Simulation of Ion Optics Using Particle-In-Cell and Treecode Methods

by

Jerold William Emhoff

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Aerospace Engineering)
in The University of Michigan
2005

Dedicated to my parents,
for their care, support,
guidance, and love.

ACKNOWLEDGEMENTS

The person who deserves the most gratitude for this thesis is undoubtedly my advisor, Iain Boyd. He has supported me in one way or another for five years, and has been the best advisor I can imagine having. I'd also like to thank Andrew Christlieb for being my "second advisor." Andrew has provided much guidance as well, and given me new and interesting problems to work on.

I am grateful to the Aerospace Engineering department for providing me with my first year of funding. For most of my graduate student career, I have been supported by grant NAG3-2497 from the NASA Glenn Research Center. I am very grateful for this support, and my technical monitor at GRC, George Soulas, has been invaluable in his willingness to work with me and help my research. The Michigan Space Grant Consortium Graduate Student fellowship has also provided me with support for the last two years, for which I am most grateful. I also want to thank the Rackham Graduate School for providing several travel grants, which I have used to attend conferences all over the country.

On a more personal note, I would like to first and foremost thank my parents, whom I love very much. I truly wouldn't be anything, anywhere, if not for them, and I mean that in a multitude of ways. They have supported me my whole life emotionally, intellectually, and financially. The last not necessarily being the least.

Thank you, Mom and Dad, for everything.

My sister Holly has been a wonderful sibling for as long as I can remember (maybe I can't remember the younger years so well). I have to thank her for being there for me whenever I needed it. I am also grateful to my brother-in-law Brian for being the family comedian and for getting me into the odd sporting event. I also have to thank the rest of my family and relatives for their love and support through the years. You all have played a part in raising me, whether you know it or not, and I appreciate it.

Next, I'd like to thank my best friends, Matt, Leigh, Ellen, and Kooj. Each of you has been there for me, in all sorts of different ways that I can't begin to count. Just know that I am grateful for each and every instance (and you know my Dutch brain keeps track of that sort of thing!). I also thank the rest of my friends, mostly from the FXB second floor. You guys have been great fun to be around for the last five years.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF APPENDICES

CHAPTER I

Introduction

An ion thruster is a form of electric propulsion that has become increasingly popular in recent years. This is due primarily to the extremely successful Deep Space 1 mission, where an ion thruster provided the primary spacecraft propulsion. These engines produce a very low amount of thrust ($\sim$ 200 mN) at a very high efficiency. Thus, they are required to operate a very long time ($\sim 20,000$ hours) in order to accomplish the desired goals of a mission. This long operation time means that the thruster must be very robust and its failure modes must be well-defined. It is extremely expensive and time-consuming to operate a thruster in an experimental vacuum facility for its entire life before using the thruster for a mission. An alternative is to use computer simulations to model the failure modes of the thruster and predict when it will fail.

One of the primary failure modes is the erosion of the ion optics. The optics are a set of two or three metal grids with thousands of apertures. An electric field is applied to these grids, accelerating ions through the apertures and producing thrust. However, ions may collide with neutral particles in the domain, creating a slow-moving ion that can be drawn into the grids. When this occurs, the optics are eroded, and over time this erosion becomes severe enough to cause the thruster to fail. The goals of an ion optics simulation are first to accurately simulate performance of the optics and second to accurately simulate erosion. The purpose of this work is to develop methods to accomplish both of these goals.

1.1 Electric Propulsion

Electric propulsion is a form of propulsion for spacecraft that relies on the use of electricity to add energy to a flow, rather than the conventional use of chemical energy from a combustion reaction. The advantage is that the limit on the electric energy that can be added to the flow is given by the amount of available electricity rather than the chemical make-up of the propellant. Typically this means that thrust from electric propulsion is very efficient and very low compared to chemical propulsion.

There are several ways to use electrical energy to create propulsion. Arcjets and resistojets heat the propellant flow using either an electric arc or by the powering of a heating element. Electrostatic thrusters, including both ion thrusters and Hall-effect thrusters, create ions using bombardment by magnetically confined electrons, and then extract the ions using an imposed electric field. Electromagnetic thrusters use both electric and magnetic fields to accelerate ions. This type of thruster includes pulsed-plasma thrusters and magnetoplasmadynamic (MPD) thrusters. High-energy electric propulsion is also under development. The Variable Specific Impulse Magnetoplasma Rocket (VASIMR) uses a magnetic field to contain a plasma heated by radio waves. This plasma is then exhausted, and depending on whether more energy is put into heating the ions or into creating the ions, a higher efficiency or higher thrust may be obtained.

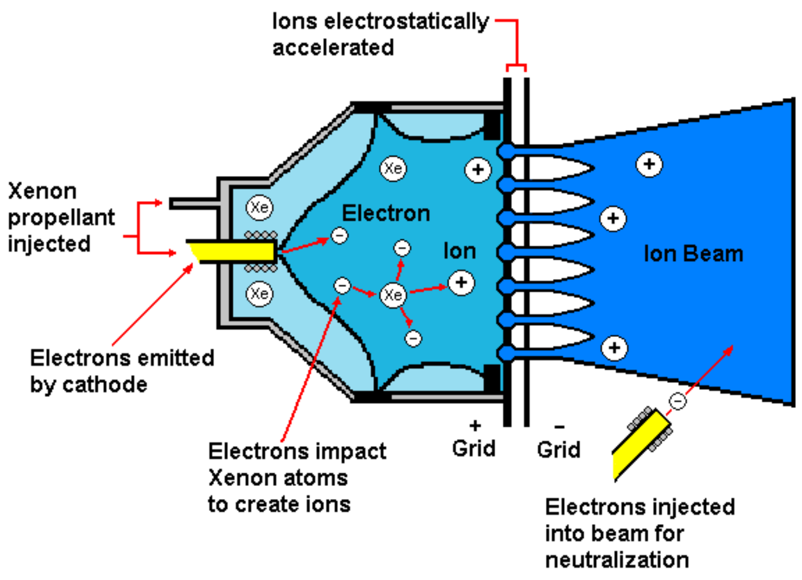

Ions electrostatically accelerated
Xenon propellant injected
Xe
Electron
Ion
Ion Beam
Electrons emitted by cathode
Electrons impact Xenon atoms to create ions
+ Grid
- Grid
Electrons injected into beam for neutralization

Figure 1.1: Diagram of an ion thruster. In the discharge chamber, neutral atoms collide with magnetically confined electrons to create ions. These ions are then accelerated by the ion optics to create thrust. Image courtesy of NASA.

1.2 Ion Thruster Operation

An ion thruster consists of a discharge chamber, a set of ion optics, and a neutralizer cathode. A basic schematic of an ion thruster is shown in Figure 1.1. The discharge chamber contains a neutral plasma consisting of neutral particles, ions, and electrons. A cathode in the discharge chamber emits electrons into a neutral gas. Inside the chamber, a magnetic field contains the electrons, which collide with the neutral gas to create ions. The magnetic field strongly affects the electron density distribution, which in turn affects the radial ion density distribution. For example, Figure 1.2 shows the radial current density distributions of the NSTAR and NEXT ion engines (*Soulas et al.* (2002)). The NEXT thruster is basically a scaled up version of the NSTAR thruster. However, the discharge chamber magnetic field for the NEXT thruster was designed to give a much flatter profile. This is advantageous from

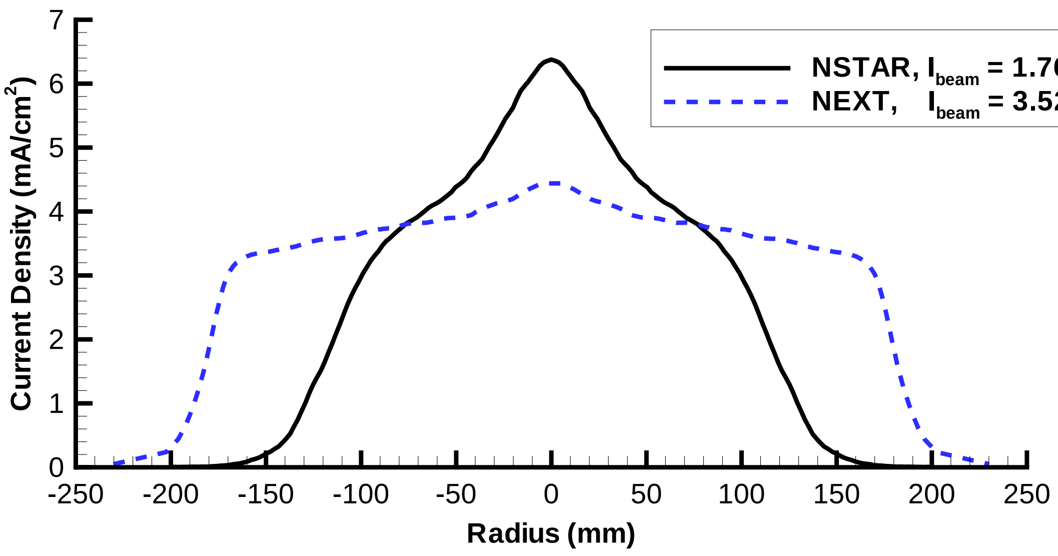


Figure 1.2: Current density profiles for the maximum operating points of the NSTAR and NEXT ion engines, from *Soulas et al.* (2002).

a performance standpoint, as the thrust and specific impulse will be more uniform. It also leads to a lower total erosion rate due to a lower peak current density.

The ion optics are a set of two or three grids that typically contain many thousands of apertures arranged in a hexagonal pattern. The first grid, the screen grid, floats at a potential about 25 V below the discharge potential, which is on the order of a thousand volts. The second grid is the accelerator grid, and it typically has a negative potential of about -200 V. A third grid, called a decelerator grid, may also be present. This grid is held at about zero volts and serves to mitigate erosion on the downstream side of the accelerator grid. However, the focus of this thesis is on the simulation of 2-grid ion optics. For larger sets of ion optics, such as in the NSTAR and NEXT thrusters, the grids are dished slightly outwards in order to provide increased structural and vibrational stability.

The neutralizer cathode produces a current of electrons equal to the ion beam current of the thruster in order to neutralize the beam and prevent a charge from building on the spacecraft. The purpose of the accelerator grid is to prevent these electrons from flowing into the discharge chamber of the thruster, thus this grid has a negative potential that repels the plume electrons.

The maximum amount of thrust an ion engine is capable of is limited by Child's Law:

$$j = \frac{4\varepsilon_0}{9}\left(\frac{2e}{m}\right)^{1/2} \frac{\Delta V^{3/2}}{d^2} \ , \tag{1.1}$$

where j is current density through a finite area, ΔV is the voltage drop, d is the distance between the screen grid and the accelerator grid, and m is the ion mass. This law states that j is the maximum current density that can be extracted through an aperture, given the potential drop ΔV and the grid gap d. As such, the performance of any given ion optics geometry and power level is limited. This is a disadvantage compared to other types of electric propulsion such as Hall thrusters, which can produce a relatively higher amount of thrust for a given thruster area, although at a lower efficiency.

Electrons from the discharge chamber are prevented from flowing through the optics by the imposed electric field, forming a sheath in the screen grid aperture. Ions are focused by this sheath and are accelerated by the electric field to produce beam current and thrust. In the downstream region, neutralizing electrons in the plasma form another sheath downstream of the accelerator grid. The negative potential of the accelerator grid prevents electrons from flowing into the discharge chamber from the plume.

There are two primary failure modes of an ion thruster. First, the discharge cathode is constantly bombarded with ions from the discharge chamber and may become so eroded that it can no longer operate. Second, erosion of the accelerator grid can progress until it either fails structurally or becomes ineffective. As ions flow through the optics, they may collide with neutral particles. Some of these collisions result in charge-exchange (CEX), where an electron from the neutral particle moves to the ion. This creates a fast-moving neutral particle, and a slow moving CEX ion. Due to its low energy, the CEX ion may then be drawn to the accelerator grid, impacting with an energy approximately equal to the potential on the grid. Typically, this energy is about 200 eV.

CEX erosion on the surface inside of the aperture, called the barrel, increases the diameter of the aperture. If the aperture becomes too large, then electrons from the plume are accelerated into the discharge chamber. This causes large amounts of erosion, resulting in thruster failure. However, the thruster may still be able to operate at a lower power if electron backstreaming occurs at a high-power operating point. A low-power operating point will have a lower total beam current, as well as a lower accelerating voltage, giving more margin against electron backstreaming.

CEX erosion on the downstream surface of the ion optics causes the formation of pits and grooves on the surface. Pits form at the center of three apertures, while grooves form along the midpoint between two apertures. The end effect is that hexagons are etched into the grid, with the grooves forming the edges and the pits forming the vertices. If these pits and grooves become deep enough, they may compromise the structural integrity of the grid and cause it to collapse.

Another failure mode is also caused indirectly by CEX erosion. Eroded grid material is ejected into the domain around the ion optics. If it impinges on the optics, it generally will recombine with the grid material. However, this leads to the formation of flakes of grid material that can spall off the surface. If a flake is large enough, it may span the distance between the screen and accelerator grids, causing a short. A grid short will generally cause the thruster to fail. However, this failure mode has largely been mitigated by the use of a grid-clearing circuit in ion thrusters (*Goodfellow et al.* (1999)). This circuit applies a large amount of current to the two grids, vaporizing the flake causing the short. Even so, it is possible that a flake may be large enough that it does not vaporize, but instead fuses to the ion optics. In this case, the thruster cannot be restarted.

1.3 History of Ion Thrusters

The electrostatic ion thruster is the original idea for electric propulsion. Nearly a century ago, both Robert Goddard and Konstantin Tsiolkovsky postulated the possibility of using electrostatically accelerated electrons to achieve thrust, as detailed in *Choueiri* (2004). Once ions were discovered to exist, Goddard realized that they would provide better thrust than the low-mass electrons. He was even granted a patent in 1920 for an engine using electrostatically accelerated ions to produce thrust. However, it was soon realized that enormous accelerating potentials are required to obtain any sizeable thrust, so ion thrusters and electric propulsion in general were ignored for many years.

In the 1960's, spacecraft were being developed, and the usefulness of a low-thrust, very high efficiency, form of propulsion became clear. Harold Kaufman popularized the use of the ion thruster at this time (*Kaufman* (1960)), while concurrently Hall-effect thrusters were being developed in the Soviet Union (*Zhurin et al.* (1999)) as well as in the United States (*Choueiri* (2001)). NASA's Space Electric Propulsion Test (SERT) program was started in the 1960's with the goal of developing and verifying the operation of ion thrusters in space. The SERT II ion engine demonstrated operation in space for several thousands of hours, providing valuable performance and spacecraft interaction data (*Sovey et al.* (2001)).

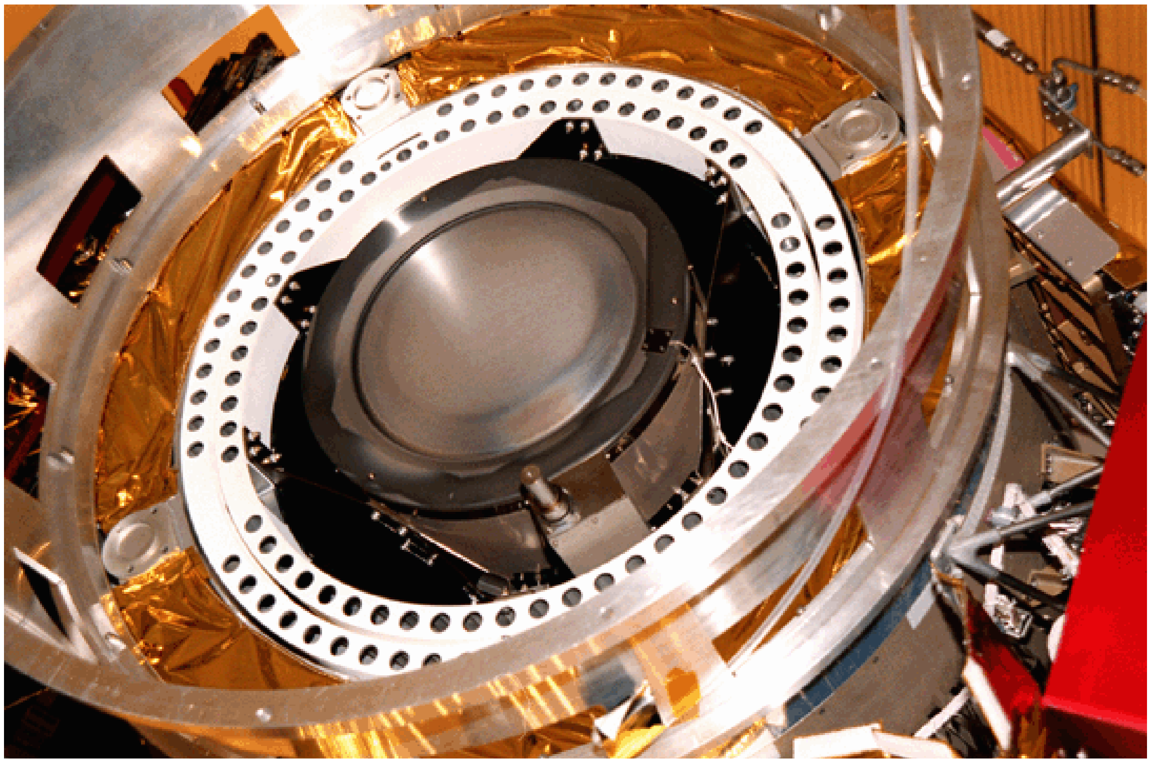


Figure 1.3: The NSTAR engine integrated into the Deep Space 1 spacecraft.

The first use of an ion engine as primary propulsion for a spacecraft was in the NASA Solar Technology Application Readiness (NSTAR) program. The 30 cm diameter NSTAR thruster used xenon gas as a propellant, rather than the mercury used in previous ion thrusters. The engine was integrated on the Deep Space 1 (DS1) spacecraft as the main source of propulsion, as Figure 1.3 shows. The primary mission was for DS1 to encounter the asteroid Braille, and the NSTAR thruster worked flawlessly to accomplish this. The DS1 mission was then extended to include an encounter with the comet Borrelly, after which the mission was extended further in order to test the thruster operation. At the end of DS1's mission, the NSTAR thruster had successfully operated for over 16,000 hours (*Brophy et al.* (2002)). On the ground, the flight spare NSTAR thruster was operated in a vacuum tank for more than 30,000 hours before the test was ended voluntarily (*Sengupta et al.* (2003)).

This major success produced a great deal of interest in ion thrusters, and in their failure modes. NASA's Evolutionary Xenon Thruster (NEXT) is an ion engine based on NSTAR, with a larger 40 cm beam extraction diameter. This thruster has completed a 2000 hour wear test (*Soulas et al.* (2004)) and is currently being prepared for a long-duration wear test. A picture of the NEXT thruster is shown in Figure 1.4.

The European Space Agency's ARTEMIS telecommunications satellite is equipped with four ion engines for station-keeping purposes. Problems with launch of the satellite required the use of the ion engines to bring the satellite into its intended geostationary orbit, even though the thrusters were not intended for use as main propulsion (*Killinger et al.* (2003)). The HAYABUSA spacecraft of the Japan Aerospace Exploration Agency also uses four ion engines for propulsion (*Kuninaka et al.* (2004)). This spacecraft has a goal of collecting and returning a sample from an asteroid. Also, the Boeing Corporation Xenon Ion Propulsion System (XIPS) has been used for station-keeping purposes on commercial satellites since 1997.

In the future, nuclear power may once again become an option for spacecraft, so higher power ion thrusters have recently been developed to take advantage of this. The High Power Electric Propulsion (HiPEP) ion engine is in development at NASA Glenn Research Center as one option (*Foster et al.* (2004)). This thruster is unique for its rectangular shape, enabling the thruster to be clustered easily. Another high-power thruster is the Nuclear Electric Xenon Ion System (NEXIS) in development at Jet Propulsion Laboratory (*Randolph and Polk* (2004)). Both thrusters are designed for specific impulses exceeding 6,000 seconds.

