/dx] \tag{2.6}$$

Here, Dec is a function returning the decimal part of a real number. The triangle is flushed if $cf <$ FLT_EPS or $cf > 1 -$ FLT_EPS, where FLT_EPS is the floating point epsilon.

2.3.3 Consistency Check

Consistency check is performed to verify the intersections. Interface intersections must satisfy three criteria:

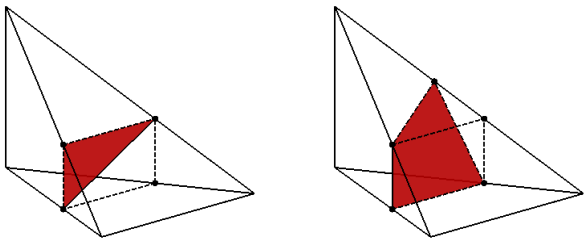


Figure 2.6: Invalid interface cuts: incomplete and non-planar cuts

1. Interface cut must divide the tetrahedron into sub-volumes
2. Interface cut must be planar
3. Continuity requires matching intersection nodes on shared edges

The first case is checked for by computing the plane equation for each type-3 tet. This plane is then intersected with the 6 edges of the tet. In some rare cases, a fourth intersection point will surface. This procedure assures that all intersection cuts are completely filled. An example of incomplete cut can be seen in figure 2.6.

The second check is for type-4 cuts that do not resolve into a flat plane. The code computes normal vectors at each intersection point by crossing all possible vectors. Bad intersection points will not share the normal vector with the rest of the plane.

Lastly, the code compares intersection points on shared edges. Mismatches result in warning messages being displayed, however no corrective action is taken.

2.3.4 Numerical Implementation

The above algorithm had to be adapted to work within the binary computer finite precision representation of real numbers. Since main COLISEUM module operates in single-precision arithmetic, mesh intersection must also use the single precision. The use of double precision to compute intersection points while obtaining the surface definition in a single precision would yield a "high-precision garbage".

The influence of truncation errors can be reduced by performing an initial normalization of both the mesh and the surface definition. Both are scaled by the average cell size, resulting in the code not working with dimensions of meters, but instead in cell lengths. By bringing the lengths closer to unity, the actual precision of the numbers is increased, since the spacing between two consecutive representable floating numbers is decreased.

However, scaling will not completely eliminate truncation errors. To allow for these errors, most comparisons are done with an incorporated fudge-factor. Hence, instead of checking

```
if (R==0.0)
{
 ...
}
```

the code checks

```
if (fabs(R)<FLT_EPS)
{
 ...
}
```

The subtraction and addition operations are of highest concern, since they are most susceptible to truncation errors. For this reason, a single-precision subtraction operation was defined as:

$$\begin{aligned} \vec{r} &= \vec{a} - \vec{b} \\ m_i &= \tfrac{1}{2}\left(|a_i| + |b_i|\right) \\ r_i &= 0, \ \forall \ |r_i| \leq m_i \cdot \text{FLT\_EPS} \end{aligned} \tag{2.7}$$

where FLT-EPS is a machine specific constant indicating the smallest number which can be added to 1 and index i ranges from 1 to 3.

Similarly, the dot-product was implemented as

$$r = \vec{a} \cdot \vec{b} \tag{2.8}$$

$$r = 0, \ \forall \ |r| \leq 2\text{FLT\_EPS} \tag{2.9}$$

The factor of 2 accounts for the two additions in the dot product operation.

2.4 Location classification

2.4.1 Algorithm

VOLCAR next classifies all nodes as external or internal. The node location flag is used by finite-difference Poisson solvers to establish internal Dirichlet boundaries. Elements are also classified as internal, external or interface. The particle boundary check uses the element flag to determine which particles collided with objects.

Location classification (LC) requires that the orientation of triangle surface vectors is consistent, with the normal vectors pointing outward. The surface normal vector is computed by COLISEUM as $\vec{n} = \vec{v}_{12} \times \vec{v}_{23}$, where $\vec{v}_{ij}$ is the vector joining triangle vertices i and j. This formulation follows the standard right-hand rule for normal orientation, and is implemented by listing the triangle connectivity in the surface definition file using the right-hand node ordering.

By requiring that the normal vectors point outward, the location algorithm becomes conceptually very simple. If $\vec{x}$ is the location of the spatial point which needs to be classified, and $\vec{c}$ is the centroid of a surface triangle *visible* from $\vec{x}$, then a vector $\vec{r} = \vec{c} - \vec{x}$ can be formed, as shown in fig. 2.7. The angle between $\vec{r}$ and the triangle normal $\hat{n}$ is given by

$$cos(\alpha) = \frac{\vec{v} \cdot \hat{n}}{|\vec{v} \cdot \hat{n}|} \tag{2.10}$$

and the node is external if $\alpha \in [0 : 90^\circ)$.

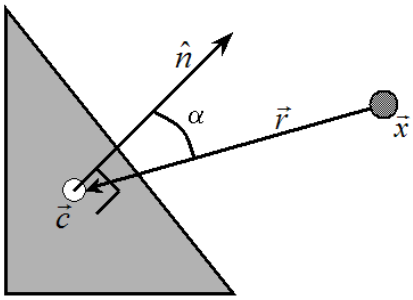


Figure 2.7: Location at a point $\vec{x}$ is determined by casting a ray to a centroid of a *visible* surface triangle. Location is determined according to the angle between the ray and the triangle normal vector.

2.4.2 Implementation and Optimization

The implementation difficulty arises from the need to find a visible surface triangle. Since only centroid information is needed, a triangle is deemed visible if its centroid is visible. For the centroid to be visible, there must not be any other triangle intersecting the ray $\vec{r}$. This calculation can become a computational nightmare if a large number of surface triangles is used. To illustrate this point, let's assume that a continuous surface is specified by n_{el} triangles. Checking the visibility of any particular surface triangle requires up to $n_{el}-1$ calls to the LTI algorithm. On average, $\frac{n_{el}}{2}$ triangles may need to be checked until a visible triangle is found. Hence, to classify the location of a *single* node, $O\left(n_{el}^2\right)$ calls to the LTI algorithm may be needed. The operation count will be smaller for the average node, however, a typical simulation contains around 10^6 nodes, and five times as many elements, which immediately shows why this algorithm cannot be used without some optimization.

Two such optimizations are employed in the code. First, the likelihood that a triangle will be visible decreases as the distance to its centroid increases. This is due to the fact that more triangles will be located in the space between the node and the centroid. The LC algorithm divides all surface triangles into "bins", according to the distance to their centroid. A visible triangle is first searched in the closest bin, if none is found, the code moves to the second bin, and so on.

Second optimization is based on the fact that all nodes internal to an object will be separated from the external nodes by the interface elements or by the flushed surface zones. Before classifying the location according to eq. 2.10, the code checks whether location could be copied from a neighboring node. This optimization is highly dependant on a successful execution of the intersection algorithm. If too many bad intersections are formed, the interface will contain holes, and the location type will "leak-out".

Element location is classified according to the node location, when possible. If all four vertices of a tetrahedron have the same location type, the tetrahedron must clearly also have the same location type. If mixed vertices exist, and the element is non-interface, as may be the case for a tetrahedron along a surface snapped to the Cartesian grid, the location is determined by applying the described LC algorithm to tetrahedron's centroid.

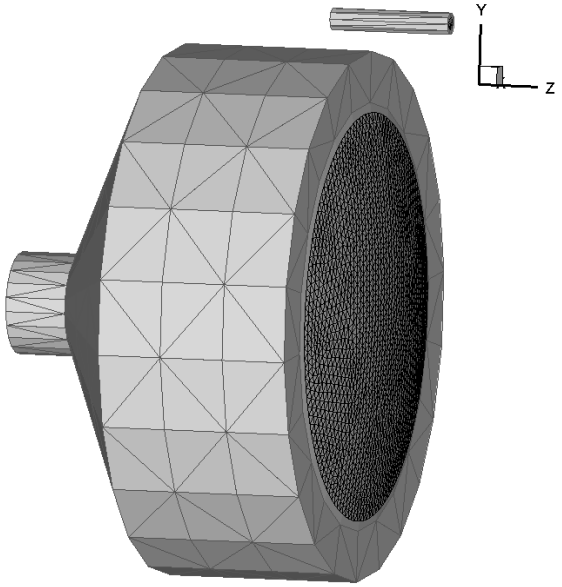


Figure 2.8: Isometric view of the 40cm NEXT thruster model

2.4.3 Consistency Check

The LC algorithm terminates by performing a consistency check. Each interface element contains four nodes, of which at least 1 must be located on the opposite side of the interface cut. The check is performed by counting the number of external nodes in each interface element. Count of zero or four indicates all internal or external nodes, respectively, which is not possible for an interface element. The interface flag on the element is replaced with the location of the nodes.

Second consistency check involves veryfing that all nodes on the same side of the interface cut have the correct location flag. This check is performed by substituting the node position into the plane equation of the interface cut. Normal vectors of the interface cut point outward, and thus all external nodes should have a non-negative RHS.

2.5 Example: a 40cm Ion Thruster

Figures 2.8 and 2.9 show the surface mesh of the NASA's 40cm NEXT ion thruster. The surface mesh was generated using MSC.Patran, using the dimensions specified on the plots. It should be noted that an actual schematic drawing including real dimensions of the thruster was not available. Instead, the dimensions were obtained by collecting data from references [13] and [14], and by direct measurements of plots 1a through 1c in [4].

The interface representation of the surface is shown in 2.10. Triangular surface cuts are shown in red. Planar (four-point) cuts are shown in green. The interface description is

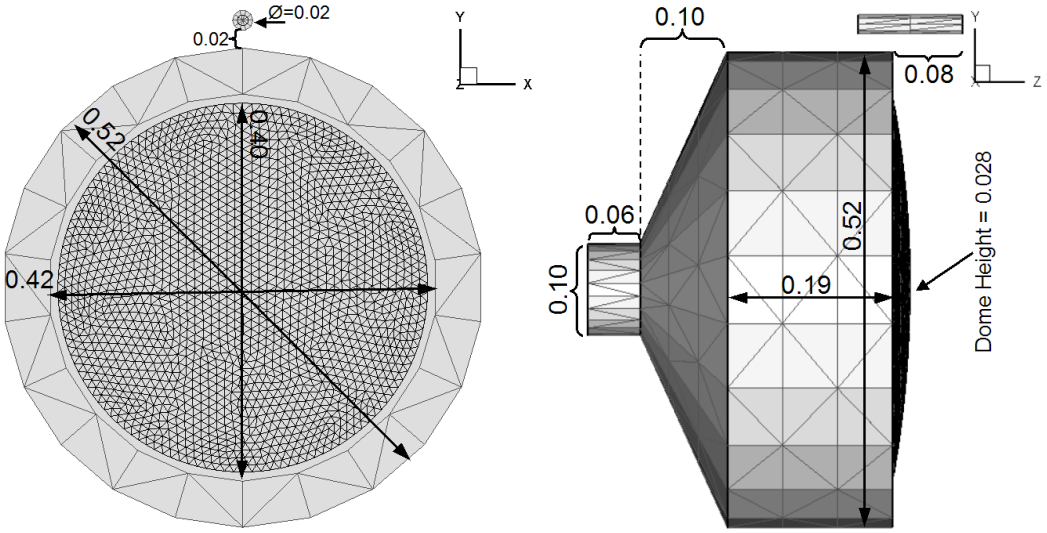


Figure 2.9: Front and side view of the 40cm NEXT thruster model, dimensions in meters

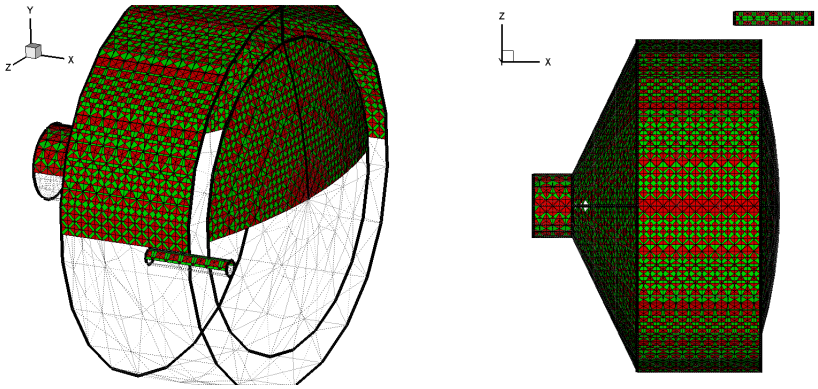


Figure 2.10: Interface representation of the 40cm thruster, illustrating that surfaces flushed with Cartesian cells do not generate interface cuts

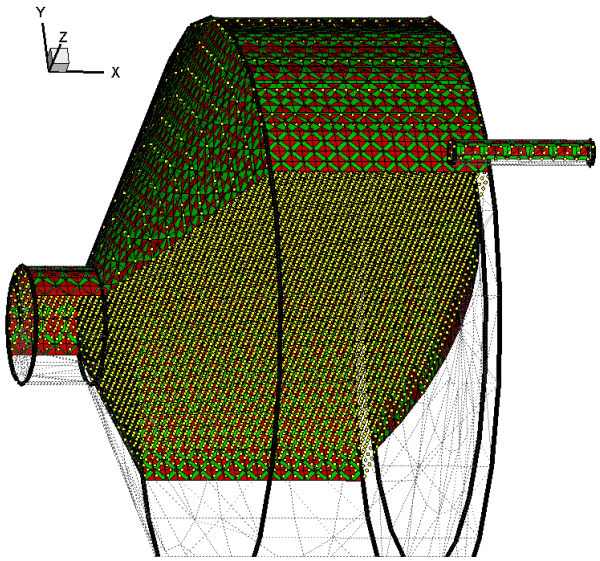


Figure 2.11: Nodes classified as internal by the LC algorithm.

capable of a very accurate representation of the surface mesh. Surface zones flushed with the Cartesian cells do not generate any interface elements. Such is the case with the flat ring surrounding the curved optics. The intersection is not perfect, as can be seen from the presence of two small holes in the back region of the ionization chamber (fig. 2.10(b)). This region was outside the simulation domain used in the neutralization studies, and thus did not affect the solution.

Figure 2.11 shows the result of the LC algorithm on the test thruster configuration. The yellow circles indicate Cartesian nodes that were determined to be internal to the thruster. Two slices of mesh contours are also included for clarity. The contours in figure 2.12 show the value of the node "object" flag. Value of zero indicates that the node is external. Green contour shows all nodes with object flag set to 1. In this case, object 1 is the thruster. Similarly, the red contour shows nodes of object 2, which is the cathode. The stair-case effect is clearly visible in the yz plot. Several misclassified nodes can also be observed, especially in the back of the thruster. Once again, this region was excluded from further simulations and thus the misclassification does not play a significant role.

2.6 Mesh refinement

Numerical simulations often impose local maximum limits on the mesh size, to either resolve variation in the surface, or to correctly model some local physical behavior. Adjusting the cell dimensions for the entire mesh according to the minimum local parameters is often too prohibitive, as an extremely large number of cells would have to be used. Not only does this increase the memory requirements for the storage of the mesh, decreasing the mesh size

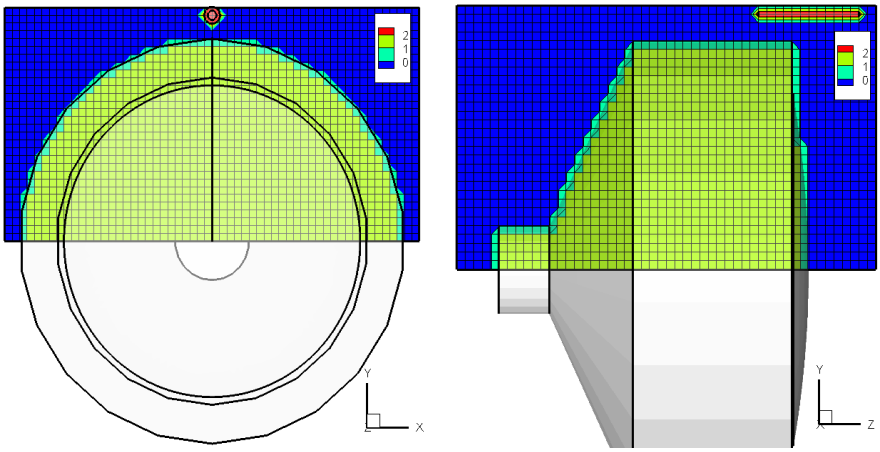


Figure 2.12: Contour plots of node-centered location flag. The flag indicates the object index. Green contour shows all nodes belonging to object 1 (thruster). Red contour indicates nodes of object 2 (cathode).

results in an increased time required to solve the Poisson equation. Unless the number of macroparticles is very large, a very fine mesh will further increase the simulation noise.

For these reasons, it is desired to resolve the mesh in high-density regions independently from the main mesh. VOLCAR contains support for mesh refinement by allowing the main mesh to contain a number of sub-meshes. The mesh-refinement is not adaptive - the user needs to specify the span of the local sub-meshes explicitly.

One application of mesh refinement is in describing surface geometry with a higher detail than is possible with the main coarse mesh. In such a case, the surface may be completely enclosed by the fine mesh and the coarse mesh is used to represent the "vacuum" region. Then, the coarse mesh can be specified to be a simple Cartesian and will not contain the overlaid tetrahedral mesh.

Mesh refinement could be applied to the thruster mesh by specifying a finer mesh around the cathode. An example of such a mesh is shown in figure 2.13. However, as is described in chapter 3, mesh refinement was not used in the neutralization study due to a lack of a good potential solver capable of retaining continuity of solution along mesh boundaries.

2.7 Data Output

Simulation results are saved by calling VOLCAR's mesh-save functions. Data is outputted at the node locations in either Tecplot or capVTE formats. VOLCAR supports both 3D and 2D output, however the 2D output is limited to the Cartesian planes. The variables to output are specified by the user through the COLISEUM input file. The variables are evaluated prior to the output and the memory for all non-PIC critical variables is de-allocated after the file save.

Besides simulation mesh, VOLCAR also contains functionality to output the interface

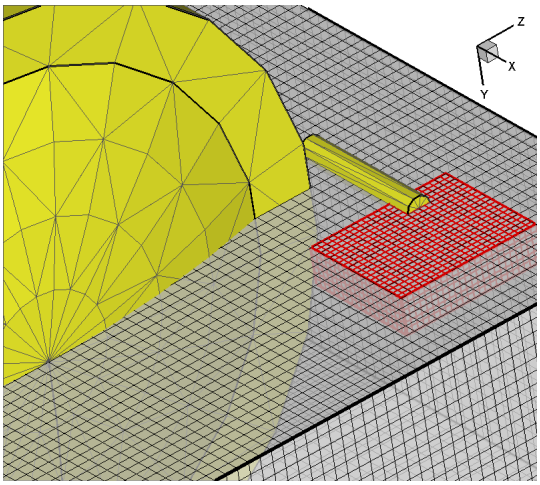


Figure 2.13: Refined mesh could be used to resolve high-density region near the cathode exit

mesh. This output contains the cut planes along with the internally-flagged nodes and elements. The cut normal vectors are also saved. This functionality is primarily used to verify the mesh generation.

Visualization of results on the original surface mesh is possible by extrapolating Cartesian cell data onto the surface nodes. This extrapolation is performed automatically at the end of each simulation. Simulation results can also be mirrored along user specified face. This feature is used to restore the full domain for visualization of configurations containing one or more planes of symmetry.

Chapter 3

DRACO ES-PIC Algorithm

3.1 Introduction

The first version of DRACO was a direct C translation of the F77 PLUME code developed by Dr. Joseph Wang. Since the PLUME code used analytically defined object geometries, DRACO was soon modified to work with an arbitrary surface definition using the VOLCAR interface mesh. This process led to a completely new surface interaction checking algorithm.

A large amount of time was spent on interfacing of DRACO with COLISEUM. The original PLUME simulation was controlled by a single large input file, containing a sequence of numbers, which were read by the initialization functions. Performing the initialization in a different order, or removing certain initialization functions required modification of the input file. DRACO simulation is instead controlled by commands specified in the `coliseum.in` script file. Material properties, and object parameters are listed independently in the material and component files.

DRACO can read input in either SI or non-dimensional units. While the actual simulation is still performed using non-dimensional quantities, the normalization process is done internally by the code. Results are automatically un-normalized at the end of the simulation. The PLUME results were saved in the normalized quantities, but the normalization values were often discarded. This made it difficult to compare old results to experimental data. The internal normalization also assures that all values are normalized consistently.

Another big change involved the internal data management. All memory in the PLUME code was statically pre-allocated. This was primary a limitation of the programming language, since F77 does not support dynamic arrays. Still, the reliance on static memory required code re-compilation prior to each simulation, limiting the transformation of the code from the research to the production environment. Data organization is discussed in a greater detail in the following section.

DRACO was also made more flexible by allowing the user to choose the modeled physics without rebuilding the code. The user can select from a variety of potential solvers, which can operate in both the linear (fully-kinetic) and non-linear (Boltzmann electrons) modes. The collision module can be activated as desired. All collision pair information is specified in the material interaction input file.

An attempt was made to support mesh refinement. Although majority of the code has been implemented, a satisfactory Poisson solver is still lacking. The developed solver uses a multi-domain formulation. Although all domains converge individually, continuation across the mesh boundaries is generally not guaranteed.

Finally, a large number of new diagnostics was added. The PLUME code was capable of outputting the potential, charge density, electric field and charge density components. The format of the output file was also not compatible with the two visualization tools used at VT's CapLAB: Tecplot (by Amtec Engineering) or capVTE [15]. The output was hence modified to support both of these formats. The list of available diagnostics was extended to include the velocity components, number densities, polytropic and Maxwellian temperatures, and surface flux. Calculation of these properties is described later in this chapter.

3.2 Numerical Implementation

3.2.1 Normalization

Due to the finite form of computer representation of floating point numbers, numerical errors are possible when mathematical operations are performed on operands with a large difference in magnitude. These errors can be minimized by *normalizing* all physical quantities. Normalization replaces the physical values (in SI units), with non-dimensional quantities with values on the order of one.

The normalization is performed automatically using user specified reference values. It is self-consistent, and a linear relationship exists between non-dimensional units and their physical counterparts. Simulation results are automatically un-normalized at the end of the run.

Particle masses are normalized by the mass of a reference particle:

$$\hat{m} = \frac{m}{m_0}$$

Charge is normalized by the elementary charge, e:

$$\hat{q} = \frac{q}{e}$$

Length is normalized by the characteristic plasma dimension, the Debye length:

$$\hat{x} = \frac{x}{\lambda_D}$$

$$\lambda_D = \sqrt{\frac{\varepsilon_0 k T e_0}{n_0 q^2}}$$

Time is normalized by the plasma frequency:

$$\hat{t} = t\omega_p$$

$$\omega_p = \sqrt{\frac{n_0 q^2}{\varepsilon_0 m_0}}$$

Velocity is normalized by the sonic speed, which is the result of combining the length and time normalizations:

$$\hat{v} = \frac{v}{\lambda_d \omega_p} = \frac{v}{\sqrt{\frac{kTe}{m_0}}} = \frac{v}{C_s}$$

Charge density is normalized by a reference charge density:

$$\hat{\rho} = \frac{\rho}{e n_0}$$

The unit of potential, volts, can be rewritten using basic SI units as $1\text{V} = \frac{\text{kg}\cdot\text{m}^2}{\text{coul}\cdot\text{s}^2}$, leading to the following normalization:

$$\hat{\phi} = \frac{e}{m_0 \lambda_D^2 \omega_p^2}\phi = \frac{e}{kTe}\phi$$

The factor ε_0 disappears from the normalized Poisson equation, due to the second spatial derivative of ϕ:

$$\begin{aligned}
\nabla^2\phi &= -\frac{\rho}{\varepsilon_0} \\
\nabla^2\hat{\phi} &= \frac{e}{m_0\omega_p^2}\nabla^2\phi = \frac{\varepsilon_0}{e n_0}\nabla^2\phi \\
\frac{e\cdot n0}{\varepsilon_0}\nabla^2\hat{\phi} &= -e n_0\frac{\hat{\rho}}{\varepsilon_0} \\
\nabla^2\hat{\phi} &= -\hat{\rho}
\end{aligned} \tag{3.1}$$

The electric field, which is given in V/m, is normalized as:

$$\hat{E} = \frac{e}{m_0 \lambda_D^2 \omega_p^2}\lambda_D E = \frac{\varepsilon_0}{\lambda_D n_0 e}E$$

The background magnetic field is normalized using:

$$\hat{B} = \frac{q}{m_0 \omega_p}B$$

3.2.2 Data Structure

DRACO stores all volumetric plasma parameters on the nodes of the interface Cartesian mesh created by VOLCAR. The data is stored by using a list of dynamically allocated *properties*. Each property holds a single scalar variable, such as potential, charge density, or a single component of the electric field.

The data structure of each property contains some basic information about the variable, such as its name, units, and the associate particle specie (if needed). Also saved are pointers to three functions: *Init*, *Eval* and *FreeData*. The actual data is saved in a one-dimensional dynamically allocated array.

This formulation allows for an *on-demand* memory allocation and evaluation. Some properties, such as potential, charge density or the electric field, are needed during the actual PIC process. However, majority of the available grid properties are simple analysis tools. As discussed later, DRACO is capable of computing the Maxwellian temperature, or the velocity vectors. While these variables help in comprehension of the results, they are not necessary to obtain the solution. Thus, these properties are evaluated only as requested by the user, and the allocated memory is freed after the file output.

The property data are saved using double precision. Since COLISEUM reads most user specified quantities with only single precision, the far significant digits can become corrupted by mixing of single and double precisions. However, the double precision format was retained to minimize the possibility of truncation due to numerical round-off. Since the mantissa of the double precision float contains more bits, it is less susceptible to truncation.

Although the data represents a 3D volume, it is saved internally in a one dimensional format. This method was chosen for both the convenience and as an attempt to speed-up the code. The C language does not contain a ready support for multi-dimensional dynamic arrays. However, a 3D mesh can be represented using a one-dimensional i-j-k indexing as:

$$N3D[i][j][k] = N1D[k * nx * ny + j * nx + i]$$

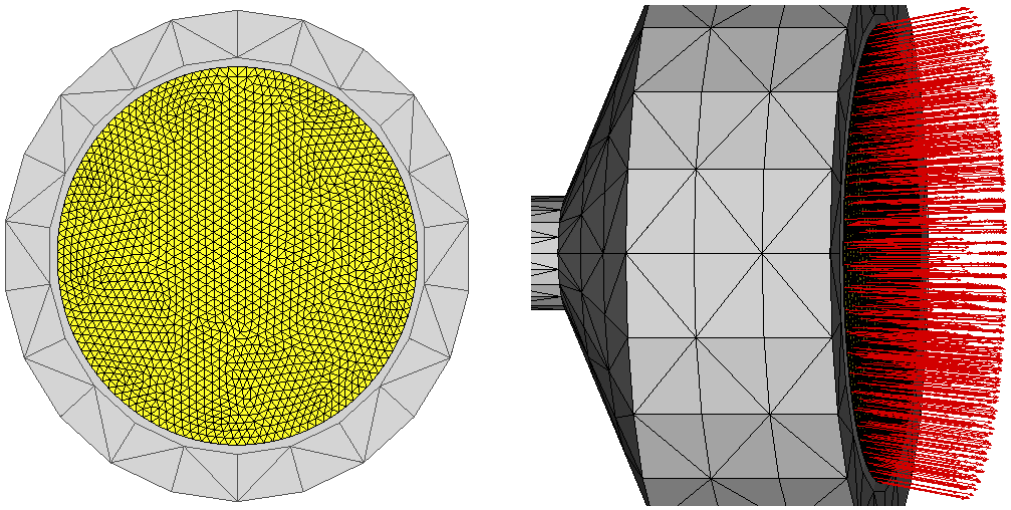
where nx and ny hold the number of nodes in the x and y directions. Accessing node neighbors is very straightforward:

$$\begin{aligned} N3D[i+1][j][k] &= N1D[n+1] \\ N3D[i][j+1][k] &= N1D[n+nx] \\ N3D[i][j][k+1] &= N1D[n+nx*ny] \end{aligned}$$

where $n = k * nx * ny + j * nx + i$.

This method was also chosen in an attempt to optimize DRACO. However, as subsequent tests showed, memory access using the three-dimensional array can actually be slightly faster, depending on the used architecture and compiler. This may seem obvious from the above formulation, since the slow-down could be attributed to the additional multiplication steps needed to generate the 1D index. However, the 3D array seemed to be faster even for a simple matrix-scalar multiplication, in which the 1D array was accessed using a single `for (n=0;n<nodes;n++)` loop. The 3D process used separate loops for the three indexes. The speed-up seemed to be heavily dependant on the compiler and platform used. Additional studies need to performed here, possibly leading to an overhaul of DRACO's internal data structure.

The particles are stored in a single dynamically allocated array, with each array member containing information on an individual particle. Among saved parameters are the particle's position, velocity, specie type and source. Retaining the source index allows the particles to be filtered during their output, allowing the user to, for instance, graphically visualize mixing of two ion beams. The particle position and velocity are stored using single precision floats. Presence of "holes" due to removed particles is avoided by moving the data for the last particle into the newly created hole. The particle array is thus contiguous, and memory jumps are avoided during particle access.



(a) source elements (b) source normal vectors

Figure 3.1: Although the yellow-shaded thruster optics is just a single component, it consists of a high-number of surface triangles. Associating a source type with the optics component results in each source triangle producing particles. Particles are generally injected relative to the element's normal vector, shown in (b).

3.3 Particle sampling

Particles are introduced into the simulation domain by *sampling* COLISEUM's sources. Sources are specified through the `coliseum.in` file using the `source_specify` command:

```
source_specify component type specie [source arguments]
```

where the *component* specifies the surface component (collection of surface triangles) which this source will associate itself to. The component is typically the front face of a thruster. The rules for particle injection are linked to the used source *type*. The material which will be injected by this source is specified by the *specie* tag.

Many various source types exist in COLISEUM. Several more were also added during the many stages of this research to model more specialized cases. The most versatile source type is MAXSTREAM, which emmits particles following the drifting Maxwellian velocity distribution. Each particle is born at a random position on the source triangle and the Maxwellian velocity shift is applied relative to the normal vector of the source element.

The *source arguments* needed by MAXSTREAM are $\dot{m}$ (kg/s), *temp* (K), v_{mean} (m/s) and n_{pel}. The n_{pel} quantity is used in the projection integration step, which is described in a later section. For instance, injection of 1.2 A of singly charged Xenon ions (1.634×10^{-6} kg/s) at 0.1eV (1160.4K) and flowing with a mean velocity of 34,400 m/s can be specified using:

```
source_specify optics MAXSTREAM XE+ 1.634e-6 1160.4 34400 20
```

The source type is specified on the component level. However, the actual particle injection is done on the surface triangle level. Each surface triangle belonging to the source component becomes part of the source. The distinction can be seen in figure 3.1. The yellow region is a single component, called *optics*. However, the component is described with a very fine surface mesh. Since the particles are injected relative to the normal vector of each surface triangle, shown in fig. 3.1(b), an actual curvature of the surface mesh will naturally introduce a beam divergence.

Each source surface element retains information on its type, associated surface element, injection specie and the local mass flow rate. The $\dot{m}$ produced by each source is scaled by the area of the associated surface element such that each source triangle is producing the same particle flux, $\Gamma_0 = \Gamma_s = \dot{m}_s A_s$.

DRACO adds particles by looping through all source elements. For each element, it computes the *real* number of macro-particles from the local $\dot{m}$ using

$$np_r = \frac{\dot{m}\Delta t}{m} \tag{3.2}$$

where m is the mass of the particle in kilograms. However, the particle masses are specified in COLISEUM using the atomic mass units. Particle mass can be converted easily to kilograms by dividing by the Avogadro's number:

$$m(kg) = \frac{m(amu)}{Na} = \frac{m(amu)}{6.022 \times 10^{26} amu/kg} \tag{3.3}$$

Generally, np_r will not be integral. This is a problem, since a fractional particle cannot be loaded. The real np_r is transformed to an integer by adding a random number in the range [0:1] and truncating the decimal part. This method, given a statically sufficient number of sources and time steps, is equivalent to loading of the extra fractional particles.