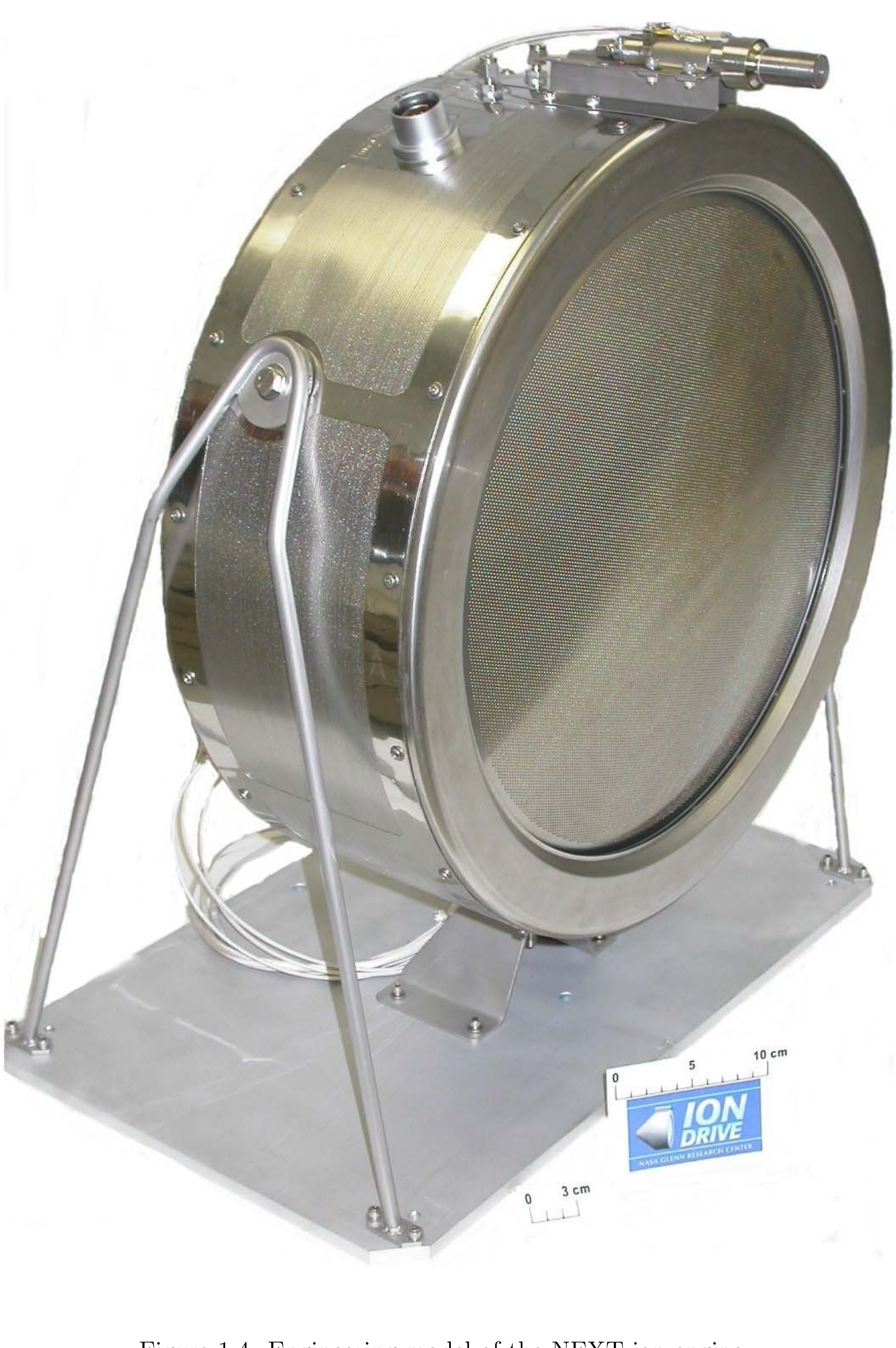


Figure 1.4: Engineering model of the NEXT ion engine.

1.4 Ion Thruster Modeling

Computer simulation of ion thruster operation is an important part of any current development program. Fabrication and testing of ion thrusters can be extremely costly and time-consuming, so the use of low-cost and relatively fast computer models is essential. Almost any part of the ion thruster may be simulated— the discharge chamber and cathode, the ion optics, and the plume and its interaction with a spacecraft are all areas with active development of computer models.

The focus of this work is on the modeling of the ion optics. The optics are responsible for creating the thrust and performance of the thruster, and their erosion is one of the primary life-limiting factors of the ion engine. Erosion is particularly difficult to measure experimentally, as this occurs very slowly, and the thruster may have a life in the tens of thousands of hours. A simulation has the ability to predict erosion and thruster end-of-life in a matter of minutes or hours.

Any ion optics simulation must perform certain tasks accurately in order to give reliable results. The potential field of the domain must be solved, including both the imposed potential of the optics and the space-charge potential of the ions and electrons. The ion, neutral, and electron species must all be simulated. For ions and neutrals, both densities and velocities are needed, but density is sufficient for electrons. This is because the ions and neutrals both contribute to thrust and mass flow, while electrons do not. Charge-exchange collisions between ions and neutrals must be simulated accurately, and this also requires both densities and velocities. The slow-moving ions that result from charge-exchange collisions must be tracked accurately in order to determine the erosion and current collected on the accelerator grid of the optics.

Simulations have been developed that are either axisymmetric single-aperture simulations or three-dimensional, multiple-aperture models. In each case, there are two main types of ion optics models. These are discussed briefly below.

1.4.1 Gun Models

The simplest and fastest type of simulation is called a "gun" code. In this type of simulation, the potential field is frozen while charged particles are accelerated through the domain. As the particles cross the domain, their charge is deposited on a mesh. The charge is then taken into account the next time the potential field is solved. This process continues until the potential field converges.

The advantage to this type of simulation is that it is very fast, typically running in minutes on a modern computer. This allows fully 3-D domains to be simulated without problem. However, gun simulations cannot model the random motion of the neutral gas directly. Usually a fluid model of some sort is imposed on the domain to estimate the neutral density field. Also, collisions between particles are not simulated directly, but are instead predicted given the ion and neutral densities in the domain.

Gun-type simulations are currently in use at NASA Glenn Research Center (*Malone and Soulas* (2004)), Jet Propulsion Laboratory (*Anderson et al.* (2004)), Colorado State University (*Farnell et al.* (2003)), and in Germany (*Tartz et al.* (2004)).

1.4.2 Particle-In-Cell Models

Particle-In-Cell (PIC) simulations are slower and more complex than a gun code, but they also produce a larger amount of information. A PIC simulation models ions, neutrals, and possibly electrons as particles in the domain. Time is advanced in steps— at each step the potential field is calculated, then all particles are accelerated and advanced in space accordingly.

These simulations may require a very small time-step, and thus large numbers of iterations, to reach a converged flow field. After convergence, the simulation must further be run in order to reduce statistical scatter in the results. A typical simulation may take days to run as a result. The advantage to PIC is that the neutral gas, collisions, and particles resulting from collisions are all simulated directly. This will generally result in more accurate results.

The simulation presented in this work is a PIC model first developed for simulation of the UK T5 ion thruster (*Crofton and Boyd* (1999)). This model is 2-dimensional and axisymmetric and calculates the potential field using a standard Poisson solver. It simulates electrons as a fluid, and includes the use of Direct Simulation Monte Carlo collision modeling.

PIC is also used in a model developed at Virginia Tech by *Kafafy and Wang* (2004), where electrons are modeled as particles with a reduced charge-to-mass ratio, and the potential field is solved using an immersed finite element method. The *Kafafy and Wang* (2004) model is fully 3-dimensional with a rectangular domain, where a quarter of an aperture is simulated in opposite corners of the domain. The model used by *Okawa et al.* (2004) is also a PIC simulation, where the domain is the same as used by *Kafafy and Wang* (2004).

1.4.3 Meshless Potential Field Solution

Particle-In-Cell simulations generally use a rectangular mesh to solve for the potential field. This allows easier tracking of particle motion, and simpler solution of the field. However, such a mesh restricts the accuracy of the solution, as curved surfaces are not simulated accurately. One exception to this is the *Kafafy and Wang* (2004) model, which applies immersed boundaries and an irregular finite element method field solver.

Another approach to the problem is to use a meshless field solver, such as a treecode or a fast multi-pole method. In these schemes, particle forces are computed directly rather than by solving Poisson's equation on a mesh. The force calculation is approximated by treating groups of particles as Taylor expansions about the group center, giving an increase in computational speed. Boundaries are simulated by the boundary integral method, which can easily model any desired geometry.

The primary advantage of a meshless method is its ability to simulate irregular geometry. Ion thruster optics have several types of irregularities in their geometry. Cusp structures are formed on the inside of the aperture wall during fabrication. The grids are also dished for structural stability, adding a slight curvature. Also, as the optics erode, the surfaces become very uneven and irregular. Simulation of such geometry may be necessary in order to obtain very accurate predictions for erosion of ion thruster optics. The inter-particle force computation is also more accurate than in most meshed methods, which has the potential to produce markedly different results from a standard PIC simulation. However, the computation time may be slower than PIC in many cases.

1.5 Thesis Overview

In this thesis, the focus is placed on two primary subjects. The first is Particle-In-Cell simulation of the NEXT ion thruster optics. The operation of the simulation is discussed and the methods and models used are detailed in Chapter II. In Chapter IV, results are given for achieving the optimal accuracy of the simulation, followed by performance and erosion predictions for the NEXT ion engine.

The second subject is the use of a meshless treecode to model domains such as the ion optics. Treecodes have not previously been applied to bounded plasmas, except by *Christlieb et al.* (2004). This work also presents the first use of an axisymmetric treecode applied to any domain. The treecode method is first derived and developed in Chapter III. It is then applied to several domains in Chapter V— a 1-dimensional sheath, a 2-dimensional box domain, an axisymmetric cylinder domain, and an axisymmetric ion optics domain. This is the first known application of a treecode to an axisymmetric plasma. The accuracy and viability of the treecode are discussed, and results are compared to direct particle force computation as well as the PIC method.

Chapter VI summarizes the results and conclusions presented in this work. Possibilities for future research on ion optics simulation are also given for both PIC and the treecode.

CHAPTER II

Particle-In-Cell Ion Thruster Simulation Model

2.1 Particle Injection, Motion, and Boundary Conditions

2.1.1 Model Domain

Ion thruster optics have many thousands of apertures, but it is computationally intractable to simulate all of them at once. Instead, a single ion optics aperture is simulated, with some symmetry assumptions depending on the dimension of the domain. The model presented in this work is 2-dimensional and axisymmetric, with the centerline of an aperture as the centerline of the domain.

A mesh is applied to the domain for particle tracking, sampling, and interactions with the boundary. The mesh is rectangular and uniform, with the axial cell size determined by the Debye length based on the discharge chamber ion density and electron temperature. The cell height is set so that it is near the cell length, and may be adjusted so that the accelerator grid aperture diameter is simulated exactly. The simulation time-step is set to a value low enough such that singly-charged ions are unable to travel axially further than a single cell in one iteration.

The ion optics geometry is determined by the thruster being simulated. This includes the grid thicknesses, aperture diameters, gap between the grids, and the presence and size of cusps on the grid barrels. Cusps on the optics are approximately

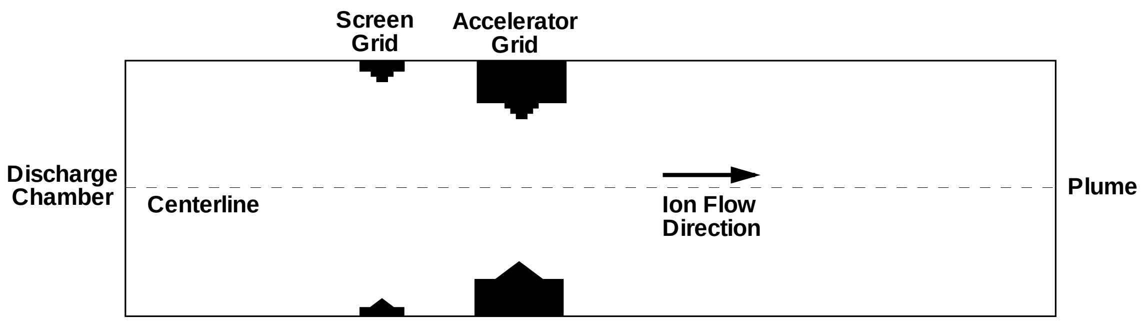


Figure 2.1: Plot of the simulation domain. The upper half is the meshed simulation domain, while the bottom half is the actual domain geometry.

triangular in shape in reality. The mesh is rectangular however, so the cusps are approximated in a stepwise fashion. Figure 2.1 shows a typical domain, with the top half of the plot showing the meshed representation, and the bottom half showing the actual dimensions. The radius of the domain is set to half the center-to-center spacing between adjacent apertures in the ion optics.

The length of the simulation domain upstream of the ion optics must be long enough to resolve the pre-sheath region of the potential field. This is discussed in more detail in Section 4.3. The downstream domain length must be sufficient to resolve the downstream sheath region, but more importantly it must be long enough to capture all charge-exchange ions that will impact on the ion optics. This length is also discussed in more detail in Section 4.4.

2.1.2 Particle Motion

Particles are advanced in space using a first-order approximation to the time-derivative of position:

$$\mathbf{x}^{t+1} = \mathbf{x}^{t} + \mathbf{v}\Delta t \ . \tag{2.1}$$

Particles have three velocities: u, v, and w, which correspond to the axial, radial, and out-of-plane velocities, respectively. Particle position is tracked axially and radially

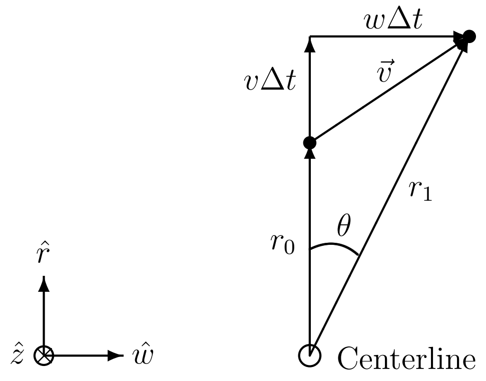


Figure 2.2: Diagram of out-of-plane particle motion and radius calculation.

only. The axial position is updated in a straightforward manner indicated by the above equation, but the radial position must take into account the out-of-plane w velocity as well as the radial velocity. This is because at each time-step, a particle is assumed to be in a plane that intersects the centerline. However, the velocity w moves the particle out of that plane, so the plane must be rotated in order to intersect the particle again. Figure 2.2 shows how this is done.

The new radius is calculated using the triangle shown in the diagram,

$$r_1 = \sqrt{(r_0 + v\Delta t)^2 + (w\Delta t)^2}\ . \tag{2.2}$$

The velocity must also be rotated by the angle θ, giving

$$w_1 = \frac{w_0 r_0}{r_1} \tag{2.3}$$

$$v_1 = \frac{(r_0 + v_0\Delta t) - w_0^2\Delta t}{r_1}\ . \tag{2.4}$$

Doing this ensures that a particle can never have a radius of zero so long as it has a finite w velocity.

2.1.3 Boundary Conditions

The discharge chamber is the upstream boundary of the domain. Neutral particles and ions, both singly and doubly charged, are introduced at this boundary. Each species to be injected has several properties: density, velocity, and temperature. The injection scheme inserts a number of particles into each upstream boundary cell in order to give the correct density. Velocities for the particles are sampled from a Maxwellian distribution centered at the species velocity and characterized by the species temperature.

Ions are given the Bohm velocity in the axial direction with a thermal distribution. The reason for this is that the region upstream of the ion optics approximates the pre-sheath region, and the Bohm criterion (*Riemann* (1991)) states that the Bohm velocity is the necessary velocity for ions to create a stable sheath. Neutral particles are injected with a thermal velocity distribution. Electrons are not modeled as particles, but as a fluid, which will be discussed in Section 2.2.2.

The downstream boundary of the domain is the plume of the thruster. If the thruster being simulated is operated in a vacuum tank with a non-negligible pressure, then neutral particles are injected at the downstream boundary in order to represent the correct pressure. Otherwise, no particles are introduced at the downstream boundary. The upstream and downstream boundaries of the domain are particle deletion boundaries. If a particle in the domain reaches either boundary, it is deleted from the simulation.

The upper domain boundary is largely considered reflective— any particles that impact on this boundary are reflected back into the domain with the opposite radial velocity. It can be shown that this will cause errors in the radial and out-of-plane velocity of particles due to the hexagonal arrangement of apertures. Figure 2.3 shows

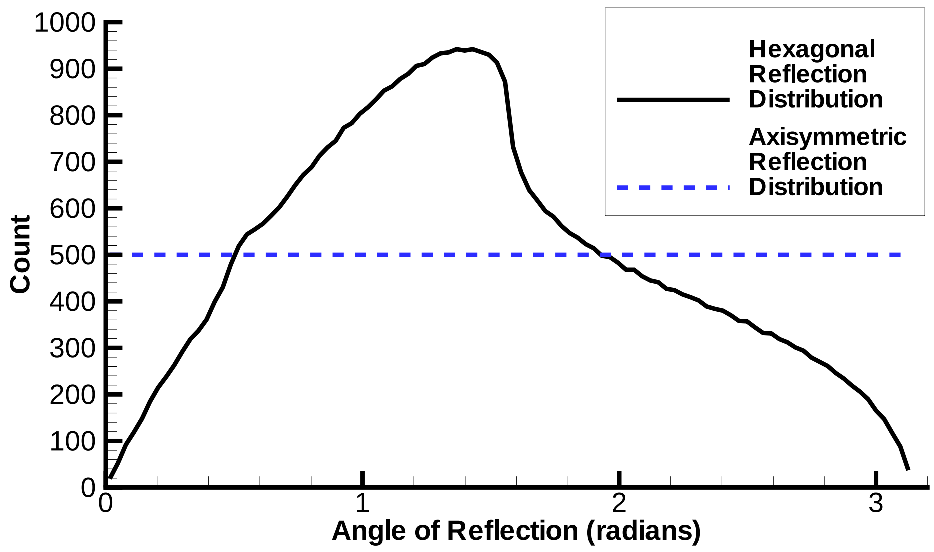


Figure 2.3: Distribution of reflected particle angles when in a hexagonal domain as opposed to an axisymmetric one. The input distribution is the same as the axisymmetric reflection distribution.

the distribution of angles at which particles leaving one aperture in the hexagonal arrangement of apertures will re-enter another aperture. The initial distribution of angles is uniform, so the resulting axisymmetric distribution is uniform as well. However, particles are more likely to have high angles of incidence in the true hexagonal geometry than in the axisymmetric geometry.

This means that a particle that reaches the boundary with a high out-of-plane velocity and a low radial velocity is more likely to be reflected back into the domain with a high radial velocity and a low out-of-plane velocity. The axisymmetric domain does not allow for this, as reflected particles will have the same out-of-plane velocity and the radial velocity will have the same magnitude. The end result in the simulation is that the average radial velocity of reflected particles is lower than if the true geometry were modeled. This most likely has some effect on the radial density distributions of the simulated species. However, the effect is not expected to be significant.

In addition to the reflective condition at the upper boundary, a particle deletion condition may also be used. In the region downstream of the ion optics, some particles may have a negative velocity, primarily due to collisions. If such a particle reaches the upper boundary, it is correct to delete it if the particle's trajectory does not intersect the entire thruster. In other words, the model simulates a single aperture on a larger thruster, so if a simulated particle would never reach the accelerator grid in reality, it should not remain in the domain. Reflecting such a particle back into the domain artificially contains it. Neutral particles are simply deleted if their current trajectory does not intersect the accelerator grid. Ions, however, must consider the potential of the accelerator grid, and so are deleted if their trajectory does not intersect the sheath downstream of the accelerator grid. In both cases, the aperture is assumed to be in the center of the ion optics. This matches the axisymmetric assumption of the domain, and allows a simpler calculation.

Also note that there is a finite divergence angle for the ion beamlet of each individual aperture. The simulation does not allow for this divergence, so the ion density in the region downstream of the ion optics will not decrease due to beam divergence. This means that the potential field is constant as well in this region, but in reality the potential will decrease as the ion beam diverges. This leads to formation of a potential hill in the downstream region which may prevent some charge-exchange ions from reaching the accelerator grid. The simulation is expected to allow more charge-exchange current due to the lack of this potential hill. However, this effect is most likely small, as most of the charge exchange current will originate closer to the ion optics, before there is a significant potential hill.

The ion optics surfaces absorb and re-emit impinging particles at the temperature of the surface. In this process, ions are neutralized and emitted as neutral particles. All particle impacts on the optics are evaluated for the amount of erosion caused. This is performed using a combination of curve fits to experimental sputter yield data. *Polk et al.* (1993) gives the following curve fit to experimental data for xenon sputtering molybdenum:

$$Y = 0.0413781E^{0.565464}\left(1 - \frac{48}{E}\right)^{3.21022}\left(1 + \frac{48M_{Xe}}{E}\right) - 0.0431433\ . \tag{2.5}$$

Here Y is the yield in atoms of molybdenum per atom of xenon, E is the energy of impact in eV, and M_{Xe} is the atomic weight of xenon in amu. This equation applies only to impacts normal to the surface, so it is then scaled by an angular yield to obtain the actual amount of sputtered material. The angular yield for 300 eV ion impacts is given by *Polk* (2002):

$$\frac{Y_\phi}{Y_{\phi=0}} = 1 + 0.251633\phi + 0.6\phi^2 + 0.6\phi^3 \quad , \quad \phi < 0.69813 \tag{2.6}$$

$$\frac{Y_\phi}{Y_{\phi=0}} = -0.057 + 1.90\exp\left[-\left(\frac{\phi - 0.8201}{0.401}\right)^2\right] \quad , \quad \phi \geq 0.69813\ . \tag{2.7}$$

ϕ is the angle of impact in radians, $Y_{\phi=0}$ is the yield obtained from equation 2.5 , and Y_ϕ is the final sputter yield in atoms of molybdenum per atom of xenon. ϕ is always scaled to be between $\pi/2$ and zero. These equations approximate the total amount of material eroded from the ion optics surface when a particle impacts. Since one equation assumes a constant angle and varying energy, and the other assumes a 300 eV ion energy and varying angle, it is likely that there is some error in the computation.

The eroded material can be modeled by injecting molybdenum particles into the simulation. A large number of these particles are created at each impact because the impacts do not occur often. Creating multiple particles enables better statistics to be obtained from the sputtered material. When molybdenum particles impact on a grid surface, they are not re-emitted, but instead are deposited onto the surface. Collisions between these particles and other species in the domain are not calculated in the current model. The density of the sputtered material is orders of magnitude lower than other species densities, so such collision events would be very rare. However, in a study of spacecraft contamination from these particles, collisions with other gas particles would be important.

The simulation also supports an active erosion simulation scheme. In this scheme, each ion optics cell is assigned a number of molybdenum particles at the beginning of the simulation. When all of these particles are sputtered via erosion, that cell is no longer considered an ion optics cell by the simulation. Also, if enough sputtered molybdenum recombines on a given cell, that cell will turn into an ion optics cell. This allows a single simulation to model the erosion pattern of the ion optics over a long time period by appropriately varying the initial number of molybdenum particles per cell. However, the scheme has not been verified by comparison to experimental data, and thus is not used in most simulations.

2.1.4 Radial Weighting Scheme

Each particle injected into a cell has the same numerical weight, indicating the number of actual atoms it represents. However, this means that the number of injected particles must be increased as the radius increases in order to maintain a constant density. The number of particles needed to maintain the correct density in a radial volume slice is given by

$$N_p = \frac{n_0 V}{W} = \frac{n_0 2\pi r \Delta r \Delta t u_0}{W}, \tag{2.8}$$

where N_p is the number of injected particles, n is the density, u_0 is the species velocity, Δt is the time-step, W is the particle weight, r is the radius of the cell center, and Δr is the radial cell dimension. So, if the weight W is held constant, as r increases the number of particles increases linearly. For example, if the centerline cell injects 15 particles, the 10th cell from the centerline will need to inject 315 particles.

To alleviate this problem, the domain is split into weighting levels, where each level has twice the weight of the previous level. For example, the centerline level has a weight of one, the second level has a weight of two, and the third level has a weight of four. This increases the factor W in Eqn. 2.8 so that N_p does not become large. In the simulation, a new weighting level is typically set every time the cell volume increases by a factor of ten. This sets weighting levels at the 6^{th}, 11^{th}, and 21^{st} cells. Also, the factor W is divided into three separate quantities: W_0, W_p, and W_r. W_0 is the base weight of all particles, and is usually referenced to the weight of an ion on the centerline. W_p is the weight of the particle itself. For ions, this is one, but neutrals generally have a higher value so that fewer neutral particles are needed. W_r is the radial weighting factor, and is the actual factor that doubles or halves when a particle crosses a weighting level. Multiplying these three factors together gives the original factor W.

The method of stepped particle weighting means that particles crossing the weighting level must be handled in a special way. There are twice as many particles on the lower side of a weighting boundary as there are above it, so when a particle crosses the boundary while increasing in radius, it must be combined with another particle also crossing the boundary so that the number of particles in the upper weighting level does not increase. Conversely, when a particle crosses the weighting boundary while decreasing in radius, it must be split into two particles to maintain the same number of particles in the weighting level.

Initially, particles were given a 50% chance of deletion as they crossed a weighting level moving upwards. However, when combined with the cloning of particles when crossing the level moving downwards, this caused some simulation problems. If a particle at the very top of the domain had a very negative radial velocity and low axial velocity, then it would cross all of the weighting levels, reach the centerline, then bounce back to the top of the domain, again crossing all of the weighting levels. Assuming three weighting levels, the particle and its clones were cloned three times as it reached the centerline, so that on the centerline there are 8 particles with exactly the same properties. With the random particle deletion as the particles move back upwards, not exactly four will be deleted at the first level, or two at the second. So once the particle group reaches the top of the domain again, there may be two or three particles with the same properties. The group again bounces to the centerline, and now there may be 16 or 24 particles. This process can occur many times, depending on the particle's radial velocity and the length of time it stays in the domain, resulting in a very large grouping of particles with the same exact properties. This is undesirable for many reasons, among them statistical relevancy and the ability of the simulation to handle a large number of particles in a single cell.

To keep this problem from occurring, particles moving upwards through the domain are first paired up by the cell they are in when crossing a weighting level. One of the two particles in each pair is then deleted, so that the number of particles halves exactly. In the case of an odd number of particles, the remaining particle has its weight W_p halved. Then, if that particle crosses a weighting level moving downwards at a later time, it is not cloned, but W_p is doubled instead. This ensures that the total number of particles is neither increased nor decreased on average by the weighting levels.

2.2 The Particle-In-Cell Method

The Particle-In-Cell (PIC) method for solving a potential field operates by applying particle charge onto a mesh, and then solving for the potential field on that mesh. This method is described in *Birdsall and Langdon* (1991), and has been used extensively in varied areas of research for many years (*Sulsky et al.* (1995), *Lasinski et al.* (1999), *Qiang et al.* (2000)).

2.2.1 Derivation of the Non-Dimensional Poisson's Equation

The following derivation and non-dimensionalization of the governing equations was first performed by *Roy* (1995). Poisson's equation is

$$\varepsilon_0 \nabla^2 \phi = e(n_e - n_i) \ . \tag{2.9}$$

Here ϕ is the potential, and n_e and n_i are the electron and ion densities, respectively. In axisymmetric coordinates, this becomes

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial \phi}{\partial r}\right) + \frac{\partial^2 \phi}{\partial z^2} = \frac{e}{\varepsilon_0}(n_e - n_i) \ . \tag{2.10}$$

To non-dimensionalize this equation, representative time and length scales are defined from the plasma frequency and Debye length of the plasma:

$$\tilde{t} = \omega_{p,i0} t \quad , \quad \omega_{p,i0} = \sqrt{\frac{e^2 n_{i,0}}{\varepsilon_0 m_{i,0}}} \tag{2.11}$$

$$\tilde{x} = \frac{x}{\lambda_{D,0}} \quad , \quad \lambda_{D,0} = \sqrt{\frac{\varepsilon_0 T_{e,0}}{e n_{i,0}}} \ . \tag{2.12}$$

$n_{i,0}$ is the reference ion density in m^{-3}, $m_{i,0}$ is the reference ion mass in kg, and $T_{e,0}$ is the reference electron temperature in eV. Next, the potential field and electron temperature at any point are non-dimensionalized by the reference electron temperature, and the densities are non-dimensionalized by the reference ion density.

$$\tilde{\phi} = \frac{\phi}{T_{e,0}}, \quad \tilde{T_e} = \frac{T_e}{T_{e,0}}, \quad \tilde{n} = \frac{n}{n_{i,0}} \ . \tag{2.13}$$

This non-dimensionalization gives almost the same exact equation as in Eqn. 2.10:

$$\frac{1}{\tilde{r}}\frac{\partial}{\partial \tilde{r}}\left(\tilde{r}\frac{\partial \tilde{\phi}}{\partial \tilde{r}}\right) + \frac{\partial^2 \tilde{\phi}}{\partial \tilde{z}^2} = (\tilde{n}_e - \tilde{n}_i) \ . \tag{2.14}$$

2.2.2 Electron Fluid Derivation

The electron density field is simulated as a fluid via the Boltzmann relation. Although the Knudsen number of the domain is too large to indicate validity of a fluid formulation, the low mass and high temperature of the electrons allows the assumption that, in an ion time-step, the electron population is at thermal equilibrium. In other words, the electron population undergoes enough collisions in an ion time-step for equilibrium to be reached. Thus, the fluid equations can be applied, with a focus on the electron momentum equation. In the absence of magnetic field, collisions, and current, this equation is:

$$\frac{d(n\mathbf{v})}{dt} + \nabla \cdot (n\mathbf{v}\mathbf{v}) = \frac{en_e\nabla\phi}{m} - \frac{\nabla p}{m} \ . \tag{2.15}$$

The pressure is replaced by the perfect gas law $p = n_e e T_e$, where T_e is in eV. Taking the limit as mass goes to zero leaves only the force terms, which can be rearranged like so:

$$\frac{\nabla\phi}{T_e} = \frac{\nabla n_e}{n_e} \ . \tag{2.16}$$

Assuming constant temperature, this can be integrated and non-dimensionalized to obtain the Boltzmann relation:

$$\tilde{n}_e(\tilde{\phi}) = \tilde{n}_{e,ref} \exp\left(\frac{\tilde{\phi} - \tilde{\phi}_{ref}}{\tilde{T}_{e,ref}}\right) . \tag{2.17}$$

Substituting into Eqn. 2.14 gives an equation that depends only on ϕ and n_i, along with constant or reference properties.

2.2.3 Discretization of Poisson's Equation

The next step is to discretize Poisson's equation so that it can be solved on a mesh. This is performed using Gauss' Law as given by *Birdsall and Langdon* (1991):

$$\begin{aligned} -\rho_{i,j} &= \frac{2r_{j+1/2}\left(\phi_{i,j+1} - \phi_{i,j}\right)}{(r^2_{j+1/2} - r^2_{j-1/2})(r_{j+1} - r_j)} - \frac{2r_{j-1/2}\left(\phi_{i,j} - \phi_{i,j-1}\right)}{(r^2_{j+1/2} - r^2_{j-1/2})(r_j - r_{j-1})} \\ &\quad + \frac{\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j}}{(z_{i+1/2} - z_{i-1/2})^2} . \end{aligned} \tag{2.18}$$

Here $\rho_{i,j}$ is the charge density of cell i, j. In a uniform mesh, with a constant Δz and Δr, this equation reduces to the following:

$$\begin{aligned} -\rho_{i,j} &= \frac{r_{j+1/2}\left(\phi_{i,j+1} - \phi_{i,j}\right)}{r_j \Delta r^2} - \frac{r_{j-1/2}\left(\phi_{i,j} - \phi_{i,j-1}\right)}{r_j \Delta r^2} \\ &\quad + \frac{\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j}}{\Delta z^2} . \end{aligned} \tag{2.19}$$

For the special case of $j = 0$, the following is used instead:

$$-\rho_{i,0} = \frac{2\left(\phi_{i,1} - \phi_{i,0}\right)}{r_{1/2} \Delta r} + \frac{\phi_{i+1,0} - 2\phi_{i,0} + \phi_{i-1,0}}{\Delta z^2} . \tag{2.20}$$

The charge density involves the Boltzmann relation for the electron density, making Eqn. 2.19 non-linear in potential. To account for this, a Newton-Raphson method is applied as given in *Hockney and Eastwood* (1981). This method gives quadratic convergence of the discretized Poisson's equation at a point using the following iterative scheme:

$$\left.\frac{\partial N}{\partial \phi}\right|^{(t)} \phi^{(t+1)} = -N(\phi^{(t)}) + \left.\frac{\partial N}{\partial \phi}\right|^{(t)} \phi^{(t)} . \tag{2.21}$$

To use this scheme, the system in question is first recast:

$$N(\tilde{\phi}) = \tilde{\nabla}^2\tilde{\phi} - \tilde{n}_e(\tilde{\phi}) + \tilde{n}_i(\mathbf{x}) = 0 \ . \tag{2.22}$$

This equation is then differentiated with respect to the potential:

$$\frac{\partial N}{\partial \tilde{\phi}} = \tilde{\nabla}^2 - \frac{\tilde{n}_e(\tilde{\phi})}{\tilde{T}_{e,ref}} \ . \tag{2.23}$$

Substituting Eqns. 2.22 and 2.23 into Eqn. 2.21 gives the iteration for the system:

$$\left(\tilde{\nabla}^2 - \frac{\tilde{n}_e(\tilde{\phi}^t)}{\tilde{T}_{e,ref}}\right)\tilde{\phi}^{t+1} = \tilde{n}_e(\tilde{\phi}^t)\left(1 - \frac{\tilde{\phi}^t}{\tilde{T}_{e,ref}}\right) - \tilde{n}_i(\mathbf{x}) \ . \tag{2.24}$$

2.2.4 Alternating-Direction Implicit Method

The simulation solves Eqn. 2.24 on a mesh using one of two methods. The first is the Alternating-Direction-Implicit (ADI) method. In this method, horizontal and vertical strips of the domain are solved implicitly in succession. ADI is generally more computationally intensive than other methods, but it has a high rate of convergence and also provides better stability than an explicit solver (*Hirsch* (1997)). Eqn. 2.24 can be written in finite difference form using Eqn. 2.19, with all quantities now non-dimensionalized:

$$\begin{aligned}
&\frac{r_{j+1/2}}{r_j\Delta r^2}\phi^{t+1}_{i,j+1} + \frac{r_{j-1/2}}{r_j\Delta r^2}\phi^{t+1}_{i,j-1} + \frac{1}{\Delta z^2}\phi^{t+1}_{i+1,j} + \\
&\frac{1}{\Delta z^2}\phi^{t+1}_{i-1,j} - 2\left(\frac{1}{\Delta z^2} + \frac{1}{\Delta r^2} + \frac{n_e(\phi^t_{i,j})}{2T_{e,ref}}\right)\phi^{t+1}_{i,j} \\
&= n_e(\phi^t_{i,j})\left(1 - \frac{\phi^t_{i,j}}{T_{e,ref}}\right) - (n_i)_{i,j} \ .
\end{aligned} \tag{2.25}$$

This equation is then transformed so that the update to the potential field is calculated rather than the new field values. First, define the following:

$$\Delta\phi = \phi^{t+1} - \phi^{t} \tag{2.26}$$

$$A^{C} = 2\left(\frac{1}{\Delta z^2} + \frac{1}{\Delta r^2}\right) \tag{2.27}$$

$$A^{C*} = A^{C} + \frac{n_e(\phi^t_{i,j})}{T_{e,ref}} \tag{2.28}$$

$$A^{S} = \frac{1}{\Delta z^2} \tag{2.29}$$

$$A^{T}_{j} = \frac{r_{j+1/2}}{r_j \Delta r^2} \tag{2.30}$$

$$A^{B}_{j} = \frac{r_{j-1/2}}{r_j \Delta r^2} \ . \tag{2.31}$$

Equation 2.25 can now be written

$$\begin{aligned}
&A^T_j \Delta\phi_{i,j+1} + A^B_j \Delta\phi_{i,j-1} + A^S \Delta\phi_{i+1,j} + A^S \Delta\phi_{i-1,j} - A^{C*} \Delta\phi_{i,j} = \\
&-\omega \left[A^T_j \phi^t_{i,j+1} + A^B_j \phi^t_{i,j-1} + A^S \phi^t_{i+1,j} + A^S \phi^t_{i-1,j} - A^C \phi^t_{i,j} - n_e(\phi^t_{i,j}) + (n_i)_{i,j}\right] \\
&= K^t_{i,j} \ .
\end{aligned} \tag{2.32}$$

Here, ω is a relaxation factor that may be used to increase the convergence rate of the solver. In this work, ω is assumed to be one, unless noted otherwise. $K^t_{i,j}$ represents the right-hand side of Eqn. 2.32, and is a known quantity at a node at each iteration.

ADI operates by first solving for the potential in each inner row of the domain:

$$A^S \Delta\phi_{i-1,j} + A^S \Delta\phi_{i+1,j} - A^C \Delta\phi_{i,j} = K^t_{i,j} \ . \tag{2.33}$$

This tridiagonal system is solved using the Thomas algorithm, given in *Hockney and Eastwood* (1981). Each system of equations corresponds to a row in the domain with i varying from 2 to $N_z - 1$. The radial index j ranges from 2 to $N_r - 1$, covering the entire domain. The same process is then applied to the columns of the domain with this system of equations:

$$A_j^B \Delta\phi_{i,j-1} + A_j^T \Delta\phi_{i,j+1} - A^C \Delta\phi_{i,j} = K_{i,j}^t \ . \tag{2.34}$$

In this case, i is constant in each column and j varies between 2 and $N_r - 1$.

Although the ADI method converges quickly compared to other solvers, it has some disadvantages. The solver performs most efficiently on rectangular blocks of cells in the domain, but geometry such as the stair-stepped cusps seen in Fig. 2.1 causes the solver to work on a line-by-line basis. Also, for fine meshes, the solver becomes very slow.

2.2.5 Multi-grid Potential Field Solver

The other method used to solve the potential field in the domain is a multi-grid potential solver (*Briggs et al.* (2000)). This type of method solves for the potential on a set of meshes, starting with the finest mesh and moving to the coarsest possible mesh, before moving back to the fine mesh. On a fine mesh, high frequency error is most efficiently eliminated. As the mesh becomes coarser, error that was previously low-frequency becomes high-frequency, and is thus easier to eliminate.

Multigrid does not need a large number of iterations to converge at each mesh level, so performing less work in a single iteration produces the best results. It is more advantageous to have a fast solver rather than an expensive but quickly converging one such as ADI. So, the Gauss-Seidel (GS) method (*Atkinson* (1993)) is used to solve the potential field at each mesh level. The discretized Poisson's equation is similar to that used in the ADI method with the exception that only the central node has an updated value:

$$-A^{C*} \Delta\phi_{i,j} = K_{i,j}^t \ . \tag{2.35}$$

Rearranging this gives the update for each node at each time-step:

$$\phi_{i,j}^{t+1} = \phi_{i,j}^{t} - \frac{K_{i,j}^{t}}{A^{C*}} \ . \tag{2.36}$$

Multigrid methods generally operate strictly on the residual at each node, rather than the absolute value of the equation. This is because solving for the potential on the coarsest mesh does not solve the correct problem. So, at each mesh level, the residual is either restricted to the next coarser mesh or prolonged to the next finer mesh. However, the system given in Eqn. 2.25 contains a non-linear term for the electron density contribution. This does not allow a clean residual formulation, so instead the full approximation storage (FAS) scheme must be used. In this scheme, the potential field is solved at each mesh level, but with adjustments to the source term according to the residuals of the next finer mesh. The problem can be represented as

$$Au = f \ . \tag{2.37}$$

Here, A is the update applied to the potential field u, and f is the ion density source term. After n iterations of the GS solver on mesh 1, the finest mesh, the residual will be

$$R_1 = Au_1(n) - f_1 \ . \tag{2.38}$$

The residual and $u(n)$ are restricted to the next coarsest mesh, and the initial restricted potential is stored. Then the restricted residual and initial coarse-mesh residual set the source term on the coarser mesh. This gives the new system components to be iterated on:

$$\begin{aligned}
&u_2(0) = I_1^2 u_1(n) \\
&\text{Update Boundary Conditions} \\
&f_2 = -I_1^2 R_1 + A u_2(0) \\
&R_2 = A u_2(n) - f_2 \ . \qquad (2.39)
\end{aligned}$$

I_i^j is a restriction or prolongation operator that moves a field from mesh i to mesh j. Neumann boundary cells are set by the node nearest to the boundary, so they must be updated because restriction removes the node next to the boundary. The process in Eqn. 2.39 can be repeated up to the coarsest possible mesh. Once the coarsest mesh has been reached, the field is prolonged back down to the finer meshes. This is done by subtracting the difference between the converged field and the initial field at that mesh level, then prolonging that difference to the fine mesh. Finally, the prolonged difference is added to the fine mesh potential field and the field is iterated on again.

$$\begin{aligned}
&\Delta u_2 = u_2(n) - u_2(0) \\
&u_1 = u_1 + I_2^1 \Delta u_2 \\
&A u_1 = f_1 \ . \qquad (2.40)
\end{aligned}$$

Again, this process is repeated until the finest mesh is reached.

The method described above moves directly from the finest mesh to the coarsest, then back to the finest. This is called a V-cycle in the multigrid method. Another scheme is to move in a zigzag pattern, first going up to the coarsest mesh, then down one level, then back up, then down two levels and so on. This is called a W-cycle, and is the pattern used in the simulation. It provides better performance than the

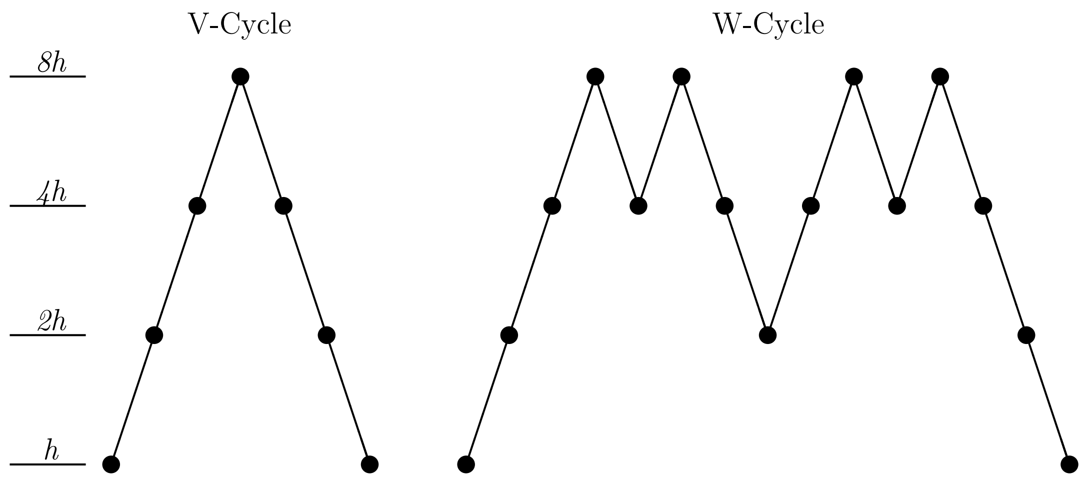


Figure 2.4: Illustration of the V- and W-cycles for traversing multi-grid meshes. h is the cell width at the finest level.

V-cycle because it converges in fewer cycles, spending relatively less time on the finer meshes. Figure 2.4 shows the difference between the two types of cycles.

2.2.6 Potential Solver Boundary Conditions and Reference Quantities

The boundary conditions of the potential solver are determined by the ion thruster domain. The discharge chamber and plume of the ion beam are assumed to be neutral, so both the upstream and downstream boundaries of the domain have Neumann boundary conditions. The outer radial boundary is also Neumann, as symmetry between apertures is assumed. The ion optics are represented by Dirichlet boundaries at the potential of the grids. The discharge chamber properties are used as the reference state for non-dimensionalization. So, the dimensional reference density is the discharge ion density, and the dimensional reference electron temperature is the discharge electron temperature.

The electric field imposed by the ion optics prevents discharge electrons from reaching the plume, and also prevents plume electrons from reaching the discharge chamber. This means that there are two electron populations in the domain, the upstream population and the downstream population, each with its own reference state for the Boltzmann relation. The upstream reference state is identical to the dimension reference state: $T_{e,ref} = T_{e,0}$ and $n_{e,ref} = n_{i,0}$. Also, the reference potential is the discharge chamber potential. The downstream reference state corresponds to plume properties. The reference electron density is the average ion density in the plume because charge neutrality is assumed. The reference electron temperature is the plume electron temperature, and the reference potential is the neutralizer cathode's potential relative to ground.