As will be shown in the subsequent section, accurate integration of the equations of motion requires that the particle positions be initially advanced by a half of Δt. This process is however not explicitly defined in DRACO. Instead, the particles are advanced by a random fraction of the time step, where the fraction ranges from 0 to 1. Since the particles are being sampled at discreet time intervals, moving the particles by a constant fraction results in the particles being loaded in *sheets*. This is not a good approximation of a continuous plasma source. A large spacing between the sheets, perhaps due to a large time step, can even induce nonphysical plasma waves. The advance of particles by a random fraction of Δt better approximates the continuous firing of a thruster at the added cost of introducing errors into the equation of motion. However, given a large number of particles, the errors can be expected to cancel out, since an equal number of particles will be located ahead and behind of their desired initial positions.

3.4 Particle Motion

3.4.1 Equation of Motion

The equations of motion for a charged particle in an EM field, in the absence of gravitational, pressure or thermal forces are:

$$\frac{d\vec{v}}{dt} = \frac{q}{m}(\vec{E} + \vec{v} \times \vec{B}) \tag{3.4}$$

$$\frac{d\vec{x}}{dt} = \vec{v} \tag{3.5}$$

Numerical formulation of 3.4 is straightforward:

$$\frac{\vec{v}^{k+1} - \vec{v}^{k}}{\Delta t} = \frac{q}{m}(\vec{E} + \vec{v}^{k+0.5} \times \vec{B}) \tag{3.6}$$

$$\frac{\vec{x}^{k+2} - \vec{x}^{k+1}}{\Delta t} = \vec{v}^{k+1} \tag{3.7}$$

This formulation leads to the well known problem of PIC modeling: particle positions and velocities are not known at the same time. This problem could be perhaps avoided by storing several time steps worth of particle velocity and position data. However, to retain numerical simplicity, the particle position's are usually initially advanced by $0.5\Delta t$, leading to:

$$\frac{\vec{v}^{k+0.5} - \vec{v}^{k-0.5}}{\Delta t} = \frac{q}{m}(\vec{E} + \vec{v}^{k} \times \vec{B}) \tag{3.8}$$

$$\frac{\vec{x}^{k+1} - \vec{x}^{k}}{\Delta t} = \vec{v}^{k+0.5} \tag{3.9}$$

where the indices were adjusted to reflect that the time step k is desired to coincide with the times of known particle positions.

The *leap-frog method*, as the previously described procedure is generally known, is not accurate for large Δt. This error arises from the decoupling of the times at which the two integrands, $\vec{v}$ and $\vec{x}$, are known. However, the error decreases as $\Delta t \to 0$. The generally accepted restriction on Δt is the CFL conditions [16]

$$(\Delta t)_{CFL} = \frac{u}{\Delta x} + \frac{v}{\Delta y} + \frac{w}{\Delta z} \tag{3.10}$$

which simplifies for a uniform discretization $\Delta x = \Delta y = \Delta z = \Delta h$ to

$$(\Delta t)_{CFL} = \frac{|\vec{v}|}{3\Delta h} \tag{3.11}$$

3.4.2 Velocity Update and Charge Density Deposit

Updating velocity requires that the value of the electric and magnetic field is known at the position of the particle. Since these quantities are stored only at the grid nodes, their values must be first extrapolated to the particle's location. DRACO uses volumetric weighing, which is schematically described in figure 3.2.

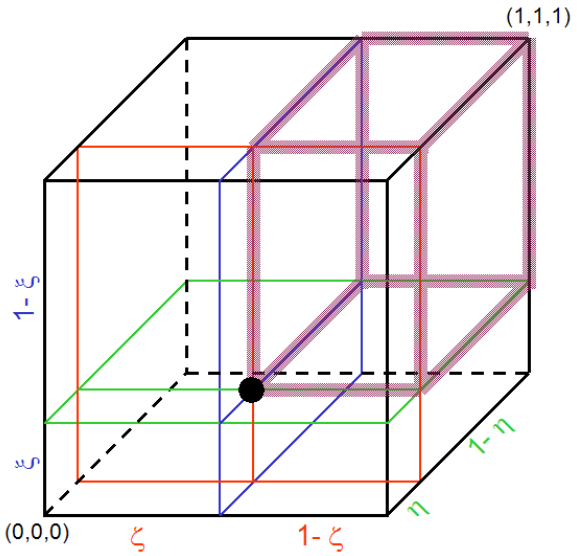


Figure 3.2: Particle weighing to the grid. Shaded region indicates the volumetric fraction deposited onto the origin.

Table 3.1: Weights associated with the eight nodes of a Cartesian cell.

Index	Node	Weight
1	(0,0,0)	$(1-\zeta)(1-\eta)(1-\xi)$
2	(1,0,0)	$\zeta(1-\eta)(1-\xi)$
3	(0,1,0)	$(1-\zeta)\eta(1-\xi)$
4	(0,0,1)	$(1-\zeta)(1-\eta)\xi$
5	(0,1,1)	$(1-\zeta)\eta\xi$
6	(1,0,1)	$\zeta(1-\eta)\xi$
7	(1,1,0)	$\zeta\eta(1-\xi)$
8	(1,1,1)	$\zeta\eta\xi$

The field is computed by adding (or *gathering*) the contributions of the 8 nodes, multiplied by an appropriate weighing factor. The factor equals to the volume of the diagonal block within a cube of volume 1. The factors are computed by translating the particle position within the cell to cell coordinates, ζ, η and ξ. Each cell coordinate varies from 0 to 1, and indicates the fraction of dx, dy or dz, respectively, which needs to be added to the cell origin to arrive at the particle position. By using the convention $\vec{\psi} =< \zeta, \eta, \xi >$, the cell coordinates are computed using:

$$\vec{\psi} = Dec\left((\vec{x} - \vec{x_0}) \cdot (1/\vec{dh})\right) \tag{3.12}$$

where $Dec()$ is a function returning the decimal portion of a real number and $\vec{dh} =< dx, dy, dz >$. The volume fractions associated with the eight cell nodes are summarized in table 3.4.2. The value of the the electric and magnetic fields at the particle position is thus given by:

$$\vec{E}(\vec{\psi}) = \sum_{i=1}^{8} W_i \cdot \vec{E}_i \tag{3.13}$$

$$\vec{B}(\vec{\psi}) = \sum_{i=1}^{8} W_i \cdot \vec{B}_i \tag{3.14}$$

Velocity update is straightforward if the background magnetic field is not loaded. Then,

$$\frac{d\vec{v}}{dt} = \frac{q}{m}\vec{E} \tag{3.15}$$

The rotational force due to the magnetic field is a function of velocity. DRACO implements the method of Boris, as described by Birdsall [6]. The velocity is first updated by adding a half of the acceleration due to the electric field. This intermediate velocity is used to perform two subsequent half rotations due to the magnetic field. Finally, the remaining half of the acceleration is added. The process is described mathematically as:

$$\vec{v}^{-} = \vec{v}^{k-1/2} + h\vec{E} \tag{3.16}$$

$$\vec{v}^{'} = \vec{v}^{-} + \vec{v}^{-} \times h\vec{B} \tag{3.17}$$

$$\vec{v}^{+} = \vec{v}^{-} + \frac{2d}{1 + |h\vec{B}|^2}\left(\vec{v}^{'} \times \vec{B}\right) \tag{3.18}$$

$$\vec{v}^{k+1/2} = \vec{v}^{+} + h\vec{E} \tag{3.19}$$

where

$$h = \frac{q}{m}\frac{\Delta t}{2} \tag{3.20}$$

Movement of particles to new positions leads to a new distribution of charge density. The charge density, ρ, is computed by weighing (or *scattering*) particles to the grid. The process is similar to the gather of electric field, but now the particle charge, multiplied by

the appropriate weight factor, is added to the eight surrounding grid nodes:

$$\begin{aligned} \rho(0,0,0) &+= W_1 \cdot \rho_m \\ \rho(1,0,0) &+= W_2 \cdot \rho_m \\ &\cdots \\ \rho(1,1,1) &+= W_8 \cdot \rho_m \end{aligned}$$

where ρ_m is the charge density contribution of the macroparticle. It is given by

$$\rho_m = \frac{sw \cdot q}{V_{cell}} \tag{3.21}$$

3.4.3 Particle Push

New velocity is used to update the particle position as

$$\vec{x}^{n+1} = \vec{x}^n + \Delta t \cdot \vec{v}^{n+1/2} \tag{3.22}$$

Computation of the destination cell is not necessary, since the particles are moved in a Cartesian domain. The cell indexes are obtained during the scatter/gather operations by the inversion of the Cartesian equation,

$$\begin{aligned} i &= int\left((x - x0)/dx\right) && (3.23)\\ j &= int\left((y - z0)/dy\right) && (3.24)\\ k &= int\left((z - z0)/dz\right) && (3.25)\\ & && (3.26) \end{aligned}$$

where i, j, k is the node index of the origin of the cell in which the particle resides and $int()$ is a function which truncates the decimal part of a real number.

3.4.4 Particle external boundary check

The particles which cross the domain boundaries need to be either removed from the particle array, or returned to the simulation domain. DRACO supports four type of external particle boundaries: *open*, *reflective*, *periodic* and *energy*. The check is performed by looping over all particles, and isolating particles with a position component outside the allowed range. An appropriate action is then taken according to the specified boundary condition for the conflicting dimension.

The open boundary is the simplest. Particles which pass through an open face are simply removed from the simulation. The particles are stored in a continuous memory block. Presence of holes, due to removed particles, is avoided by replacing the removed particle, with the last entry in the array. The position of the newly moved particle must be checked before moving to the next entry in the particle list.

Reflective boundary condition is applied along symmetric faces. Computational time for problems containing one or two planes of symmetry can be reduced by modeling only a

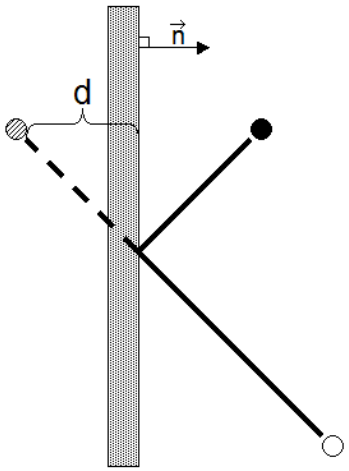


Figure 3.3: Particle passing through a reflective external boundary is elastically reflected back to the domain. The normal velocity component is reversed, and the position is shifted by 2d along the normal direction.

portion of the domain. Full domain can be restored for visualization by mirroring during post-processing.

Particles passing through reflective boundaries are elastically reflected back to the domain. The reflection is performed by reversing the velocity component perpendicular to the wall, and by shifting the position along the normal direction by the twice the distance behind the wall. This process is illustrated in figure 3.3, with the open circle denoting the pre-push position of the particle. The slashed circle shows the position after the particle push. The solid circle is the desired position, obtained by the elastic collision. Each component is checked individually; completion of the check for all three dimensions results in the particle being returned back to the domain, even if the particle left near the corner of the domain. Mathematically, the process is given by

$$\vec{v}_{new} = \vec{v}_{old} - 2(\vec{v}_{old} \cdot \hat{n}) \tag{3.27}$$

$$\vec{x}_{new} = \vec{x}_{old} + 2d(\vec{x}_{old} \cdot \hat{n}) \tag{3.28}$$

The velocity of particles crossing periodic boundaries is not affected, but their position is adjusted by the domain length. For instance, periodicity in x requires that

$$x_{new} = x_{old} - nx * dx \quad \forall\ x_{old} \in \Gamma_{XMAX,P} \tag{3.29}$$

$$x_{new} = x_{old} + nx * dx \quad \forall\ x_{old} \in \Gamma_{XMIN,P} \tag{3.30}$$

The last condition, the energy boundary condition, was developed as a response to a numerically induced instability in Full-PIC simulations. It is described in a greater detail in the 4^{th} chapter of this work.

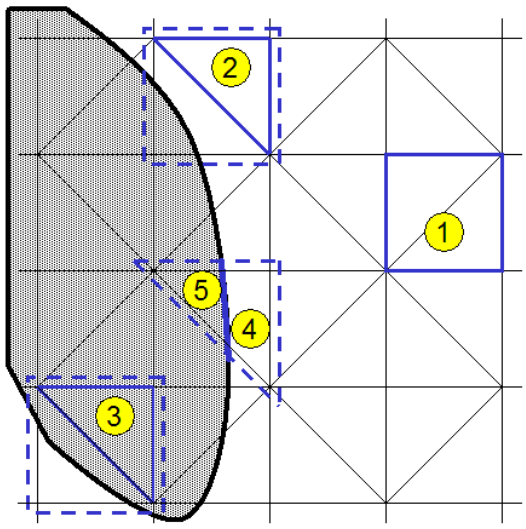


Figure 3.4: Five possible particle locations that need to be checked during surface interaction check. The case of a particle located inside a completely internal Cartesian cell can occur only if the requirenment that particles travels less than a cell per time step is violated, and is therefore not included.

3.4.5 Surface interactions and flux

Particle motion also causes collisions of particles with the surfaces. The check for surface interactions had to be fast, otherwise all benefits of the interface model would be lost. Intersecting the movement ray of all particles with all surfaces using the LTI algorithm is the most accurate method of performing the check. However, it is also very slow.

The algorithm instead assumes that all objects are larger than a cell, and that particles are not allowed to travel more than a cell length per time step. Then, any particle which collided with an object during the push will terminate inside the object. An exception is possible near edges and corners; currently the code is not capable of resolving this type of interaction.

Then, performing the surface interaction check comes down to classifying the particle's post-push position as *external* (located in the simulation vacuum) or *internal* (located inside some object). Figure 3.4 shows a 2D version of the possible locations of a particle. Majority of particles will be located within a completely Cartesian cell, which is shown by particle "1". The index of the Cartesian cell containing the particle is computed by inverting the particle's position following eq. 3.23. The location flag of the five tetrahedrons forming the cell is then analyzed. If all five tets are classified as external, the particle resides in a completely external cell. Similarly, presence of five internal tets indicates that the particle is in a completely internal cell, and is flagged for future processing. The last case will occur only if a non-interface boundary exists between the object and the simulation domain, as is

the case with cell-flushed geometry faces.

As a particle nears a surface, it will cross into an interface Cartesian cell. This condition is shown by particle "2". The cell in which it resides contains tets of mixed location type. Particle location can no longer be determined from the Cartesian cell, and the tetrahedron containing the particle must be uniquely identified. The approach taken is to loop over the five possible choices, and check whether the particle (a point) is located within the tetrahedron. The check uses the volumetric formulation, in which the tet is divided into four sub-tets, each formed by joining three original vertices with the particle position. The particle is located inside the tetrahedron only if the combined volume of the four sub-tets equals the volume of the original tetrahedron. In other words, a point is located in a tetrahedron if

$$\left| V_{tet}\left(1,2,3,4\right) - \sum_{i=1}^{4} V_{tet}\left(p, v_{i1}, v_{i2}, v_{i3}\right) \right| \leq \epsilon \tag{3.31}$$

where a volume of a tetrahedron, V_{tet}, given by vertices $\left(x_i, y_i, z_i\right)\big|_1^4$ is

$$V_{tet}\left(1,2,3,4\right) = \frac{1}{6} \begin{vmatrix} \left(x_2 - x_1\right) & \left(y_2 - y_1\right) & \left(z_2 - z_1\right) \\ \left(x_3 - x_1\right) & \left(y_3 - y_1\right) & \left(z_3 - z_1\right) \\ \left(x_4 - x_1\right) & \left(y_4 - y_1\right) & \left(z_4 - z_1\right) \end{vmatrix} \tag{3.32}$$

and $\left(x_i, y_i, z_i\right)_p$ denotes the position of the particle. The tolerance term, ϵ, is used to account for the inexactness of computer arithmetic. The three vertices of the original tetrahedron used to construct a sub-tet are given by the circular indexes v_i[1].

Three actions are possible according to the location type of the owning tetrahedron. If the tetrahedron is external, no collision is assumed to have happened, which is the case for particle "2". On the other hand, particle "3" is located in an internal tetrahedron, and the particle is flagged for later processing.

The final possibility is that the particle resides in an interface element. An additional test must be performed to determine which side of the interface cut the particle is on. VOLCAR stores the normal vector for each interface cut, and the normals are oriented outwards. Thus, the last check requires computing the angle between the interface normal and the ray to the cut's centroid, in a manner similar used by VOLCAR to classify node locations. Particles located on the internal side of the cut are flagged. The two possibilities are shown by particles "4" and "5".

DRACO also contains basic support for thin plates. Thin plates are special components, for which the actual line-triangle intersection is performed. The intersection is performed only for the particles located within the component's bounding box, but nevertheless, the process is slow, especially if a large number of surface triangles describes the component. Hence, it is important to use the coarsest surface mesh capable of resolving the curvature of the thin plate. Only two surface triangles should be used to describe a flat plate.

If the particle impacted a surface, several physical responses are possible according to the particle's type, its impact velocity and the material property on the surface. A neutral particle can either reflect or stick. A charged particle will be neutralized, and, similarly, can either impinge to the surface or bounce back. If the impact energy is high enough, the impact can result in some surface atoms being sputtered off.

[1] $v_1 = (1,2,3)$, $v_2 = (2,3,1), \cdots$

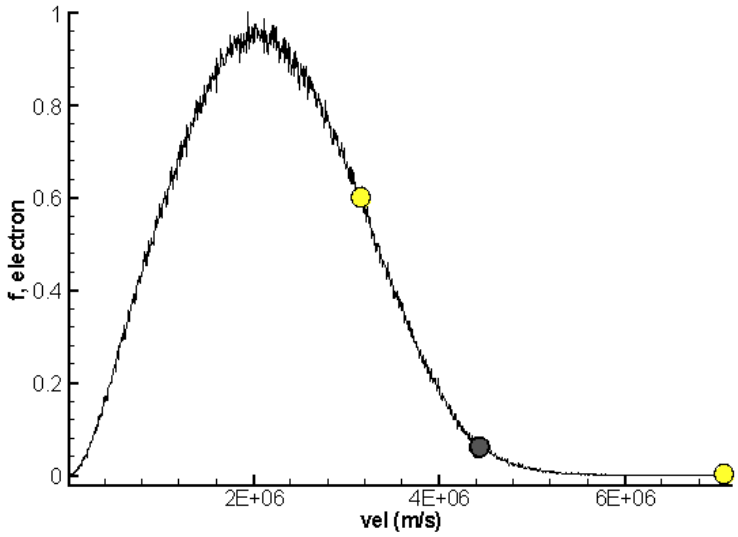


Figure 3.5: Two yellow circles indicate the mean and the maximum velocities of the reference specie. The average of these two velocities, shown by the gray circle, is used to automatically adjust the time step.

The surface physics implemented in the current version of DRACO is fairly limited. Generally, all particles hitting the surface are absorbed by the surface. If the simulation does not track neutrals, this removal is analogous to the neutralization of an ion, followed by the reflection of a neutral. A simple reflection algorithm has been implemented, but it is not yet fully integrated with the rest of the code. The neutralization study, described later, was performed with a conductor model of the surface. All absorbed electrons were re-emitted at the next time step from their original source.

Surface flux tracking is possible by weighing the density and velocity of the removed particles, since

$$\Gamma_{surf} = \frac{\vec{v}}{n\Delta t} \tag{3.33}$$

Flux should be saved directly onto the surface nodes, but since no direct relationship exists between the surface and grid nodes during a DRACO simulation, the flux is instead weighed onto the grid nodes. Time averaged values are interpolated from the grid onto the surface nodes at the end of the simulation.

3.4.6 Variable Time Step

An excessively large time step can result in non-physical simulation results. For instance, a high particle density crossing multiple cells per time step can induce numerical plasma oscillations. The surface interaction algorithm also assumes that the particle motion is limited to less than a cell per time step. Otherwise, particles may pass through objects.

However, predicting the optimal time step before the simulation starts is not an easy task, especially if highly mobile species, such as electrons, are involved. The behavior of these species is driven by the distribution of the more massive species, and they respond to any disturbances very rapidly. The results presented in the later chapter indicate that the electrons flow from the cathode into the ion beam, where they become trapped. However, instead of slowing down, the electrons tend to bounce around the beam. The speed of these trapped electrons is driven by the relative neutrality of the beam; large non-neutrality will result in highly excited electrons.

One possibility is to start the simulation with an artficially small time step. However, some time is needed to establish the ion beam. The electron motion during this period may not be as rapid, and a large number of time steps will be performed with electrons moving just a tiny fraction of the cell. The excessively small initial time step results in a large increase of the wall time of the simulation.

A better approach is adjust the simulation time step in response to the plasma behavior. DRACO contains such a feature. The user specifies the desired cell fraction traveled by the reference (most mobile) specie, and the code adjusts the time step according to the motion during the previous step. As figure 3.5 shows, the electron distribution often displays a long high velocity tail, containing a statistically insignificant number of particles. Adjusting the time step according to the velocity at the end of the distribution tail is not desired, since these particle's can be attributed to simulation noise. Furthermore, using this high velocity would mean that the bulk of the particles were traveling distance much smaller than what was specified by the user.

Ideally, the code should compute an offset from the mean velocity equal to several σ. Unfortunately, computating the distribution function at each time step is not feasible from the performance standpoint. Instead, the code approximates this offset by first computing the average velocity of the particles of the reference specie. This value is shown by the first yellow circle in 3.5. The second yellow circle indicates the maximum velocity in the distribution tail. The velocity used to adjust the time step is computed as $0.5 * (v_{mean} + v_{max})$. It is shown by the gray circle. This simple approach sufficiently captures majority of the particles. By setting the desired distance traveled to below 75% of the cell size, almost all particles can be assured to travel less than a cell per time step.

3.5 Potential Solvers

3.5.1 Boundary Conditions

Once the charge density is known, the potential can be computed by solving the Poisson equation:

$$\nabla^2 \phi = -\frac{\rho}{\varepsilon_0} \tag{3.34}$$

Unique solution of the elliptic Poisson equation requires specification of the field boundary conditions on all external and internal boundaries. The external boundaries are formed by the six walls of the Cartesian simulation domain. The surface of the solid objects placed into the simulation domain forms the internal boundaries.

The external boundary conditions are divided into Dirichlet and Neumann. Specifying a Dirichlet B.C. along a wall requires providing the value of potential, ϕ, along that boundary.

The Neumann boundary conditions specifies the value of the normal derivate, $\partial\phi/\partial n$. Since $\vec{E} = -\nabla\phi$, this value corresponds to the component of the electric field normal to the boundary. DRACO currently implements only the "Neumann zero" condition, $\partial\phi/\partial n = 0$.

From a physical perspective, the Dirichlet boundary states that the wall potential is fixed. Generally, this condition is used to model the flow of plasma towards a charged plate. The "Neumann zero" condition on the other hand specifies that the electric field is constant. Since $\varepsilon_0\nabla\vec{E} = \rho$, this condition indicates that the boundary is located beyond the influence of the beam. The Neumann condition is used to model plasma expansion in the space vacuum. However, since $\rho(p-1) = \rho(p+1)$, where $i = p$ is the position of a symmetric plate, the Neumann condition also exists on symmetric faces.

3.5.2 Quasi-Neutral Approximation

Performing a fully-kinetic plasma modeling, in which both the ions and electrons are tracked, can lead to numerical difficulties, as is demonstrated later in this work. Furthermore, the electrons move very rapidly, requiring very small time steps. From the viewpoint of the ions, the electrons respond almost instantaneously to any fluctuations in the charge density. The kinetic description can thus be replaced with a set of fluid equations.

DRACO does not solve the complete set of the electron fluid equations, instead it uses the *Boltzmann relationship* to approximate the electron density. Following the formulation in Chen [17], the relationship can be derived from the electron momentum equation

$$m_e n_e \left[\frac{\partial \vec{v}_e}{\partial t} + (\vec{v}_e \cdot \nabla)\vec{v}_e\right] = e n_e \left(\vec{E} + \vec{v}_e \times \vec{B}\right) - \nabla p_e \tag{3.35}$$

where $p = nkT_e$ is the pressure term, derived by assuming the plasma is Maxwellian and isothermal. Then, if the $\vec{v}_e \times \vec{B}$ is negligible (as is the case in motion parallel to $\vec{B}$), the above equation can be simplifies to:

$$m_e n_e \left[\frac{\partial \vec{v}_e}{\partial t} + (\vec{v}_e \cdot \nabla)\vec{v}_e\right] = e n_e \vec{E} - \nabla p_e \tag{3.36}$$

Isolating the z velocity component (w) leads to

$$m_e n_e \left[\frac{\partial w_e}{\partial t} + (\vec{v}_e \cdot \nabla)w\right] = e n_e E_z - \frac{\partial (n_e k T_e)}{\partial z} \tag{3.37}$$

Then, by assuming that spatial variations in w are small, the above relationship is simplified to

$$m_e \frac{\partial w}{\partial t} = e E_z - \frac{K T_e}{n_e}\frac{\partial n_e}{\partial z} \tag{3.38}$$

Relative to the ions, the electrons can be assumed to be massless. The term on the left side of eq. 3.38 then drops out, and a relationship between the electric field and density gradient can be obtained

$$e E_z = \frac{K T_e}{n_e}\frac{\partial n_e}{\partial z} \tag{3.39}$$

However, since $\vec{E} = -\nabla\phi$, equation 3.39 is equivalent to

$$e\frac{\partial\phi}{\partial z} = \frac{KT_e}{n_e}\frac{\partial n_e}{\partial z} \tag{3.40}$$

Integration leads to

$$\begin{aligned} e\left(\phi - \phi_0\right) &= kT_e\left[\ln(n_e) - \ln(n_0)\right] \\ &= kT_e \ln\left(\frac{n_e}{n_0}\right) \end{aligned} \tag{3.41}$$

Finally, solving for n_e leads to the Boltzmann relationship for electrons

$$n_e = n_0 \exp\left[\frac{e(\phi - \phi_0)}{kT_e}\right] \tag{3.42}$$

Since $\rho_j = q_j n_j$, the Poisson's equation can be rewritten as

$$\nabla^2\phi = -\frac{\rho}{\varepsilon_0} = -\frac{e(Z \cdot n_i - n_e)}{\varepsilon_0} \tag{3.43}$$

where Z is the charge state of the ions. Then by assuming quasi-neutrality, $n_e \sim n_i$, the form of the Poisson's equation which needs to be solved is

$$\nabla^2\phi = -\frac{e\left[Z \cdot n_i - n_0 \exp((\phi - \phi_0)/kT_e)\right]}{\varepsilon_0} \tag{3.44}$$

where n_i = is the ion number density (m$^-$3), from tracked particles
n_0 = reference plasma density (m$^-$3)
T_e = reference electron temperature (K)
ϕ_0 = reference plasma potential (V)

The three reference parameters are provided by the user through the `coliseum.in` file. DRACO also allows the user to swith the Poisson solver to the *linear* mode, in which the linear Poisson's equation (eq. 3.43) is solved. The linear equation should be used whenever a fully-kinetic simulation is performed.

Equation 3.44 is *non-linear*, since the right handside vector in the $A\vec{x} = \vec{b}$ formulation depends on the solution vector $\vec{x}$. Hence, an iterative method must be used to obtain the solution, unless the potential is obtained by the direct Boltzmann inversion.

3.5.3 Boltzmann Inversion

The inverted Boltzman relationship, eq. 3.41 leads to an interesting conclusions: the local plasma density is a sufficient criterion to determine the local potential. Of course, this conclusion is valid only for those cases in which the assumptions taken to derive the relationship hold.

This method of obtaining the potential is called *Boltzmann inversion*. Compared to the full solution of the elliptic Poisson's equation, it is very fast, and as simulation experience showed, the solutions in the high density, quasi-neutral ion beam are usually equivalent.

However, the solutions outside the beam usually diverge, since quasi-neutrality can no longer be assumed. More specifically, the Boltzmann inversion is not capable of resolving the plasma sheath. Also, since the node at each potential is computed individually, no guarantee is made to the continuity of the potential. Lack of continuity can lead to incorrect numerical discretization in the electric field. Due to these limitations, the Boltzmann inversion should be used primarily to test the simulation setup.

3.5.4 DADI

The DADI solver uses the Dynamic Alternate-Direct-Implicit method of solving the Poisson's equation. The solver was part of the original PLUME code, and was simply translated to C, and modified to work with new data structures. The solver is described in a greater detail in [18].

3.5.5 Gauss-Seidel

For many cases, the DADI solver converges very rapidly. However, some configurations lead to a highly-oscillatory "convergence." Increasing the number of solver iterations should lead to a better solution, but this is not generally true for the DADI solver. For this reason, a simple SOR optimized Gauss-Seidel solver was written. The solver requires a longer time to converge than the DADI, but the solution generally does not oscillate. Taylor's expansion of the Poisson's equation on an uniform mesh leads to

$$\frac{\phi_{i-1,j,k} - 2\phi_{i,j,k} + \phi_{i+1,j,k}}{(\Delta x)^2} + \frac{\phi_{i,j-1,k} - 2\phi_{i,j,k} + \phi_{i,j+1,k}}{(\Delta y)^2} + \\ + \frac{\phi_{i,j,k-1} - 2\phi_{i,j,k} + \phi_{i,j,k+1}}{(\Delta z)^2} = -\frac{\rho}{\varepsilon_0} \tag{3.45}$$

Potential at node i, j, k can be pulled out

$$\phi_{i,j,k} = \frac{(\Delta x)^2 + (\Delta y)^2 + (\Delta z)^2}{2} \left[\frac{\rho}{\varepsilon_0} + \frac{\phi_{i-1,j,k} + \phi_{i+1,j,k}}{(\Delta x)^2} + \right. \\ \left. + \frac{\phi_{i,j-1,k} + \phi_{i,j+1,k}}{(\Delta y)^2} + \frac{\phi_{i,j,k-1} + \phi_{i,j,k+1}}{(\Delta z)^2} \right] \tag{3.46}$$

The Neumann boundary is implemented using a one-sided, first order accurate, finite differencing of the $\nabla\phi = -h$ as

$$\phi_{0,j,k} = \phi_{1,j,k} - \frac{1}{\Delta x} h_{XMIN} \tag{3.47}$$

$$\phi_{nx-1,j,k} = \phi_{nx-1,j,k} + \frac{1}{\Delta x} h_{XMAX} \tag{3.48}$$

where h is the prescribed boundary condition. Similar formulation exists for the Neumann boundaries along the y and z directions. Setting $h = 0$ leads to the previously described "Neumann zero" condition.

As can be seen, except on the boundaries, the solution at node i, j, k depends on the six surrounding nodes. Strictly speaking, these nodes hold the values obtained at the previous

iteration, or the initial guess, if $it = 1$. The Boltzmann inversion provides a good starting point for the solver.

The Gauss-Seidel formulation uses only a single array to retain the potential solution; newly computed values of $\phi^n_{i,j,k}$ immediately replace $\phi^{n-1}_{i,j,k}$. The GS method tends to converge faster than the Jacobian formulation, in which two separate arrays are stored, and ϕ^n is copied to ϕ^{n-1} only after all nodes were evaluated at time n.

Convergence can be increased by adding successive over-relaxation (SOR) method to GS. The SOR is a simple method which attempts to predict the future value of ϕ based on linear interpolation. By labeling the value computed by eq. 3.46 as $\phi^*_{i,j,k}$, and if $\phi^{n-1}_{i,j,k}$ is the previous value, then SOR method gives

$$\phi^n_{i,j,k} = \phi^{n-1}_{i,j,k} + \omega \left(\phi^*_{i,j,k} - \phi^{n-1}_{i,j,k} \right) \tag{3.49}$$

where ω is the SOR acceleration factor. The optimal value will depend on the mesh topology, but values from 1.4 to 1.6 tend to work well.

The GS solver iterates until the solution converges to a specified tolerance, or the maximum number of iterations is exceeded. Some speed up was obtained by checking the convergence only after each fifth iteration. Convergence is checked by comparing the L2 norm of $\mathbf{A}\vec{\phi} + \vec{\rho}/\varepsilon_0$ to the user specified tolerance.