2.2.7 Particle Weighting to the Mesh

In order to obtain a charge density field for the charged particles in the simulation, the particles must be weighted to the mesh. This is done using a charge-conservative weighting scheme developed by *Ruyten* (1993) and implemented by *Roy* (1995). The scheme uses the area of the cell a particle is in to determine the amount of charge to assign to each of the four cell nodes. First, four lengths corresponding to the distances from the particle to the cell vertices are defined:

$$S_j = \frac{(r_{j+1} - r_p)(2r_{j+1} + 3r_j - r_p)}{2\Delta r (r_{j+1} + r_j)} \tag{2.41}$$

$$S_{j+1} = \frac{(r_p - r_j)(3r_{j+1} + 2r_j - r_p)}{2\Delta r (r_{j+1} + r_j)} \tag{2.42}$$

$$S_i = \frac{z_{i+1} - z_p}{\Delta z} \tag{2.43}$$

$$S_{i+1} = \frac{z_p - z_i}{\Delta z} \; . \tag{2.44}$$

A constant cell size is assumed here, and the particle's position is given by z_p, r_p. These lengths are multiplied to obtain the weighting on each node. So, the weight to node i, j is given by $S_i S_j$, the weight for node $i, j+1$ is $S_i S_{j+1}$, and so on. Once the charge of the particle has been distributed to each node, the charge density is determined by dividing by the volume of the node, which at node i, j is $2\pi r_j \Delta r \Delta z$. Calculating this weighting for each particle in the simulation gives the ion density field $(n_i)_{i,j}$.

2.2.8 Electric Field and Force Calculation

The electric field is calculated at each node by taking the gradient of the potential field. A central-differences scheme is used, giving this stencil:

$$E_s = -\frac{\phi_{k+1} - \phi_{k-1}}{2\Delta s}, \tag{2.45}$$

where s is either z or r, and k is the corresponding index i or j. The field accelerates particles by applying the area weighting back from the nodes to the particle position. The electric field at the particle is determined by summing each node's area contribution times that node's electric field. The particle velocity is then updated using a first order approximation to the velocity time derivative:

$$\mathbf{v}_p^{t+1} = \mathbf{v}_p^t + \frac{q}{m}\mathbf{E}\Delta t \ . \tag{2.46}$$

2.3 Direct Simulation Monte Carlo

Collisions in the simulation are calculated using the Direct Simulation Monte Carlo (DSMC) method, developed by *Bird* (1994). This method pairs up particles in a cell, then randomly collides these pairs depending on the collision cross-sections. Charge-exchange collisions are considered between ions and neutrals, while momentum-exchange collisions are evaluated between all particles except sputtered grid material. The collision cross-sections for charge-exchange collisions between

Xe^+ and Xe and Xe^{++} and Xe are given by *Miller et al.* (2002) as the following:

$$\sigma_{Xe^+,Xe} = 10^{-20}\left[(87.3 \pm 0.9) - (13.6 \pm 0.6)\log(E)\right] \tag{2.47}$$

$$\sigma_{Xe^{++},Xe} = 10^{-20}\left[(45.7 \pm 1.9) - (8.9 \pm 1.2)\log(E)\right], \tag{2.48}$$

where E is the collision energy in eV, and σ is the collision cross section in m^2.

It has been shown by *Boyd and Dressler* (2002) that the momentum-exchange collision cross-sections are identical to the CEX cross-sections for ion-neutral collisions. However, simulation erosion rate results are closer to experimental values using momentum-exchange cross-sections given by *Dalgarno et al.* (1958) (see Section 4.2). This discrepancy has not been resolved as of yet, so the Dalgarno cross-sections are used in most simulations. These cross-sections are much smaller than the CEX cross-sections, so there will be many fewer momentum-exchange collisions.

In a collision, it is very possible that the two particles have different weights, especially since neutrals and ions have different base weights in the simulation. To account for this, the particle with the higher weight is split into two particles before the collision, such that one of the new particles has the same weight as the collision partner. This process is described in *Boyd* (1996).

Collisions are not processed in every time step of the simulation. The densities in the domain are generally too low for collisions to occur very often, so instead collisions are processed every 100 iterations. The collision probability is then increased by this same factor in order to obtain the correct number of collisions.

2.4 Simulation Operation

The simulation begins by reading in a set of three input files that contain the required information for the simulation. This includes gas species densities, velocities and temperatures, cell sizes and numbers, ion optics dimensions, simulation time-step, the number of iterations to sample, etc. From these quantities, the code sets up the necessary variables used in the simulation.

Next, the model initializes neutral flow by filling the domain with neutral particles, then running with normal neutral injection until the flow is steady. The time-step used here is much higher than the normal simulation time-step because the neutrals only have thermal velocities, which are much lower than the accelerated ion velocities in the main simulation. In each time-step, the particles are simply advanced in space, and every 100 iterations collisions are calculated.

Once this is complete, the potential field is initialized using the ADI potential solver. ADI is used instead of multigrid because the potential field begins at zero in all interior regions, and thus the error is large. The high stability of ADI allows the field to be easily solved at this point. Once the field is initialized, the electric field is computed.

The main iteration now begins. First, charged particles are weighted to the mesh, then the potential field is solved using multigrid. The charged particles are then accelerated, after which the electric field is updated using the new potential field. Particles are then moved in space, and weighting level changes are processed. New particles are injected, and then boundary interactions are processed. After every 100 iterations, collisions are processed at this point. When enough iterations have been run to establish a steady state in the flow, particle properties are sampled for averaging in each cell. After the simulation is finished, the sampled particle properties are averaged and processed to obtain performance results and information on the flow field.

CHAPTER III

The Treecode Method

3.1 Types of Potential Field Solvers

Many methods have been developed for the solution of the potential field in a domain. Each scheme has different advantages and disadvantages in areas such as accuracy and computational efficiency. The primary methods are outlined below.

3.1.1 Direct Summation

Direct summation is the most basic method for computing the electric force on a particle due to other particles. To find the total force on a single particle, the force between the target particle and every other particle in the domain is computed. This process requires on the order of N computations for each particle, where N is the number of particles. To find the force on all particles then takes $O(N^2)$ computations. This is the most accurate method for force calculation, as no approximations are made, but it is also the slowest. Direct summation also requires an external scheme to represent boundary conditions, such as a boundary integral method.

3.1.2 Particle-In-Cell

The Particle-In-Cell method approximates the forces in the domain by applying a mesh, weighting the charge of each particle to the mesh, then solving for the potential field. The electric field in each mesh cell then provides the force on the particles in that cell. This method is much faster than direct summation and many other methods as well. The process of particle weighting to a regular mesh and particle force calculation is only $O(N)$, while the solution of the potential field is $O(N_x N_y)$, where N_x is the number of nodes in the x-direction, and N_y is the number of nodes in the y-direction. However, the use of an irregular mesh requires $O(NN_{mesh})$ computational work instead of $O(N)$ in order to weight the particles to the mesh nodes, where N_{mesh} is the total number of mesh nodes. The field solution is independent of the number of particles.

So long as the mesh is regular and not too fine, PIC will outperform most other methods. The disadvantage to this approach is that inter-particle forces are not described within a mesh cell, and so the force calculation is not extremely accurate. This is mitigated somewhat in situations where particles are Debye shielded, or when the imposed boundary conditions are much stronger than the inter-particle forces.

One other advantage for PIC is that the boundary conditions are easily integrated into the mesh edges, so that all aspects of the potential field are solved at once. However, this limits the boundary to the mesh itself, unless a scheme such as cut-cells is used. If a rectangular mesh is used, then an irregular boundary will not be simulated accurately. A finite-element method could also be used for the solution of the potential field, but this has the additional computational cost of an irregular mesh.

The PIC method is discussed in further detail in Section 2.2.

3.1.3 Particle-Particle/Particle-Mesh

The Particle-Particle/Particle-Mesh (P^3M) method was developed by *Hockney and Eastwood* (1981) as a hybrid algorithm mixing PIC's efficiency and direct summation's accuracy. In this method, the potential field is computed on a mesh as in PIC. However, in addition to the force provided by the mesh, inter-particle forces are calculated for particles near the target particle. These particles are found by use of a "chaining mesh": a coarse mesh containing an ordered list of particles in each cell. This allows the near-neighbor particles to be found relatively easily. The added work only slightly increases the computation cost of PIC, depending on the amount of inter-particle forces computed. The computational efficiency can range from $O(N)$ to about $O(N\log(N))$, with the same amount of work for the mesh solution as in PIC. P^3M also shares the advantages and disadvantages of PIC with regard to boundary conditions. The imposed mesh allows straightforward application of boundary conditions, but that mesh also limits the complexity of the boundary.

3.1.4 The Treecode Method

The treecode method for particle force and potential computation was developed by *Barnes and Hut* (1986) as a way to reduce the computation cost associated with direct sum force calculation. The idea behind the treecode method is that, at large distances, a cluster of particles appears to be a single particle located at the center of the cluster, with a total charge equal to that of its members. This is only a first order approximation— if a Taylor expansion is performed about the center of the cluster, then the force contribution of the cluster may be computed as accurately as desired. For near-neighbor particles, the force contribution is computed directly. This reduces the total computation cost to $O(N\log(N))$ while maintaining accurate inter-particle forces. Also, unlike meshed methods, no computation is wasted on empty space—forces are only generated on particles. The main disadvantage of the Barnes-Hut treecode is that, in its most basic form, the computational work is higher than in other methods. Treecodes also require an additional method for the simulation of boundary conditions. However, the treecode may also be applied to the boundary method, as is discussed in Section 3.4.2.

Treecodes can be applied to any problem where a Green's function is used for interactions between objects. This applies to electrostatic interactions (*Christlieb et al.* (2004)), gravitational interactions (*Alimi et al.* (2003)), and vortices in fluids (*Lindsay and Krasny* (2001)).

3.1.5 Fast-Multi-Pole Method

The Fast-Multi-Pole (FMM) Method for particle force computation was developed by *Greengard and Rokhlin* (1987). This method is a close relative to the treecode method, except the force is computed between two clusters at a time, rather than between a single particle and a single cluster. Taylor expansions are performed about the center of both the target cluster and the source cluster. These expansions interact with each other, determining the force on all particles in the target cluster at once. This allows a reduction in computation cost to approximately $O(N)$.

The FMM method requires large amounts of memory for storage of the moments of the Taylor expansions, and also adds a further level of approximation to the force computation. FMM's structure also allows applications to acoustical scattering (*Rokhlin* (1990)). Also, FMM can provide faster solution of problems than fast-Fourier transforms, while maintaining the same accuracy. FMM boundaries may be represented by boundary integrals. As in the treecode, FMM can be used to evaluate the integrals efficiently.

3.2 General Treecode Derivation

The treecode method is developed in this work for application to ion thruster optics modeling. The treecode was chosen as an alternative to the PIC method normally used because of its more accurate particle force calculation. The ability to use the boundary integral method to accurately simulate the boundaries of the domain is also an advantage, as ion thruster optics can have complex geometry.

To derive the treecode algorithm, first begin with Poisson's equation:

$$\varepsilon_0 \nabla^2 \phi = \sum_{j=1}^{N} q_j \delta(\vec{x} - \vec{x}_j), \tag{3.1}$$

where $\vec{x}_j$ is the position of particle j, q_j is its charge, and $\delta(x)$ is the Dirac delta function. Integrating twice gives the potential field at a point $\vec{x}$:

$$\varepsilon_0 \phi(\vec{x}) = \sum_{j=1}^{N} q_j G\left(\vec{x}, \vec{x_j}\right), \tag{3.2}$$

where $G\left(\vec{x}, \vec{x_j}\right)$ is the Green's function for whatever dimension of domain is being considered. Next, a Taylor expansion is performed about a point $\vec{x_c}$, which will be the center of the treecode cluster. In a 1-D domain, this is the following:

$$\phi_c(x) = \sum_{j=1}^{N} \left[\frac{q_j}{\varepsilon_0} \sum_{k=0}^{\infty} \frac{(x_j - x_c)^k}{k!} G^{(k)}(x, x_c) \right] . \tag{3.3}$$

Here $G^{(k)}(x, x_c)$ is the k^{th} derivative of the Green's function, and $\phi_c(x)$ is the potential at point x due to cluster c.

The Taylor expansion is performed up to n_t terms in practice. Accounting for this and rearranging Equation 3.3 gives the following:

$$\phi_c(x) \cong \sum_{k=0}^{n_t} \left[\frac{G^{(k)}(x, x_c)}{\varepsilon_0 k!} M_c^k \right] . \tag{3.4}$$

M_c^k is defined as

$$M_c^k = \sum_{j=1}^{N_c} q_j (x_j - x_c)^k, \tag{3.5}$$

where c refers to the cluster of particles being expanded about, and N_c is the number of particles in that cluster. M_c^k is independent of the position x, so this term only needs to be evaluated once in order to obtain the potential at many different points. Also, in some cases a recursion relation can be obtained for the derivatives of the Green's function, allowing fast computation of high order Taylor expansions.

3.3 Treecode Operation

Using the treecode to determine forces or potentials in a domain involves several steps. First, the tree and its clusters must be formed. This is done by starting with a box that encompasses the domain, whether it be 1-D, 2-D, or 3-D. The box is then decreased in size such that its position and dimensions correspond to maximum and minimum positions of the particles in the domain. Next, the box is divided into two, four, or eight sub-boxes, depending on the dimension of the domain. These boxes are in turn reduced to their particle limits, and if no particles are in a box, then it is not added to the tree. This process continues until the lowest level box has fewer than the maximum allowable number of particles per box. This limiting number is defined as the parameter m_{max}. Each cluster has a set of properties, including its dimensions, radius, particle members, and child clusters. These properties are all calculated during the tree formation. Figure 3.1 shows the process of tree formation and the resulting tree structure.

It is possible that, in the division process, a cluster may form that has a poor aspect ratio, i.e. it is very long and thin. The Taylor expansion of such a cluster has poor convergence, so when division of boxes is carried out some boxes may not be

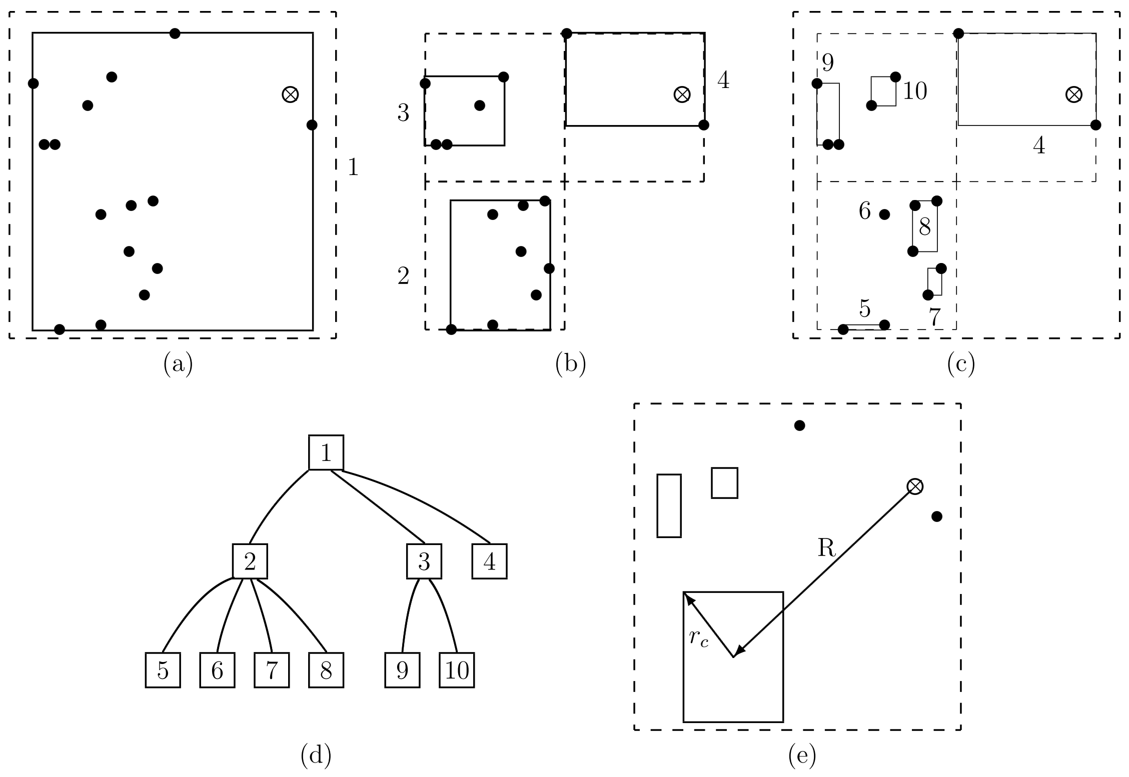


Figure 3.1: Illustration of the tree formation process. m_{max} is set to three in this case. In (a), the initial cluster is shrunk to fit the particles. Then in (b) this box is divided into four, and those four are shrunk to fit their particles. The empty box is ignored. In (c), the clusters are further divided and shrunk. (d) shows the resulting tree structure. Diagram (e) illustrates the view of the domain for the $\otimes$ particle, with indications of the distance to the center of cluster 2, and the radius of cluster 2.

divided in a given direction if they have a poor aspect ratio. In other words, a cluster whose length is twice its height will only be divided into left and right boxes, not into upper and lower boxes as well. Obviously, this does not apply to 1-dimensional problems.

The next step is the computation of the cluster moments. As this process is relatively fast compared to other processes in the treecode, all moments for each cluster are computed, regardless of whether that moment will actually be used. The moments are computed up to a maximum possible number of terms in the Taylor expansion.

Finally, the tree is traversed recursively in order to compute the desired quantity. For reference, let $\vec{x}_p$ be the point at which the potential or force is to be computed, $\vec{x}_c$ is the center of cluster c, and $\vec{x}_j$ is the location of any particle in the domain. r_c, the radius of cluster c, is calculated as the distance between the midpoint of the cluster and a corner of the cluster. At each cluster in the tree, the value of $R = |\vec{x}_p - \vec{x}_c|$ is computed. If the ratio of the cluster radius to R is less than the acceptance parameter α, the cluster is "accepted." α is between zero and one, with higher values indicating that more clusters will be accepted. The higher the ratio is between r_c and R, the worse the cluster approximation becomes, so more terms in the Taylor expansion are necessary to achieve a constant accuracy.

If a cluster is accepted, then the n_t-term Taylor expansion is computed about the center of that cluster as in Eqn. 3.4. Otherwise, the child clusters of that cluster are evaluated for acceptance individually. When a cluster is reached that is not accepted, and has no child clusters, the potential or force contribution is computed by directly summing over the member particles $\vec{x}_j$. The total potential or force is a sum of the contributions of each tree branch.

As an example, consider the particle represented by $\otimes$ in Figure 3.1(e). The treecode function is called starting at cluster 1. This box is not accepted because the target is a member, and thus $r_c/R > 1$. The treecode function now calls itself for clusters 2, 3, and 4. Cluster 2 is accepted as it is far enough away, with a small enough radius. As such, its child clusters do not need to be considered, as their contribution is accounted for by cluster 2. Cluster 3 is processed next, and is not accepted, so it calls the treecode function for clusters 9 and 10. Both of these clusters are accepted, and their contribution is added to the total. Finally, cluster 4 is reached. It has no child clusters, and is not accepted, so the force from the two members other than the target is computed directly. Instead of 15 direct sum computations, two particle interactions and three cluster interactions are processed.

Note that the treecode methodology applied above may not be the most efficient scheme. Another approach is to consider acceptance of clusters based on how many Taylor terms are required to achieve the desired accuracy. If the number of terms needed results in more computational work than using direct summation or descending to child clusters, then the cluster is not accepted. When the cluster is accepted, the number of Taylor expansion terms is determined by the amount of desired accuracy. In this case, it is not necessarily faster to compute the moments of all clusters before performing the force computation. Instead, the moments are computed as they are needed during the force computation.

3.4 The Boundary Integral Method

A boundary integral method is used to represent boundary conditions (*Katsikadelis* (2002), *Banerjee* (1994)) in the treecode. In this method, the boundary is divided into panels that have a constant charge distributed along them. When the charge is integrated on each panel, the correct boundary conditions are obtained.

Poisson's equation is first divided into a particular solution corresponding to the free-space charged particle contribution, and a homogenous solution corresponding to the boundary contribution:

$$\phi(\vec{x}) = \phi_h(\vec{x}) + \phi_p(\vec{x}) \tag{3.6}$$

$$\nabla^2 \phi_p = \rho \tag{3.7}$$

$$\nabla^2 \phi_h = 0 \ . \tag{3.8}$$

Here ρ is the charge density in the domain. Green's second identity gives

$$\int_\Omega \left(v\nabla^2 u - u\nabla^2 v\right) d\Omega = \int_\Gamma \left(v\nabla u - u\nabla v\right) \cdot \hat{n} ds \ . \tag{3.9}$$

v is chosen such that $\nabla^2 v = \delta(P - Q)$, where P is some point in the domain Ω at which the potential is evaluated and Q is any point in Ω. u is set to the homogenous potential ϕ_h. In this case, v will be the free-space Green's function of the domain. Substituting into Eqn. 3.9 gives

$$\int_\Omega \Big(v\nabla^2 \phi_h - \phi_h \delta(P - Q)\Big) d\Omega = \int_\Gamma \left(v\nabla \phi_h - \phi_h \nabla v\right) \cdot \hat{n} ds \ . \tag{3.10}$$

The first half of the first integral is zero due to Eqn. 3.8. The property of the Dirac delta function can be used to evaluate the second half of the integral as $-\phi_h(P)$. The result is an expression for the potential due to the boundary anywhere in the domain:

$$\phi_h(P) = \int_\Gamma \left(\phi_h(q)\frac{\partial G(P,q)}{\partial n_q} - G(P,q)\frac{\partial \phi_h(q)}{\partial n_q}\right) ds \ . \tag{3.11}$$

The variable q is the integration variable on the boundary Γ, while $\phi_h(q)$ and $\partial\phi_h(q)/\partial n_q$ are the potential and potential slope on the boundary. The notation $\partial f/\partial n$ represents $\nabla f \cdot \hat{n}$.

The normal derivative of Eqn. 3.11 is

$$\frac{\partial \phi_h(P)}{\partial n_P} = \int_\Gamma \left(\phi_h(q) \frac{\partial^2 G(P,q)}{\partial n_q \partial n_P} - \frac{\partial G(P,q)}{\partial n_P} \frac{\partial \phi_h(q)}{\partial n_q} \right) ds \ . \tag{3.12}$$

This gives the slope of the potential at any point in the domain, and thus gives the electric field in the direction specified by the normal n_P.

If the boundary is Dirichlet, then $\phi_h(q)$ is known, while $\partial \phi_h(q)/\partial n_q$ is known on a Neumann boundary. Note that each quantity is determined by combining the potential boundary conditions and the particular solution on the boundary, depending on whether the boundary is Dirichlet or Neumann:

$$\phi_h(\vec{x}_i) = \phi(\vec{x}_i) - \phi_p(\vec{x}_i) \tag{3.13}$$

$$\frac{\partial \phi_h(\vec{x}_i)}{\partial n_i} = \frac{\partial \phi(\vec{x}_i)}{\partial n_i} - \frac{\partial \phi_p(\vec{x}_i)}{\partial n_i} \ . \tag{3.14}$$

Equation 3.13 is applied for Dirichlet boundaries, while Eqn. 3.14 is used for Neumann boundaries. The particular solution is determined by charged particles in the domain, and is computed by the treecode.

Equation 3.11 is only valid for points in the interior of the domain, so a different relation is needed for points on the boundary. The limit is taken as the point P approaches a boundary point B. The integral in 3.11 is also split to separately integrate over the boundary point B, which is contained in the boundary β_ε. This separates out the singularity that occurs when $P = B$:

$$\begin{aligned} \lim_{P \to B} \phi(P) = \lim_{P \to B} \int_{\Gamma - \beta_\varepsilon} \left(G(P,q) \frac{\partial \phi_h(q)}{\partial n_q} - \phi_h(q) \frac{\partial G(P,q)}{\partial n_q} \right) ds \ + \\ \lim_{P \to B} \int_{\beta_\varepsilon} \left(G(P,q) \frac{\partial \phi_h(q)}{\partial n_q} - \phi_h(q) \frac{\partial G(P,q)}{\partial n_q} \right) ds \ . \end{aligned} \tag{3.15}$$

The limit of the left-hand side is simply $\phi(B)$, while the Cauchy principal value theorem is applied to the second integral on the right-hand side. Finally, the limit is taken as β_ε goes to zero, giving

$$\phi(B) = \int_\Gamma \left(G(B,q)\frac{\partial \phi_h(q)}{\partial n_q} - \phi_h(q)\frac{\partial G(B,q)}{\partial n_q} \right) ds + \frac{\phi(B)}{2} \ . \tag{3.16}$$

The one-half term is moved to the left-hand side to give the final result. The normal derivative of this equation gives the slope of the potential field on the boundary.

3.4.1 Computation of the Panel Charge

Equation 3.11 can be discretized such that the boundary is divided into panels:

$$\phi_h(P) = \sum_{j=0}^{N_p} \int_{\Gamma_j} \left(\phi_h(j)\frac{\partial G(P,q)}{\partial n_j} - G(P,q)\frac{\partial \phi_h(j)}{\partial n_j} \right) ds \ . \tag{3.17}$$

Here N_p is the number of panels and Γ_j is the part of the boundary the panel j represents. Either the potential or the slope is known on each panel, depending on the boundary condition. The panels are assumed to be flat with a constant charge, such that each integral is simply a line integral in a two-dimensional domain. The point P is set to the midpoint of a panel, allowing construction of a linear system of equations. P_i is at the center of the i^{th} panel, and the sum of the line integrals over N_p panels is equal to either the potential or slope at P_i, depending on which is known.

For example, consider a system of two panels, where the first is Dirichlet and the second is Neumann. This produces the following equations:

$$\begin{aligned} \phi_{h,1} &= \phi_{h,1}\int_{S_1} \frac{\partial G(\vec{x}_1,\vec{x})}{\partial n_1} dS + \frac{1}{2}\phi_{h,1} + \frac{\partial \phi_{h,1}}{\partial n_1}\int_{S_1} G(\vec{x}_1,\vec{x}) dS \\ &+ \phi_{h,2}\int_{S_2} \frac{\partial G(\vec{x}_1,\vec{x})}{\partial n_2} dS + \frac{\partial \phi_{h,2}}{\partial n_2}\int_{S_2} G(\vec{x}_1,\vec{x}) dS \end{aligned} \tag{3.18}$$

$$\begin{aligned} \frac{\partial \phi_{h,2}}{\partial n_2} &= \phi_{h,1}\int_{S_1} \frac{\partial^2 G(\vec{x}_2,\vec{x})}{\partial n_1 \partial n_2} dS + \frac{\partial \phi_{h,1}}{\partial n_1}\int_{S_1} \frac{\partial G(\vec{x}_2,\vec{x})}{\partial n_2} dS \\ &+ \phi_{h,2}\int_{S_2} \frac{\partial^2 G(\vec{x}_2,\vec{x})}{\partial n_2^2} dS + \frac{\partial \phi_{h,2}}{\partial n_2}\int_{S_2} \frac{\partial G(\vec{x}_2,\vec{x})}{\partial n_2} dS + \frac{1}{2}\frac{\partial \phi_{h,2}}{\partial n_2} \ . \end{aligned} \tag{3.19}$$

The terms $\frac{1}{2}\phi_{h,1}$ and $\frac{1}{2}\frac{\partial\phi_{h,2}}{\partial n_2}$ arise from a panel's self-effect, as noted previously. These equations can be rearranged into a 2×2 linear system:

$$\begin{bmatrix} \int_{S_1} G(\vec{x}_1, \vec{x}) dS & \int_{S_2} \frac{\partial G(\vec{x}_1,\vec{x})}{\partial n_2} dS \\ \int_{S_1} \frac{\partial G(\vec{x}_2,\vec{x})}{\partial n_2} dS & \int_{S_2} \frac{\partial^2 G(\vec{x}_2,\vec{x})}{\partial n_2^2} dS \end{bmatrix} \begin{bmatrix} \frac{\partial\phi_{h,1}}{\partial n_1} \\ \phi_{h,2} \end{bmatrix} = \begin{bmatrix} \phi_{h,1}\left(\frac{1}{2} - \int_{S_1} \frac{\partial G(\vec{x}_1,\vec{x})}{\partial n_1} dS\right) \\ \frac{\partial\phi_{h,2}}{\partial n_2}\left(\frac{1}{2} - \int_{S_2} \frac{\partial G(\vec{x}_2,\vec{x})}{\partial n_2} dS\right) \end{bmatrix} . \tag{3.20}$$

Here $\frac{\partial\phi_{h,1}}{\partial n_1}$ and $\phi_{h,2}$ are the unknowns. After solving this system, both the potential and electric field are known at each panel center. However, it is not always necessary for these quantities to be known on each panel, and this method requires that two integrals are computed for every panel interaction, so the total number of integrals evaluated is $2N_p^2$.

A different method for approaching the problem is to replace $\phi_{h,i}$ and $\partial\phi_h/\partial n_i$ with constants σ_i and μ_i on the right-hand side of Eqns. 3.18 and 3.19. For Dirichlet panels μ_i is set to zero and for Neumann panels σ_i is set to zero. This gives the following system:

$$\begin{bmatrix} \int_{S_1} \frac{\partial G(\vec{x}_1,\vec{x})}{\partial n_1} dS + \frac{1}{2} & \int_{S_2} G(\vec{x}_1, \vec{x}) dS \\ \int_{S_1} \frac{\partial^2 G(\vec{x}_2,\vec{x})}{\partial n_1 \partial n_2} dS & \int_{S_2} \frac{\partial G(\vec{x}_2,\vec{x})}{\partial n_2} dS + \frac{1}{2} \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \mu_2 \end{bmatrix} = \begin{bmatrix} \phi_{h,1} \\ \frac{\partial\phi_{h,2}}{\partial n_2} \end{bmatrix} . \tag{3.21}$$

The integral over the second-order derivative in the above is singular if $\vec{x}$ is equal to $\vec{x}_2$. However, this will not occur as the panel self-effect is not computed using this term. Once the values of σ_i and μ_i are found, they may then be used to determine the potential anywhere in the domain due to the boundaries:

$$\phi_h(\vec{x}_i) = \sum_{k=1}^{N_{p,N}} \mu_k \int_{S_k} G(\vec{x}_i, \vec{x}_j) dS + \sum_{k=N_{p,N}+1}^{N_p} \sigma_k \int_{S_k} \frac{\partial G}{\partial n_k} dS . \tag{3.22}$$

$N_{p,N}$ is the number of Neumann panels. This is the method currently used in the treecode simulation to determine the panel-induced potential in the domain.

The linear system may be quite large for a given domain, especially since smaller panels provide more accurate solutions. However, for a domain with static boundaries, the matrix of panel effects does not change. As such, it may be inverted completely as an initial step, even though this is very computationally intensive ($O(N_p^3)$). At each iteration, the inverted matrix simply needs to be multiplied by the updated homogenous boundary conditions vector to obtain the panel source strengths. In the case where either the number of panels is too high or the number of iterations is too low to make direct inversion reasonable, a Generalized Minimal Residuals (GMRES) algorithm is used to solve the system (*Trefethen and Bau* (1997)).

3.4.2 Panel Integration Methods

The integrals in Eqn. 3.22 may be solved analytically in 2-D, but not in an axisymmetric domain. The resulting functions of the integration in 2-D are logarithms or arctangents, and as such are computationally inefficient. It is faster in all cases to perform numerical integration over the panel instead. This is done using a standard Gaussian Quadrature method with a linear transformation of the panel coordinates (*Atkinson* (1993)). The result is that each panel has a set of integration points distributed at the Gaussian nodes, with the charge of the panel multiplied by the corresponding weight for each node. These integration points do not need to be evaluated as part of the integral, but can instead be evaluated as individual point charges. In other words, the contribution from each integration point may be computed in any order, so long as all points are eventually accounted for.

For Neumann panels, the integration points are exactly the same as a point charge, while Dirichlet panels require an initial normal derivative of the Green's function. This is split into the component directions, such that there is a set of points corresponding to the x or z-derivative, and another set of points for the y or r-derivative. Equation 3.22 can then be expressed as

$$\phi_h(\vec{x_i}) = \sum_{j=1}^{N_{BP,N}} \mu_j w_j G(\vec{x_i}, \vec{x_j}) + \sum_{j=1}^{N_{BP,D}} \sigma_j w_j n_{x,j} \frac{\partial G}{\partial x} + \sum_{j=1}^{N_{BP,D}} \sigma_j w_j n_{y,j} \frac{\partial G}{\partial y} \ . \tag{3.23}$$

Here $N_{BP,N}$ is the number of Neumann boundary points, $N_{BP,D}$ is the number of Dirichlet boundary points, w_j is the point's Gaussian Quadrature weight, $n_{x,j}$ and $n_{y,j}$ are the panel normal components, and σ_j, and μ_j are the panel strengths for the panel that the point belongs to. The treecode may then be applied to these panel integration points to reduce the computation cost involved in the same way as with point charges in the domain. A separate tree is constructed for each type of point—particles, Neumann panel points, Dirichlet-x (or z) panel points and Dirichlet-y (or r) panel points. Each tree is then handled in a slightly different way— particles must make sure not to include themselves in the field calculation, but Neumann panels do not require this, as the fields are not computed exactly on the Neumann points. The Dirichlet points require that the Taylor expansion starts with one derivative of the Green's function in either the x or y-direction.

One note to the Boundary Integral Method is that it too uses the Green's function for evaluation of forces and potentials. This means that when a particle is close to the boundary, the Green's function computation is nearly singular. If the boundary integrals are exact, then this is not a problem. However, if a numerical integration scheme is used, then the exact nature of the singularity is not captured. This causes a large error when computing a force or potential near the boundary. This problem and methods for dealing with it are addressed in further detail in Section 5.3.1.

3.5 Two-Dimensional Treecode Derivation

The treecode is relatively easy to implement in a two-dimensional domain. All integrals can be evaluated analytically if desired, and a recursion relation may be obtained for derivatives of the Green's function.

First define

$$r_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}\ . \tag{3.24}$$

The 2-Dimensional Green's function is then

$$G(x_i, y_i, x_j, y_j) = -\frac{1}{2\pi}\log(r_{i,j})\ . \tag{3.25}$$

Taylor expanding the potential field equation (Eqn. 3.2) about a cluster center x_c, y_c gives

$$\phi_c(x_i, y_i) = \sum_{j=1}^{N}\left[\frac{q_j}{\varepsilon_0}\sum_{k=0}^{\infty}\sum_{m=0}^{k}\left[\frac{(x_j - x_c)^m (y_j - y_c)^{(k-m)}}{m!(k-m)!}\frac{\partial^k G(r_{i,c})}{\partial x^m \partial y^{(k-m)}}\right]\right]\ . \tag{3.26}$$

This is the potential field at a point x_i, y_i due to the cluster c. The derivatives of $G(r_{i,c})$ can be taken out of the summation over j, leaving a derivative term and a moments term:

$$\phi_c(x_i, y_i) = \sum_{k=0}^{\infty}\sum_{m=0}^{k}\left[\frac{\partial^k G(r_{i,c})}{\partial x^m \partial y^{(k-m)}} M_c^{k,m}\right]\ . \tag{3.27}$$

3.5.1 2-D Recursion Relation

The derivatives in Eqn. 3.27 can be computed using a recursion relation. Leibniz's Rule for differentiating the product of two functions is

$$\frac{\partial^{(k-1)}}{\partial x^{(k-1)}}(fg) = f\frac{\partial^{(k-1)}g}{\partial x^{(k-1)}} + \binom{k-1}{1}\frac{\partial f}{\partial x}\frac{\partial^{(k-2)}g}{\partial x^{(k-2)}} + \binom{k-1}{2}\frac{\partial^2 f}{\partial x^2}\frac{\partial^{(k-3)}g}{\partial x^{(k-3)}}$$
$$+\cdots+\binom{k-1}{s}\frac{\partial^s f}{\partial x^s}\frac{\partial^{(k-s-1)}g}{\partial x^{(k-s-1)}} + \cdots + \frac{\partial^{(k-1)}f}{\partial x^{(k-1)}}g\ . \tag{3.28}$$

Differentiating the Green's function with respect to x_i and rearranging gives

$$r_{i,c}^2 \frac{\partial G}{\partial x_i} = \frac{x_i - x_c}{2\pi} \; . \tag{3.29}$$

If $f = r_{i,c}^2$ and $g = \partial G / \partial x_i$, then there are only three non-zero terms in Eqn. 3.28, since $r_{i,c}^2$ has a zero third derivative with respect to x_i. Also, the second derivative of fg will be zero, as Eqn. 3.29 indicates. Eqn. 3.28 will then be equal to zero for $k > 2$. So, for $k > 2$, Eqn. 3.28 becomes

$$r_{i,c}^2 \frac{\partial^k G}{\partial x^k} + 2(k-1)(x_i - x_c)\frac{\partial^{(k-1)} G}{\partial x^{(k-1)}} + (k-1)(k-2)\frac{\partial^{(k-2)} G}{\partial x^{(k-2)}} = 0 \; . \tag{3.30}$$

This can be rearranged to give the k^{th} derivative as a function of the $k-1$ and $k-2$ derivatives. A similar equation can be written for derivatives in y. However, when derivatives in both x and y are needed, Leibniz's Rule must be applied to each of the three terms in Eqn. 3.30. This gives a general equation for all derivatives

$$\begin{aligned} \frac{\partial^{(k+m)} G}{\partial x^k \partial y^m} &= \frac{-1}{r_{i,c}^2}\left[2n(y_1 - y_c)\frac{\partial^{(k+m-1)} G}{\partial x^k \partial y^{(m-1)}} \right. \\ &+ 2(k-1)(x_1 - x_c)\frac{\partial^{(k+m-1)} G}{\partial x^{k-1} \partial y^m} + m(m-1)\frac{\partial^{(k+m-2)} G}{\partial x^k \partial y^{(m-2)}} \\ &+ \left. 2m(k-1)\frac{\partial^{(k+m-2)} G}{\partial x^{(k-1)} \partial y^{(m-1)}} + (k-1)(k-2)\frac{\partial^{(k+m-2)} G}{\partial x^{(k-2)} \partial y^m} \right] . \end{aligned} \tag{3.31}$$

Here $k > 2$ and $m > 2$. Only the first two orders of the Taylor expansion derivatives need to be calculated directly.

3.6 Axisymmetric Treecode Derivation

The axisymmetric domain adds a great deal of complexity to the treecode. The Green's function involves an elliptic integral, which cannot be evaluated analytically. Also, a full recursion relation for the derivatives of the Green's function is not known.

3.6.1 The Axisymmetric Free-Space Green's Function

The axisymmetric Green's function for free space can be derived by integrating the 3-D Green's function in the θ direction in cylindrical coordinates. The 3-D Green's function is

$$G(\vec{x}_0, \vec{x}_s) = \frac{1}{4\pi}\left[(x_0 - x_s)^2 + (y_0 - y_s)^2 + (z_0 - z_s)^2\right]^{-\frac{1}{2}} \ . \tag{3.32}$$

The point $\vec{x}_0$ is the point at which the potential is computed, and $\vec{x}_s$ is the source point providing the potential. First, convert to cylindrical coordinates using the following:

$$x_i = r_i \cos(\theta_i) \tag{3.33}$$

$$y_i = r_i \sin(\theta_i) \ . \tag{3.34}$$

Substituting into Eqn. 3.32 gives

$$\begin{aligned} G(\vec{x}_0, \vec{x}_s) = \frac{1}{4\pi}\Big[& r_0^2\cos^2(\theta_0) + r_s^2\cos^2(\theta_s) - 2r_0 r_s\cos(\theta_0)\cos(\theta_s) \\ & + r_0^2\sin^2(\theta_0) + r_s^2\sin^2(\theta_s) - 2r_0 r_s\sin(\theta_0)\sin(\theta_s) \\ & + (z_0 - z_s)^2\Big]^{-\frac{1}{2}} \ . \end{aligned} \tag{3.35}$$

This can be simplified by using these identities:

$$\sin^2(\theta) + \cos^2(\theta) = 1 \tag{3.36}$$

$$\cos(\theta_1)\cos(\theta_2) + \sin(\theta_1)\sin(\theta_2) = \cos(\theta_1 - \theta_2) \ . \tag{3.37}$$

Equation 3.35 then becomes

$$G(\vec{x}_0, \vec{x}_s) = \frac{1}{4\pi}\left[r_0^2 + r_s^2 - 2r_0 r_s\cos(\theta_0 - \theta_s) + (z_0 - z_s)^2\right]^{-1/2} \ . \tag{3.38}$$

In an axisymmetric domain, the values of θ_0 and θ_s do not matter, only the difference between them does. Define this difference as

$$\psi = \theta_0 - \theta_s \ . \tag{3.39}$$

The source particle at $\vec{x}_s$ is a point charge in 3-D space, but in axisymmetric space it is a ring charge of constant density. So, Eqn. 3.38 is integrated over ψ in order to obtain the effect of the entire ring charge:

$$G(\vec{x}_0, \vec{x}_s) = \frac{1}{4\pi} \int_0^{2\pi} \left[r_0^2 + r_s^2 - 2r_0 r_s \cos(\psi) + (z_0 - z_s)^2\right]^{-\frac{1}{2}} d\psi \ . \tag{3.40}$$

Define the following:

$$L = (r_0 + r_s)^2 + (z_0 - z_s)^2 \tag{3.41}$$

$$m = \frac{4r_0 r_s}{L} \tag{3.42}$$

$$\psi = 2\theta \ . \tag{3.43}$$

Substituting and changing the integration limit gives

$$G(\vec{x}_0, \vec{x}_s) = \frac{1}{4\pi\sqrt{L}} \int_0^{\pi} \left[2 - m(1 + \cos(2\theta))\right]^{-\frac{1}{2}} d\theta \ . \tag{3.44}$$

Next, the identity $1 + \cos(2\theta) = 2\cos^2(\theta)$ is applied:

$$G(\vec{x}_0, \vec{x}_s) = \frac{1}{2\pi\sqrt{L}} \int_0^{\pi} \left[1 - m\cos^2(\theta)\right]^{-\frac{1}{2}} d\theta \ . \tag{3.45}$$

The integration is from zero to π, so $sin^2(\theta)$ may be substituted for $cos^2(\theta)$ without changing the value of the integral. The integral is also symmetric about $\pi/2$, so multiplying by 2 allows the upper limit to be changed to $\pi/2$. The final equation is then

$$G(\vec{x}_0, \vec{x}_s) = \frac{1}{\pi\sqrt{L}} \int_0^{\frac{\pi}{2}} \frac{d\theta}{\sqrt{1 - m\sin^2(\theta)}} = \frac{K(m)}{\pi\sqrt{L}}, \tag{3.46}$$

where $K(m)$ is the complete elliptic integral of the first kind. Properties of elliptic integrals, and the methods for calculating them can be found in *Abramowitz and Stegun* (1974). In the simulation, this function is evaluated using an arithmetic-geometric mean algorithm written by *Shure and Bonnevier* (2001).

One note to this derivation is that it is possible to define $L = (r_0 - r_s)^2 + (z_0 - z_s)^2$ instead, leading to exactly the same final equation. In this form, the parameter m can go to infinity when $z_0 = z_s$ and r_0 approaches r_s. However, in the form derived above, this does not occur— instead, m will go to one as r_0 approaches r_s. This causes the elliptic integral to go to infinity instead.

3.6.2 Derivatives of the Axisymmetric Green's Function

Although a simple recursion relation for the derivatives of the Green's function may be obtained for both 2-D and 3-D domains, a full recursion is not known for the axisymmetric Green's function. A partial recursion can be found in *Strickland and Amos* (1992), but this still requires calculation of some of the derivatives at each level of the Taylor expansion.