3.5.6 Immersed Finite Element Solver

The immersed finite element (IFE) solver is based on a method developed by Kafafy and Lin [9]. It operates on the interface tetrahedral mesh, described previously. Currently, it is DRACO's only solver capable of resolving the curvature of the surface. The finite-difference solvers, described previously, solve the potential on the stair-case degenerated surface.

The solver was implemented by Kafafy using the Fortran 90 programming language. Due to language differences, the solver has not yet been fully integrated into DRACO. Direct passing of data structures and pointers from C to F90 was attempted but was not successful. The initial mesh is instead passed to IFE using a temporary data file. Subsequent communication (to send ρ and ϕ) is done by packing and unpacking of data buffers. This method allows DRACO to communicate with the solver, at the added cost of computational performance. Of greatest concern is the added memory usage. Since IFE does not interface directly with VOLCAR, it must re-create the mesh structure already existing in memory. Additional work is needed to complete the IFE integration, but the integration may require a complete re-write of the solver in the C language.

3.6 Electric Field Update

The electric field is obtained from

$$\vec{E} = -\nabla\phi \tag{3.50}$$

Since the potential values are stored on the nodes of a Cartesian grid, the numerical implementation is straightforward:

$$Ex_{i,j,k} = \frac{\phi_{i-1,j,k} - \phi_{i+1,j,k}}{\Delta x} \tag{3.51}$$

$$Ey_{i,j,k} = \frac{\phi_{i,j-1,k} - \phi_{i,j+1,k}}{\Delta y} \tag{3.52}$$

$$Ez_{i,j,k} = \frac{\phi_{i,j,k-1} - \phi_{i,j,k+1}}{\Delta z} \tag{3.53}$$

This general scheme needs to be modified along the external boundaries. Along Dirichlet boundaries, a one sided Taylor expansion is performed to obtain

$$Ex_{0,j,k} = \frac{3\phi_{0,j,k} - 4\phi_{1,j,k} + \phi_{2,j,k}}{2\Delta x} \tag{3.54}$$

$$Ex_{nx-1,j,k} = \frac{-3\phi_{nx-1,j,k} + 4\phi_{nx-2,j,k} - \phi_{nx-3,j,k}}{2\Delta x} \tag{3.55}$$

with similar discretization used for the remaining spatial directions.

The Neumann zero condition requires that the perpendicular E-field component is zero along the face. The discretization is performed by pre-computing the "left" and "right" indices for eq. 3.51. The numerical implementation of eq 3.51 can be approximated by:

$$Ex[n] = \frac{\phi[xm[n]] - \phi[xp[n]]}{\Delta x} \tag{3.56}$$

$$Ey[n] = \frac{\phi[ym[n]] - \phi[yp[n]]}{\Delta y} \tag{3.57}$$

$$Ez[n] = \frac{\phi[zm[n]] - \phi[zp[n]]}{\Delta z} \tag{3.58}$$

The actual implementation is a bit more involved, since it avoids the double array indexing by relying on memory pointers. Regardless of the implementation, the Neumann zero condition can be directly incorporated into 3.56 by setting the left index on the MIN boudnary to the right index, and vice versa for the MAX boundary. Hence:

$$xm[n] = xp[n] \quad \forall\ n \in \Gamma_{XMIN,N} \tag{3.59}$$

and

$$xp[n] = xm[n] \quad \forall\ n \in \Gamma_{XMAX,N} \tag{3.60}$$

The periodic boundary condition in x means that the computational nodes $n_{0,j,k}$ and $n_{nx-1,j,k}$ represent the same point in space. The left neighbor of $n_{0,j,k}$ is $n_{nx-2,j,k}$. Similarly, the right neighbor on the XMAX boundary is $n_{1,j,k}$. This is implemented using

$$xm[n] = N(nx-2, j, k) \quad \forall\ n \in \Gamma_{XMIN,P} \tag{3.61}$$

and

$$xp[n] = N(1, j, k) \quad \forall\ n \in \Gamma_{XMAX,P} \tag{3.62}$$

3.7 Monte-Carlo Collision Modeling

3.7.1 CEX Collisions

DRACO contains a rudimentary MCC collision model for CEX collisions. Additional models for ionization and scatter, were added by Tran[19] and Pierru[20].

The Monte-Carlo modeling performs collisions by colliding particles with a background target "cloud." Unlike the DSMC method, developed by Bird [21], the standard MCC does not require sampling of two particles from the same cell. Retaining a list of particles belonging to each cell is commonly performed in PIC implementations on unstructured grids, to expedite the particle push algorithm. DRACO does not maintain such a list, since the particle push is performed on a structured grid. The search of a particle pair would thus be a time consuming operation.

However, since only one particle is involved in an MCC collision, it is difficult to conserve momentum. One commonly used approach is to adjust the mass of the background cloud. DRACO does not perform such adjustment. Instead, it assumes that background cloud is very massive, such that it is not affected by collisions. Validation of this assumption, by a comparison of MCC and DSMC results, is left for future work.

The collision details are specified in the material interaction file, `mat_mat.txt`. For each collision, the source specie and the background target specie are given. Also specified is the type of collision, and a model for the collision cross-section, σ. The appropriate number of coefficients for the σ calculation is also given.

For CEX collisions, the neutral velocity is assumed to be constant and negligible compared to the ion velocities, v_i. The collision frequency, ν, is then computed using

$$\nu = n_n \sigma v_r \tag{3.63}$$

where ν = collision frequency, (1/s)
σ = collision cross-section, (m^2)
n_n = neutral density, weighed onto particle position, (m^3)
v_r = relative velocity, $v_r \sim v_i$, (m/s)

The probability of the collision, P, is given by

$$P = 1 - e^{-\nu \Delta t} \tag{3.64}$$

The code then computes whether a collision occurred by comparing P to a random number. For a CEX collision, the velocity of the ion is decreased to the thermal velocity of the neutrals. Imposing a Maxwellian velocity distribution, the new ion velocity is given by:

$$v_i = 2 v_n (R1 + R2 + R3 - 1.5) \tag{3.65}$$

where v_n is the neutral thermal velocity and $R1$, $R2$ and $R3$ are three random numbers, ranging from 0 to 1.

3.7.2 Source Projection

The MCC formulation requires the density distribution of the background target specie. A direct approach is to track the neutral particles, and compute their densities prior the MCC

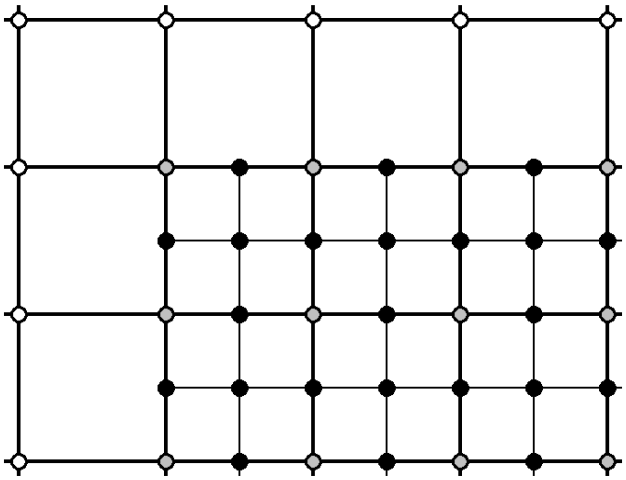


Figure 3.6: 2D representation of DRACO's mesh-refinement capabilities. Only 1:2 refinement is supported, and the external boundaries of the fine mesh must coincide with the coarse cells. White circle indicates a node on the coarse mesh. Gray circle is a node which is shared by both the fine and coarse mesh. Black nodes are specific to the fine mesh.

step. However, such an approach is not very effective from the computational perspective. In the absence of scatter collisions, the neutral particles will simply move in straight-line trajectories. However, their positions still need to be updated and memory must be allocated for storage of neutral data.

A more attractive approach is to replace the particle tracking of the neutrals by an volumetric description of the background neutral plume. DRACO allows for this by *projecting* the particle source onto all simulation grid nodes.

COLISEUM sources implement a *projection* function, which extrapolates the mean velocity and flux from the source onto an arbitrary point in space. Source projection is loaded by looping through all grid nodes, and using the source projection function to compute the mean particle velocity, v_m and the particle flux, Γ at the position of the node. The density is then given by

$$n = \frac{v_m}{\Gamma} \tag{3.66}$$

3.8 Support for Mesh Refinement

Mesh refinement allows the user to specify local regions containing cells smaller than those used on the primary simulation mesh. The addition of the finer mesh can help to resolve the Debye length in high-density regions. It also increases the amount of detail which can be captured by the interface Cartesian mesh.

An attempt was made to incorporate mesh refinement into DRACO. Although a significant progress had been made, the code lacks a good Poisson solver capable of operating, and converging, on the refined domain. Of course, this is not a problem if the Boltzmann

inversion is used to obtain the solution. However, the Boltzmann inversion does not resolve the plasma sheath, which is usually the primary reason for including the refined mesh.

DRACO currently supports only a 1:2 refinement, in which each cell edge is cut in two. Figure 3.6 shows a 1:2 refinement on a 2D mesh. The 1:2 refinement was chosen for its relative simplicity. The refined mesh shares a large number of its nodes with the parent mesh, which simplifies the interpolation of values between the two meshes. The shared nodes are shown using gray circles.

3.8.1 Particle Motion

Particle motion is performed on the finest available mesh. Tracking on the finest mesh is necessary to properly resolve surface interactions. Small "kinks" in the surface may not be captured by the coarse mesh, especially if they completely reside within the cell. Electric field from the fine mesh is also needed to properly account for near-surface potential effects. Particle charge density is also deposited onto the finest mesh, so that local potential can be calculated.

The charge density must be interpolated from the fine mesh onto the coarse mesh to obtain the initial guess at the potential, which also sets the boundaries on the fine mesh. However, depositing the charge density on the fine grid results in particle-shrinkage and a "loss" of charge density on the overlapped nodes. In other words, $\rho^f_{i,j,k} < \rho^c_{i,j,k}$, where ρ^f and ρ^c are the values of charge density on the fine and coarse mesh respectively, both taken at the same spatial position. This result is clear if the actual weighing process is considered. A particle contributes to all eight nodes of the cell it resides in, regardless of the size of the cell. Refining of the coarse mesh replaces the mesh with eight smaller cells and only the particles located within the cells attached to i, j, k will contribute to the density. If the particle density is uniform, $\rho^c_{i,j,k} \sim 8\rho^f_{i,j,k}$.

The value of ρ^c is correctly computed by weighing values at internal nodes onto the coarse nodes. This process is analogous to weighing of particle's charge density to the grid, where the position of the particle is replaced by the position of the internal nodes. The value that is weighed onto the coarse nodes is the value of charge density on the fine node.

3.8.2 Multi-Domain Potential solver

As was mentioned previously, no satisfactory potential solver capable of working on the refined mesh has been developed. An attempt was made by incorporating some ideas from the multi-grid method. However, the crucial difference between a standard multi-grid method and the mesh-refinement concept used in DRACO is the treatment of boundaries. In a standard multi-grid method, the entire coarse grid is overlaid with a fine mesh. The two grids share the external boundaries. This is not the case in DRACO, where the fine mesh may cover only a small region of the coarse mesh. The region can further be completely enclosed inside the coarse mesh.

A unique solution of the Poisson's equation is possible only if the external boundary conditions are specified. Since the fine mesh may be completely internal, the boundary information is not available. In fact, the fine mesh does not have any physical boundaries, since the mesh is simply a piece of the global solution of the Poisson's equation.

Poisson's equation could be solved if the coarse and fine nodes were coupled together using some non-standard finite differencing. Then, the solution would be of $\mathbf{A}\vec{x} = \vec{b}$ form, and external boundaries would be preserved. However, this method has two drawbacks. First, such a finite-difference form was not readily available. Even assuming that the time was spent to derive the form, this formulation still required that the solution is performed using a uniform solver.

The single solver restriction was the primary reason why a different approach was investigated. Simulations performed with DRACO typically look at plume effects on some satellite. The simulation volume for such a study typically contains the satellite enclosed in a large amount of free space. There is no need to overlay the free space with the tetrahedral interface mesh, since the tetrahedrons are used only to resolve surfaces. Therefore, it is desired to create the tetrahedral mesh only in the vicinity of the spacecraft. This multi-domain formulation can be easily achieved with mesh refinement. A Cartesian-only "coarse" mesh is used to describe the simulation domain. Secondary refined tetrahedral mesh is created around the spacecraft. The description of the surface by the interface elements is used to track the particles and to accurately resolve the near-surface potential field.

However this formulation requires coupling of two potential solvers. A finite difference solver is used on the coarse mesh. The finite element IFE is used in the near-surface domain. The two solutions must be continuous along the mesh boundaries, which showed to be problematic. The current approach is to obtain a solution on the coarse mesh, according to the prescribed external boundary conditions. A staircase representation is used to approximate the satellite. The external faces of the fine mesh are then set as Dirichlet, with potential values obtained by interpolation of the coarse grid values. The internal region is then solved, giving new values of potential near the surface. This potential fix is applied to the coarse mesh by interpolation. All coarse nodes which are covered by the *non-boundary* part of the fine mesh are also set as Dirichlet. A new solution is obtained on the coarse mesh, including new values along the boundary of the fine mesh. These values are used to once again fix the boundary of the fine mesh, and the process is repeated. Ideally, after several repetitions, the process should arrive at some continuous solution. Unfortunately, this is often not the case, and the solutions converge independently from each other. Additional work is obviously needed.

3.9 Time-loop Termination

The main time loop is terminated once a user-specified exit condition is satisfied. DRACO supports three conditions: maximum number of time steps, plasma run time and steady state.

The first condition is straightforward. The second condition causes the run to terminate, after the total simulated time ($\sum \Delta t$) exceeds a prescribed value. Automatic time step adjustment results in a variation of the step size, and thus the total run time cannot be computed from $t = nt * \Delta t$, where nt is the total number of time steps.

The last method terminates if a steady-state solution has been reached. A steady-state is characterized by the number of particles leaving through the external boundaries matching the number of newly born particles at the thruster. This is implemented by requiring that

Table 3.2: List of simulation variables available in DRACO

Name	Specie	Description
phi	N	potential, V
efx,efy,efz	N	electric field components, V/m
rho	N	charge density, C/m^3
rho_back	N	constant background charge density, C/m^3
nd	Y	number density, #/m^3
ne	N	Boltzmann electron number density, #/m^3
u,v,w	Y	average flow velocity, m/s
jx,jy,jz	Y	current density, C/(m^{2}s)
PolyTe	Y	polytropic temperature, eV
MaxTe	Y	Maxwellian temperature, eV
mpc	Y	macro-particles per cell, #
flux	Y	surface flux, #/(m^{2}s)

the change in number of particles between two consecutive iterations is less than 0.01%, or

$$d = \frac{|np^k - np^{k-1}|}{np^k} \tag{3.67}$$

$$ss = \begin{cases} \text{true}, & d < 1 \times 10^{-4} \\ \text{false}, & d \geq 1 \times 10^{-4} \end{cases} \tag{3.68}$$

where np^k and np^{k-1} is the total number of particles at the current and previous time steps, respectively, and ss is a flag indicating steady-state.

3.10 Post-Processing

3.10.1 Volumetric Plasma Diagnostics

The grid centered simulation results can be outputted in both the Tecplot and capVTE formats. The actual file output is handled by VOLCAR. However, DRACO supplies a list of evaluation functions which are called by VOLCAR to update the desired variables.

To save memory and to increase performance, DRACO retains only the minimum crucial data in memory during the simulation run. These include the potential, electric field and charge density. Collision modeling will add the number density of all target species and source projection will add a background charge density. However, the majority of available grid variables are initialized and evaluated only prior to the disk output.

The variables which can be outputted by DRACO are summarized in table 3.2. Many variables are specie specific, since, as an example, the number density of the Xe+ and Ar+ may be desired individually. These variables are referenced using a dot convention, where the

variable type is separated from the specie name by a dot. For instance, the number density of Xe+ is referenced to using "nd.xe+".

The simulation noise can be removed from the results by averaging the values over several iterations. Typical approach is to run the simulation until the steady state, followed by a restart over several tens of iteration, during which the averaging is performed. The average values are updated using

$$A_1^k = \frac{1}{k}\left[(k-1)A_1^{k-1} + a^k\right] \tag{3.69}$$

where A_1^k is the average of some quantity a over k time steps. In other words,

$$A_1^k = \frac{1}{k}\sum_{i=1}^{k} a^i \tag{3.70}$$

The processes used in the computation of the potential, electric field and charge density were described in previous sections. The number density is computed in a fashion similar to the charge density scatter. Instead of depositing charge per cell volume, density is computed by depositing the number of *real* particles per cell volume. In other words, the contribution of each macro-particle to the density is sw/V_{cell}. Number of macro-particles per cell is simply the scatter of 1.

The average u velocity component is obtained by first computing the one dimensional particle flux, $\Gamma = u \cdot n$, by scattering $u \cdot sw/V_{cell}$ for all particles of the variable specie. Average velocity is the obtained by dividing the flux on each node by the corresponding number density, $u_j = \Gamma_j/n_j$, where j is a node number. Same formulation is used for the other two velocity components. Current density is computed by scattering $q_j n_j v_j$.

The polytropic temperature is computed from

$$T_P = T_e\left(\frac{n}{n_0}\right)^{\gamma-1} \tag{3.71}$$

where γ is the polytropic constant, provided by the user and T_e is the reference electron temperature.

The Maxwellian temperature computation requires the greatest amount of work. First, the average velocity magnitude is computed at each node using

$$c = \sqrt{u^2 + v^2 + w^2} \tag{3.72}$$

where u, v and w are the average velocity components. Next, the thermal component of particle velocity,$\vec{v}_p$ is deposited on the grid

$$t = |\vec{v}_p| - c \tag{3.73}$$

Average thermal velocity is obtained by dividing the aggregate value by the number of particles. Finaly

$$T_M = \frac{\bar{t}}{kT_e} \tag{3.74}$$

Table 3.3: List of outputted time-dependant variables

Name	Specie	Description
it	N	time step number, V
np	Y	total number of macro-particles, #
loss rate, comp	Y	particle loss rate, for each component, #/s
loss rate, outer	Y	particle loss rate to outer boundaries, #/s
time	N	total elapsed time, s
dt	N	time step length, s
p	N	total momentum, kg*m/s
p-avg	N	momentum per particle, kg*m/s
etot	N	total energy, J
ek	N	total kinetic energy, J
ee	N	total field energy, J

3.10.2 Particle Sampling

Particle positions and velocities can also be sampled to a Tecplot file. The sampled particles are filtered according to user specified specie and/or source. For instance, in a simulation containing multiple thrusters and multiple particle species, individual zones can be created for particles of specie 1, particles emitted by thruster 1 and particles of specie 2 only emitted by thruster 2.

The particles are sampled randomly, to prevent apparent periodicity in the particle phase space due the linear loading of particles. The sampling frequency is computed from the requested number of outputted particles as

$$f = n_{req}/n_{tot} \tag{3.75}$$

where n_tot is the number of particles which qualify for output by meeting the specie and/or source requirements. The outputted particles are determined by looping through all qualifying particles, and drawing a random number, r, from 0 to 1. The particle is outputted if $r \leq f$.

3.10.3 Time-dependant Diagnostics

DRACO can also output global state diagnostics at each time step. The list of outputted variables is summarized in table 3.3. Plot of these values versus simulation time provides additional information which may not be observable from the final grid output. Usually of high importance is development of an oscillatory behavior in the solution. This behavior will demonstrate itself as a periodic variation of particle counts, and also in oscillations in momentum and energy. Particle loss rate provides information on the absorbtion of particles by surfaces and the external walls. This data was used to analyze the current collected by the anode plate in the cathode model.

Chapter 4

Neutralization Modeling Approach

4.1 Problem Overview

The plasma environment outside an ion thruster is dominated by the interaction of ions, electrons and neutrals. High velocity beam ions flow out of the thruster optics and are responsible for the production of thrust. The ionization of the neutral propellant within the thruster yields secondary electrons, which are screened out by the ion optics grid. These electrons are instead absorbed by the thruster walls, and are emitted into the beam by a hollow neutralizer cathode located near the edge of the beam. Without an effective neutralization, a high potential would develop in the beam, possibly reversing the flow of the ions, and thus canceling the thrust. Emission of electrons from the cathode also helps retain the charge neutrality of the spacecraft.

Ionization chambers operate at limited efficiency levels, and thus only a fraction of the supplied fuel will become ionized. The neutral gas can exit the chamber by a random motion across the accelerator grid holes. Although the rate of neutral mass is much lower than the rate of ion emission, the density of the neutral plume near the thruster exit often exceeds the ion density, due to the low velocity of the neutral gas. The neutrals interact with the ion particles through collisions. However, the interaction of neutrals was ignored in this study, and the plume region was assumed to consist of only ions and electrons. Furthermore, due to DRACO's limited collision capabilities, the Coulomb collisions between the electrons and ions were also ignored.

Modeling the neutralization process is not a simple task, since not much is known about the plasma dynamics in this region. Fully kinetic approach, in which both electrons and ions are tracked as macro-particles, offers a viable solution, since it relies only on fundamental first-principle physical laws. This is contrasted to the fluid electron model, in which the electron motion is obtained from the concurrent solution of the momentum, continuity and energy equations. The fluid equations are influenced by the correct selection of *diffusion* and *mobility* coefficients, but the actual values of the coefficients may not be known a priori.

However, the physical simplicity of the fully kinetic model is also coupled with implemen-

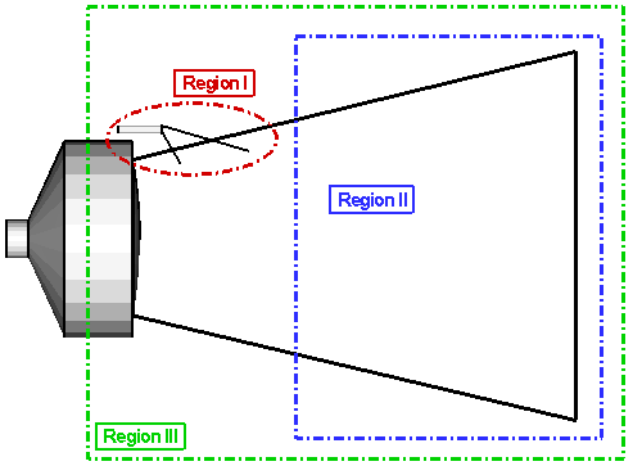


Figure 4.1: Overview of the simulation process. The electron dynamics in an already neutralized beam (Region I) was studied first. Next, the flow of electrons from the cathode (Region II) was modeled. Finally, the two simulations were combined (Region III).

tation difficulties. During the course of this research, three major issues were identified:

- Since the quasi-neutral approximation can no longer be used, the simulation mesh must be able to resolve λ_D.
- High electron velocities require very small Δt, leading to a large number of total time steps.
- Improper treatment of electrons at grid boundaries introduces a numerical "pump" instability.

These problems could be partly avoided by using an artificially high electron mass. However, using $kT = 3eV$ as the reference plasma temperature, the thermal velocity

$$v_{th,j} = \sqrt{\frac{2kT}{m_j}} \tag{4.1}$$

of the ions and electrons is approximately 10^3 and 10^6 m/s. The mean ion and electron velocities, v_j, for the case studied here were approximately 3×10^3 and 5×10^5 m/s. Characteristic plasma speed ratio (similar to the Mach number in neutral gas dynamics) is defined by

$$S_j = \frac{v_j}{v_{th,j}} \tag{4.2}$$

Quick calculation shows that the plasma is meso-thermal, since $S_i \gg 1$ while $S_e \ll 1$. Using a non-physical mass ratio of perhaps $m_i = 100m_e$ would destroy the meso-thermal relationship, and instead of ion beam neutralization, the model would approximate the interaction of two beams [22].

Instead of attempting to model the entire neutralization process initially, the problem was split into a series of steps. From the modeling perspective, the area outside the ion thruster can be divided into three distinct regions, as shown in figure 4.1. First is the near cathode region, which is dominated by a high electron density. The simplistic model used here ignored the presence of high-energy ions which have been observed in cathode plumes.

A *plasma bridge* forms between the cathode and the beam, aiding in the transfer of the electrons into the beam. Beyond the plane of the cathode lies the next region, Region II. The beam here is assumed to be fully neutralized. Of primary interest in this region are the forces leading to electron capture in the beam, and the consequent electron dynamics. The third regions combines the dynamics of the two subregions in a single simulation.

Large amount of time was spent on Region II modeling. The pre-neutralization was approximated by emitting both electrons and ions from the ion optics. The main objective of the study of this region was a development of a better understanding of electron dynamics, and the problems which may be encountered in a fully kinetic modeling. Several questions had to be answered, including whether DRACO was actually capable of resolving the electron containment. The apparently simple task of injecting the ions and electrons proved to be difficult, since a virtual anode developed in the original configuration. A dimensional scaling approach was used to overcome this difficulty. The finite domain dimensions proved to have a profound impact on the simulation results. Removal of the electrons at the grid edge lead to development of a "plasma pump" instability, which effectively sucked electrons out of the beam. The instability was corrected by introducing a new particle boundary behavior, based on the electron kinetic energies.

Simulation of Region I was performed primarily to obtain an understanding of the electron emission process. As described in following sections, exact modeling of the cathode proved to be a difficult task, limited to a large extend by the numerical inability to resolve the potential near the tip. Two modes of operation were clearly visible, with the electrons being emitted in either a "plume" or a "spot" mode. However, neither mode was capable of producing electrons at velocity sufficiently low to allow electron turning into the beam, while retaining physically reasonable anode/cathode drop. The problem was resolved by fixing the near-tip charge density.

Lessons learned from modeling of Regions I and II were applied to a complete simulation of the neutralization process in the NEXT ion thruster. The simulation looked at neutralization of both a single and cluster configuration. Both individual and a single central cathode were investigated. These results are presented in the following chapter.

4.2 Computational Platforms

The extent of available computational resources largely impacts the level of detail which can be simulated. Selection of computers was primarily influenced by the processor speed and the amount of available memory. The operating system did not play a significant role, since with the exception of MPI, the code is completely self-contained. The MPI version [20] was still under development, and was not used in this study. Any computer with a C [1] compiler was capable of running DRACO.

[1] and F90, if IFE is used

Fortunately, a large number of computers was available to run the simulations, thanks to a great work of the department's system administrator Luke Scharf. Results presented in this chapter were obtained on the following machines:

- Toshiba laptop with a 2.53GHz Pentium IV processor and 512Mb of RAM running Windows XP
- Dell desktop with a 2.8GHz Pentium IV processor and 1Gb of RAM running Windows XP
- Server with two 1.6GHz AMD Opteron CPUs and 3Gb of RAM running Whitebox Enterprise Linux 3.0
- Dell workstation with dual 3GHz Intel Xeon processors and 2Gb of RAM, running Redhat Enterprise Linux 3.0
- A four node Apple cluster, with two 2GHz PowerPC G5 CPUs and 2Gb of RAM per node, running Mac OS X 10.3

The 512Mb of available memory on the PC laptop proved to be limiting in many cases, and thus the laptop was used primarily for code development, debugging and testing. However, ignoring the memory limitation, the performance of this off-the-shelf computer was very good, indicating that full scale plasma simulations no longer require massive investments in computational resources. However, having access to a large number of computational machines allowed for multiple simulation to executive concurrently, thus reducing the total wall time needed to obtain results.

Shifting the simulations from the laptop onto the Linux/OSX stations increased the amount of available memory. Memory usage is divided chiefly among the storage of the computational mesh and the storage of particle positions and velocities. Theoretically, having access to a machine with larger available memory allows for simulations to proceed on a much finer mesh, and/or to use a larger number of macro-particles. However, a larger number of nodes also increases the time needed to solve the Poisson equation. Usually, the solver time, and not the available memory, were responsible for the upper limit on the mesh size. Similarly, a higher number of macro-particles could be used to decrease simulation noise, but only minor improvements were observed for cases using over 100 macro-particles per cell [23].

4.3 Ion Optics Source Model

The thruster operating parameters were obtained from Table 1 in [24]. The parameters of interest are summarized in table 4.1.

Injection velocity was computed from the specific impulse as

$$u_{eq} = \mathrm{Isp} \cdot g_e \sim 34,400 m/s \tag{4.3}$$

where $g_e = 9.807 m/s^2$ is the gravitational acceleration at the Earth's surface. The ions were assumed to follow the Maxwellian distribution at 0.1eV of thermal energy.

Table 4.1: Thruster operating parameters

Parameter	Value
Input Power, kW	2.2
Beam Current, A	1.20
Beam Voltage, V	1570
Accel. Voltage, V	-257
Total Voltage, V	1820
Specific Inpulse, s	3510

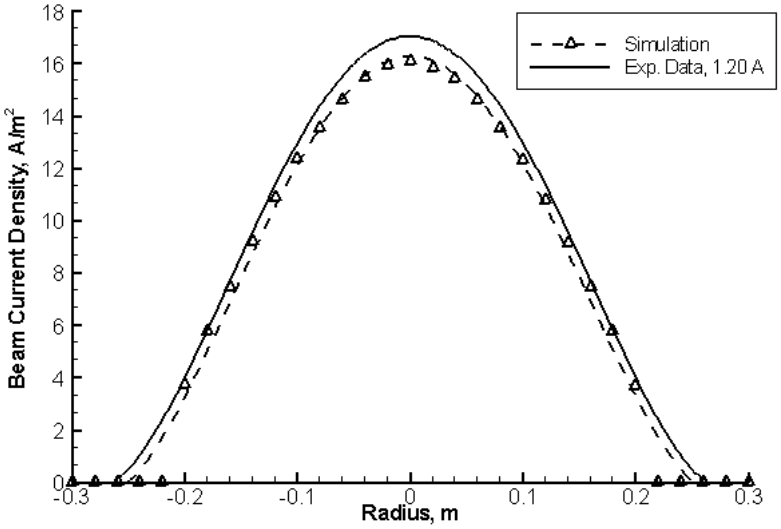


Figure 4.2: Comparison of simulation radial beam current density profile to a curve fit of measurement of fig. 4 in [24], 5 cm downstream of the accelerator grid center. Source parameters were based on the 1570 V power supply voltage and 1.2A beam current operating condition.