The partial recursion is derived by simply taking the derivative of the axisymmetric Laplace's equation with respect to the radius of either the source or the reference point. This results in two equations:

$$\begin{aligned}
\frac{\partial^{k+m+n}G}{\partial z^k \partial r_0^m \partial r_s^n} &= -\frac{\partial^{k+m+n}G}{\partial z^{k+2} \partial r_0^{m-2} \partial r_s^n} \\
&\quad - \sum_{j=0}^{m-2} \frac{(-1)^j}{r_0^{j+1}} \frac{(m-2)!}{(m-2-j)!} \frac{\partial^{k+m+n-j-1}G}{\partial z^k \partial r_0^{m-j-1} \partial r_s^n} \qquad (3.47) \\
\frac{\partial^{k+m+n}G}{\partial z^k \partial r_0^m \partial r_s^n} &= -\frac{\partial^{k+m+n}G}{\partial z^{k+2} \partial r_0^{m} \partial r_s^{n-2}} \\
&\quad - \sum_{j=0}^{n-2} \frac{(-1)^j}{r_s^{j+1}} \frac{(n-2)!}{(n-2-j)!} \frac{\partial^{k+m+n-j-1}G}{\partial z^k \partial r_0^{m} \partial r_s^{n-j-1}} \,. \qquad (3.48)
\end{aligned}$$

Here $z = z_0 - z_s$, so taking the derivative with respect to z_s requires a multiplication by -1. The first equation is only valid if $m \neq 0, 1$, while the second equation is valid for $n \neq 0, 1$.

Four derivatives must be calculated directly at each derivative order $p = k+m+n$:

$$\frac{\partial^p G}{\partial z}, \quad \frac{\partial^p G}{\partial z^{p-1} \partial r_0}, \quad \frac{\partial^p G}{\partial z^{p-1} \partial r_s}, \quad \frac{\partial^p G}{\partial z^{p-2} \partial r_0 \partial r_s} \ . \tag{3.49}$$

These derivatives can become quite complex at higher orders, so they have only been derived up to the fourth order. A full listing of all derivatives may be found in Appendix B. The limit on the highest order Taylor expansion, and the fact that the derivatives are calculated directly both detract from the computational speed of the treecode in the axisymmetric case.

CHAPTER IV

Particle-In-Cell Simulation Results

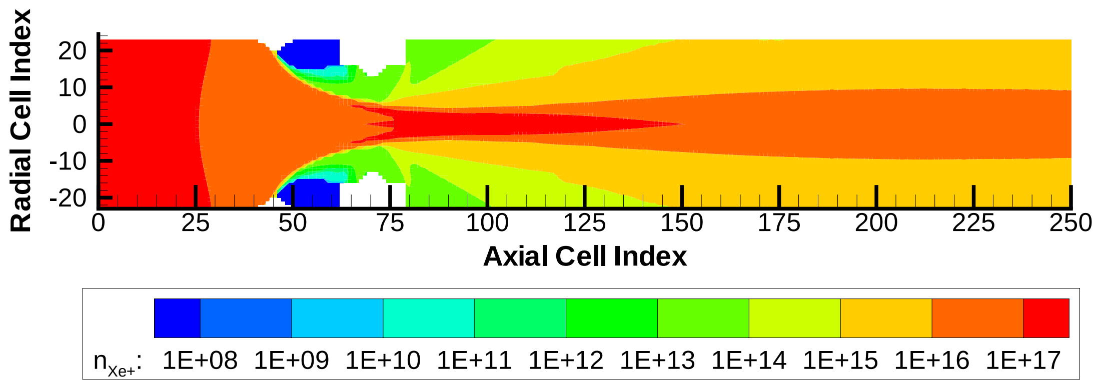
4.1 NEXT Ion Optics Simulation Domain

The simulations presented here model a single aperture in the ion optics. For the NEXT thruster simulation cases, the 3.52 A operating point is modeled, with an 1800 V discharge potential (*Soulas et al.* (2002)). This operating point has a mass flow rate of 5.87 mg/s, the screen grid potential floats at about 1776 V, and the accelerator grid potential is nominally -210 V. The utilization efficiency for the NEXT thruster is held at 0.9, and the ratio of Xe^{++} current to Xe^{+} current has been estimated at about 0.04 based on measurements made for the NSTAR thruster (*Soulas et al.* (2004)). The electron temperature in the discharge chamber is estimated to be 6 eV, the plume electron temperature is estimated at 1 eV. These values are based on measurements made on the NSTAR thruster discharge chamber and plume by *Foster et al.* (2000). More recent measurements of the NEXT thruster give similar numbers for the plasma potential and plume electron temperature (*Kamhawi et al.* (2004)). Also, *Herman and Gallimore* (2004) gives discharge electron temperatures for the NSTAR thruster that support the estimated 6 eV discharge electron temperature. Cusps are simulated in all cases unless noted otherwise. The cusps are first approximated as triangular structures on the aperture wall. If a cell center is inside the triangle, then that cell is assigned as an optics cell.

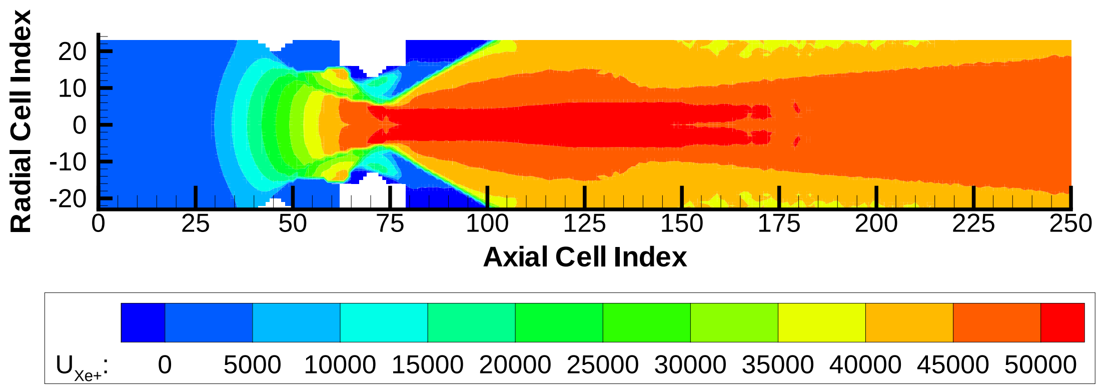
Representative flow field quantities for the centerline aperture of the 3.52 A operating point are plotted in Figures 4.1 and 4.2. The beamlet current in this case is 0.167 mA. Figure 4.1 shows the ion density and velocity in the domain, as well as the potential field. The neutral particle density and velocity, and the CEX production rate are plotted in Figure 4.2. As in all plots of the ion optics domain, the white regions extending into the domain represent the screen and accelerator grids.

The ion population is focused as it travels through the optics in Figure 4.1(a), resulting in a region of high ion density in the center region of the domain. In Figure 4.1(b), the upstream ion injection surface can be seen in the axial velocity contour upstream of the screen grid. The downstream neutralization surface is seen in the plot of potential in Figure 4.1(c) as the last contour in the downstream region.

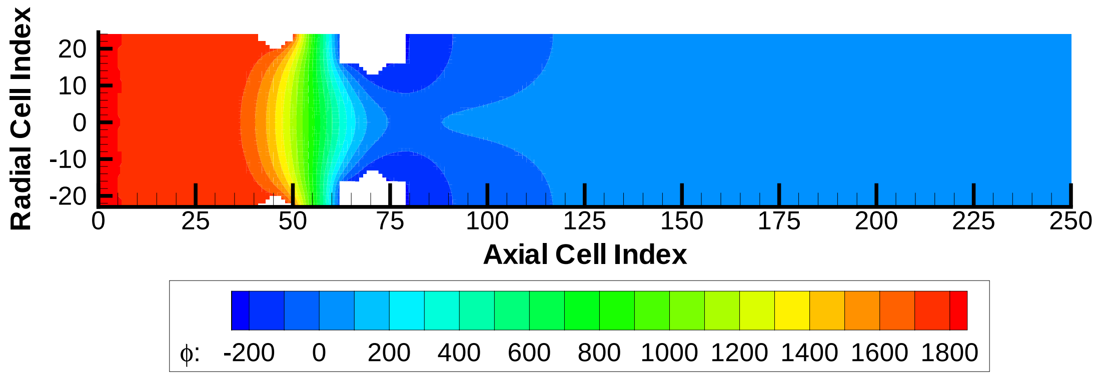
The neutral density shown in Figure 4.2(a) is collimated to an extent by the ion optics, leaving regions of lower density behind the accelerator grid. In the discharge chamber, the neutral density is higher away from the centerline because ions that neutralize on the screen grid are re-emitted as neutral particles, increasing the neutral density in front of the screen grid. The average neutral velocity in the downstream region is uniform and positive, but the average velocity is approximately zero in the discharge chamber, as Figure 4.2(b) shows. Figure 4.2(c) plots the CEX collision rate in each cell, with the regions of highest rate being in the discharge chamber and along the regions of highest ion density. In the downstream region, the collision rate is approximately constant everywhere, although the lower ion and neutral density behind the optics lowers the rate at the domain edge. Note that, because the rate is plotted for each computational cell rather than per unit volume, the rate will increase with increasing radius if all other quantities are held constant. This accounts for the



(a) Ion Density (m^{-3})



(b) Ion Axial Velocity (m/s)



(c) Potential (V)

Figure 4.1: Flow field quantities at a beamlet current of 0.167 mA. The (a) ion density, (b) ion axial velocity, and (c) potential field are shown. Each plot is mirrored about the domain centerline— only the half of the field for $r \geq 0$ is simulated.

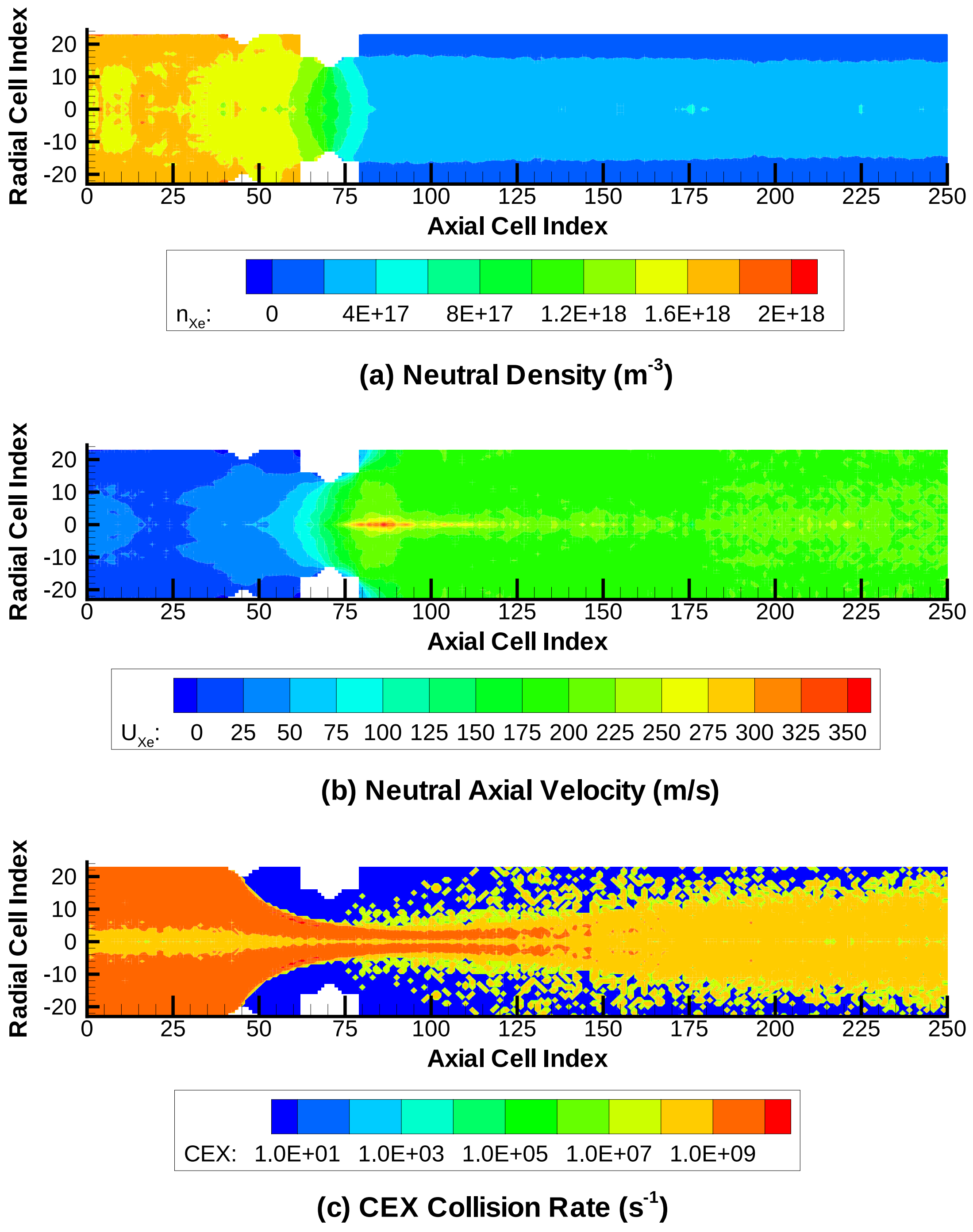


Figure 4.2: Flow field quantities at a beamlet current of 0.167 mA. (a) Neutral density, (b) neutral axial velocity, and (c) charge-exchange collision rate are shown.

region of low collision rate on the centerline of the domain.

4.2 Potential Solver Mesh Refinement Study

The model determines the size of the mesh for Particle-In-Cell based on the Debye length of the plasma in ion optics simulations. This is done because an ion's charge is shielded beyond the Debye length by electrons in a neutral plasma, and thus inter-particle forces are not important. However, this does not consider the other, non-neutral portions of the ion optics domain, such as in the inter-grid region. As such, the mesh size for the neutral plasma is not necessarily appropriate for everywhere in the domain. A mesh refinement study must be performed to determine the correct or sufficient mesh size in order to obtain accurate results at all points in the domain.

Two meshes may be considered in the domain: a particle mesh and a potential solver mesh. The particle mesh tracks the location of particles for sampling and weighting purposes. This mesh sets the boundary locations and also determines the number of particles needed for statistical convergence. The potential solver mesh is used to solve for the potential field everywhere in the domain. This mesh cannot be coarser than the particle mesh, as the boundaries might not be represented correctly otherwise.

The typical particle cell size as determined by the Debye length is taken as the reference cell size. The potential solver mesh can be refined from the reference by halving the cell size in both the radial and axial directions. This produces a mesh twice as fine as the reference. The halving process can be continued until the desired mesh fineness is reached. This same process can be applied to the particle mesh, giving progressively finer particle meshes. Both particle and potential solver meshes may be refined simultaneously, such that a particle mesh twice as fine as the reference

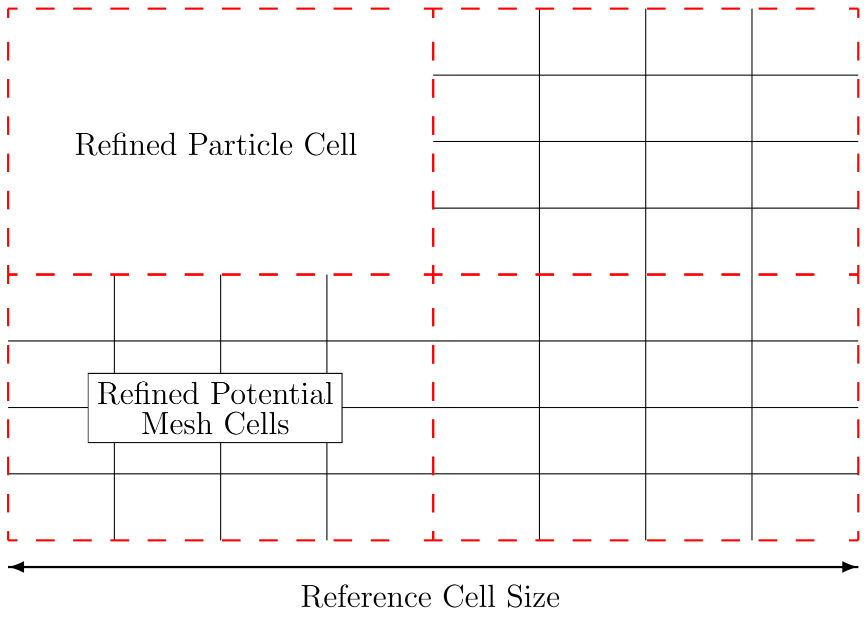


Figure 4.3: Illustration of the mesh refinement levels. The entire box is the reference cell size. The dashed grid indicates a 2x refined particle mesh, and the solid grid indicates a potential mesh refinement of a factor of four on top of the particle mesh's refinement. This gives a total refinement of 8x for the potential solver mesh.

may have a potential mesh that is four times finer than the reference. Each level of refinement is referenced by the factor used to reach it from the reference mesh: the first level of refinement is 2x, the second is 4x, and so on. The total level of refinement is the amount by which the potential solver mesh is refined from the reference mesh. Figure 4.3 illustrates this process and the relevant definitions.

Assuming that the error in the potential solver decreases as the mesh is refined, the total error at any mesh level may be estimated by comparing to the finest mesh obtained. This is done by simply taking the difference between the potential on the coarser mesh and the finest mesh at each coarse mesh point. Figure 4.4 shows this difference as plotted in the domain for a representative case. Typically, the difference in potential over the entire field is not considered— for compactness only

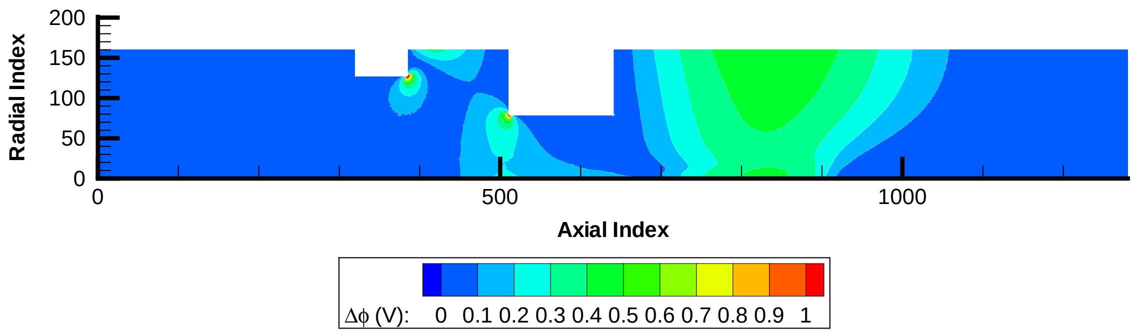


Figure 4.4: The difference in potential between a 4x refined case and a 8x refined case in volts. Coarseness of the mesh causes the error near the ion optics corners and centerline, while residual error in the potential solver causes the error downstream of the optics.

the maximum difference is used.

Two different types of error are seen in Figure 4.4. The first is mesh size error, which occurs primarily at the inside corners of the ion optics, extending to the accelerator grid aperture centerline. This error occurs because the mesh is not fine enough to resolve the gradients in potential that occur in the domain, especially at the optics corners. Mesh size error is reduced by refining the mesh, allowing the gradients to be more accurately resolved. The second type of error is residual error, seen most strongly in the region downstream of the ion optics in Figure 4.4. This error is due to incomplete convergence of the potential field solver. Typically, residual error is highest in the regions with the strongest gradients in electron density, which are the regions immediately upstream and downstream of the ion optics. This type of error is reduced by running the potential solver for a larger number of iterations.

The simulation domain used to perform the mesh refinement study has cell dimensions that are fixed to $5 \cdot 10^{-5}$ m both radially and axially. The ion optics dimensions are set to values that fit the mesh exactly. A beamlet current of 0.104 mA is simulated, and the grid potentials are the same as for the NEXT domain, although the grid geometry is not the same as in NEXT. The screen grid is 0.4 mm thick with a 1.6 mm diameter aperture, while the accelerator grid is 0.8 mm thick and the aperture is 1.0 mm in diameter. The domain diameter is set to 2.0 mm, the distance between the grids is 0.8 mm, the length of the domain upstream of the ion optics is 2.0 mm, and the downstream domain length is 4.0 mm. Cusps are not simulated.

4.2.1 Mesh Size Error

Figure 4.5 shows the maximum difference in potential as the mesh is refined from 1x to 16x. The 16x refined potential mesh case, with 4x refined particle cells, is used as the reference for finding the maximum difference. Each line corresponds to a different level of particle mesh refinement. The maximum difference in potential is caused by mesh size error in these cases: the potential field is converged in each case, giving a residual error that is not significant. The maximum difference in the potential field is quite high at the coarsest mesh— over 20 V. This difference drops by a factor of approximately two at each level of refinement. At a refinement level of 8x, the maximum difference in potential is about 2.5 V, so the error in the 16x case can be estimated to be below 1.5 V. This error will be at the inside corners of the ion optics, but as Figure 4.4 shows, it will decrease sharply further from the optics. Ions rarely approach the inside corners of the optics, so the centerline potential error is much more important. If the actual error at the ion optics corners is estimated to be 2.5 V, then the error on the centerline will be on the order of a volt. This is a sufficient amount of accuracy for most simulations, so a refinement of 8x is assumed in all other results, unless specified otherwise.

As is readily apparent in Figure 4.5, the particle mesh refinement has little effect on the potential solver error. This indicates that the particle mesh does not need

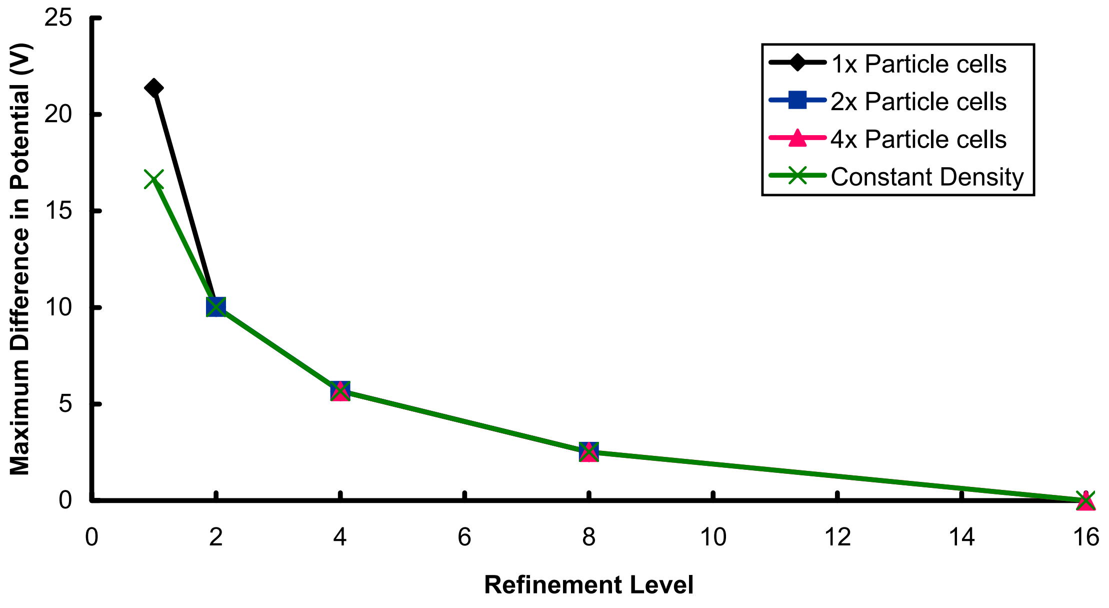


Figure 4.5: Maximum difference in potential as a function of potential solver mesh refinement level. Each line corresponds to a different level of particle mesh refinement, except the green line, which corresponds to a constant ion density field.

to be refined in order to obtain an accurate potential field. However, if the particle numbers are not increased as the potential mesh is refined, then poor statistics may cause errors when the particle charge is weighted to the potential mesh. This problem is avoided by weighting the particle charge to the particle mesh nodes, giving a smooth density field on those nodes. This field is then linearly interpolated onto the potential solver mesh, maintaining a smooth ion contribution.

The green line in Figure 4.5 represents cases run with a constant ion density field. In the other cases, a full simulation is performed to obtain the converged potential field. However, the constant ion density case simply uses an ion density field obtained from a previous converged simulation to determine the potential. This process is much quicker than running a full simulation, as the potential solver simply needs to converge once, and no particle movement is processed. Since the error given by these cases is the same as in the full runs, they may be used in lieu of full simulations.

4.2.2 Residual Error

To reduce the residual error, the solver must be run for a larger number of iterations. Although the ADI method (Section 2.2.4) reduces the residual more per iteration than other methods, the computational cost is also higher per iteration. The computational cost per iteration also goes up by a factor of four each time the mesh is refined by a factor of two. Further adding to the computation cost is the fact that higher refinement levels require more iterations to converge in the ADI method.

To mitigate these problems, the multigrid method (Section 2.2.5) is used instead. The particle mesh is used as the coarsest mesh and the potential solver mesh is the finest mesh. This allows the multigrid method to use n meshes, where 2^{n-1} is the refinement level.

Figure 4.6 plots the computation time as a function of residual for several refinement levels for both multigrid (MG) and ADI. The multigrid cases are represented by solid symbols and lines, while the ADI cases have open symbols and dashed lines. For each data set, the residual error is computed by comparing to the most converged solution for that refinement level. These cases are all performed using a constant ion density field.

As the plot shows, the multigrid method converges much more rapidly than the ADI method. At a refinement of 4x, the time to converge to any given error less than one is more than two orders of magnitude lower for multigrid. The time for multigrid to compute a converged potential at a refinement level of 64x is still lower than the time required for ADI to compute a converged potential at a refinement

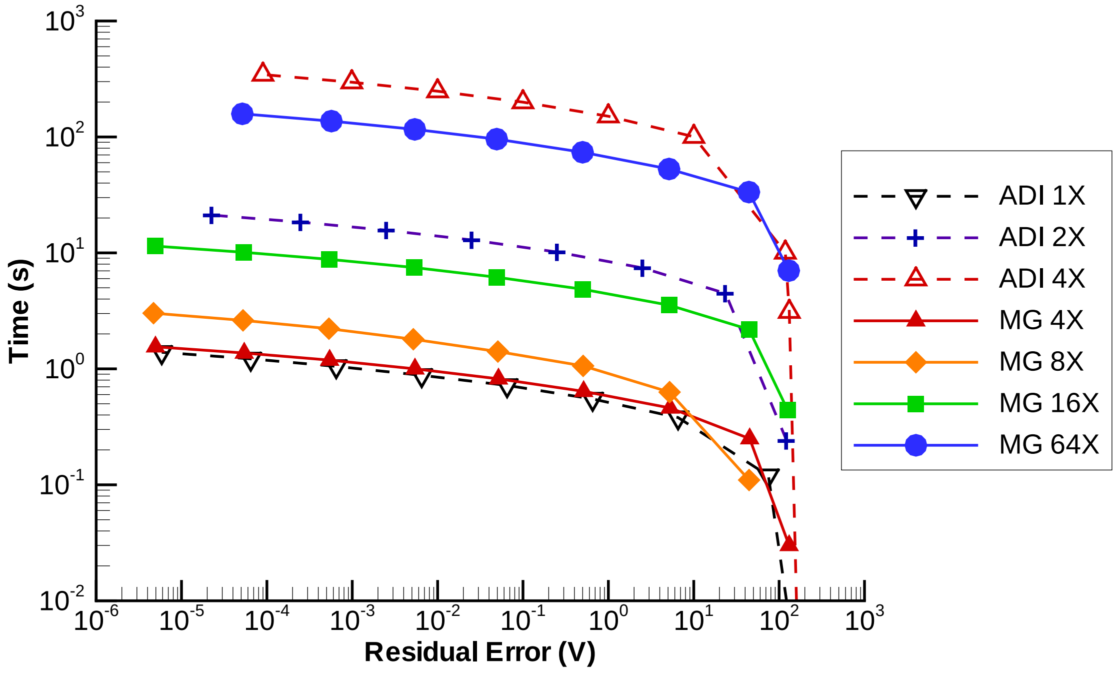


Figure 4.6: Computation time as a function of residual error for ADI and multigrid (MG). The multigrid method requires much less time to converge than the ADI method.

level of 4x. Given these results, the multigrid solver is used in the simulation to solve for the potential field. It outperforms the ADI method by a large margin with no loss in accuracy, allowing simulations with refined potential field meshes to be run in a much shorter time.

The multigrid scheme, as used in ion optics simulations, performs one iteration of the Gauss-Seidel solver on each mesh level except the coarsest mesh. At this mesh, the solver converges to the residual limit. This scheme provides the most efficient convergence. Also, at the beginning of a simulation, the ADI solver is used to solve the potential field on the coarse mesh. The converged solution is interpolated onto the finer mesh used by the multigrid solver. This approach ensures stability of the multigrid solver.

4.2.3 Multigrid Residual Requirement

In order to obtain an accurate potential field, both the mesh size error and the residual error must be low. The 8x refinement level will reduce the mesh size error to acceptable levels, but the residual error must still be kept low. However, an ion optics simulation typically requires iteration numbers on the order of tens of thousands in order to initialize the ion and neutral flow before data is sampled. Thus, it is not necessary for the potential field to be completely converged at each early iteration. The most efficient scheme would be for the potential field to converge as the ion flow does, such that both are converged at the same time, at which point sampling begins.

The multigrid potential solver uses a residual limit parameter to determine the number of cycles to run at each overall simulation iteration. After a solver cycle, the multigrid algorithm checks to see if the residual is lower than the parameter. If

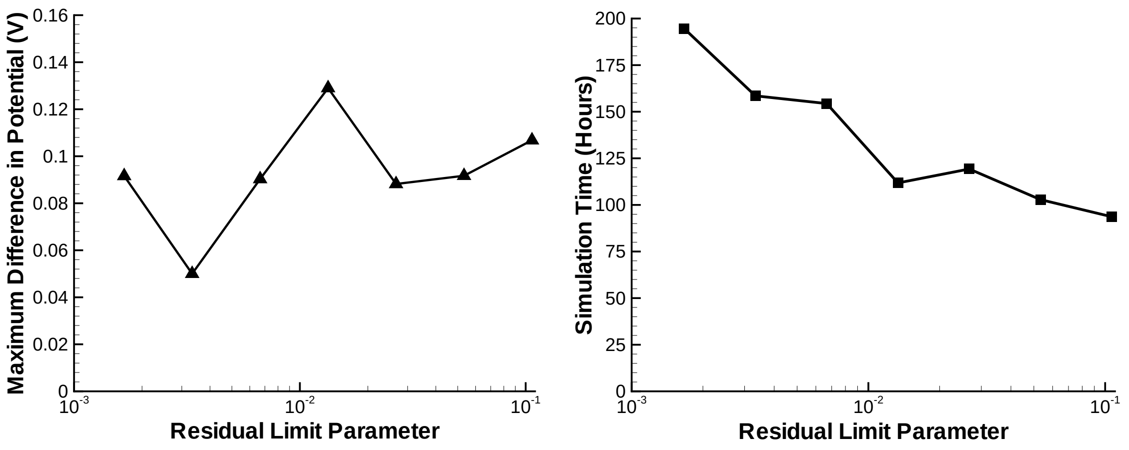


Figure 4.7: The maximum difference in potential and the simulation time as a function of the multigrid residual limit parameter. The potential difference does not change greatly, but the simulation time drops rapidly as the parameter is increased.

it is, then the solver is done, otherwise a new cycle is started. The actual potential field error will change linearly with this parameter, although the scaling changes depending on the domain in question. A typical scaling value may range from 600 to 400, such that the actual error due to the residual is 600 to 400 times the residual calculated by the multigrid solver.

To find the optimal residual limit parameter for the multigrid solver, several ion optics cases are run with varying limits. These cases include a neutral flow and a long domain downstream of the ion optics. This means that approximately 100,000 iterations are required to initialize the flow before sampling begins, after which sampling continues for 100,000 iterations. These iteration numbers are typical for most ion optics simulations presented in this thesis.

In Figure 4.7 the maximum difference in potential and the simulation time are plotted as a function of the multigrid residual limit parameter. The difference in potential is referenced to the potential at a parameter value of $8.3 \cdot 10^{-4}$. The maximum difference in potential is very small for all values of the parameter, and shows no trend either up or down as the parameter increases. However, the simulation time decreases greatly as the parameter increases. This indicates that use of the highest parameter possible will give the most efficient simulation. Other simulation results such as beam current and thrust vary less than 1% as the residual limit parameter changes.

The reason the use of a high parameter works efficiently is that the multigrid solver will always run one cycle for each overall simulation iteration, no matter what the residual is. This ensures that, if the potential field is stable and does not change greatly between iterations, the solver will slowly converge to the correct potential field. However, at the beginning of the simulation, the ion flow is not yet steady, so the potential field changes rapidly as ions reach the ion optics and sheaths form. Using a high residual limit parameter ensures that the solver does not waste too much computation time converging the potential field when it will simply change by a large amount in the next overall iteration. The only requirement is that the potential field is converged enough to be stable. The data above indicate that a multigrid residual limit parameter value of approximately 0.11 gives stable, fast, and accurate results, so this is the value used in all subsequent simulations.

4.3 Upstream Domain Length

As the neutral discharge chamber plasma approaches the screen grid of the ion optics, ions are extracted from the plasma and accelerated through the apertures. A sheath forms at the screen grid surface as electrons impinge and pass current to the grid, decreasing the screen grid potential to about 25 V below the discharge potential. The electron population is repelled by the voltage drop across the grids,

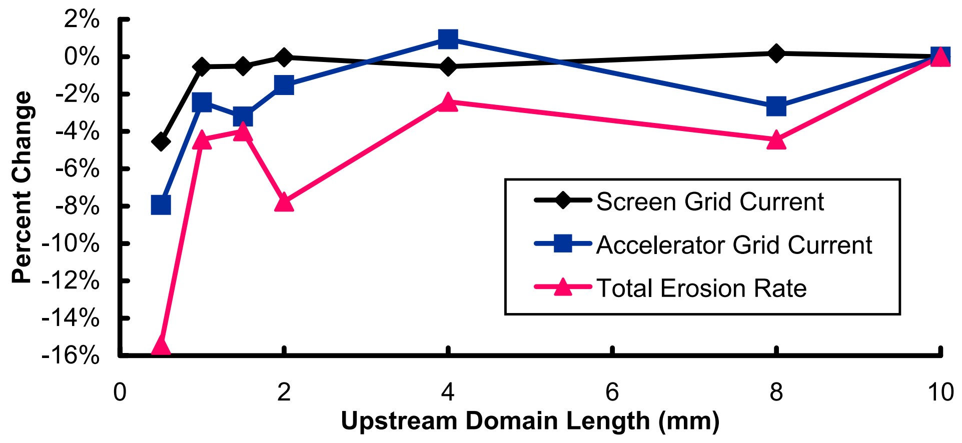


Figure 4.8: Percent change in accelerator grid current, screen grid current, and erosion rate as a function of the upstream domain length. All values are referenced to the 10 mm length case, and the beamlet current in each case is 0.168 mA. A domain length of 1 mm is sufficient for the results to converge to within statistical scatter.

and also by the sheath potential drop. A curved ion extraction surface forms at the point where the electron density begins to decrease and the ions begin to accelerate. In order to accurately simulate ion optics behavior, the model must ensure that these flow structures upstream of the screen grid are able to form correctly. The model simulates electrons self-consistently, so ion extraction will occur correctly so long as enough of the domain is modeled upstream of the screen grid.

In order to determine a sufficient length, several simulations are performed with varying upstream domain lengths. These cases are operated at 0.168 mA, the peak beamlet current condition of the NEXT thruster, with the NEXT ion optics geometry. Neutral flow is included, and the downstream domain length is set to 4 cm.

Figure 4.8 plots the percent change in screen grid current, accelerator grid current, and erosion rate as a function of upstream domain length. The comparison is made against the 10 mm upstream length case. The only aberrant case is at a length of 0.5 mm, where all quantities are much too low. Otherwise, variations are within the

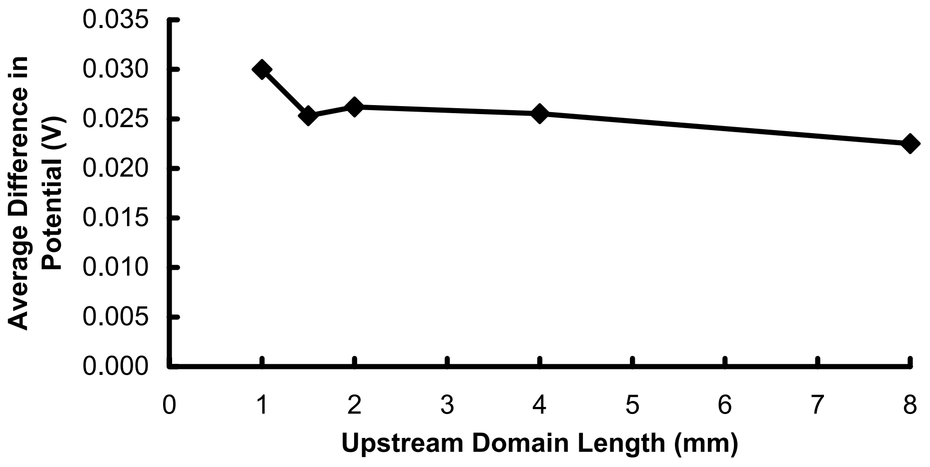


Figure 4.9: The average difference in potential as a function of the upstream domain length at a beamlet current of 0.168 mA. The potential for each length is compared to the 10 mm case potential field.

range of expected statistical scatter.

Figure 4.9 shows the average difference in potential from the 10 mm length case as a function of length. The case at a length of 0.5 mm is not shown, as its difference is an order of magnitude higher than the cases plotted. Very little change is seen in these cases, so for this operating condition a length of 1 mm is most likely sufficient for accurate simulation of ion extraction. To allow for some buffer space, a length of 2 mm is set as the base upstream length.

The required upstream domain length will increase as the beamlet current decreases. This is because a lower density of ions will be affected more strongly by the potential surrounding the grids. Thus, the ion extraction surface will form further upstream of the optics. In Figure 4.10, the maximum difference in potential is shown for a beamlet current of 0.0185 mA as a function of upstream domain length, referenced against the 5 mm length results. Here, the potential difference is dropping steadily as the domain length increases. A length of 2.5 mm drops the potential difference drastically, while a length of 4 mm gives a very small difference in potential.

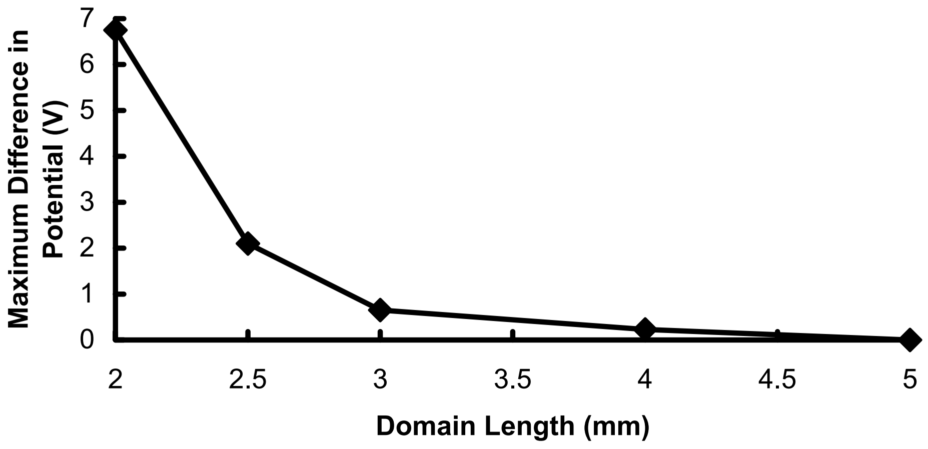


Figure 4.10: The maximum difference in potential as a function of the upstream domain length at a beamlet current of 0.0185 mA. The potential for each length is compared to the 5 mm case potential field.

Also, in performance results such as screen grid current, the 2 mm domain length is the only case that gives anomalous results. To allow for an even smaller beamlet current, and also to reduce the error at the current level tested, an upstream domain length of 4 mm is used for beamlet currents below 0.02 mA. However, the previously set length of 2 mm is sufficient for higher beamlet currents.

4.4 Downstream Domain Length

The domain simulated downstream of the accelerator grid serves two important purposes. First, it allows the formation of a sheath downstream of the accelerator grid. Second, it enables the creation of charge-exchange (CEX) ions, which can be drawn to the accelerator grid, causing erosion. If the domain is too short for the sheath to form, the beam ions will not be decelerated to the vacuum potential and the simulated performance will be too high. A short domain will also not account for all of the CEX ions that are created in the downstream region and then impinge on the accelerator grid, thus the accelerator grid current and erosion rate will not be simulated accurately.

The downstream sheath typically forms within a few millimeters of the downstream face of the accelerator grid. However, the required length to capture all the necessary CEX ions is on the order of centimeters. For this reason, the current and erosion rate of the accelerator grid are the primary indicators of whether the domain is long enough.

CEX ions in the downstream region are created by collisions between neutral particles and ions. In such a collision, one electron is transferred from the neutral to the ion, or two electrons in the case of a doubly-charged ion. This results in a fast moving neutral particle that will quickly travel out of the domain, and a slow moving ion with a random velocity. Beyond the downstream sheath region, there is no significant accelerating force on ions towards either the ion optics or the downstream boundary. So, whether the ion impinges on the optics depends primarily on its initial velocity.

Some CEX ions in the downstream region may have negative axial velocities, but their trajectories are such that they will not impinge on any part of the entire thruster. When these particles reach the upper domain boundary, they are deleted from the simulation. The further the ion is from the ion optics, the smaller the viewing angle for impingement becomes. Thus, it is expected that the current and erosion rate from CEX ions will converge as the domain length increases, as fewer of the CEX ions created will reach the grid. For the purposes of the deletion algorithm, the simulated aperture is always assumed to be in the center of the ion optics. This is in line with the axisymmetric assumption, and also simplifies the calculation greatly.

In Figure 4.11, the accelerator grid current is plotted as a function of downstream domain length. These cases are for a beamlet current of approximately 0.168 mA,

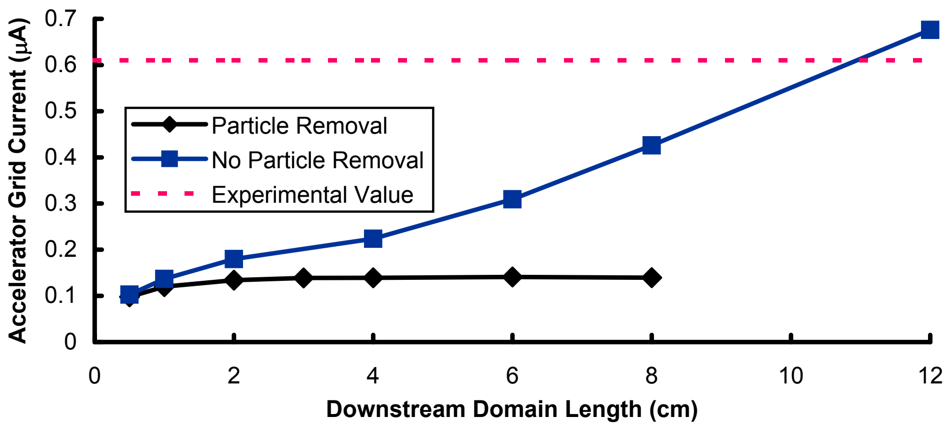


Figure 4.11: The accelerator grid current collected as a function of the downstream domain length at a beamlet current of 0.168 mA. Cases with and without particle deletion at the upper boundary are shown, as is the experimentally measured accelerator grid current for this beamlet current.

corresponding to the peak beamlet current of the 3.52 A operating point of the NEXT ion engine. Cases are shown both with and without particle removal at the upper domain boundary. The experimentally measured accelerator grid current (*Soulas et al.* (2004)) is also shown. As expected, with particle removal at the boundary, the current converges past a length of 4 cm. However, the converged current is more than 4 times lower than the experimentally measured value. Without particle removal, the current continues to increase linearly, reaching the experimental value at a domain length of about 12 cm.