Table 4.2: Radial beam current density values used to describe the NEXT ion beam profile, with assumed radial symmetry

Radius, m	J, A/m^2
0	17
0.1	13
0.2	4
0.3	0

The source production rate is specifying in COLISEUM using $\dot{m}$ (kg/s). Current can be converted to mass flow rate easily using:

$$\dot{m} = \frac{I}{Ze} m \tag{4.4}$$

where Z is the charge state of the particle and m is the mass, in kilograms. The ratio between the currents due to the doubly and singly charged ions is approximately 0.04[5]. In this simulation, any effects due to doubly charged ions were ignored and the beam was assumed to consist of singly-charged ions only. The required mass flow rate is then

$$\dot{m} = \frac{1.2A}{1.602 \times 10^{-19} C} 2.182 \times 10^{-25} kg = 1.634 \times 10^{-6} kg/s \tag{4.5}$$

The MAXSTREAM source distributes the total mass flow rate onto all source elements by scaling the total $\dot{m}$ by the fraction of the source element's area:

$$\dot{m}_i = \frac{\mathbb{A}_i}{\mathbb{A}_0} \dot{m}_0 \tag{4.6}$$

where $\dot{m}_i$ = mass flow rate produced by source element i, kg/s
$\mathbb{A}_i$ = area of the source element i, m^2
$\mathbb{A}_0$ = total source area, m^2
$\dot{m}_0$ = total mass flow rate, kg/s

This formulation equally distributes the source particle flux, Φ_i, since

$$\Phi_i = \frac{\dot{m}_i}{\mathbb{A}_i} = \frac{\dot{m}_0}{\mathbb{A}_0} \tag{4.7}$$

producing a flat beam. However, despite the advancements in ion thruster technology, the NEXT ion beam profile is not completely flat. A larger flux is produced near the centerline of the optics. The procedure undertaken to replicate the beam profile of an actual NEXT thruster was to bias the production mass flow rate according to a radial distribution function. Soulas [24] measured the beam current density for the NEXT thruster operating at various operating conditions. Data from figure 4 in [24] was used to obtain a polynomial relationship of current density. The experimentally measured current density indicates that the source flux is not radially symmetric. However, in this research a radial symmetry was imposed, and the measured values were adjusted to remove the lack symmetry. The radial dependence on current density is listed in table 4.3. These values were entered into a spreadsheet, and a polynomial curve fit was obtained. A polynomial of order fourth seemed to produce the best ratio between interpolation accuracy and the extent of oscillation near the endpoints. The polynomial was computed as a function of normalized radius, $\hat{r} = r/0.2m$, as:

$$J(\hat{r}) = 4.3837\hat{r}^4 - 17.434\hat{r}^2 + 17.043 \quad (A/m^2) \tag{4.8}$$

The odd terms do not appear in the formula due to the symmetry about the $\hat{r} = 0$ axis. Then, since

$$I = \mathbb{A}J \tag{4.9}$$

and

$$I = \frac{\dot{m}}{m} q \tag{4.10}$$

we have, for each source element,

$$\Phi_i = \frac{\dot{m}}{\mathbb{A}_i} = \frac{m}{q} J(\hat{r}_i) \tag{4.11}$$

where $J(\hat{r}_i)$ is the current density at the centroid of the i^{th} source element. This relationship was implemented in DRACO by adding a new source type, called ION_THRUSTER. The source parameter list contains the standard MAXSTREAM parameters ($\hat{m}$, v, T and projection elements) followed by the density polynomial. The order of polynomial is given first, followed by the respective number of coefficients. The source acts as top-layer wrapper for MAXSTREAM. It initializes MAXSTREAM, and modifies the $\dot{m}_i$ for all source elements. The complete line used to describe the ion source for the NEXT ion thruster, operating at the conditions listed in table 4.1 was:

```
source_specify optics ION_THRUSTER Xe+ 1.634e-10 1160.4 34400 20 4 4.3837 ...
... 0 -17.434 0 17.043
```

The source model was tested by running a simulation on a mesh enclosing the optics near region. The total current density, $J = \sqrt{Jx^2 + Jy^2 + Jz^2}$, was plotted 5cm downstream from the accelerator grid. The comparison versus the polynomial approximation of the experimentally measured current density is plotted in figure 4.2. The experimental data were collected 4.5cm downstream. Comparison was performed at 5cm since the size of the simulation cell was 1cm, and was thus not able to resolve the 4.5cm distance. Overall, a good agreement is observed, however the simulation seems to slightly under-produce the current density.

4.4 Potential Boundary Conditions and the Poisson Solver

4.4.1 Field Boundary Conditions

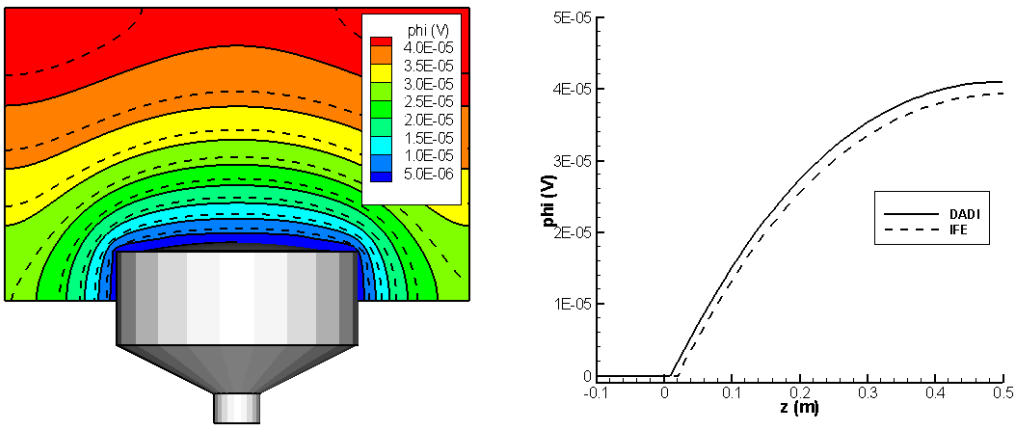
The thruster was assumed to operate in a space vacuum. The variation in space charge density disappears at some distance outside the zone of the beam influence. Then, since

$$\nabla \vec{E} = -\frac{\rho}{\varepsilon_0} \tag{4.12}$$

the electric field on the external boundaries is constant. This condition is specified by the Neumann boundary condition, $\partial\phi/\partial n = 0$.

Majority of the cases studied here exhibited at least one plane of symmetry. Computational time was reduced by simulating just the minimum required region. The charge density does not vary across the plane of symmetry, since $\rho(p-1) = \rho(p+1)$, where $i = p$ is the position of the symmetric plane. The Neumann condition is thus also applicable to symmetric faces.

The thruster was firing towards the ZMAX face. The condition of negligible ρ variation generally cannot be assumed along this face, especially if the simulation continues until the



(a) plane of symmetry

(b) beam centerline

Figure 4.3: Initial potential obtained with linear DADI (solid line) and IFE (dashed line) for a small uniform background charge. Only minor discrepancies are observed in the solution, but DADI converged 3 times faster and required less RAM to operate

steady state. However, if the wall is placed sufficiently far from the thruster, the local axial variation in the charge density should be negligible. Furthermore, it was found that the simulation needs to be terminated prior to the beam reaching the edge of the simulation domain. Stopping the beam early was necessary to prevent the loss of the leading electron sheath.

Field solvers included with DRACO support only the Dirichlet boundary condition on the internal boundaries (the enclosed objects). However, DRACO supports multiple object types, with one of them being "sink". The sink objects are not resolved by the field solver, but they act as boundaries for particle flow. Hence, setting an object to be a sink is analogous to setting a floating potential on the object. This condition is used on the hollow cathode, as is described later. The thruster was given a fixed potential of 0V.

4.4.2 Poisson Solver

Correctly resolving the surface curvature requires the use of the IFE field solver. However, as can be seen from figure 4.3, the agreement between the DADI and IFE solution is very good. The picture shows the initial field around the thruster, computed using a small uniform background charge density. DADI resolves the thruster as a collection of Dirichlet nodes. However, the curvature of the optics means that the actual surface extends some fraction of the cell past the last Dirichlet node. IFE *does* resolve this difference, since its formulation is based on the tetrahedral cuts. Due to this difference, the thruster, as seen by DADI, is smaller than the thruster seen by IFE by a fraction of a cell size. This discrepancy leads to the DADI potential *leading* the IFE potential, as is clearly visible in 4.3.

The discrepancy translates to approximately 5% difference in the potential between the two solvers. However, the IFE solver required 300Mb of additional RAM, and the solver took

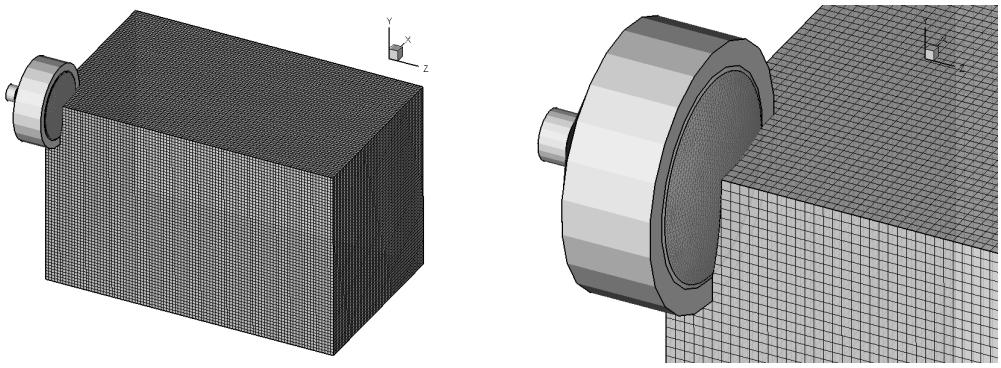
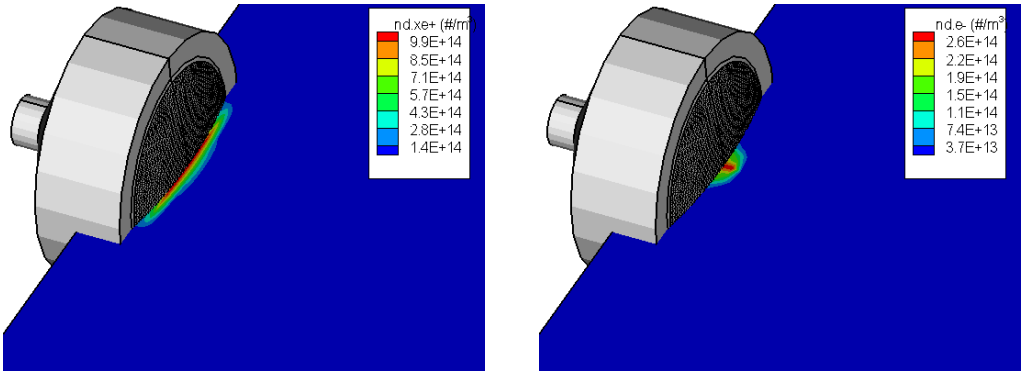


Figure 4.4: Initial simulation domain with an uniform cell size of 2cm.



(a) Ion number density

(b) Electron number density

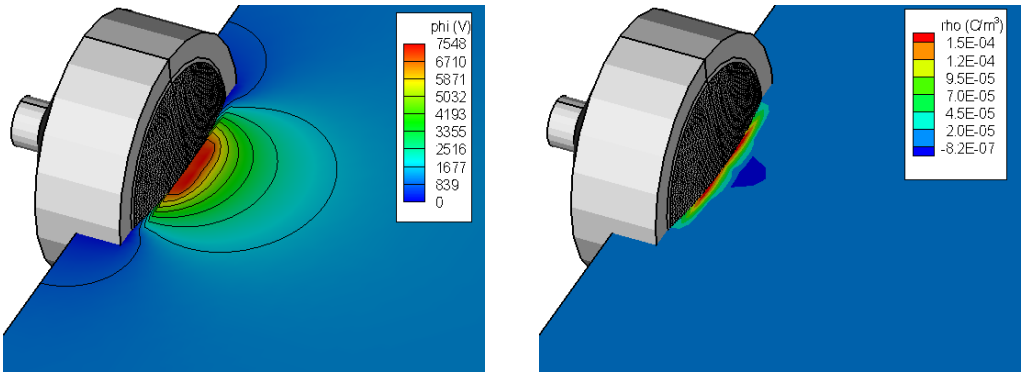
Figure 4.5: Ion and electron number density after 30,000 time steps, 2cm cell size

3× longer to converge. The large added performance cost, with only a small improvement in potential, lead to the decision to use DADI as the potential solver in the study shown here.

4.5 Induced Virtual Anode and Dimensional Scaling

One of the first tasks undertaken in modeling of the ion beam neutralization was a study of the containment of electrons in the ion beam. The flow was assumed to be already neutralized, which was approximated by emitting both the ions and electrons from the optics. The electrons were emitted at the ion velocity (34,400 m/s) and their temperature was 1eV. Only a quarter of the symmetric domain (figure 4.4) was modeled using a $40 \times 40 \times 60$ grid, with a uniform cell size of 2cm.

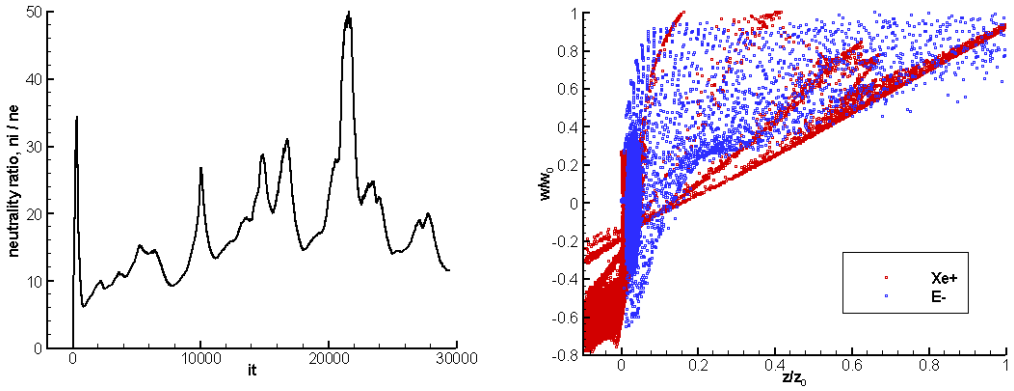
The simulation ran for approximately 30,000 time steps, with time step automatically adjusted to 75% of cell size. The ions were expected to form a distinct beam, with the electrons forming a neutralizing cloud surrounding the beam. However, as figure 4.5 shows,



(a) Potential, V

(b) Charge density, $\hat{q}/\mathrm{m}^{-3}$

Figure 4.6: Potential and normalized charge density after 30,000 time steps, 2cm cell size



(a) nr, ni/ne

(b) z-w phase plot

Figure 4.7: Neutrality ratio (number of ions / number of electrons) and normalized axial velocity phase plot for both ions and electrons. Velocity was normalized by the maximum velocity of each specie

the result was surprisingly much different. Not only are the electrons confined to a small region near the optics, the expected ion beam is completely absent.

The thruster seems to operate in the "virtual anode" mode [25]. The emitted ions lack the kinetic energy to pass over the potential hill of the virtual anode and are reflected back to the thruster. This reflection can be seen by the presence of negative velocity values in the axial velocity phase plot, figure 4.7(b). However, the question that needs to be answered is why the anode developed in the first place. The phenomenon of a virtual anode is associated with cases involving a high space charge, but the beam in this case was assumed to be well neutralized.

Yet, although a matching electron and ion current were injected from the thruster, the resulting plasma is highly non-neutral. The neutrality ratio is shown versus the iteration number in figure 4.7(a). At one instance, the simulation contains almost 50 times as many ions as electrons. This discrepancy is tied to the concept of the Debye length. Plasma is a collection of charged particles, which, given a large-enough length scale, can be assumed to be quasi-neutral. The minimum length at which quasi-neutrality can be assumed is given by the Debye length:

$$\lambda_D = \sqrt{\frac{\varepsilon_0 k T_e}{n e^2}} \tag{4.13}$$

where λ_D = Debye Length (m)
ε_0 = permitivity of free space ($8.885 \times 10^{-12}\mathrm{A \cdot s/V \cdot m}$)
k = Boltzmann constant ($1.38 \times 10^{-23}\mathrm{J/K}$)
T_e = electron temperature (K)
n_e = electron number density (m^{-3})
e = elementary charge (1.6×10^{-19} C)

Using the thruster operating parameters from table 4.1 indicates that $\lambda_D \sim 2 \times 10^{-4}$m, or roughly 1% of the used simulation cell size. The size of the simulation cell provides the minimum distance at which unique characteristics about the plasma are known. A cell which is 100 times larger than λ_D implies that the plasma within the cell is quasi-neutral. However, a fully kinetic modeling requires that the actual local non-neutrality of plasma is resolved, otherwise the electron motion cannot be traced correctly.

Solution to this problem seems simple: the cell size needs to be decreased to the order of λ_D. However, decreasing the cell size 100 times, while retaining the span of the simulation domain, would require a million-fold increase in the total number nodes. Not only would the memory storage of such a massive mesh be problematic, the increased number of unknowns would drastically decrease the convergence rate of the Poisson solver. Using the ideal $n\log(n)$ convergence scaling indicates that time needed to obtain a field solution would increase 6 million times. Time per time-step would thus grow from several seconds to several years.

Obviously, performing a neutralization study on the full-scale geometry is not computationally feasible. Simulations presented in this paper were performed on *scaled-down* thrusters. A decrease in the physical dimensions of the geometry allowed for a decrease in the span of the simulation domain. The length of a Cartesian cell, dx, is related to the span, S, by $dx = S/nx$ and thus decreasing the geometry size by a factor f allowed for a decrease of the cell size by the same factor f, while retaining the total number of mesh nodes.

A fundamental requirement of scaling was that plasma dynamics is not altered by the scaling process. The scaled-down thruster had to produce plasma environment identical to

that produced by the full-sized device. The scaling thus had to satisfy two conditions:

- Dimensions of the scaled device should be small enough so that plasma Debye length can be resolved by the mesh
- The scaled configuration must produce plasma environment identical to the full-scale model

The characteristic fluid equations describing a collision-less, non-magnetized two specie plasma are summarized by Chen [17] as:

$$\rho = n_i q_i + n_e q_e \tag{4.14}$$

$$\vec{j} = n_i q_i v_i + n_e q_e v_e \tag{4.15}$$

$$\varepsilon_0 \nabla \cdot \vec{E_p} = \rho \tag{4.16}$$

$$m_j n_j \left[\frac{\partial \vec{v_j}}{\partial t} + (\vec{v_j} \cdot \nabla)\vec{v_j} \right] = q_j n_j (\vec{E_p} + \vec{E_a}) - \nabla p_j \tag{4.17}$$

$$\frac{\partial n_j}{\partial t} + \nabla \cdot (n_j \vec{v_j}) = 0 \tag{4.18}$$

The first equation, 4.14 shows that charge density, ρ, will be retained across scaling if $n_j^* = n_j$, where the star indicates the scaled down values. Current density, given by eq 4.15 is automatically retained if the $n_j^* = n_j$ condition is satisfied. However, since for the ion source

$$\vec{j} = \frac{\vec{I}}{\mathbb{A}} \tag{4.19}$$

where $\vec{j}$ = current density (A/m$^2\cdot$ s)
$\vec{I}$ = current (A)
$\mathbb{A}$ = source area, πr^2, (m^2)

equation 4.15 imposes condition $\vec{I}^* = \vec{I}/f^2$, where f is the geometry scaling factor.

Next relationship which had to be satisfied, and which is given by eq. 4.16, is Maxwell's first equation for vacuum. This equation is self-consistent, since the electric field arises according to the charge density, which is retained across scaling. The fluid equation of motion, given by eq. 4.17 indicates that velocity change is imposed by electric field and pressure. The pressure gradient, $\nabla p_j = \nabla n_j kT$, is retained by conservation of number density. The electric field term appearing on the RHS of the motion relationship is due to two sources: the plasma induced field, $\vec{E_p}$, and externally applied field $\vec{E_a}$. The plasma induced field is computed self-consistently and does not need to be adjusted. The externally applied field component appears only if a constant background field is specified, using DRACO's `draco_load_field` command, and would need to be scaled by f. Background fields were not used in this simulation. The final fluid equation is the conservation of mass, eq. 4.18, and it is retained by the number density equality.

Hence, physical parameters are retained self-consistently, but initial conditions need to be adjusted according to table 4.5.

The scaling factor used in this study was 100:1, which reduced the diameter of the beam from 40cm to 4mm. The cell size was reduced to 2×10^{-4}m, or $\sim \lambda_D$.

Table 4.3: Scaling Parameters

Scaled	Not-scaled
n^*	n
$\vec{v^*}$	$\vec{v}$
$\vec{I^*}$	$\vec{I}/f^2$
$\vec{E_p^*}$	$\vec{E}_p$
$\vec{E_a^*}$	$\vec{E}_a/f$
ϕ_a^*	ϕ

Near-surface particle interactions are influenced by the thickness of the plasma sheath. The presence of a charged object in the plasma results in the formation of a sheath around the object. The primary role of the sheath is to shield out any potential disturbances, by balancing the flow of charged particles to the object. Outside the sheath, the plasma potential is undisturbed, and the particle motion is influenced only by the local potential fluctuations (thin-sheath limit). However, inside the sheath, the particle motion is dominated by the potential difference between the sheath edge, and the charged object (thick-sheath limit).

Hastings [26] shows that the sheath thickness is related to the Debye length by

$$S = \frac{2}{3}\left(\frac{\sqrt{2}}{K^*}\right)^{1/2} \lambda_D \left(\frac{|q\phi_s|}{kT}\right)^{3/4} \tag{4.20}$$

where K^* ranges from $1/\sqrt{2\pi}$ for far field Maxwellian to 1 for far field monoenergetic distribution. Using the worst case $K^* = 1/\sqrt{2\pi}$ with an artificially high sheath potential, ϕ_s to temperature, kT, ratio of 10:2, results in

$$S \sim 4\lambda_D \tag{4.21}$$

The diameter of the scaled thruster is approximately $20\lambda_D$. The plasma can still be expected to operate in the thin-sheath mode.

4.6 Particle Boundaries

4.6.1 Surface Collisions

Since the thruster is not grounded, it acts as an isolated probe, which will collect current [17]. The current balance can be written for the case studied here as [26]

$$I_e(V_s) - I_i(V_s) - I_b(V_s) = I_{net} = 0 \tag{4.22}$$

Table 4.4: Electron currents for cases R1 through R6

Case	I_i	I_e
R1	1.2	0.6
R2	1.2	1.2
R3	1.2	1.5
R4	1.2	1.8
R5	1.2	2.4
R6	1.2	4.8

where V_s = potential on the thruster, V
I_e = incident electron current, A
I_i = incident ion current
I_b = current due to the ion beam

The above formulation retained only terms significant to the simulation. Equation 4.22 states that a potential on the thruster will change until the total incoming current equals the total emitted current. The change of potential needed to retain current equality results in the so-called *spacecraft charging*.

Proper solution of the spacecraft charging problem was outside the scope of DRACO's functionality. The current equality was instead retained by assuming a perfect conduction of the surface. Any electrons impacting the thruster were sent back to their original source and were re-emitted at the next time step. Ion surface collisions were not important, since the beam was expanding into an empty domain. The production of electrons at time step k can then be described by:

$$m_e^k = \dot{m}_{e,s}\Delta t^k + \frac{I_{e,a}^{k-1}}{q} m_e \tag{4.23}$$

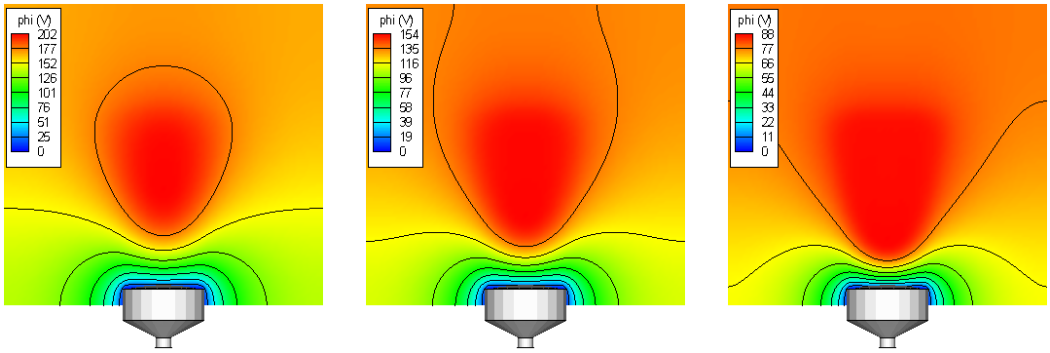
where m_e = mass of electrons to emit (kg)
$\cdot m_{e,s}$ = electron source production rate (kg/s)
Δt^k = time step duration (s)
$I_{e,a}^{k-1}$ = electron current absorbed at previous time step (A)
m_e = electron mass (kg)

4.6.2 Initial Results with Open Boundaries

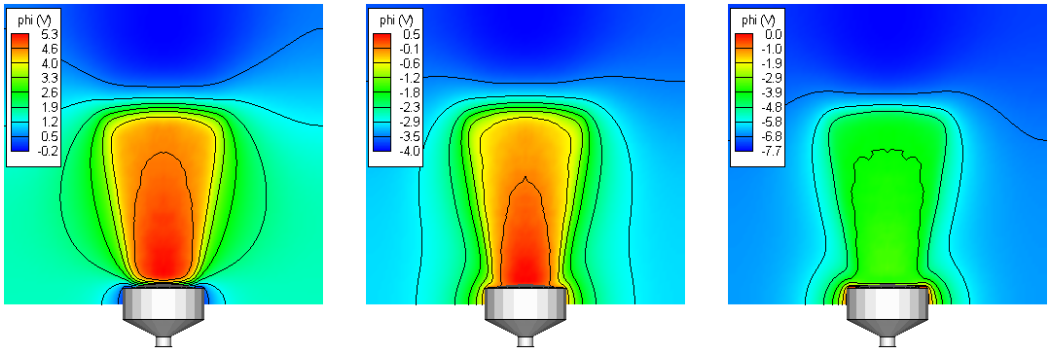
The external particle boundaries were initially set as *open*. Any electrons that passed through the external boundary were simply removed from the simulation domain. Yet, it soon became apparent that this treatment of the external boundaries was responsible for an introduction of a highly destructive numerical instability.

A good example of the effect of the instability can be shown by comparing results for six cases in which the electron-to-ion current was varied. These cases are labeled as R1 through R6. In each case, the ion current was 1.2A [2]. The electron current varied from 0.6A to 4.8A, as shown in table 4.6.2.

[2]The actual current was 1.2×10^-4 A, due to scaling, but for simplicity, all future references to input parameters will be made using their not-scaled equivalents.

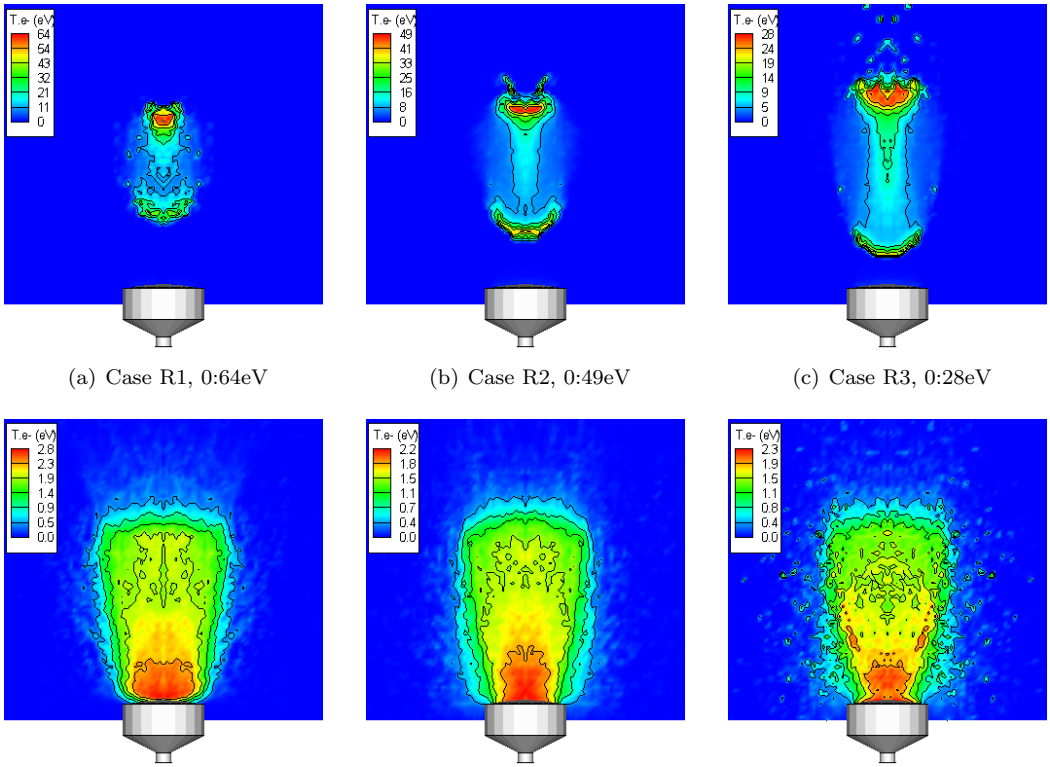


(a) Case R1, 0:220V (b) Case R2, 0:154V (c) Case R3, 0:88V



(d) Case R4, -0.2:5.3V (e) Case R5, -4:0.5V (f) Case R6, -7.7:0V

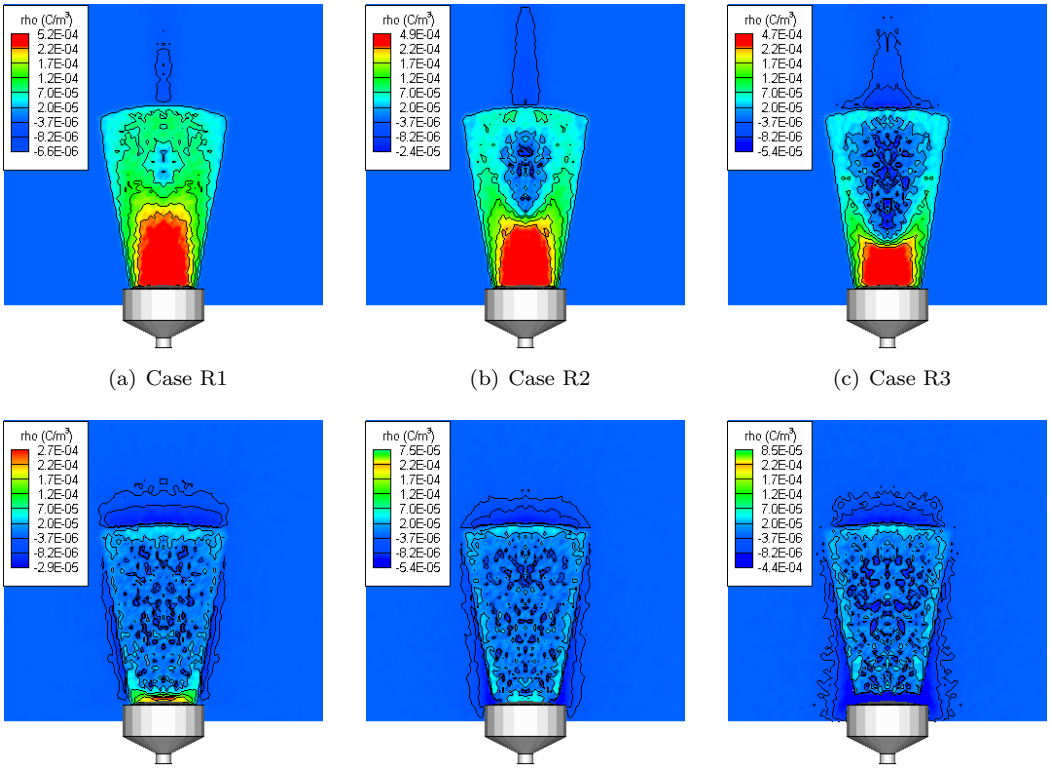
Figure 4.8: Potential (V) contour plots for cases R1 to R6 with range in potential values listed below each figure. Case R4 ($I_e = 1.5I_i$) develops the most physically reasonable solution.



(a) Case R1, 0:64eV (b) Case R2, 0:49eV (c) Case R3, 0:28eV

(d) Case R4, 0:2.8eV (e) Case R5, 0:2.2eV (f) Case R6, 0:2.3eV

Figure 4.9: Contours of Maxwellian electron temperature (eV) for cases R1 through R6. A significant electron heating can be observed for the cases lacking sufficient number of electrons to neutralize the beam. Electron temperature does not seem to be strongly influenced by over neutralization.



(a) Case R1 (b) Case R2 (c) Case R3

(d) Case R4 (e) Case R5 (f) Case R6

Figure 4.10: Contours of charge density (C/m^3). Two additional contours were manually added to highlight the formation of an electron sheath surrounding the ion beam. Charge density ranges from -4×10^{-4} to 5×10^{-4} C/m^3 in the core of case R1.

Plot of the electric potential after 3×10^{-7} plasma seconds for the 6 cases is shown in figures 4.8(a) through 4.8(f). Potential as high as 220V develops for case R1. Increase in electron injection current tends to lower the maximum potential, but surprisingly, the beam structure does not become apparent until case R4. Case R4 also seems to produce the most reasonable potential solution, with potential ranging from -0.2 in the sheath surrounding the thruster to about 5.3V in the core of the beam. The electron current is further increased in cases R5 and R6, but the impact on the beam potential is minimal. The electrons are highly mobile, and the extra electrons can easily be shed from the beam in cases R4 to R6. The same is not true for the ions. Due to their high mass, the ions do not respond rapidly to the increased beam potential. The high potential develops as a response to the buildup of a large positive charge.

Figures 4.9(a) through 4.9(f) show the effect of beam neutrality on the electron temperature. The lack of neutrality in case R1 leads to electrons heating to over 60eV. Interesting is the clear indication of two distinct electron families on both sides of the potential hill. The electron heating then seems to be induced by the flow of the electrons over the hill with subsequent flow reversal in the trout. The maximum temperature corresponds to the flow inflection point. Existence of two reversal points is clearly visible up to case R3, however the distance traveled by the oscillating electrons increases, which agrees with the observation of flattening of the potential gradient. Electron dynamics completely changes in case R4, shown in plot 4.9(d). The electrons no longer seem to oscillate rapidly around a potential hill, instead they follow the motion of the ions. Local heating occurs near the injection point, which follows from some electrons being trapped in a local high potential region. However, the electrons undergo a cooling as the beam expands. Electron temperature ranges from 2.8eV near the optics to about 1eV along the beam edge. Interestingly, over-neutralization in cases R5 and R6 has only a minor effect on the electron temperature, with temperature range remaining almost identical to case R4. Of importance is the introduction of radial fluctuation in case R6, which seems to be induced by electrons trapped between the sheath and the negative beam potential.

This behavior is further illustrated in plots 4.10(a) through 4.10(f). These plots show the beam charge density, ρ, in C/m^3. Same contour levels were retained for all 6 plots to better illustrate the effect of charge neutrality. The beam in case R1 is very poorly neutralized, with the bulk of the electrons residing in a central "bubble", corresponding to the location of the potential hill. Electron concentration grows in cases R2 and R3, but a good neutralization is not achieved until case R4. Cases R5 and R6 are not very different from R4, despite the increase in electron current. Interesting is the presence of an electron "jet" in cases R1, R2 and R3. This jet is not visible in case R4, R5 and R6. Instead, the beam is surrounded by what seems to be a sheath. This sheath structure is absent in the first three cases.

4.6.3 Numerical "Pump" Instability

A closer inspection of the potential, charge density and temperature results from the previous section clearly indicates that the results fall into two categories. First group consists of cases R1, R2 and R3, and is dominated by an extremely high potential, a non-neutral beam, and a high temperature plasma. Probably of greatest concern is the lack of any discrete plume structure in the potential solution. The plume structure becomes suddenly clearly visible in cases R4, R5 and R6. These cases form the second category of results. Suddenly, not

only does the beam structure become apparent, the charge density solution indicates a well neutralized beam, and electron temperature is within experimentally measured bounds.

Case R4 corresponds to $I_e = 1.5I_i$. On the other hand, the case which simulates the neutralization process of a real thruster is case R2, in which $I_e = I_i$. Judging from the solution, it seems that injection of an equal electron and ion currents should yield a highly non-neutral beam, with potential reaching 150V and temperature reaching 50eV. Of course, this simulation result is incorrect, since a large number of experimental measurements indicate that plume potential and temperature should be $\sim 5V$ and $\sim 5eV$.

An insight into this discrepancy is offered by the presence of the electron jet, which was mentioned in previous section. This jet is visible in cases R1, R2, and R3, but is not seen in R4, R5 and R6. Cases in which the jet did not develop resulted in a stable solution, and vice versa.

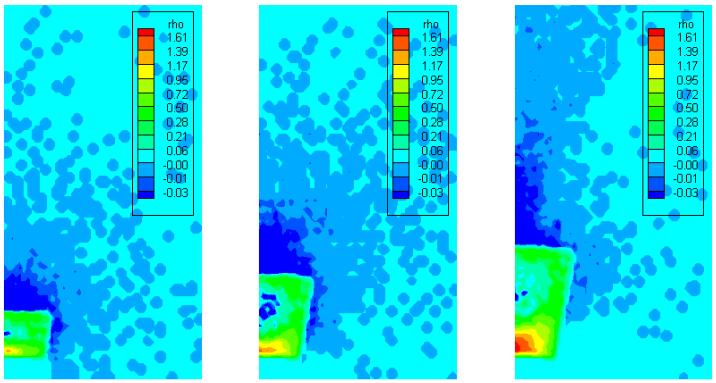
This jet has a profound effect on the simulation. It represents a region of fast moving electrons which are being "sucked-out" of the simulation domain. The result resembles a presence of a vacuum pump located beyond the ZMAX face; hence the *pump instability* label for the behavior. This effect can be seen in the time snapshots of charge density for case R2, which are shown in figure 4.11. Similar plots for potential are shown in 4.12. Initially, the beam is surrounded by the electron cloud, with some additional electrons randomly propagating through the simulation domain. The potential beam structure is also well developed. However, starting with iteration 1500, the electrons start flowing towards the ZMAX plane. The jet is well developed by iteration 2000, and by iteration 3000, no background electrons remain. This period is also associated with a rapid growth of the plume potential, and the loss of the well defined plume profile.

The time evolution of the plasma properties can also be seen in figures 4.13(a) through 4.13(d). These figures show the profile of potential, Maxwellian temperature, and ion and electron densities along the beam centerline. The axial dimension is shown in terms of the simulation cell index. The simulation contained 90 cells, with the nodes ranging from 0 to 90. The particles were injected at $k = 5$.

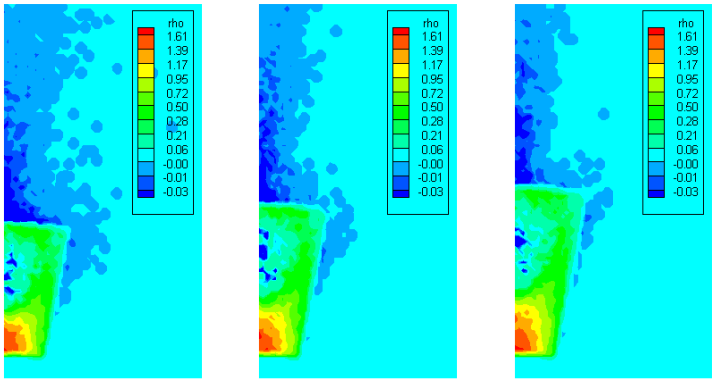
The instability is seen as the growth of the potential hill, without a corresponding drop-off in the post-hill region. Electrons introduced at $k = 5$ gain a large amount of kinetic energy as they travel up the hill. They overshoot the peak, and start traveling downwards. However, due to a lack of corresponding rapid potential drop on the downward side of the hill, the electrons are removed by the open boundary before they get a chance to return back to the domain.

The growth of the instability is also demonstrated by figure 4.13(b). Initially, the electron temperature follows the beam profile. However, starting with iteration 2000, a second hump can be seen and is clearly developed by iteration 4000, indicating the presence of two electron families. The first family is concentrated around $k = 30$, which seems to correspond to the center of the potential hill. Author's explanation is that these are the initial electrons which were introduced within the first 1000 iterations. These electrons were not affected by the growth of the potential gradient between the thruster and the beam, since they were already oscillating around the potential hill. In other words, these are the initial electrons with about 5eV of kinetic energy, which is not sufficient to travel down the post-hill grade.

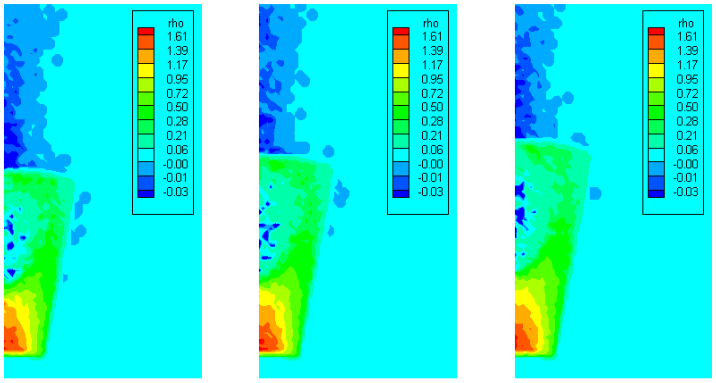
However, any electrons introduced after the first 1000 iterations *do* see the potential gradient, since they originate at $k = 5$. These electrons rapidly fly through the beam, and since the retarding potential drop is not sufficient, they are removed at the simulation



(a) it = 0500 (b) it = 1000 (c) it = 1500

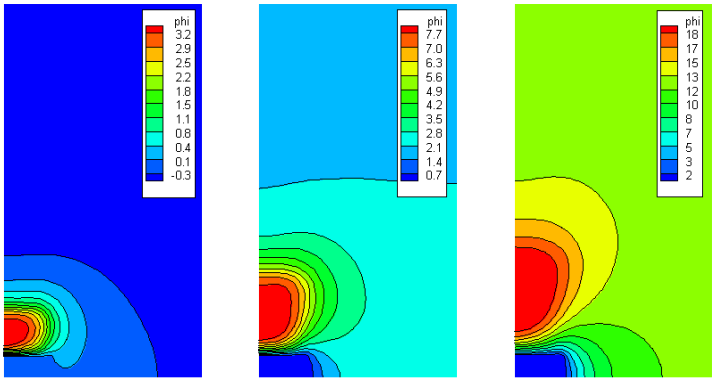


(d) it = 2000 (e) it = 2500 (f) it = 3000

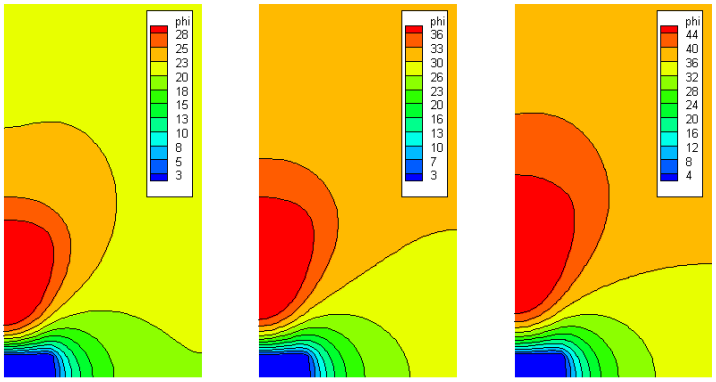


(g) it = 3500 (h) it = 4000 (i) it = 4500

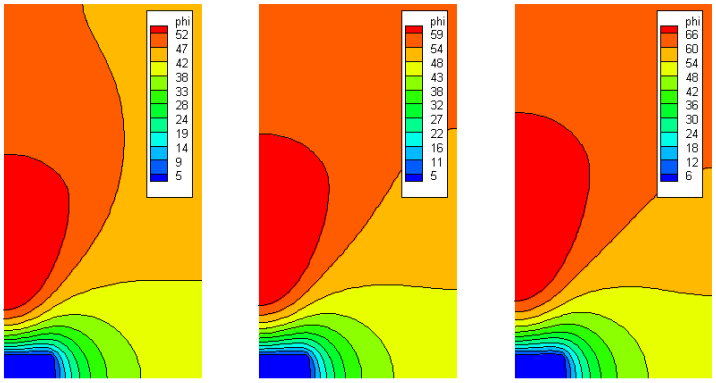
Figure 4.11: Charge density contours versus iteration number for R2. Removal of electrons at the boundaries introduces an instability which tends to "suck-out" the electrons from of the beam.



(a) it = 0500 (b) it = 1000 (c) it = 1500

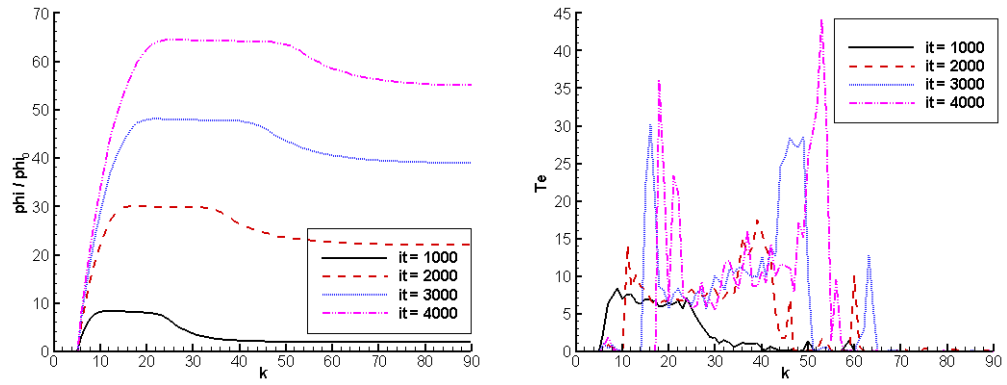


(d) it = 2000 (e) it = 2500 (f) it = 3000



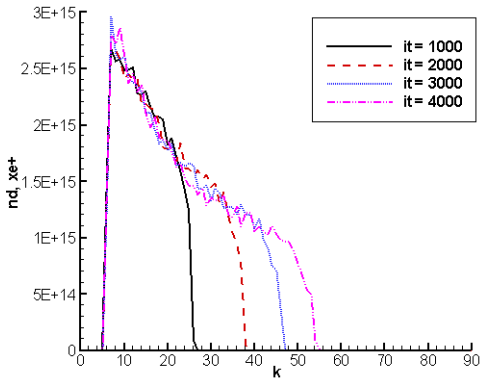
(g) it = 3500 (h) it = 4000 (i) it = 4500

Figure 4.12: Potential contours versus iteration number. The numerical instability results in a rapid growth of potential.

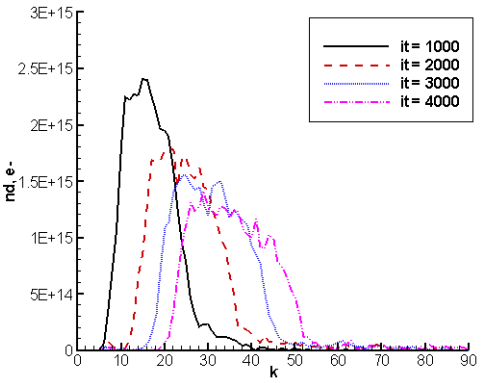


(a) Potential

(b) Maxwellian Temperature

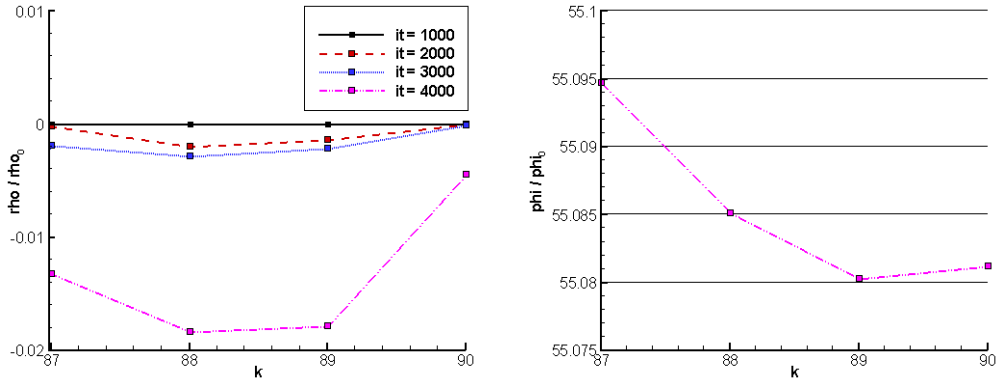


(c) Ion Density



(d) Electron Density

Figure 4.13: Growth of instability, shown by the axial profiles of plasma parameters versus iteration number. Axial distance, k, refers to the cell index, with $k = 90$ being the ZMAX edge of the simulation domain.



(a) Charge Density

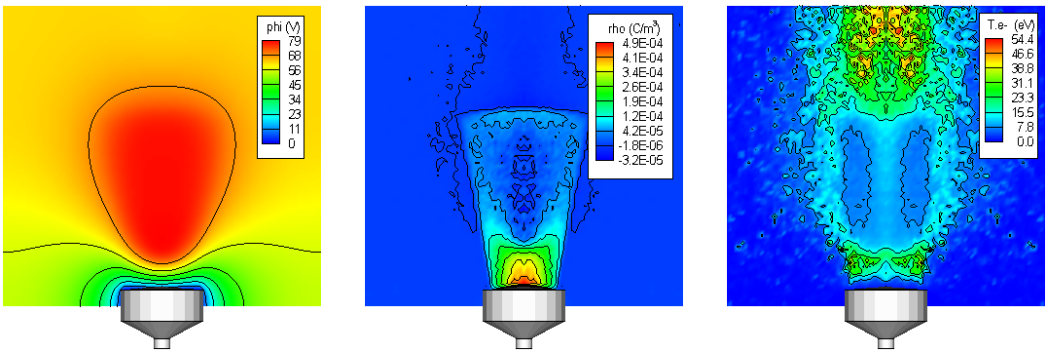
(b) Potential, it=4000

Figure 4.14: The instability is attributed to the removal of electrons at the simulation boundary. The removal results in the last cell ($k = 89 : 90$) containing fewer electrons than the second to last cell. Resulting potential gradient directs the electrons into the last cell, where they are also removed by the open boundary.

boundary.

These plots provide a good understanding of the influence of the instability on the solution. The instability results in a high beam potential, and a corresponding increase in electric field seen by injected electrons. Due to their high mobility, the electrons accelerate rapidly, but due to the finite domain, overshoot the beam, and are removed from the simulation. The ions do not respond very rapidly, and thus the potential grows even higher. The question which remains is what caused the initial instability?

Answer to this question lies in the numerical profile of charge density along the domain boundary. This plot is shown in figure 4.14(a). Initially, the charge density at the ZMAX face is zero, since no electrons are present. The random motion of the background electrons soon lowers this value. Assuming a constant background distribution, $\rho_{90} = \rho_{89} = -C$. However, the random motion through the open boundary will soon start depleting the number of electrons present in the last cell. Thus, very soon the charge density relationship changes to $\rho_{90} > \rho_{89}$, meaning that the second-to-last cell contains more negative charge. A small electric field develops, directing electrons from this cell to move to the last cell. This field can be seen if figure 4.14(b). These electrons will have a tendency to overshoot the last cell, and will also fall off the edge of the finite domain. This process thus starts depleting ρ_{89} and soon the effect cascades to cell at $k = 88$ and so on. The electron jet which has been described previously is simply due to the plasma attempting to correct an initial disturbance. The attempted fix is hindered by the open boundaries, and the disturbance grows rapidly. The seemingly good solution for case R4 is simply due to an over-saturation of the simulation with electrons. The production rate of new electrons is sufficiently high to overcome the electron loss at the boundaries.



(a) Potential (b) Charge Density (c) Temperature

Figure 4.15: Plasma parameters for case R2 with reflective boundary condition. ZMIN face was left as open, to retain a particle sink in the simulation.

4.6.4 Particle Reflection and Thermalization

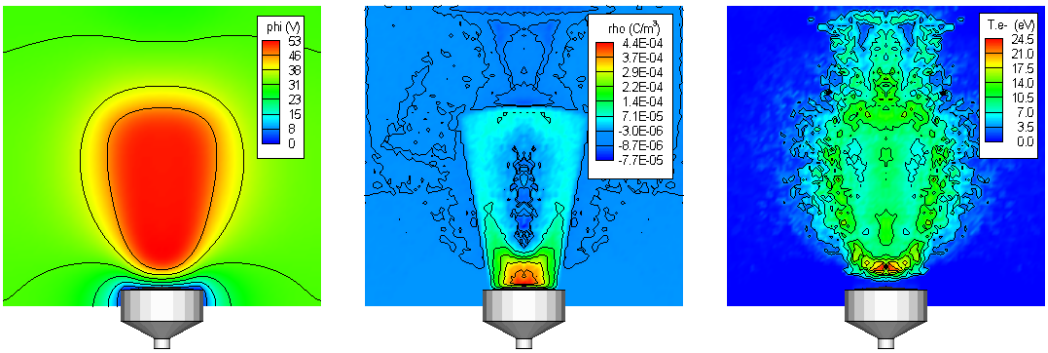
Several methods for particle treatment at the boundaries were investigated. In the early stages of this research, DRACO could handle only two types of particle conditions: open and reflective. Setting the external boundaries as reflective would aid in the electron retention, but could not be justified physically. Nevertheless, this method was investigated first.

As was mentioned previously, the spacecraft charging problem was avoided by re-emitting all incident particles. Setting all external boundaries as reflective would then be problematic, since no particle sinks would be exist in the domain. As cases R5 and R6 showed, plasma is capable of shedding extra unneeded electrons. Since this study was also concerned with the beam response to over or under neutralization, it was important to allow the extra electrons to leave from the domain.

Hence, one face of the domain was left open. The face chosen for this was ZMIN, the face behind the thruster. Figure 4.15 shows the potential, charge density and temperature achieved for $I_e = I_i$. A slightly larger neutralizing electron cloud is achieved, but the results are still non-physical. The reflective boundary at ZMAX simply seems to simply reflect the electrons back towards the thruster, where they escape through the ZMIN face.

Reflecting the electrons may help delay the onset of the instability, but it does not help to overcome it once it develops. The injected electrons are given a large velocity boost from the high beam potential and are flowing at high velocities by the time they reach the ZMAX boundary. Reflection simply results in fast electrons moving towards the thruster. Assuming no energy losses, an electron which was injected exactly along the beam centerline will have just enough energy to reach the thruster. Here it is absorbed by the conducting thruster, and subsequently re-emitted at the next time step. The process thus continues. The presence of reflective boundaries has the effect of introducing electrons which rapidly oscillate through the beam, without being retained by the beam's potential hill. Any small deflection from the original path will result in the electrons being absorbed by the open ZMIN boundary.

Thus, another approach had to be taken. The idea was to *thermalize* the electrons at the boundaries. Removed electrons were reintroduced at their previous position, but



(a) Potential (b) Charge Density (c) Temperature

Figure 4.16: Plasma parameters for case R2 with thermal boundary condition. ZMIN face was left as open, to retain a particle sink in the simulation.

their velocity was replaced by a random Maxwellian thermal velocity, based on an user prescribed wall temperature. Reasoning for this formulation was based on the assumption that the initial removal of electrons at the boundary is due solely to the random motion of background "thermal" electrons. Reinjecting the electrons with some other random velocity simply continued the random motion in and out of an additional reservoir.

Results for this case can be seen in figure 4.16. Once again, the ZMIN boundary was left open. The results are still incorrect, but they start to approach the expected behavior. For instance, the second population disappears from the temperature distribution. The beam is hottest near the thruster, and cools in response to the density decrease. However, thermal electrons along the edges shared with ZMIN will still eventually escape through the open boundary. This loss leads to a depletion of the thermal electrons along the remaining boundaries.

4.6.5 The Energy Boundary Condition

Previous two runs indicated that the instability can be controlled by altering the particle boundary conditions. Yet, it also became apparent that the open boundary could not be retained on any of the faces, at least not as long as the existing simulation domain was used. Moving the ZMIN face further behind the thruster would allow for more electrons to reverse directions and flow back to the beam, but the added computational cost was too great.

It became clear that yet another model had to be developed. Let's assume that an electron is introduced at a bottom of a potential hill with zero velocity. The potential hill is time-invariant. This situation is sketched in figure 4.6.5. Conservation of energy requires that

$$\frac{1}{2}m_e v_e^2 - e\Delta\phi = E_0 \tag{4.24}$$

where E_0 is a constant. Initially, the entire potential energy term dominates. However, as the particle moves up the potential hill, it trades potential energy for kinetic energy. The

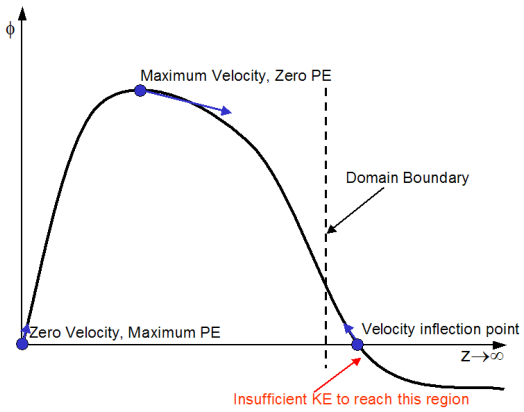


Figure 4.17: Conservation of energy in electron dynamics. Placing the domain boundary before the electron velocity inflection point results in a loss of electrons which are trapped in the potential hill.

potential energy is zero at the top of the hill, and the particle is moving at the fastest velocity it can achieve.

As the particle starts falling down the hill, the flow of energy reverses. The kinetic energy is transferred to potential energy, which manifests itself by slowing down of the particle. After traveling down the initial $\Delta\phi$, the particle will come to a stop. It cannot continue any further down the potential slope, since that would require a negative KE. The particle velocity will thus inflect, and the particle will start to travel up the hill. The electron is trapped in the potential hill gradient.

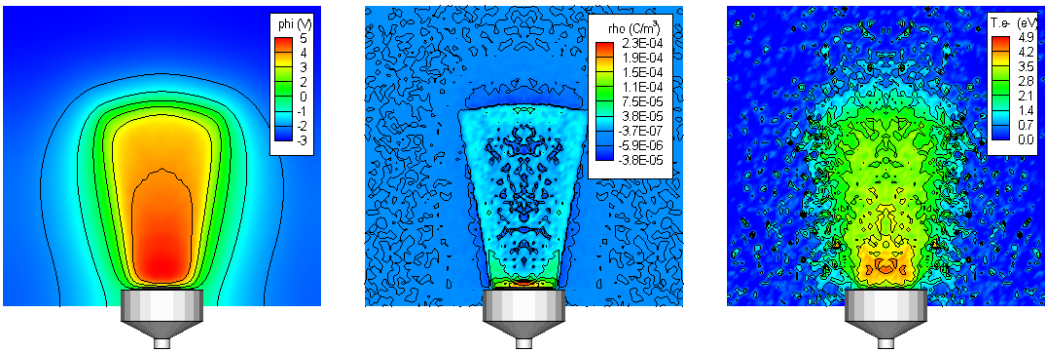
Due to finite domain dimensions, the inflection point may not be located within the simulation domain. Then, electrons which should remain trapped in the beam, are removed by the open particle boundaries. The newly developed energy particle boundary uses conservation of energy to reflect the trapped particles back to the simulation domain.

The potential at the inflection point is fixed by the user. Value of 0V was used during this simulation, which produced good results. However, comparison to measurements taken on board a real spacecraft would require a difference of several volts to account for increased spacecraft potential due to charging. The maximum potential energy for an electron is then

$$PE_{max} = q\left(\phi_{max} - \phi_{\infty}\right) \tag{4.25}$$

where ϕ_{max} is the potential at the core of the beam, and ϕ_{∞} is the user specified potential at infinity. The kinetic energy of the particle is computed from $KE = 1/2 m_e v_e^2$. If $KE \leq PE_{max}$ the particle is reflected back. Otherwise, it is removed. Hence, this boundary acts as a filter, allowing high energy particles to escape. Presence of too many electrons in the beam will bring down the beam potential, and thus decreasing PE_{max}. The kinetic energy of the newly created electrons, which over-neutralized teh beam, will be too high, and they will be removed from the simulation domain.

New results for R2 using the energy boundary condition are shown in figure 4.18. The solution is much better, with maximum potential in the beam of about 5V. These results are



(a) Potential (b) Charge Density (c) Temperature

Figure 4.18: Plasma parameters for case R2 with energy boundary condition. ZMIN face was left as open, to retain a particle sink in the simulation.

analyzed in a greater detail in the following chapter.

4.7 Hollow Cathode Model

Neutralizing electrons are provided by a hollow cathode. The device consists of a hollow chamber wrapped by an insert material. The electrons are produced by an isothermic emission from the insert material, which is manufactured from a material with a low work function, usually a barium-calcium-aluminate [27]. One side of the chamber is connected to a neutral gas supply line. The cathode is initially heated to about 1000K by externally wrapped heater coil. The emitted electrons will ionize the neutral gas, and will be drawn out of the cathode through a small orifice in the other side of the chamber. An external electrode, called the keeper, is used to regulate the flow of electrons. The extraction is initiated by applying a large positive potential difference between the keeper and the cathode common.

The actual processes governing the hollow cathode operation are not completely understood. However, several authors developed models describing the cathode operations. Salhi [28] developed an analytical model based on basic physical laws. Additional studies were done by Katz and Mandell [29].

Boyd [30] developed a detailed hybrid numerical model of the plasma environment produced by a hollow cathode. The model tracks the heavy ions and neutrals using the PIC algorithm and describes the electron dynamics by the solution of the electron continuity, momentum and energy equations. Direct Simulation Monte Carlo (DSMC) method of Bird [21] is used to introduce collisional scatter and charge exchange ions.

The model presented in this work differs from these previous models in two areas. First, a fully kinetic approach was taken, with both the electrons and ions being tracked using the PIC algorithm. Secondly, the actual plasma environment inside and near the cathode was not investigated. The primary topic of this research was the study of the electron dynamics in an ion beam neutralization. The cathode model was responsible for the introduction of the neutralizing electrons at a correct initial displacement from the ion beam, with an appropriate

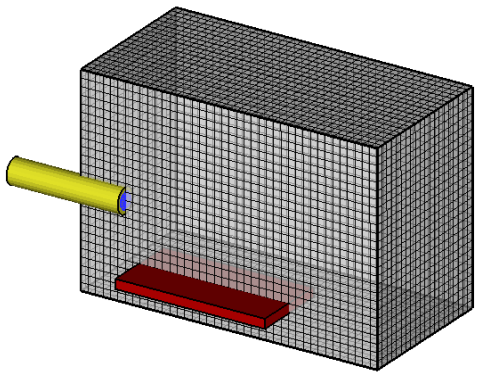


Figure 4.19: Simulation setup used to model electron flow from the cathode. The red plate indicates the target anode. The electrons are produced from the smaller blue surface on the cathode body end plate.

initial velocity distribution.

Following assumptions were made in the modeling of the cathode:

- No ions or neutrals are generated by the cathode
- Collission and friction terms do not play a significant role
- The external potential is not disturbed by the keeper electrode

A low scale modeling of the electron emission from the cathode was performed. The cathode was approximated as a 12cm long cylinder of a 2cm diameter. The electrons were injected from a source area on the front face of the cylinder. The diameter of the injection surface was 1.2cm. The ion beam was approximated by centering a 1cm thick flat plate 7cm below the cathode centerline, as illustrated in figure 4.19. Potential difference of 5V was applied between the collector plate and the cathode. The electron injection current was 1.2A. The electrons were injected with the Maxwellian distribution at 1eV.

The mesh size was set to 1×10^{-4}m, or half the cell size of the mesh size used in the ion beam model. This mesh size reduction corresponds to introducing a level 1 refined mesh around the cathode. The Neumann, $\partial\phi/\partial n = 0$ boundary condition was applied on all external faces. Electrons crossing the domain boundary were removed from the simulation. Due to symmetry, only a half-domain was simulated. The total number of cells was $20 \times 26 \times 40$.

Two distinct modes of operation developed according to the initial injection velocity, as show in in figure 4.20. The left view shows the cathode operating in a plume mode, which developed if the injection velocity was on the order of 1×10^6 m/s (about 3eV of kinetic energy). This mode is characterized by a large elongated electron region with a relatively low density. However, if the initial velocity was decreased below a critical value, the plume disappeared, and a high density spot developed at the cathode exit. The effect of these two modes on the current collected by the plate is illustrated in figure 4.21. Although the plume seems to indicate a better electron extraction from the cathode, the current collected by the

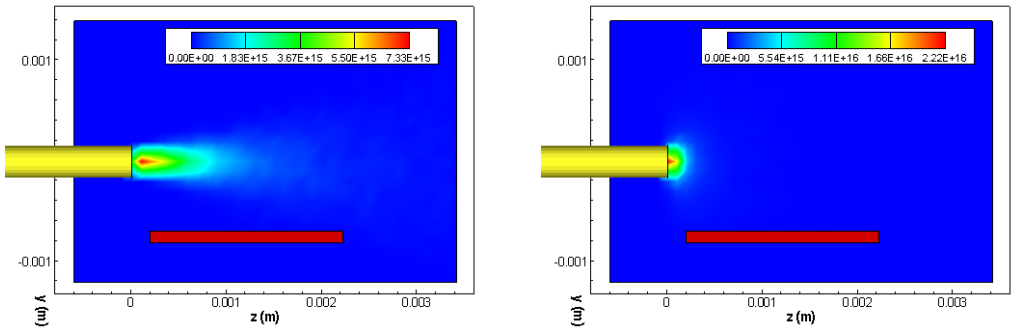


Figure 4.20: Electron density profile for the two operating modes: plume mode (left) and spot mode (right)

plate is actually lower than that collected in the spot mode. However, even in the spot mode, the maximum collected current was only about 0.3A, or a quarter of the emission current.

Figure 4.22 shows the potential distribution along the plane of symmetry. Also shown is the variation in the potential along the cathode axis for the two operating modes. A potential drop of about 1V develops in the spot mode. The drop is not well resolved, mostly due to the large cell size not being able to resolve the local λ_D. The conservation of energy

$$\Delta v = \sqrt{\frac{2q}{m}\Delta\phi} \tag{4.26}$$

thus indicates that at least 6×10^5 m/s of initial velocity is necessary for the electrons to get across the drop. This potential drop is responsible for the creation of the spot mode in this simulation. Particles without sufficiently large initial velocity are reflected back to the cathode, in a process analogous to the development of a virtual anode in space-charge limited ion flows.

Electrons with a sufficient initial velocity will next undergo a very large axial velocity increase. The height of the potential hill is about 3V, indicating a velocity gain on the order of 1×10^6 m/s. The potential gradient outside the cathode tip near region is not large enough to turn the electrons back to the anode, and most electrons escape the simulation domain. The acceleration is larger for the spot mode, indicating that the current collected in this mode should be smaller to that collected in the plume mode, which contradicts measurement of figure 4.21. This discrepancy is explained by assuming that the current collection in the spot mode is predominantly due to energetic electrons injected with a high off-axis initial velocity component.

The results from the previous paragraphs seem to indicate that the electrons are not capable of flowing into the ion beam, which directly contradicts many experimental measurements showing the plume potential to be on the order of several volts. Increasing the potential on the anode actually decreases the current collection, by increasing the axial potential gradient at the cathode tip.

The problems presented in the previous paragraphs are increased by the large cell size. As was demonstrated in the previous section, proper simulation of electron dynamics requires

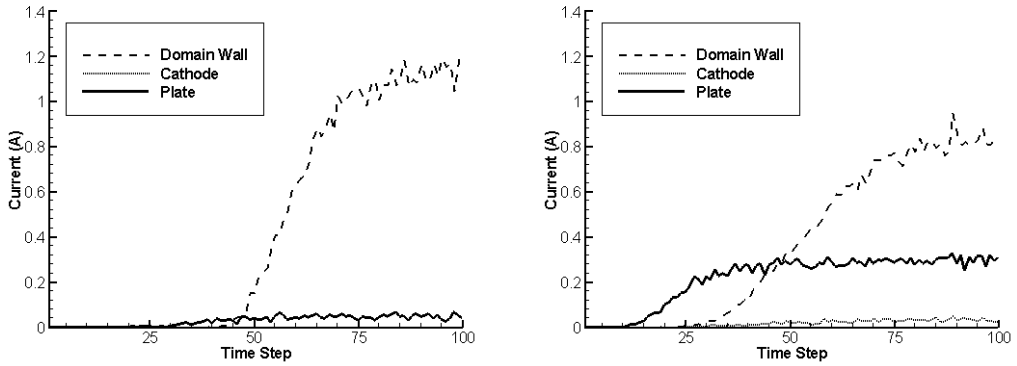


Figure 4.21: Current collection versus simulation time step for the plume (left) and spot (right) modes. "Domain Wall" indicates electron current leaving the simulation domain. Cathode emission current was 1.2A

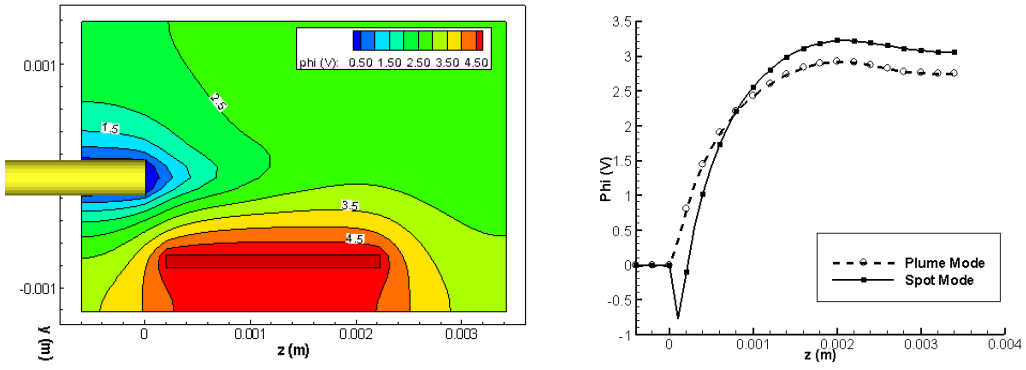
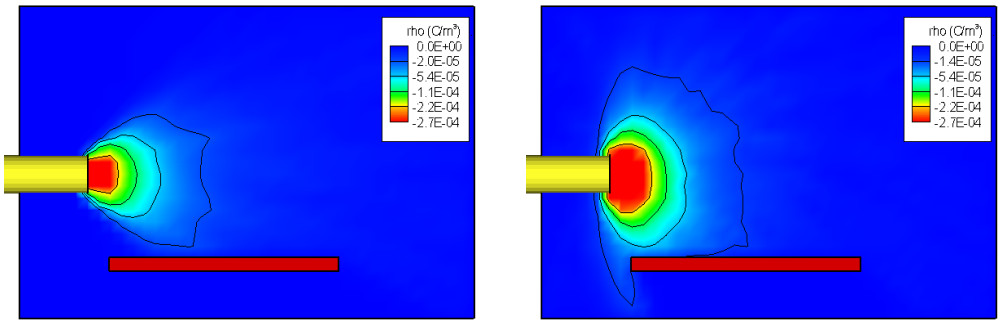


Figure 4.22: Left figure shows potential contours along the plane of symmetry for the cathode operating in the plume mode. The potential profile along cathode axis for the two modes is shown in the right figure. A discontinuity develops in the spot mode due to a large simulation cell size



(a) fixed ρ (b) fixed ρ and floating ϕ

Figure 4.23: Left figure shows the charge density after a limiting value of -2.7×10^{-4} C/m^3 has been applied. Second figure is the result of applying both the charge density limit and allowing the cathode potential to float. Floating cathode results in a good flow of the electrons into the "beam."

that the simulation cell is smaller than the local Debye length. However, decreasing the cell size in this region is not practical. It could be achieved by introducing a higher number of refined meshes. Unfortunately, the current state of the mesh refinement solver available in DRACO limits the number of mesh levels to 2. Decreasing the global mesh size was not practical as it would increase the simulation time, by both imposing smaller time steps (due to the requirement that $v_i \Delta t \leq h$, and increasing the convergence time of the Poisson solver. Additional scaling was also not feasible due to a possibility of introducing sheath effects into the solution. Decreasing the domain size would lower the ratio of L/λ_D, where L is a characteristic dimension of the thruster. Dimension scaling would thus modify the original thin-sheath problem into a thick-sheath problem.

Since the primary interest of this simulation was the transport of the electrons from the cathode location into the beam, proper resolution of the cathode near region was not of a large importance. Hence, the cathode model was modified by introducing a maximum negative charge density of $\sim -2.7 \times 10^{-4} C/m^3$. All mesh nodes with a charge density lower than this prescribed value were reset to the limit. The potential gradient was thus decreased, and the electrons could float out of the cathode even for low injection velocities. Electron density using this hack is shown in figure 4.23(a).

The electron expansion is better, but the injection velocity is still too high to allow the electrons turn to the beam. Due to the large cell size, the electric field outside the cathode exit is not properly resolved. Local variations in the electric field are smoothed out, and the electrons gain a large velocity boost in the "uphill" region which develops right at the cathode exit. Even with the limiting ρ, the electron behavior is almost identical to the case of large injection velocity!

Therefore, the potential on the cathode was allowed to float. By floating the potential, the electric field due to the difference between the object Dirichlet potential and the local charge density was eliminated. Resulting charge density is shown in figure 4.23(b). High electron density at the cathode tip induces a negative potential. The potential gradient between this region and the plate transports all electrons to the anode. This model is used

in the subsequent study of the ion beam neutralization.

Chapter 5

Study of Ion Beam Neutralization

5.1 Electron dynamics in an already-mixed beam

Simulation of the previously described Region II was used to establish a reference baseline for further comparisons with the actual neutralization modeling in which the electrons were injected from the cathode. Injection of the electrons from the optics approximates the "best-case" scenario for neutralization, in which all cathode electrons flow directly into the beam, and turn in the direction of the ion motion.

Simulation was performed on quarter domain with $50 \times 50 \times 90$ cells, with a uniform cell size of $2 \times 10^{-4} m$. The reflective particle boundary condition was applied along the two symmetric faces. The remaining walls were set as energy boundaries, with $\phi_\infty = 0$V. This domain is shown in figure 5.1. The simulation ran for 3×10^{-7} plasma seconds, with the time step automatically adjusting to 75% of reference cell travel distance.

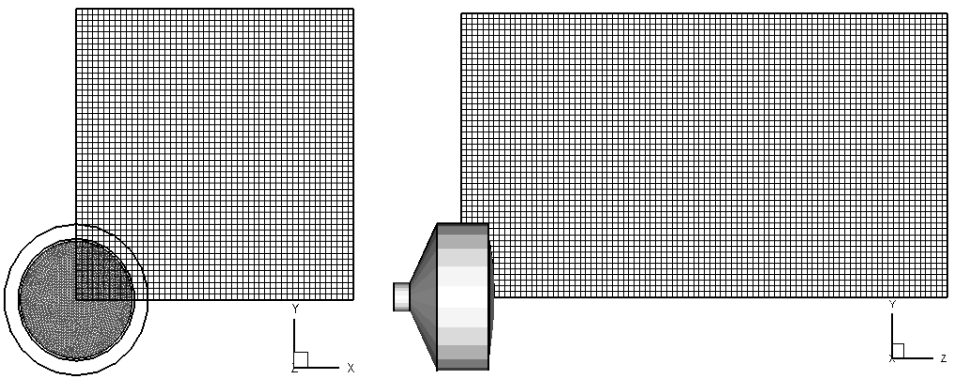
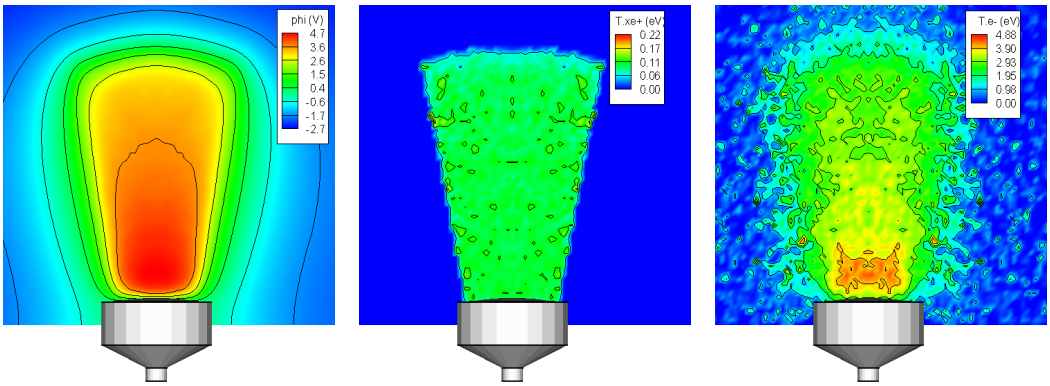


Figure 5.1: Simulation domain for modeling of the reference R2 case.



(a) Potential, -2.7:4.7V (b) Ion Temp., 0.1eV (c) Electron Temp., 0:4.9eV

Figure 5.2: Potential, and ion and electron temperature, R2 case

5.1.1 Plasma Properties

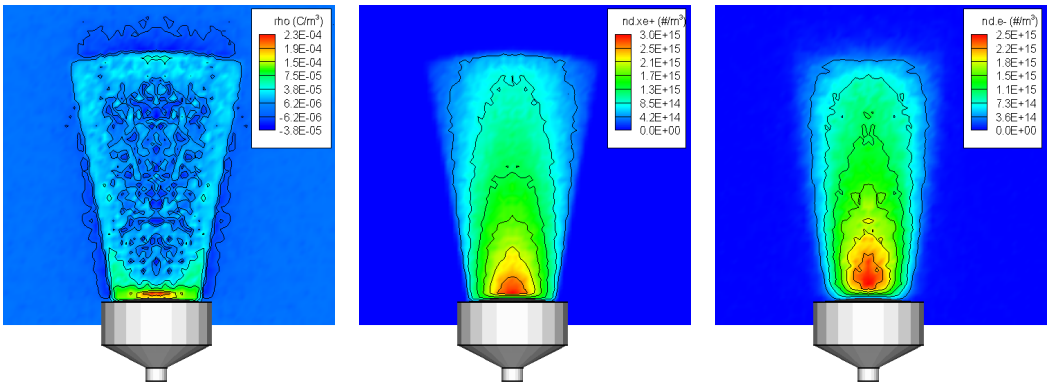
Figure 5.2 shows the potential, and ion and electron temperatures. The potential in the core of the beam is 4.7V. The ions do not undergo any significant cooling or heating, instead their temperature remains at the injection 0.1eV. The same is not true for the electrons. The electrons were injected at 1eV, however, the electron temperature near the beam core reaches 4.9eV. The temperature is seen to decrease with the decrease in beam density (due to beam divergence), indicating that the plasma follows some polytropic relationship. A more detailed analysis of the polytropic relationship is presented in a later section.

5.1.2 Electron Dynamics

An interesting observation can be made from figure set 5.3 which shows the contour plots of charge and number densities for the two species. The ion beam shows a slight divergence, which can be attributed to both the potential within the beam and the initial divergence due to the curvature of the optics. However, while the electrons demonstrate a similar divergence, the actual density profile is different, which is seen in the charge density plot. The electron cloud extends past the edge of the ion beam, forming a region of negative charge surrounding the beam. This region (a sheath) is responsible for the development of a potential gradient restricting the motion of the remaining electrons to within the beam interior.

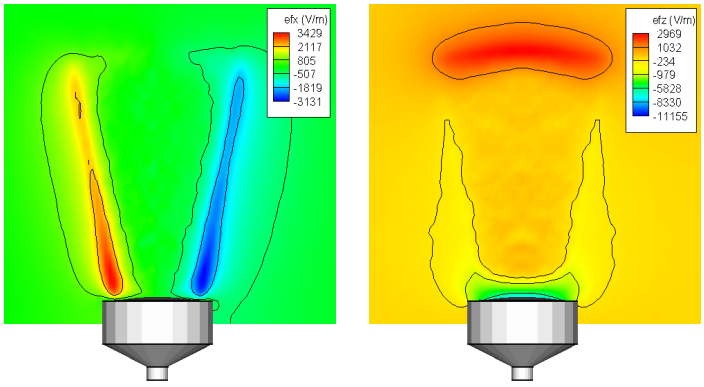
The effect of the sheath can be seen in the plots of electric field, shown in figures 5.4(a) and 5.4(b). The plots show the electric field on the plane of symmetry, where E_y is zero. Interestingly, all components of the electric field are almost zero in the bulk of the beam. The electric field is concentrated along the edges of the beam. The electrons move in straight lines at nearly constant velocity within the beam, but are reflected at the beam edge, by the sheath-induced electric field.

Unfortunately, node-centered average velocity data was not collected during the simulation. Electron trajectories are thus visualized from the velocity vectors of sampled particles. A 2D and a 3D representation of the flow field is shown in 5.5. Interesting is the amount of



(a) Charge Density (b) Ion Density (c) Electron Density

Figure 5.3: Charge and number densities. The electron cloud is seen to extend past ion beam. Resulting negative sheath retains electrons within the beam.



(a) y-component (b) z-component

Figure 5.4: Electric field components plotted on the plane of symmetry, R2 case. The y component is zero on this plane, due to symmetry. Electrons are contained within the beam by strong radial and axial components along the beam edge. The electric field is almost zero in the beam, indicating that the electrons bounce within the beam at relatively constant velocities.

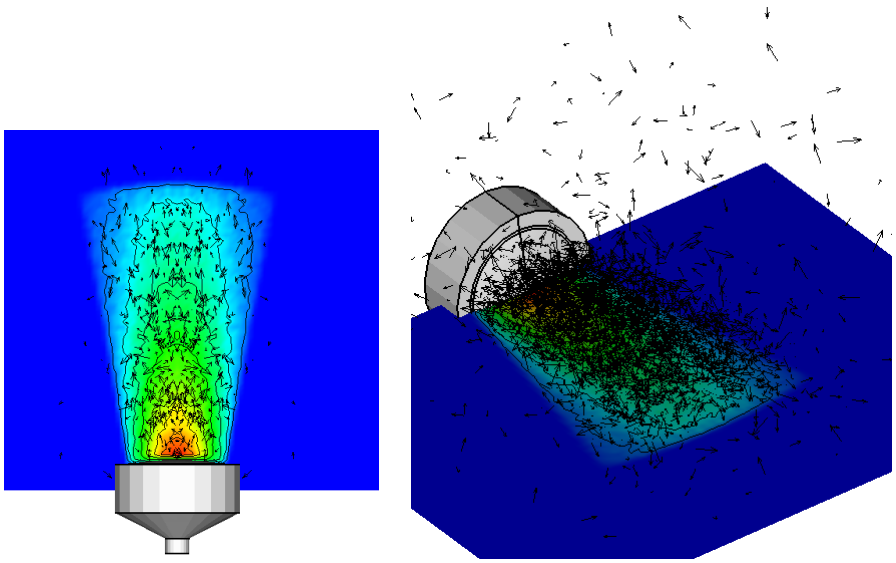


Figure 5.5: Electron velocity vectors. The ion beam profile is shown by the contour flood. The electron motion seems to be random, indicating that the electrons are bouncing inside the beam.

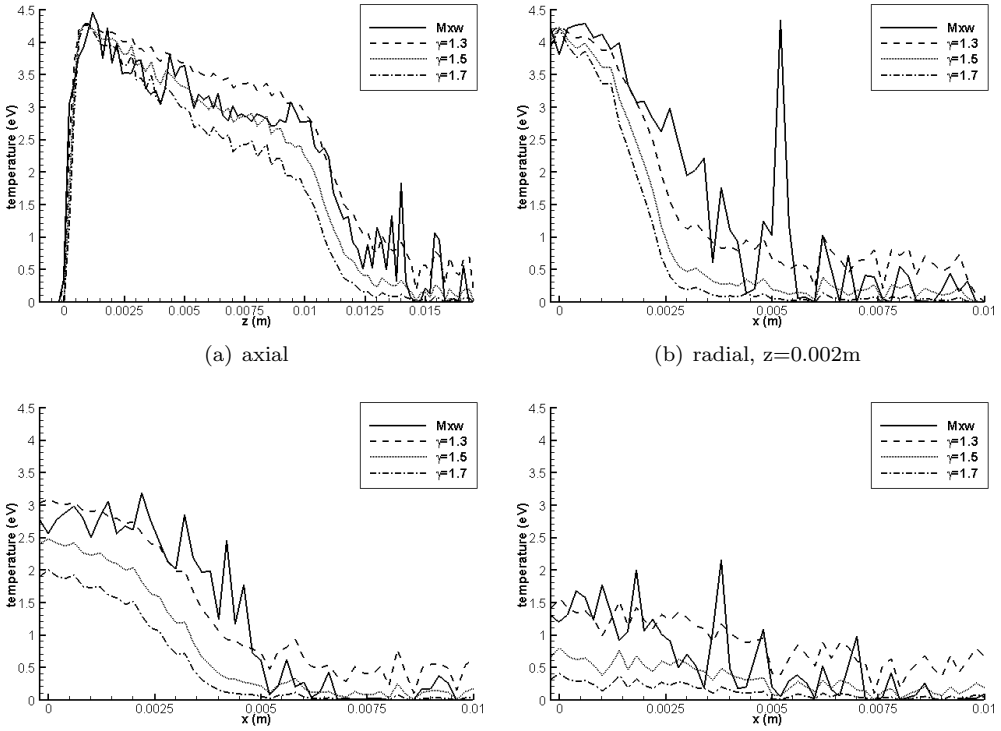
randomness in the motion of the electrons. The electrons were loaded from a 1eV Maxwellian distribution, with the mean velocity vector in the beam direction. The plot 5.5 indicates that the electrons do not have a strong "memory" of the initial loading. No z-directed motion is observed. The motion instead seems completely random, with about equal number of electrons flowing with and against the beam.

5.1.3 Polytropic temperature relationship

The polytropic relationship between gas temperature and pressure is

$$T = T_0 \left(\frac{n}{n_0} \right)^{(\gamma - 1)} \tag{5.1}$$

This relationship was plotted against the Maxwellian temperature for three values of γ along both the axial and radial directions. Reference values were chosen to correspond to the maximum electron Maxwellian temperature and density, 4.2eV and 2.5×10^{15} m^{-3}. The plots can be seen in figure 5.6. The relationship was plotted in the axial direction along the beam centerline, and at three radial positions. The radial plots roughly correspond to the beginning and end of the well defined beam, with the last cut taken in the electron sheath leading the ion beam. Neither of the three γ values was able to produce an exact match with the Maxwellian relationship, but the best agreement was obtained with $\gamma \sim 1.4$.



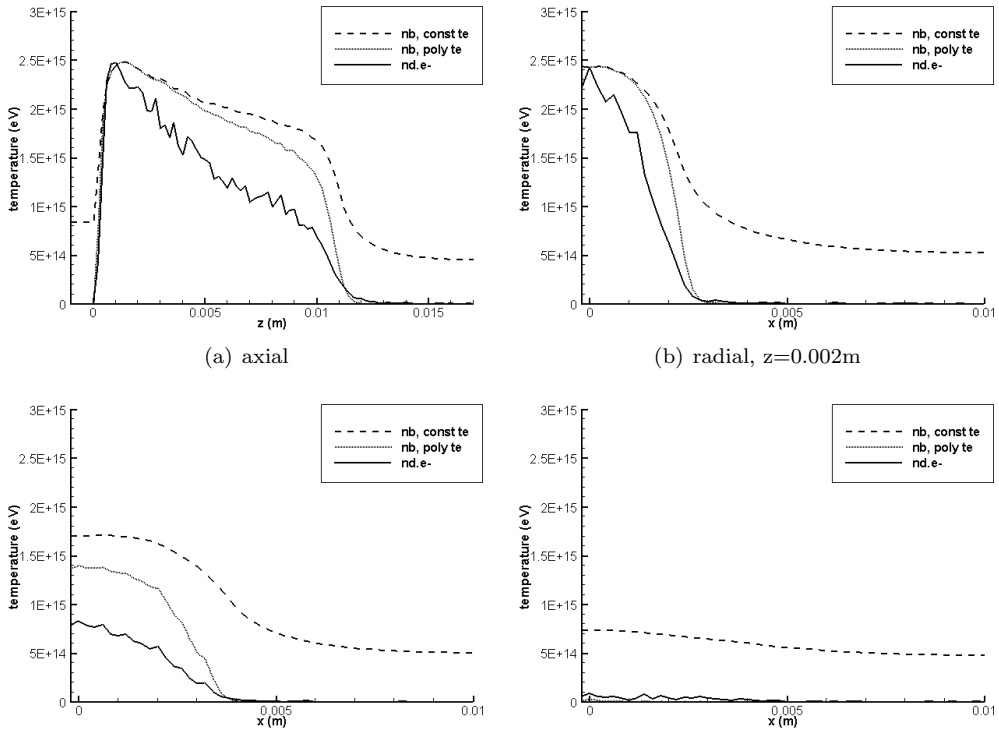
(a) axial

(b) radial, z=0.002m

(c) radial, z=0.01m

(d) radial, z=0.12m

Figure 5.6: Comparison of Maxwellian and polytropic temperatures for a several values of γ. $\gamma = 1.4$ seems to yield the best agreement.



(a) axial

(b) radial, z=0.002m

(c) radial, z=0.01m

(d) radial, z=0.12m

Figure 5.7: Comparison of simulation number density to one predicted by the Boltzmann relationship. The polytropic relationship was calculated with $\gamma = 1.4$.

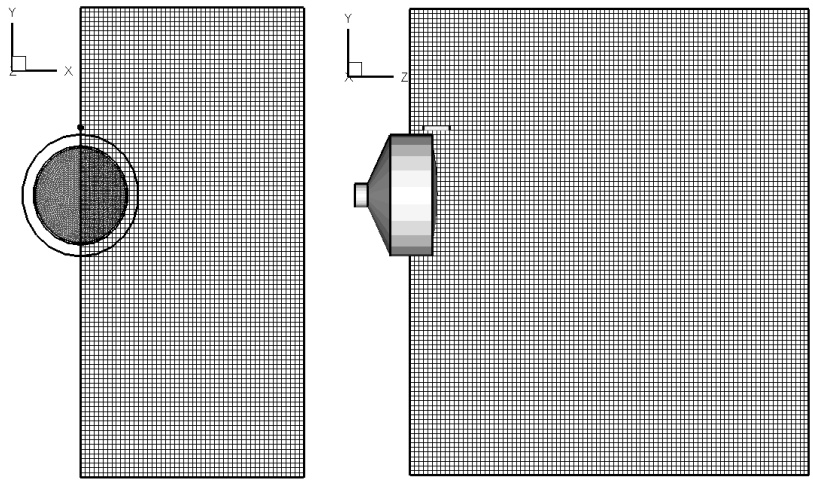


Figure 5.8: Simulation domain used to study the neutralization of a single ion thruster. Domain size is $50 \times 100 \times 90$ cells.

5.1.4 Comparison to the Boltzmann model

Simulation electron number density was also compared with the density predicted by the Boltzmann relationship. The reference density and potential were set to 2.5×10^{15} m^{-3} and 4.7V, respectively. The relationship was computed using both the constant and polytropic temperature models. In the constant case, the reference temperature from the previous section, 4.2eV, was retained. The polytropic temperature model was computed using $\gamma = 1.4$, as shown in the previous section.

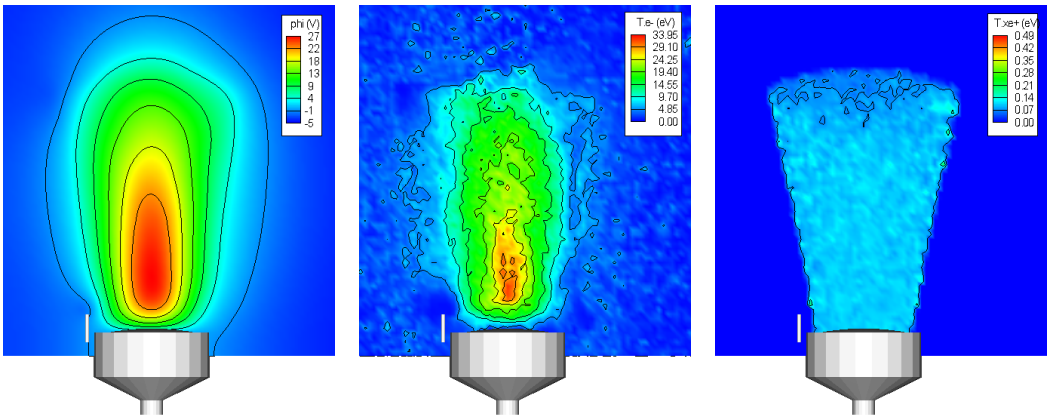
Generally, the agreement between the simulation and the Boltzmann relationship is not very good. The best agreement is reached near the core of the beam, since this location corresponds to the point at which the reference values were sampled.

5.2 Ion Beam Neutralization

5.2.1 Single Thruster

The previously described ion source was combined with the modified cathode model to simulate the entire neutralization process of an ion thruster. The simulation domain is shown in figure 5.8. The domain was symmetric along the x face, so only a half of the domain was simulated. However, a large number of nodes in the y direction was needed to capture the initial "overshoot" of the electrons. The domain size was $50 \times 100 \times 90$ cells. The simulation cell size was 2×10^{-4} m. The simulation was performed with geometry scaling ratio of 100:1. The simulation ran for 3×10^{-7} of plasma seconds, which corresponded to about 14 computational hours.

Potential, and ion and electron temperatures are plotted in figure 5.9. Although the potential solution shows a distinct beam profile, the maximum potential was about 5 times



(a) Potential, -5:27V (b) Electron Temp., 0:34eV (c) Ion Temp., 0.1eV

Figure 5.9: Potential, and ion and electron temperature, single thruster

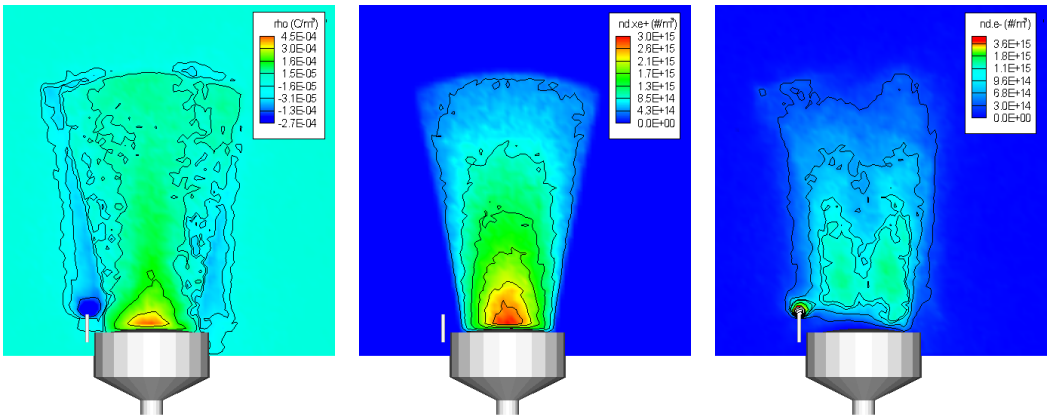
larger than what was expected from the reference pre-neutralized case. However, as was mentioned in the previous section, the potential on the cathode was allowed to float. $\Delta\phi$ between the cathode tip and the beam core is about $25V$.

Yet, the potential difference between the beam core and the thruster body is high, which seems to indicate that the neutralization is not perfect, with an insufficient number of electrons flowing into the beam. This observation is confirmed by the high electron temperatures which develop in the beam. The electrons heat up from 1eV at the cathode tip to about 15eV in the bulk of the beam, but temperatures as high as 32eV develop in the core. The ions do not undergo any noticeable heating, and their temperature remains at the injection 0.1eV.

The plots of charge and number densities are shown in figure 5.10. Although an electron sheath does develop around the ion beam, it is not as pronounced as in the case of already neutralized beam. Interesting is the clear development of the electron bridge linking the cathode with the main beam. The electrons seem to follow this bridge into the main beam, where they become trapped by they beam potential. However, the shape of the electron cloud does not accurately describe the ion beam. Instead, a strange two strip formation is seen.

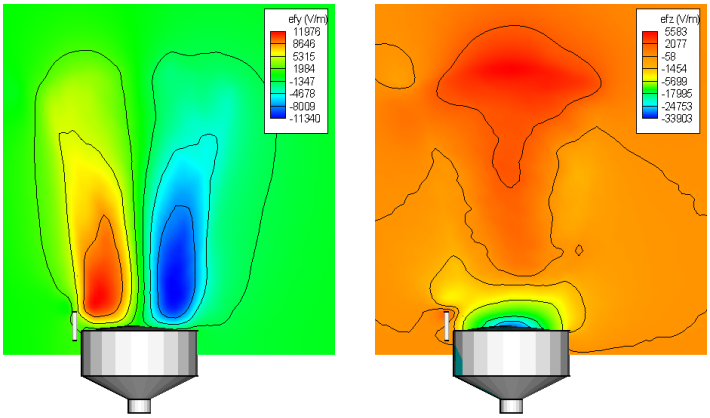
The two strips seem to be due to wedge-shaped profile of the z component of the electric field, which is plotted in figure 5.11. Another interesting behavior is the relatively insignificant influence of the cathode on the y component of the field. The motion of the electrons hence seems to be dominated by the potential which develops in the beam, with the cathode potential playing just a minor role. However, as was mentioned previously, the potential on the cathode was allowed to float. The width of the zero V/m region of E_y is also much smaller than what was observed previously. The electrons thus seem to be constantly accelerated or decelerated, with only a small portion of the beam dynamics governed by constant velocity motion. This result is most likely due to the high core beam potential.

Plot 5.12 shows the electron velocity vectors. The motion of the electrons seems to be fairly random, indicating that the electrons bounce back and forth inside the beam. Beam



(a) Charge Density (b) Ion Density (c) Electron Density

Figure 5.10: Charge and number densities, single thruster. A distinct electron bridge is observed, but the shape of the electron cloud does not correspond to the ion beam.



(a) y-component (b) z-component

Figure 5.11: Electric field components plotted on the place of symmetry, single thruster. The x component is zero on this plane, due to symmetry

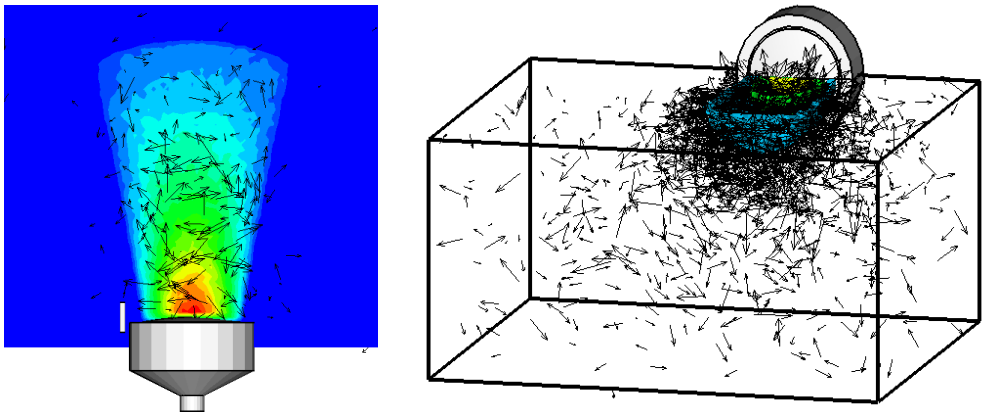


Figure 5.12: Electron velocity vectors. The ion beam profile is shown by the contour flood. The electron motion seems to be random, indicating that the electrons are bouncing inside the beam.

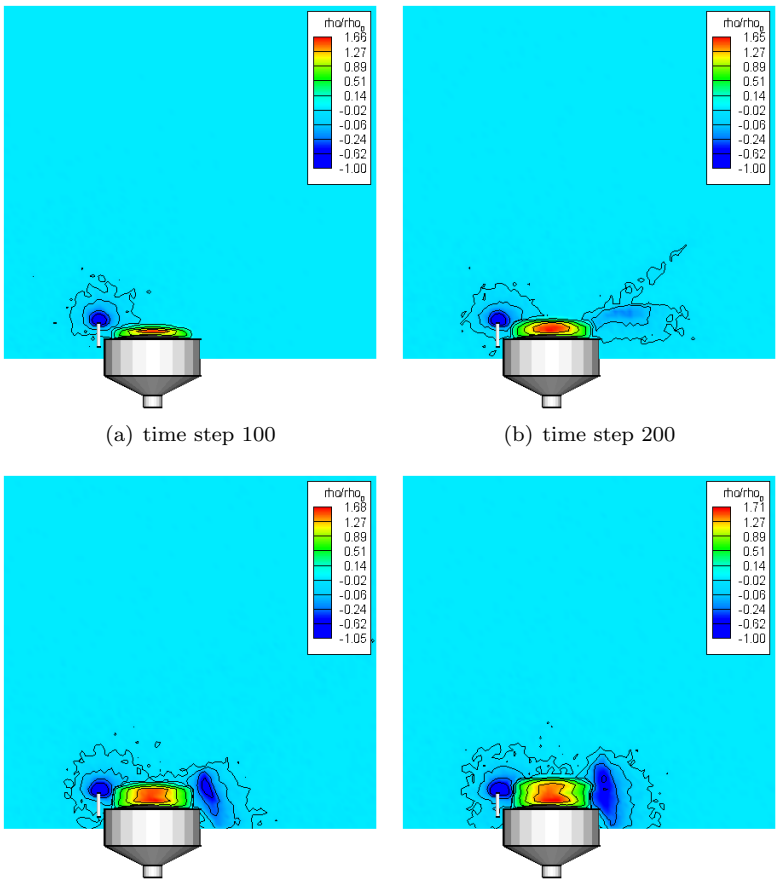
neutralization does not seem to indicate that the electrons start following individual ions. Instead, the neutralization is achieved by the fast transport of negative charge through the beam.

Solution time snapshots were used to analyze the evolution of the electron beam profile. The snapshots for the first 400 time steps are shown in the figure strip 5.13. These time steps correspond to simulation times of 1×10^{-8}, 2×10^{-8}, 3×10^{-8} and 3.7×10^{-8} seconds. The electrons are seen to initially overshoot the beam core, but are subsequently attracted back to it. After the initial ion beam profile develops, the new electrons flow directly into the beam. This behavior can be seen in the strip 5.14, which shows charge density for time steps 1000 through 4000 (8×10^{-8}, 1.4×10^{-7}, 2×10^{-7} and 2.6×10^{-7} seconds). The development of the sheath is very noisy, and the profile of the electron cloud varies widely between the snapshots.

5.2.2 Thruster Array with individual cathodes

Next, neutralization of a 2x2 cluster, shown in figure 5.15, was analyzed. The horizontal and vertical spacing between the thruster centerlines was 0.56m (0.0056m scaled). Each thruster was neutralized by an individual cathode, centered either above or below the thruster. The domain was symmetric, and hence just one quarter of the full domain was simulated. The reflective particle boundary condition was applied along the symmetric faces. The number of simulation cells was $60 \times 60 \times 100$.

The three-dimensional results were analyzed by extracting the data along two cutting planes. The first cut was made by joining the center of the cluster with the center of the simulated thruster. This *diagonal* cut is shown in 5.16(a). While this cut captures the region along the cluster centerline, it does not enclose the neutralizing cathode. The second cut was thus made through the *plane of the cathode*, by joining the cathode center with the thruster center. This cut is shown in figure 5.16(b).



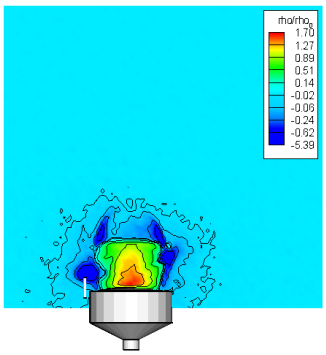
(a) time step 100

(b) time step 200

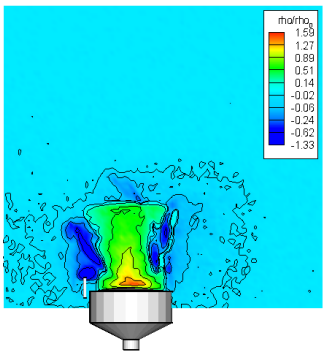
(c) time step 300

(d) time step 400

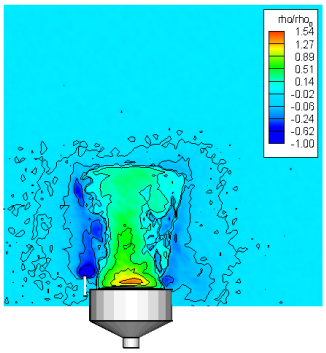
Figure 5.13: Charge density along the symmetry plane for the first 400 time steps. The dark blue regions indicate regions of high electron density. The electrons are seen to initially overshoot the beam.



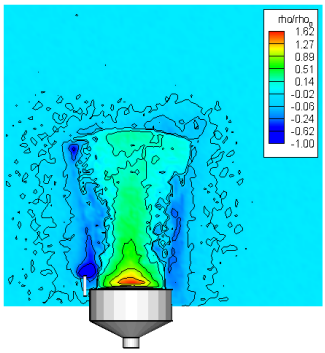
(a) time step 1000



(b) time step 2000



(c) time step 3000



(d) time step 4000

Figure 5.14: Charge density along the symmetry plane for time steps 1000 through 4000. New electrons flow directly into the beam, but the profile of the electron sheath fluctuates greatly between time snapshots.

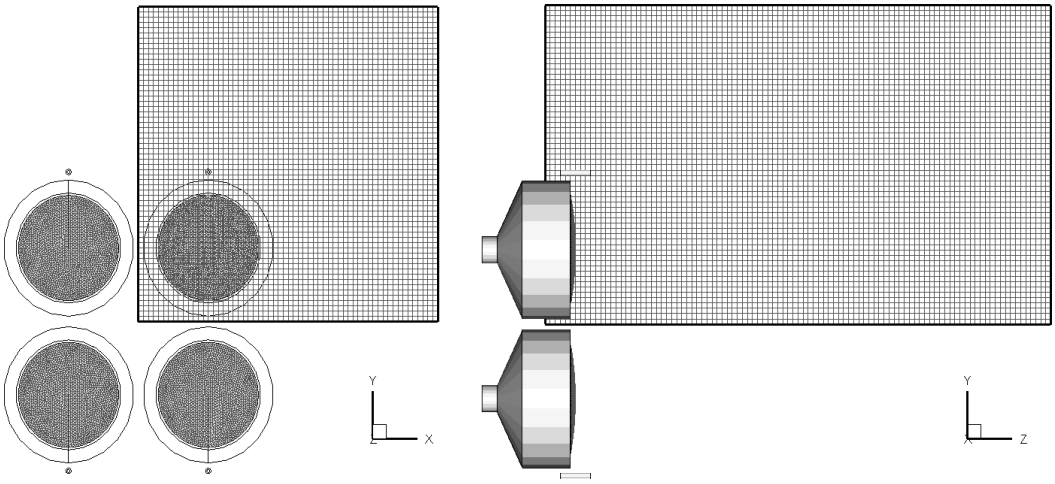
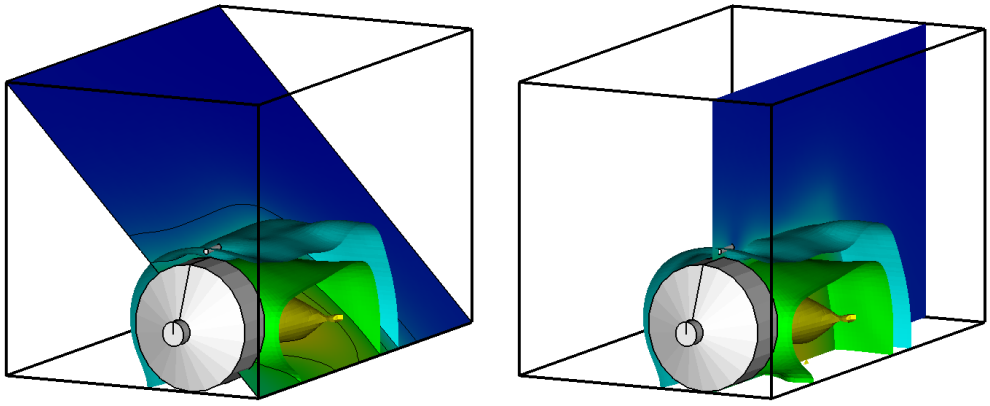
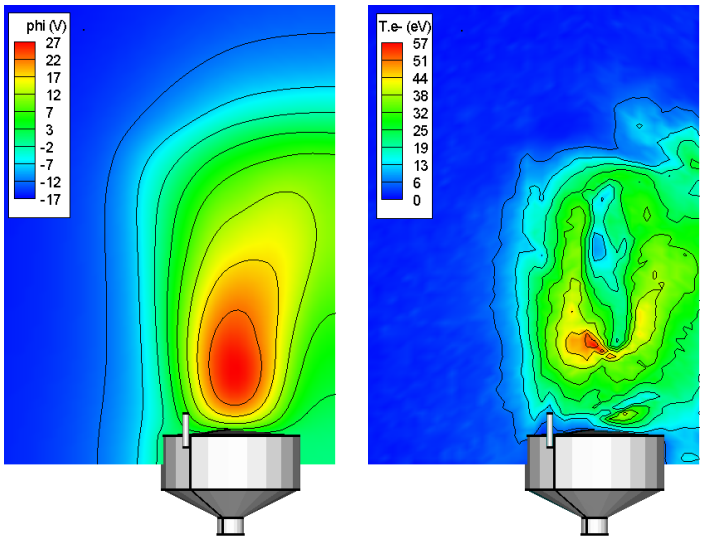


Figure 5.15: Simulation domain used to study the neutralization of a 2x2 cluster of ion thrusters with individual neutralizing cathodes. Symmetry allowed for the simulation domain to include just one thruster. The reflective particle boundary condition was applied along the symmetric faces.



(a) diagonal (b) cathode plane

Figure 5.16: Diagonal and cathode plane cuts used to visualize results for individual cathode cluster.



(a) Potential, -17:27 V (b) Electron Temp., 0:57 eV

Figure 5.17: Potential and electron temperature, plotted on the cathode plane.

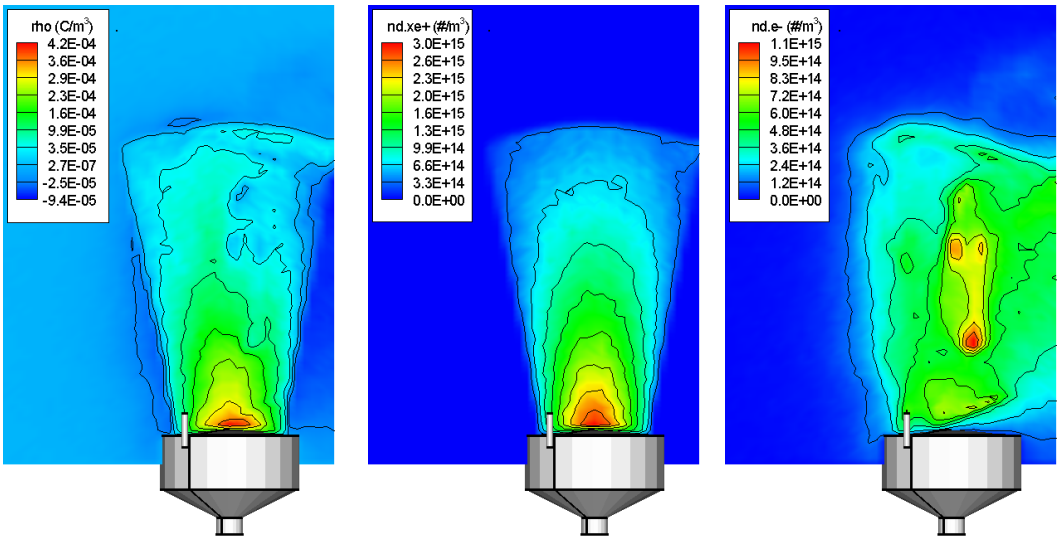
Potential and electron temperatures are plotted in figures 5.17(a) and 5.17(b). The total potential difference between the beam center and the ambient plasma has increased to 44V. The mixing of the ion beams is clearly visible by the bending of the potential profiles. Even more interesting is the presence of hot spots in figure 5.17(b). The structure of the spots is indicative of vortices, but further time-variant analysic needs to be made to further investigate this anomaly.

The electron sheath is visible in the charge density plot, 5.18(a). Interesting to note is the high concentration of electrons along the cluster centerline. An increased density was expected at the beam overlap, but the electron density concentration does not directly correspond to the increased ion density. Further, the electron density is fairly constant, with high density concentrated only in several seemingly random spots.

The electric field contours, shown in 5.19 also do not show any direct correlation between the field and the localized high electron density. The diagonal component was computed as a signed magnitude of the x and y field,

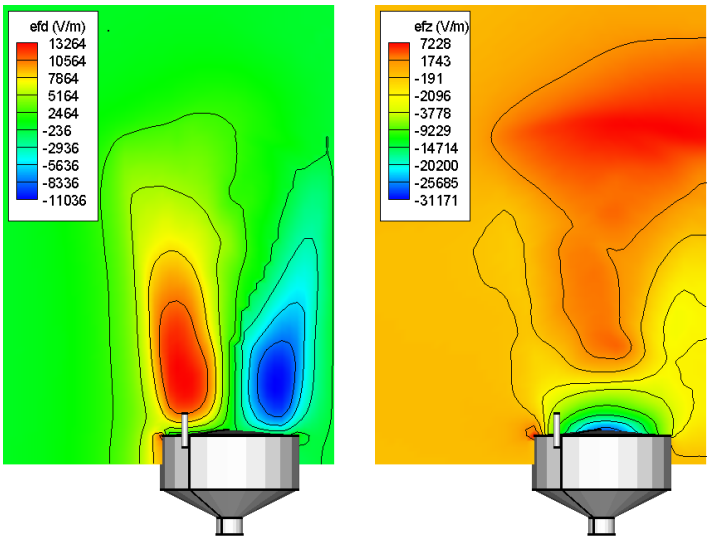
$$E_d = \text{sign}\,(E_y)\sqrt{E_x^2 + E_y^2} \tag{5.2}$$

An even more bizarre temperature profile can be seen along the cathode plane. Several low temperature streaks are seen to propagate through a mostly constant temperature region. These streaks correspond to focused electron beams. Figure 5.21(c) shows the beam to originate at the cathode, apparently reflect at the cluster centerline. However, the reflection most likely indicates a pass of the electrons from the current thruster to its neighbor. The electrons then flow towards the beam edge, where they disperse. Mixing of the ion beams is also more visible along this cut, since the vertical/horizontal distance between the plumes



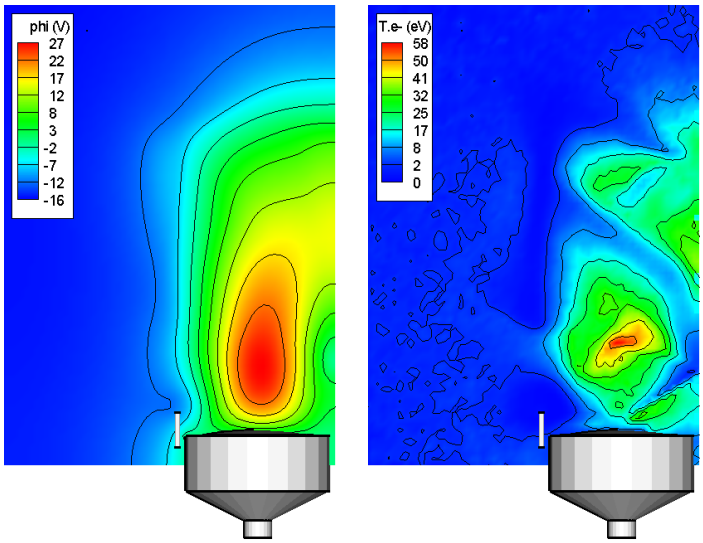
(a) Charge Density (b) Ion Density (c) Electron Density

Figure 5.18: Charge and number density, diagonal plane.



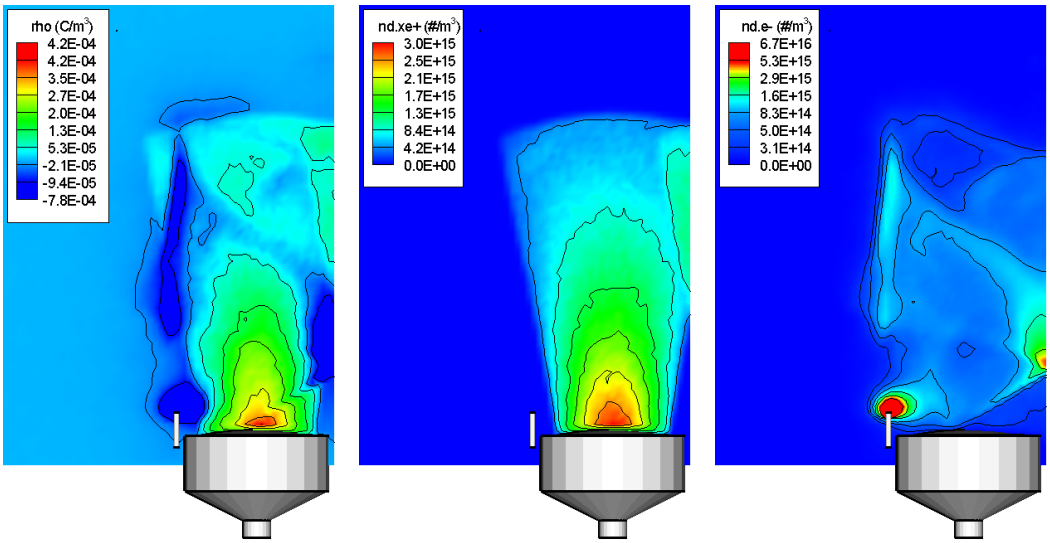
(a) diagonal component (b) z component

Figure 5.19: Electric field, shown on the cathode plane.



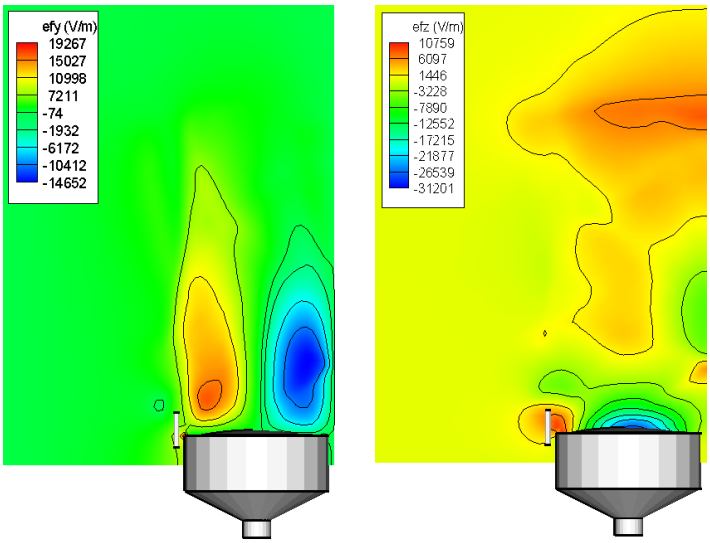
(a) Potential, -16:27 V (b) Electron Temp., 0:58 eV

Figure 5.20: Potential, and ion and electron temperature, cathode plane.



(a) Charge Density (b) Ion Density (c) Electron Density

Figure 5.21: Charge and number density, cathode plane.



(a) y-component (b) z-component

Figure 5.22: Electric field components, individual cathodes. Cut shows the cathode plane.

is smaller than the diagonal one. However, although a more pronounced high density region exists along this cut, the electron density at the mixing region is relatively small.

Field contours were also plotted for this case. The electric field along this plane is very similar to one shown with the diagonal cut.

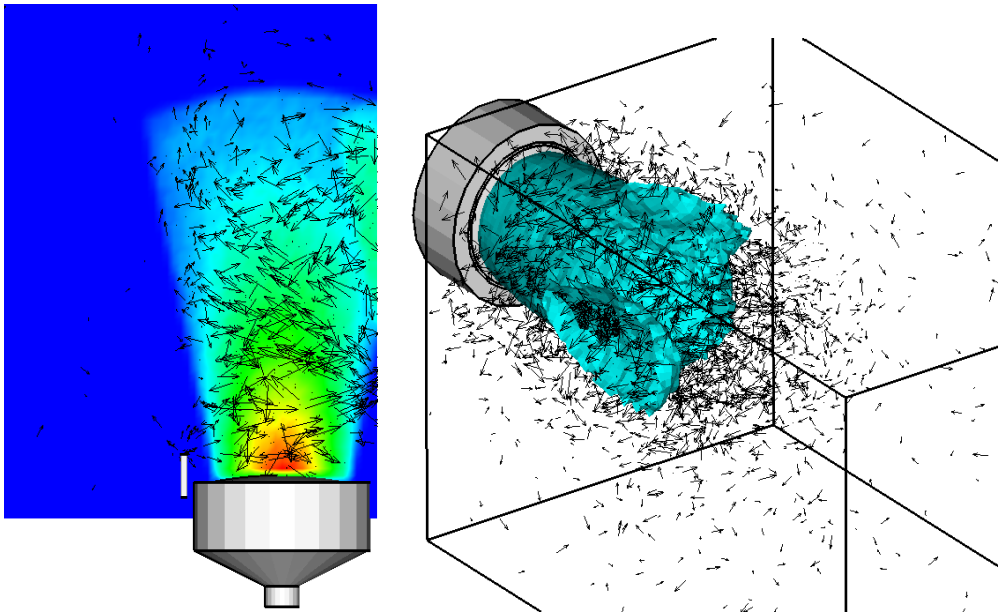
Velocity vectors for the electrons are shown in 5.23. The focused flow of the electrons from the cathode towards the centerline is clearly visible. The apparent reflection indicates the inflow of electrons from the neighbor thruster. Hence, these results indicate that the neutralization of an ion beam in a cluster configuration is due to the cathode at the *neighbor* thruster.

5.2.3 Thruster array with a single central cathode

Replacing the individual cathodes with a single central neutralizer would reduce the complexity of the assembly. Feasibility of such a configuration was modeled next, using the configuration shown in figure 5.24. Again, due to symmetry, only a quarter domain was simulated.

Since the cathode is responsible for neutralization of four thrusters, the cathode current was increased to 4.8A. The injection area was also increased by a factor of four by doubling the cathode diameter. The electron density, and subsequently λ_D, at the cathode tip thus retained their original values.

Potential and electron temperature plotted on the diagonal cut are shown in figure 5.25. Comparison to the diagonal cut for the cluster with individual cathodes, figure 5.17(a) shows a similar beam profile, however, the potential in the beam core is higher for this case. Yet, the total potential difference between the beam core and the background plasma has decreased



(a) cathode plane (b) simulation domain

Figure 5.23: Velocity vectors for the cluster with individual cathodes. Reflection at the symmetric face seems to indicate inflow of electrons from *neighbor* thruster.

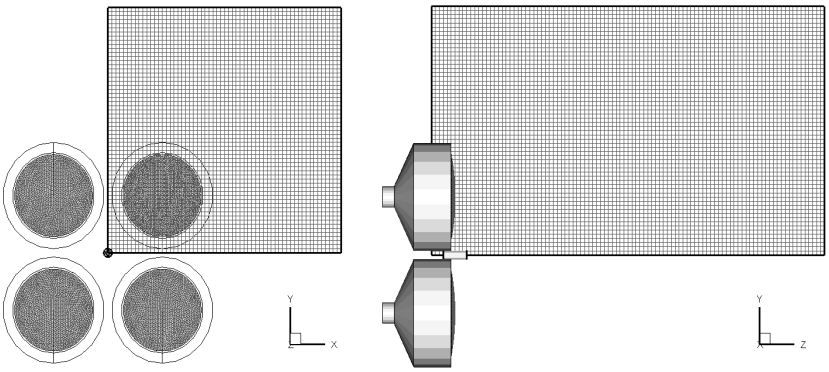
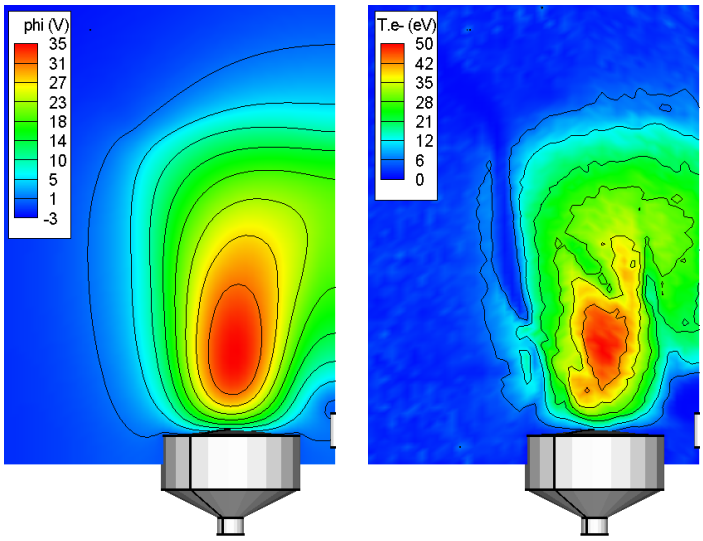


Figure 5.24: Simulation domain used to study the neutralization of a 2x2 cluster of ion thrusters with a single central neutralizer. Only a quarter of the domain was simulated due to symmetry, and the reflective particle boundary condition was applied along the symmetric faces.



(a) Potential, -3:35 V (b) Electron Temp., 0:50 eV

Figure 5.25: Potential and electron temperature for a cluster with a single neutralizer. Plots show the diagonal cut.

slightly to 38V. The electron temperature has also decreased to 50 eV.

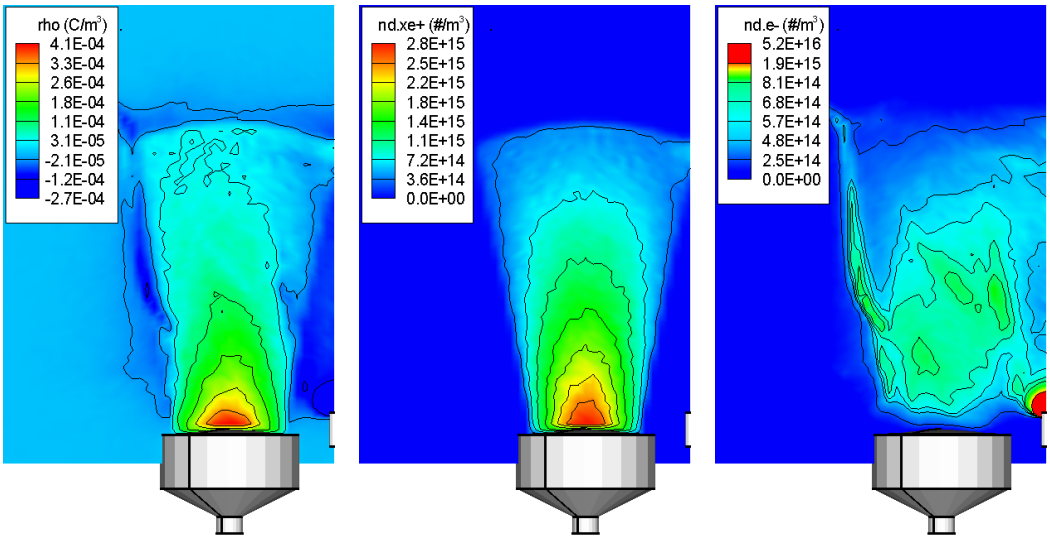
The decrease in temperature could be attributed to decreased mixing of the electron beams. All electrons originate from the same source, and simply expand towards the ion regions. This dynamics is very different from the configuration with individual cathodes, in which the electrons flow towards the cluster centerline, where they mix and then flow to any of the four thrusters.

Information about the electron dynamics can be extracted from the density plots in figure 5.26. The electron sheath is much more consistent, than the sheath seen in figure 5.18. More important is the lack of the dense strip of negative charge density. The electron density cloud is more uniform, and is more comparable to the cloud seen in the single thruster neutralization result.

The diagonal and axial electric field components are plotted in figure 5.27. The velocity vectors are shown in figure 5.28. The electrons are seen to float from the cathode to the beam, where their motion becomes random due to reflection at the beam boundary. This behavior is comparable to the single thruster neutralization.

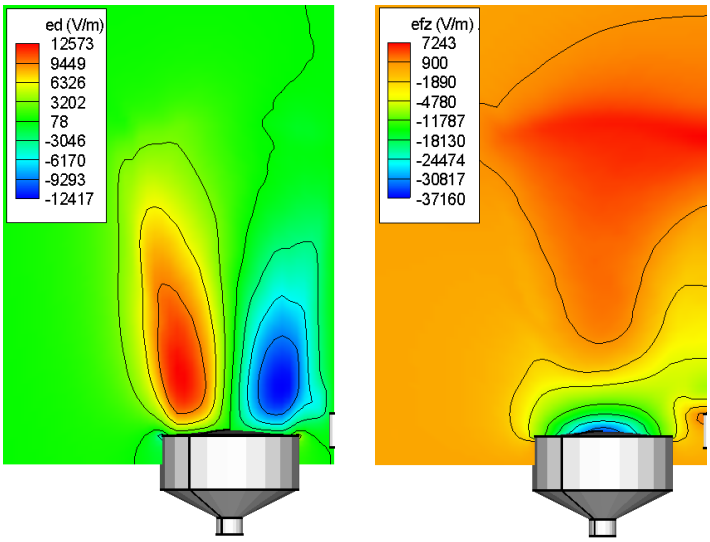
5.2.4 Increased Electron Current

The possibility that the non-physical values obtained in the cathode/beam simulations were due to an electron loss at the boundary was investigated by oversaturating the simulation with electrons. The electron current injected from the cathode was increased to 1.5 time the ion current. The resulting potential, temperature and charge density is plotted in figures 5.31 through 5.33.



(a) Charge Density (b) Ion Density (c) Electron Density

Figure 5.26: Charge and number density contour plots for the cluster with a single neutralizer.



(a) diagonal component (b) z component

Figure 5.27: Electric field components for the cluster with a central neutralizer

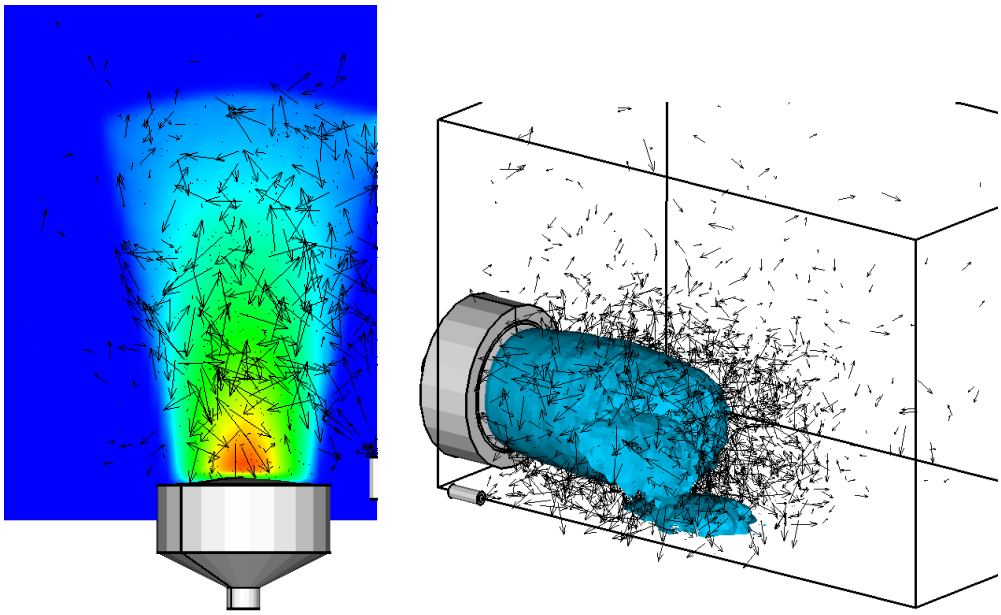


Figure 5.28: Velocity vectors, single central cathode

The potential in the core of the beam drops by about 10V. However, the $\Delta\phi$ between the core and the ambient plasma actually increased to about 50V from the previous 38V. The beam is also seen to turn towards the cluster centerline, due to the increased negative charge in that region.

Increase in electron current resulted in an increase in electron temperature. The temperature in the core observed for the $Ii = Ie$ case reached about 50eV. The electron oversaturation resulted in temperature increase to over 70eV.

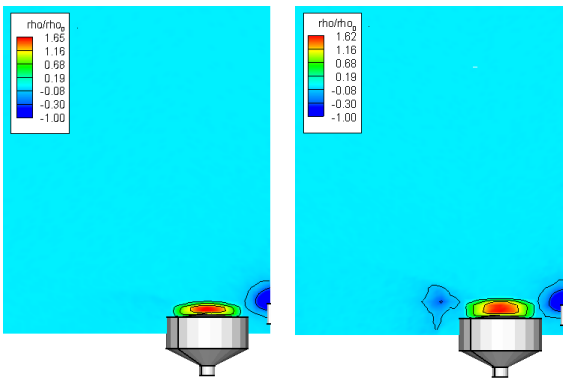
The charge density plot shows a more pronounced electron sheath. Although this may be a desired result, the increased $\Delta\phi$ and the electron temperature indicate that electron loss is not a source of the high potential.

5.2.5 Velocity Profiles

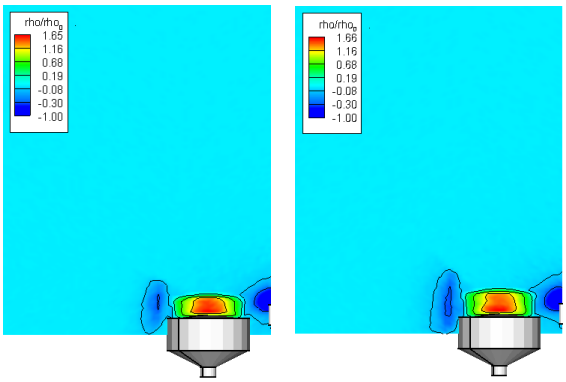
Velocity histograms can provide an additional insight into the behavior of the particle species. Figure 5.34 shows the velocity histograms for the ions and the electrons for all cases studied in this chapter. The cases are labeled as R2, NS, NI, NC1 and NC2, and correspond to the base-line initially neutralized R2 run, single thruster, cluster with individual cathodes and the two central cathode runs with $Ie = Ii$ and $Ie = 1.5Ii$, respectively.

Interesting is the departure of the ions from the Maxwellian distribution for all cathode cases. This departure is due to an increased number of high velocity ions. The mean drifting velocity is also seen to decrease, indicating a slow down of the beam due to the higher core potential.

An even greater discrepancy is seen in the electron distribution. The distribution for

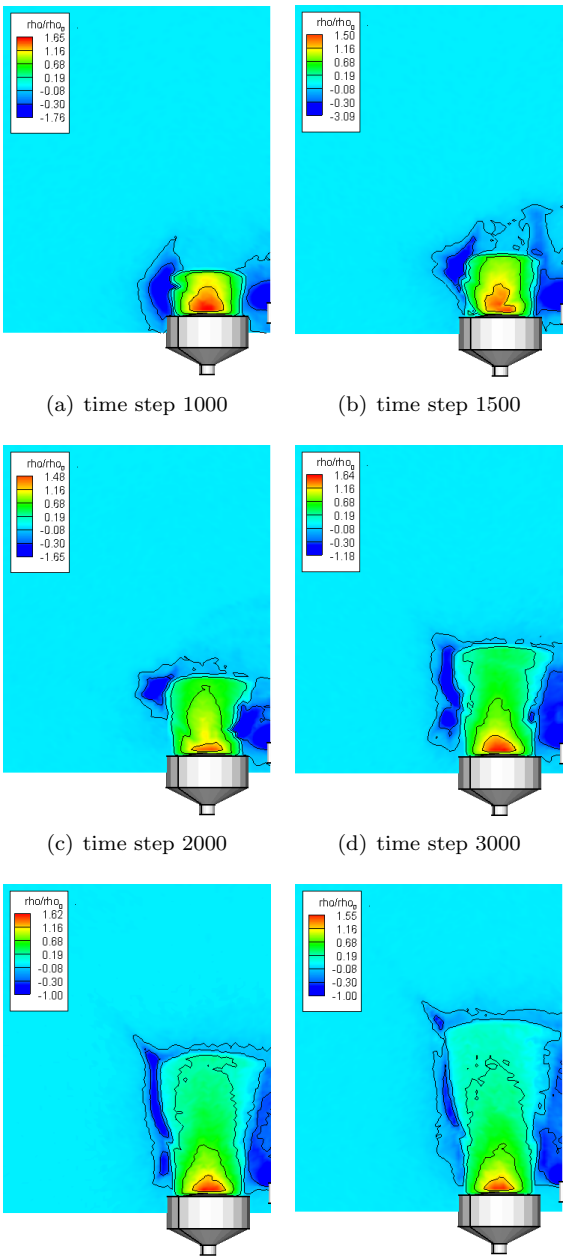


(a) time step 200 (b) time step 300



(c) time step 400 (d) time step 500

Figure 5.29: Charge density for time steps 200 through 500, single central cathode

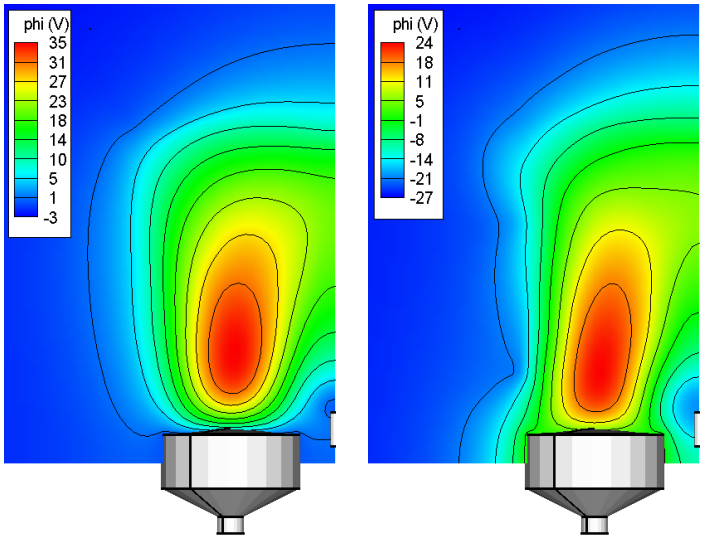


(a) time step 1000 (b) time step 1500

(c) time step 2000 (d) time step 3000

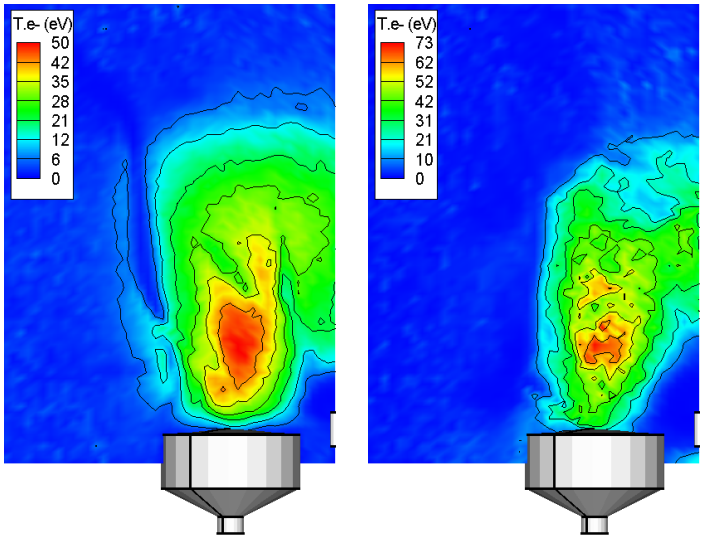
(e) time step 4000 (f) time step 5000

Figure 5.30: Charge density for time stpes 1000 through 5000, single central cathode



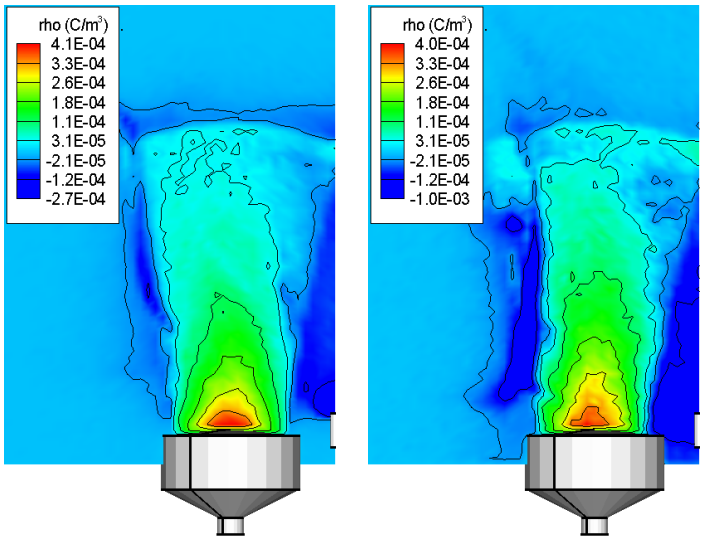
(a) $Ie = Ii$ (b) $Ie = 1.5Ii$

Figure 5.31: Effect of increased electron current on potential, single central cathode



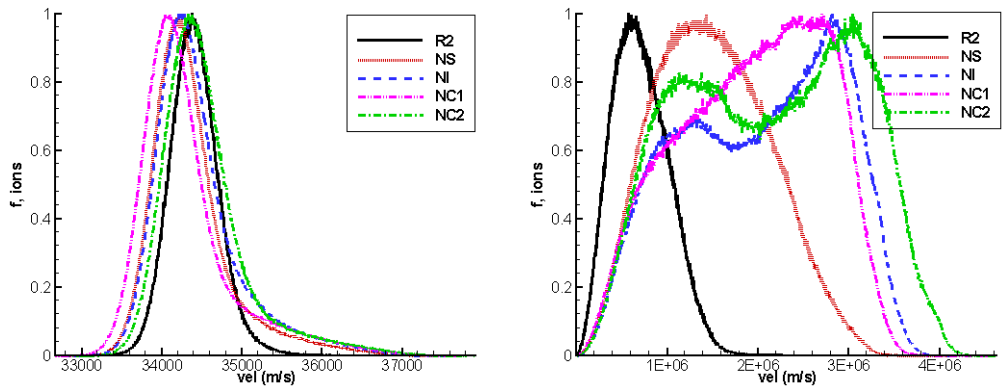
(a) $Ie = Ii$ (b) $Ie = 1.5Ii$

Figure 5.32: Effect of increased electron current on electron temperature, single central cathode



(a) $Ie = Ii$ (b) $Ie = 1.5Ii$

Figure 5.33: Effect of increased electron current on charge density, single central cathode



(a) ions (b) electrons

Figure 5.34: Velocity histogram for the ions and electrons

the single thruster shows the greatest agreement with the reference R2 case. However, the Maxwellian temperature is seen to increase, which is demonstrated by the widening of the distribution function. Further, the mean drift velocity has increased from about 1×10^6 to about 1.8×10^6 m/s. The next closest match is achieved by neutralizing the cluster with a single cathode, case NC1. This case shows a better agreement than the cluster configuration with individual cathodes. In that case, case NI, a double hump develops, indicating presence of two electron families. Very similar distribution is seen in case NC2, in which the increased electron current was emitted from the single central cathode.

Chapter 6

Conclusions

6.1 Development of the DRACO simulation module

This thesis described the development of a new ES-PIC simulation module called DRACO, based on an structured tetrahedral interface mesh. This mesh is capable of resolving surface curvature thanks to planar cuts of *interface* tetrahedrons. However, since the tetrahedrons are fitted into cells of a Cartesian mesh, particle tracking can be performed very rapidly.

DRACO was developed by integrating an existing plume simulation module into the Air Force COLISEUM framework. The simulation domain is specified using a surface triangular mesh, and the topology of the Cartesian domain. The interface mesh is created automatically by a helper meshing module, called VOLCAR.

DRACO was also coupled with several new potential solvers. A quick plume expansion modeling can be obtained by directly inverting the Boltzmann relationship. A non-neutral solution can be obtained by solving the Poisson's equation using on of three available Poisson solvers. Of these, only the IFE finite element solver is capable of resolving the curvature of the surface geometry. All Poisson solvers can operate in both the linear and non-linear modes. The linear mode uses the position of electron macro-particles to obtain the electron density. The non-linear mode assumes the Boltzmann distribution for the electron density.

Integration of DRACO into the COLISEUM framework greatly increased the flexibility of the simulation module. The simulation is completely controlled by commands in COLISEUM's input script file. Particle are introduced through surface sources. Since most source types inject particles relative to the normal vector of the injection surface triangle, an actual curvature of the surface mesh can be used to approximate the curvature of ion-thruster optics. Variable time-stepping scheme has also been developed.

6.2 Summary of Results

Development of a method for performing ion beam neutralization studies using a fully-kinetic PIC model is also described. Direct modeling of the electrons simplified the underlying physical models, however, it also introduced a number of implementation difficulties. Of these, of a great importance was the need to resolve the Debye length by using very small

cell sizes. Resolving this characteristic, while retaining the original simulation span for the full-size device was not numerically feasible. Instead, the surface dimensions were decreased by a factor of 100. Input parameters were adjusted such that this scaling did not affect the final solution.

Open particle boundaries introduced a numerical instability which sucked-out electrons from the beam. This instability was due to an initial removal of background electrons at the boundaries. This removal lead to the last cell containing less negative charge than the second to last cell. Response of the plasma is to direct more electrons to flow to the last cell, to equalize this difference. However, due to an overshoot, these electrons also fell of the simulation domain, and the problem cascaded toward the thruster.

This instability was mitigated by introducing a new particle boundary type, based on the conservation of energy. An electron moving up a potential hill exchanges its initial potential energy for an increase in the kinetic energy (velocity). As the particle passes the top of the hill and starts moving down the hill, the situation reverses. The potential energy will increase, which is demonstrated by reduction in the velocity. At some point, the kinetic energy becomes zero and the particle starts moving up the hill again. However, due to the finite domain, this inflection point may be located beyond the external walls. Hence, the open boundary removes particles which should be trapped in the beam. The energy condition instead reflects particles back to the domain that do not have sufficient energy to escape. Other particles are allow to escape the domain.

Combination of energy boundary with surface scaling lead to good results for the reference case in which the electrons were injected from the optics. The same cannot be said for the actual neutralization runs, in which the electrons were flowing from the cathode. Here, an artificially high potential and temperature developed. Best neutralization was achieved in the case of a single thruster, and in a 2x2 cluster neutralized by a single central cathode. Surprisingly, cluster neutralization with 4 individual cathodes produced very strange electron density profile. The profile, along with velocity vectors, seems to indicate that the electrons from the cathode actually flow through the first beam and cross into the neighbor beam, where they disperse. Possibility of bad neutralization due to lack of electrons was disproved by injection of extra electron current from the cathode. The resulting plasma parameters indicated *worse* neutralization than what was achieved in the initial central cluster case.

6.3 Future Work

Although DRACO is capable of performing a wide variety of simulations, a number of areas still require additional work. The list of possible improvements for DRACO includes:

- Optimization and full integration of the IFE field solver into DRACO. Current version of IFE is slow and requires a large amount of additional memory to run.
- Further debugging and testing of VOLCAR. Sharp corners and edges, and close proximity of objects generate bad intersections. VOLCAR should be able to automatically trace such problems and apply mesh smoothing.
- Investigation of a hybrid volumetric mesh. Even with mesh smoothing, the structured interface mesh will not be able to resolve the level of detail possible with body fitted

mesh. The interface tetrahedrons could be replaced with body fitted tets. The tets would fill the region between the surface boundary and the Cartesian "vacuum" cells.

- Mesh refinement. The support for mesh refinement needs to be finalized. Currently, DRACO field solvers are not capable of correctly solving the potential along the boundaries of a refined mesh.

- Additional Physics. The current amount of physics that can be simulated with DRACO is fairly limited. Additional development is required in collision modeling, sputtering and spacecraft charging. The Boltzmann electron model should also be replaced with full solution of the electron fluid equations. An electro-magnetic formulation would extend the range of problems which can be simulated with DRACO.

Further, the analysis of the ion beam neutralization results indicates that additional modeling is requires. Some items that require future investigation include:

- Cathode model. Current model used a floating potential on the cathode, and the charge density near the tip was fixed. These adjustments were needed since the mesh could not resolve the local Debye length. The cathode was also assumed to inject only electrons, while experimental measurements indicate that neutrals and ions are present in the cathode plume.

- Test of scaling. The geometry scaling approach was developed solely by analyzing the complete set of plasma fluid equations. The current formulation should be verified by repeating same test case for a range of scaling ratios.

- Effect of Energy Boundary conditions. The newly-developed energy boundary condition simply reflects all particles with insufficient energy to escape. It thus ignores the time delay associated with the particle traveling to the inflection point. Estimating this delay is however not straightforward. Further, the potential energy is computed using the drop between the maximum potential in the simulation, and a user prescribed infinity value. There is no guarantee that the particle actually passed through the region of the maximum potential. The potential energy should be computed individually for each particle, according to the particle's path.

- Study of collision effects. Electrons and ions in the current simulation interact only through the macro-scopic effects prescribed by the electric field. Influence of microscopic Coulomb collisions on the solution should be investigated. Ion-neutral collisions can also be expected to increase the divergence of the beam, possibly lowering the core potential. Presence of charge-exchange ions will modify the plasma environment around the thruster.

- Large scale modeling needs to be performed. Results shown here indicate a highly unsteady electron dynamics in the beam. Effect of simulation domain on the results should be studied. Further, the simulation should be allowed to run for a longer time period to determine the location at which a good beam neutrality can be assumed.

- Comparison to experimental results. No comparison to experimental data was made, since such data was not available to the author. Availability of the experimental data

would allow to "tweak" the input parameters as well as the particle boundaries. Good agreement with ion densities, potential and electron temperature could then be used to properly describe the electron motion.

Appendix A

Simulation Input Files

This appendix lists the simulation input files used to run the 2x2 cluster case with a single neutralizer (NC1). Injection ion current was 1.2×10^{-4}A, and the electron current was 4.8×10^{-4}A.

A.1 Coliseum Input File

This file is automatically processed by COLISEUM upon execution. The script file contains a list of commands which are executed during the simulation. Lines beginning with # are ignored (comments). The charge density fix at the cathode and the reinjection of electrons were hard-coded into the code. Future versions will allow the user to specify such functionality through the input files.

```
# load material and component files
material_load material.txt mat_mat.txt
component_load component.txt

# load surface mesh for the thruster and the cathode
# scale surface by a factor of 100, and save in Tecplot format
surface_load ANSYS next.ans
surface_load ANSYS cathode.ans
surface_scale 100
surface_save TECPLOT surface.dat

# generate interface simulation domain
# topology is specified in domain.txt
volcar domain.txt

# specify particle sources
source_specify optics ION_THRUSTER Xe+ 1.634e-10 1160.4 34400 20 ...
 ... 4 4.3837 0 -17.434 0 17.043
source_specify e_source MAXSTREAM E- 2.729e-15 11604 1000 20

# simulation parameters: initial time step and particle travel per time step
```

```
SET dt 4e-11
SET dhp_per_step .75

# run for prescribed plasma time
SET nt -2
SET run_time 3e-7

# specify frequency of restart saves, field update, PIC diagnostics
SET nt_restart_save 100
SET nt_field_update 1
set nt_pic_diag 1

# reference parameters for the Boltzmann electron distribution
# not used in this case, since linear solver is used
SET Te_Ref 2
SET Phi_Ref 0
SET Ne_Ref 1.7e15

# set reference specie, for normalization, dt update
SET specie_ref E-

# use linear DADI to solve the potential
draco_init_solver dadi_linear const 300 1e-3

# output mesh solution each 100 time steps to file f_xxx.dat
draco_set_diagnostics 100 TECPLOT f phi rho MaxTe.e-

# start the simulation
draco

# average over additional 100 steps
draco_start_averaging 0 1 phi rho nd.xe+ nd.e- MaxTe.xe+ MaxTe.e- ...
 ... jx.xe+ jy.xe+ jz.xe+

# cancel restart saves
SET nt_restart_save -1
draco restart 100

# save results
volcar_mesh_save TECPLOT field.dat phi-ave rho-ave nd-ave.xe+ nd-ave.e- ...
 ... MaxTe-ave.xe+ MaxTe-ave.e- efx efy efz jx-ave.xe+ jy-ave.xe+ jz-ave.xe+

# sample particles
draco_particle_save particles.dat ALL Xe+ 20000
draco_particle_save append ALL E- 20000
```

A.2 Material File

The material file specifies material properties. These properties are used primarily to define the particle species. Material properties also define the behavior of particles interacting with surfaces, but this feature is currently not implemented in DRACO. The interaction is specified in a similar file, `mat_mat.txt`.

```
name        molwt       spwt   charge     density
vacuum      0           0      0          1e4
Al          26.9        0      0          2700
Xe+         131.4       5e1    1          1.7e15
E-          5.49e-4     5e1    -1         1.7e15
```

A.3 Component File

The component file defines properties for surface triangles. DRACO uses this file to define potential on the objects, as well as to specify the object type (solid, source, sink, etc...)

```
Name        Num   mat_name    phi   eps         type
vacuum      0     vacuum      0     1           source
shell       1     Al          0     100000      solid
optics      2     Al          0     100000      source
cathode     3     Al          0     100000      source
e_source    4     Al          0     100000      source
```

A.4 Mesh Topology

Mesh topology is specified in a domain file. VOLCAR uses this information to create the Cartesian mesh and the overlaying tetrahedral mesh.

```
add_grid
 x0: -0.0028
 y0: -0.0028
 z0: -0.001
 dx: 2e-4
 dy: 2e-4
 dz: 2e-4
 nx: 60
 ny: 60
 nz: 100
 set_boundary X_MIN NEUMANN 0 REFLECTIVE
 set_boundary X_MAX NEUMANN 0 ENERGY 0
 set_boundary Y_MIN NEUMANN 0 REFLECTIVE
 set_boundary Y_MAX NEUMANN 0 ENERGY 0
 set_boundary Z_MIN NEUMANN 0 ENERGY 0
 set_boundary Z_MAX NEUMANN 0 ENERGY 0
end_grid
```

A.5 Surface Mesh

The surface is specified using a triangular surface mesh, saved in the ANSYS format. Due to the size of this file, only an illustratory portion is included here. All nodes are listed first. Following is the connectivity, separated by the component.

```
MP,EX,1,1
MP,EX,2,2
N, 1, 0, 0.21, 0
N, 2, 0, 0.26, 0
N, 3, -0.0591668, 0.201493, 0
N, 4, -0.0732565, 0.249466, 0
N, 5, -0.113538, 0.176661, 0
N, 6, -0.140575, 0.21872, 0
...
MAT,1
EN, 1, 4, 3, 1
EN, 2, 1, 2, 4
EN, 3, 3, 4, 5
EN, 4, 6, 5, 4
...
MAT,2
EN, 661, 484, 483, 464
EN, 662, 485, 628, 484
EN, 663, 486, 629, 485
EN, 664, 487, 630, 486
...
```

Bibliography

[1] NASA Glenn Website, *DS1: How the Ion Engine Works*, http://www.nasa.gov/centers/glenn/about/history/ipsworks.html

[2] NASA Glenn Website, *NASA's Evolutionary Xenon Thruster (NEXT)*, http://space-power.grc.nasa.gov/ppo/projects/next/accomp.html

[3] Polk, J., et al., "Validation of the NSTAR Ion Propulsion System on the Deep Space One Mission: Overview and Initial Results," *AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, 35th, Los Angeles, CA, June 20-24, 1999.

[4] Patterson, M., et. al, "NEXT: NASA's Evolutionary Xenon Thruster," *AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, 38th, Indianapolis, IN, July 7-10, 2002.

[5] Soulas, G., Domonkos, M., and Patterson, M., "Peformance evaluation of the NEXT ion engine," *AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, 39th, Huntsville, AL, July 20-23, 2003

[6] Birdsall, C., and Langdon, A., *Plasma Physics Via Computer Simulations*, Institute of Physics Publishing, Philadelphia, 2000

[7] Wang, J., Brinza, D., and Young, M., "Three-Dimensional Particle Simulations of Ion Propulsion Plasma Environment for Deep Space 1," *Journal of Spacecraft and Rockets*, Vol. 38, No.3, 2001, pp. 433-440

[8] Roy, S.R., "Numerical Simulation of Ion Thruster Plume Backflow for Spacecraft Contamination Assesment," Ph.D. Dissertation, Aeronautics and Astronautics Dept., Mass. Institute of Technology, Cambridge, MA, 1985

[9] Kafafy, R., Lin, T., Lin, Y., and Wang, J., "3-D Immersed Finite Element Method for Electric Field Simulation in Composite Materials", submitted to *Int. Journal for Numerical Methods in Engineering*, 2004.

[10] Kafafy, R., "Immersed-Finite-Element Particle-In-Cell Simulation of Ion Thrusters," Ph.D. Dissertation, Aerospace and Ocean Eng. Dept., Virginia Tech, Blacksburg, VA, 2005

[11] Gibbons, M., Kirtley, D., VanGilder, D., and Fife, J., "Flexible Three-Dimensional Modeling of Electric Thrusters in Vaccum Chambers," AIAA-2003-4872, 2003

[12] Santi, M., Cheng, S., Celik, M., Martinez-Sanchez, M., and Peraire, J., "Further Development and Preliminary Results of the Aquila Hall Thruster Plume Model," AIAA-2003-4873, 2003

[13] Patterson, M., et. al., "NEXT: NASA's Evolutionary Xenon Thruster Development Status," *AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, 39th, Huntsville, AL, July 20-23, 2003.

[14] Soulas, G., "Design and Peformance of 40 cm Ion Optics," *International Electric Propulsion Conference*, 27th, Pasadena, CA, October 15-19, 2001

[15] Wang, J., Brieda, L., Kafafy, R., Pierru, J., "A Virtual Testing Environment for Electric Propulsion-Spacecraft Interactions," AIAA-2004-0652, 2004

[16] Tanehill, J. C., Anderson, D. A., and Pletcher, R. H., *Computational Fluid Mechanics and Heat Transfer*, Taylor and Francis, Philadelphi, 1997

[17] Chen, F. F., *Introduction to Plasma Physics and Controlled Fusion, Volume 1*, Plenum Press, New York, 1984

[18] Hewett, D., Larson, W., and Doss, S., "Solution of Simultaneous Partial Differential Equations Using Dynamic ADI: Solution of the Streamline Darwin Field Equations," *Journal of Computational Physics*, Vol. 101, 1992, pp. 11-24

[19] Tran, B., "Performance Analysis of Ion Thruster Discharge Chamber Using 3D Particle-In-Cell Monte-Carlo-Collision Method", Master's Thesis, Aerospace and Ocean Eng. Dept., Virginia Tech, Blacksburg, VA, 2005

[20] Pierru, J., "Development of a parallel electrostatic PIC code and its use to model electric propulsion", Master's Thesis, Aerospace and Ocean Eng. Dept., Virginia Tech, Blacksburg, VA, 2005

[21] Bird, G., *Molecular Gas Dynamics and the Direct Simulation of Gas Flows*, Institute of Physics Publishing, Philadelphia, 2000

[22] Tajmar, M., and Wang, J., "Three-Dimensional Numerical Simulation of Field-Emission-Electric-Propulsion Neutralization," *Journal of Propulsion and Power*, Vol. 16, No. 3, 2000.

[23] Kikolski, R., and Spicer, R., "Study of macro particle weight on simulation noise," Undergraduate research report, April 2005.

[24] Soulas, G., Haag, T., and Patterson, M., "Performance evaluation of 40 cm ion optics for the NEXT ion engine," AIAA Paper 2002-3834, July 2002.

[25] Wang, J., and Lai, S., "Virtual Anode in Ion Beam Emission in Space: Numerical Simulations," *Journal of Spacecraft and Rockets*, Vol. 34, No. 6, 1997

[26] Hastings, D., and Garrett, H., *Spacecraft-Environment Interactions*, Cambridge University Press, Cambridge, England, 1996

[27] Capacci, M., Minucci, M., and Severi, A., "Simple numerical model describing discharge parameters in orificed hollow cathode devices," *AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, 33rd, Seattle, WA, July 6-9, 1997.

[28] Salhi, A., and Turchi, P., "A First-Principle Model For Orificed Hollow Cathode Operation," *AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, 28th, Nashville, TN, July 6-8, 1992

[29] Mandell, M., and Katz, I., "Theory of Hollow Cathode Operation in Spot and Plume Modes," *AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, 30th, Indianapolis, IN, June 27-29, 1994

[30] Boyd, I., and Crofton, M., "Modeling the plasma plume of a hollow cathode," *Journal of Applied Physics*, Vol. 95, No. 7, 2004.