Note that, in order to obtain the measured accelerator grid current used for comparison in Figure 4.11, the experimentally measured current for the entire thruster is scaled to a single aperture on the centerline. This is done by dividing the total accelerator grid current by the number of apertures, and by the beam flatness parameter for the operating point. This assumes that the accelerator grid current scales directly with the beamlet current. For the NEXT thruster, there are approximately $30,000$ apertures, and the beam flatness parameter at the 3.52 A operating point is

0.7.

The reason for the inaccuracy of the simulation of the accelerator grid current is unknown at this time. Although the experimental current is reached without particle deletion, the simulated current in this case would continue to increase linearly with domain length, overestimating the current past a domain length of 12 cm. Thus, particle deletion at the boundary is most likely not causing the problem, although it is possible that the deletion scheme could be tuned to match the experimental data. Also, the experimental current will not be increased due to direct impingement on the ion optics. Although this does occur at the beginning of thruster life, the high-energy ions quickly erode the optics until direct impingement is no longer occurring. The accelerator grid current discrepancy is discussed further in Section 4.5.7.

4.5 Total Thruster Simulation

In *Soulas et al.* (2003), performance data is given for the NEXT ion engine during a 2000 hour wear test. In order to obtain accurate computational results for comparison to this performance data, it may be necessary to simulate several apertures at varying radii on the thruster optics. The results from these simulations can then be integrated to give results for the entire thruster. However, the ion current determines much of the performance of the thruster, so it may be possible to estimate the thruster performance based on a single simulation. The results from that simulation can be scaled appropriately by the beamlet current to give results for all apertures.

4.5.1 Beam Current Density Profile Scaling

A beam current density profile for the NEXT ion engine operating at 3.52 A is given in *Soulas et al.* (2004), and is plotted in Figure 4.12. This profile can be used to determine the radius of a given single-aperture simulation by matching the

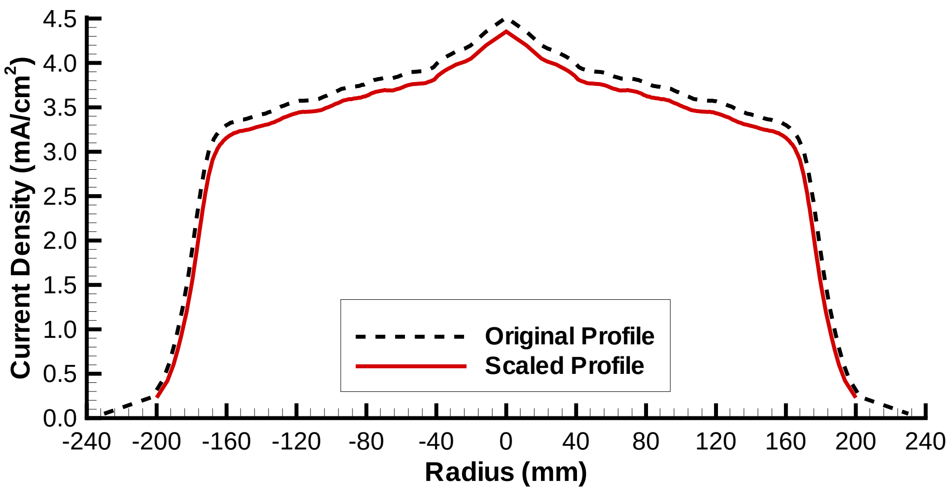


Figure 4.12: Experimentally measured and scaled beam current density profiles for the NEXT ion engine extracting 3.52 A of beam current. The scaled profile ignores the outer-most experimental point, then scales the radius of the remaining points to be within ±200 mm. The radius-scaled profile is then scaled such that the integral of the profile gives the correct beam current.

beamlet current density of the simulation to the profile. It can also be used to integrate thruster performance based on a single simulation. However, the experimental current density profile presents two problems. First, the profile extends beyond the beam extraction area of the thruster. This is resolved by scaling the radius by a factor of approximately 0.988 at each point, such that the second-to-last point on the profile is at ±20 cm. The last point is ignored completely.

Second, when integrated, the profile gives a higher beam current than the thruster is operating at. This is accounted for by scaling the profile down such that integrating it gives the correct beam current. This scaling factor is about 0.965 at the 3.52 A beam current operating point. The profile resulting from these two scaling methods has a flatness parameter of about 0.71, the same as is measured experimentally. This indicates that the scaled profile is reasonably accurate for the purposes of locating simulated apertures on the thruster. The scaled version of the profile is also shown in Figure 4.12.

4.5.2 Multiple-Aperture Simulation Sets

Three sets of multiple-aperture simulations results are given. The primary set of results presented below, Set A, uses Dalgarno momentum-exchange collision cross-sections with isotropic scattering and varying aperture diameters. The ion optics used in the NEXT 2000 hour wear test have decreasing aperture diameters as the thruster radius increases, due to the grid manufacturing process. In order to obtain accurate simulation results, the diameters should be varied in the model as well.

Simulation Set B is performed with Dalgarno MEX cross-sections and constant aperture diameters. In the final set, Set C, results are obtained using the charge-exchange collision cross-sections for momentum exchange collisions, as explained in Section 2.3. This set of simulations also uses varying aperture diameters.

In each set of simulations, the upstream domain length is set to 2 mm for all but the lowest current case in Set A. For this case, the domain length is set to 4 mm. The downstream domain length is 4 cm in each case, with particle deletion enabled. Sputtered grid material is not modeled in any of the simulations.

Figure 4.13 shows the variation of both grid aperture diameters as a function of radius, normalized to the nominal centerline aperture diameter (*Soulas et al.* (2004)). Also shown are the simulated aperture locations and diameters for Set A simulations. The experimental grid diameters were measured using a pin gauge, thus the apertures may be up to 0.001 inch larger in diameter in reality. The model uses a constant radial cell size in these simulations, so there is some error in the simulation's representation of the aperture diameters and cusp structures. Also, when a simulation is run, the exact output beamlet current is not known beforehand, so an estimate of the radius

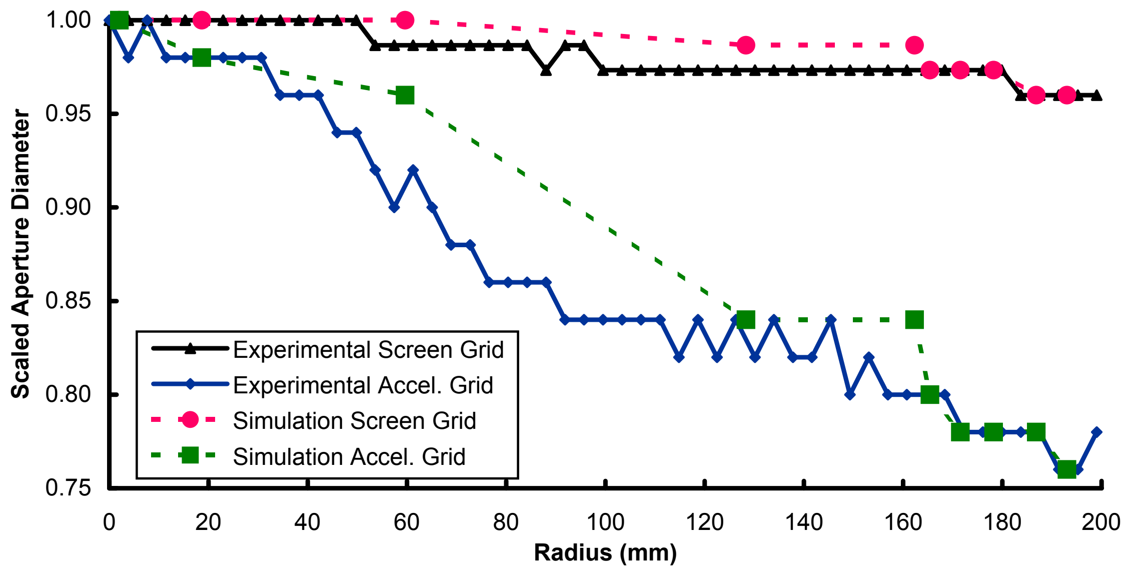


Figure 4.13: Variation of the accelerator and screen grid aperture diameters as a function of radius on the thruster. The simulated aperture diameters from Set A are also shown.

on the thruster is required in order to determine what the aperture diameters will be. However, the aperture diameters change very rapidly for mid-range thruster radii, and the beamlet current changes very slowly. The result is that a small change in beamlet current may produce a large change in aperture diameter. This explains the discrepancies seen in the plot.

4.5.3 Multiple-Aperture Simulation Results

The thrust, beamlet current, mass flow rate, and accelerator grid current are shown in Figure 4.14 as a function of radius for Set A simulations. Also shown is the accelerator grid current for Set B. All quantities are normalized to the centerline value. The thrust varies exactly linearly with the beamlet current, as the beam ions are producing almost all of the thrust. The mass flow rate is nearly linear with the beamlet current, but at low currents the mass flow rate does not drop as quickly as the beamlet current. This is because the neutral flow rate is held constant across all

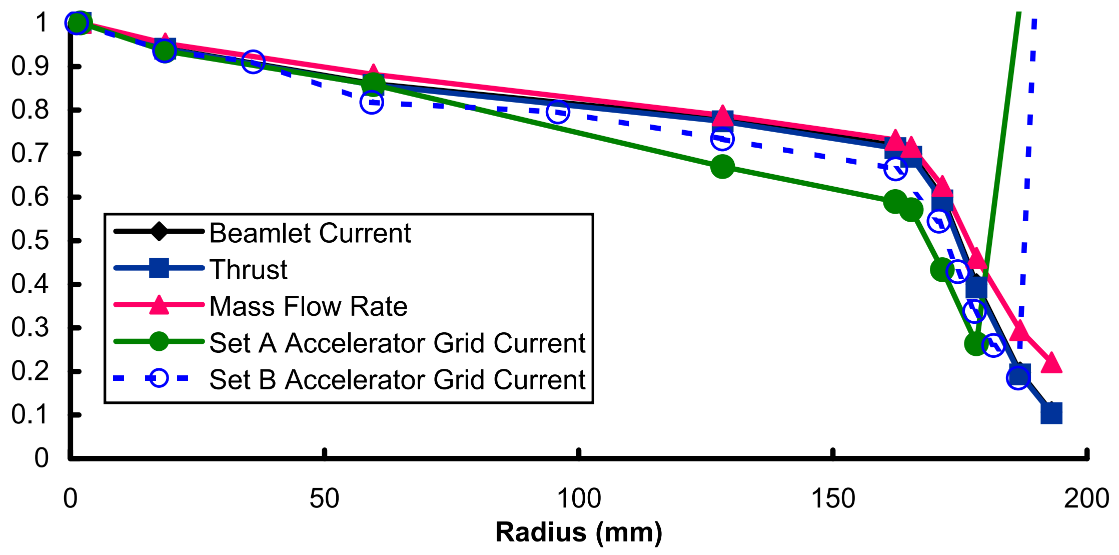


Figure 4.14: Variation of performance quantities as a function of radius for Set A data. The accelerator grid current variation of Set B is shown as well.

apertures, so the total flow rate does not drop as quickly when the ion flow rate is small.

The Set A accelerator grid current decreases more quickly as the radius increases than the beamlet current or thrust do. This is caused by the decrease in screen and accelerator grid aperture diameters. As the diameter decreases, fewer neutrals from the discharge chamber are able to flow through the optics, leading to a lower neutral density downstream of the ion optics. The CEX production rate is a linear function of both ion and neutral density, so as the neutral density decreases, the accelerator grid current will decrease as well. Also, the two simulated apertures at the highest radii have much larger accelerator grid currents than the other apertures. The low beamlet current in both cases induces direct impingement of beam ions on the accelerator grid barrel. This direct impingement will in reality quickly erode the aperture wall until the beamlet no longer impinges directly.

The Set B accelerator grid current varies much more closely with the beamlet current, as the neutral flow rate is not being restricted by smaller apertures. Also, because the outer aperture diameters are not as small, direct impingement does not occur until the lowest beamlet current point.

4.5.4 Integration Procedure

The multiple-aperture simulation results are integrated to obtain total thruster performance results. Each relevant quantity obtained for individual apertures is divided by the simulation area in order to obtain a density value instead. The density value is then assumed to vary linearly between the simulated apertures. The resulting linear approximating function is integrated between the points and over the thruster surface area to give the contribution between the points. The individual contributions are then summed over all points to obtain the total thruster performance quantities. In equation form, the process is as follows for a quantity Q:

$$\begin{aligned} q_i &= Q_i/A_{sim} && (4.1)\\ a_i &= \frac{q_i - q_{i+1}}{r_i - r_{i+1}} && (4.2)\\ b_i &= -a_i r_i + q_i && (4.3)\\ Q_{Total} &= \pi \sum_{i=1}^{N-1} \frac{2a_i}{3}\left(r_{i+1}^3 - r_i^3\right) + b_i\left(r_{i+1}^2 - r_i^2\right) \ . && (4.4) \end{aligned}$$

Here the a_i and b_i are the slopes and intercepts of the linear approximation functions, respectively, A_{sim} is the simulated aperture domain area, and N is the number of simulated points.

To approximate the total thruster performance using a single aperture, the quantities of interest are scaled according to the beam current density profile in order to obtain values at all points. The scaled values are then integrated in the same way as above in order to obtain total thruster performance results. This procedure will give the best results when the quantity of interest varies linearly with the beamlet

Table 4.1: Comparison of performance quantities for multiple aperture simulations and single aperture integration against experimental data from the NEXT ion engine 2000 hour wear test. Multiple aperture integration results are given for both Set A and Set B simulation results.

	Experimental Data	Set A Multiple	Set A Single	Set B Multiple
Mass Flow Rate (mg/s)	5.87	6.01	6.24	6.02
Thrust (mN)	237	239	241	238
Specific Impulse (s)	4117	4055	4001	4030
Beam Current (A)	3.52	3.51	3.52	3.50
Screen Grid Current (A)	0.44	0.465	0.462	0.441
Accelerator Grid Current (mA)	12.5	4.77	3.88	3.78
Erosion Rate (mg/hr)	—	26.46	5.64	19.37

current. The centerline beamlet current simulation is used in these cases as the base, but any simulation can be used. The values from the simulation are simply scaled up or down depending on the simulated aperture's position on the current density profile.

4.5.5 Total Thruster Performance Results

In Table 4.1, integrated performance results from the Set A and Set B multiple-aperture simulations are compared to experimental data. Also shown are the single-aperture simulation performance results for Set A. As the table shows, the multiple aperture integration gives excellent comparison to the experimental results in most cases. The thrust is nearly exact, the mass flow rate is only slightly high, and the specific impulse is also very close to the experimental value. The mass flow rate is high due to a slightly high neutral flow rate imposed in the simulations. Note that the experimental thrust is actually a calculated value, as the thrust is not measured directly.

The screen grid current is only higher than the experimental value by a small amount. The difference seen here may be caused by the inaccuracy of the simulation of the cusp structure on the screen grid aperture wall. Also, the accelerator grid current is about 3 times smaller than the experimental data. The beam current should be very close to the experimental value, as the experimental current density profile is used to determine the aperture radii. Thus, error in the beam current integration will indicate the integration error for all quantities. Also, the beam current for the single-simulation case should be exact, as the beam current density curve is simply integrated to obtain the value.

The single-aperture integration results do not compare as well as the multiple aperture results. This is expected, especially given that quantities such as mass flow rate and accelerator grid current do not vary linearly with the beamlet current. However, there is still excellent prediction of thrust and screen grid current. There is some difference from the multiple-aperture results in accelerator grid current and erosion rate, much of which is due to the direct impingement seen in the low current apertures.

The Set B results compare very well to the experimental data as well, even though the aperture diameters are not varied. The screen grid current is almost exact, and the other performance quantities are about the same as seen in Set A. The accelerator grid current is even more deficient however, due to less direct impingement at the outer apertures.

4.5.6 Screen Grid Current Correction

The simulated screen grid currents listed in Table 4.1 are corrected values. The current actually simulated by the model is approximately 0.13 A in each case. However, the axisymmetry of the simulation does not account for the hexagonal nature of the domain. Figure 4.15 shows this difference by illustrating the simulated domain

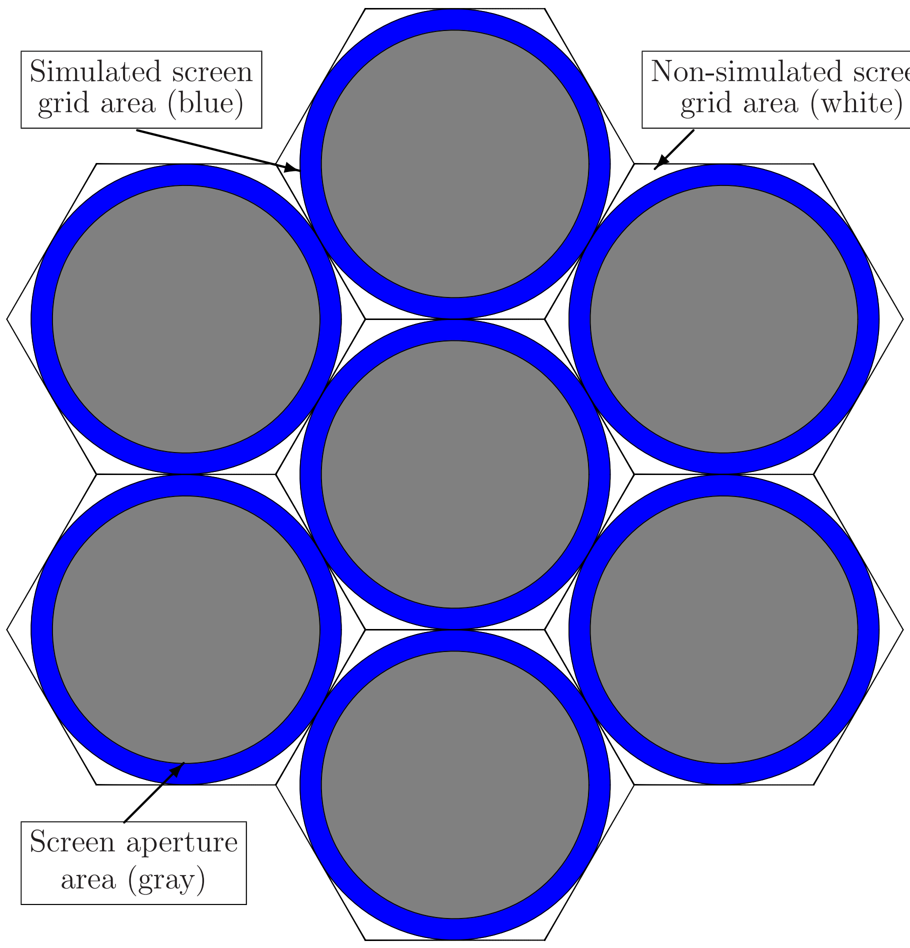

Simulated screen grid area (blue)
Non-simulated screen grid area (white)
Screen aperture area (gray)

Figure 4.15: Illustration of the difference between the simulated screen grid area versus the actual screen grid surface area.

area versus the actual thruster area of the screen grid.

The screen grid current collected in the white areas in the plot is not accounted for by the simulation. Given the reflective upper boundary in the simulation, it can be assumed that all non-simulated ions must impinge on the screen grid. In other words, the simulation boundary is reflective on both sides, and the simulation bounds all sides of the non-simulated area, so anything inside that area cannot leave it.

The current collected in the non-simulated area around a single aperture will be equal to $n_i v_{Bohm} e A$, where n_i is the discharge ion density, v_{Bohm} is the Bohm ion velocity, and A is the area of the six non-simulated corners. This area is equal to approximately $0.08063L_c^2$, where L_c is the distance between aperture centerlines. Adding this current to the simulated screen grid current gives the corrected screen grid current.

The correction to the current is about twice as large as the simulated amount of current. In Table 4.1, approximately 30% of the current is simulated for each set of simulated results, while the corrected current provides the remaining 70%.

4.5.7 Accelerator Grid Current Discussion

The area of poorest comparison between simulation and experiment, for both multiple and single aperture integration, is in the accelerator grid current. In both cases, the simulated accelerator grid current is low by a factor of 3. The deficiency in current should be seen in the erosion rate as well, as both depend on CEX ions. If the simulated erosion rate is three times too low as well, then the simulation is clearly operating incorrectly.

Figure 4.16 shows the increase in accelerator grid aperture diameters following 2000 hours of thruster operation. Experimentally measured pin gauge results and Set A simulation results are plotted. Also shown are the aperture diameters assuming an erosion rate three times higher than is simulated, and the results for simulations from Set C. The erosion in the Set A simulations compares well to the experimental data, as it is generally higher than the measured value by approximately 0.001 inches or less. There is more error for apertures near the thruster edge because these apertures experience direct impingement due to beamlet cross-over, as well as non-circular erosion due to misalignment of the screen and accelerator grid apertures. The axisymmetric simulation is unable to accurately model this type of erosion.

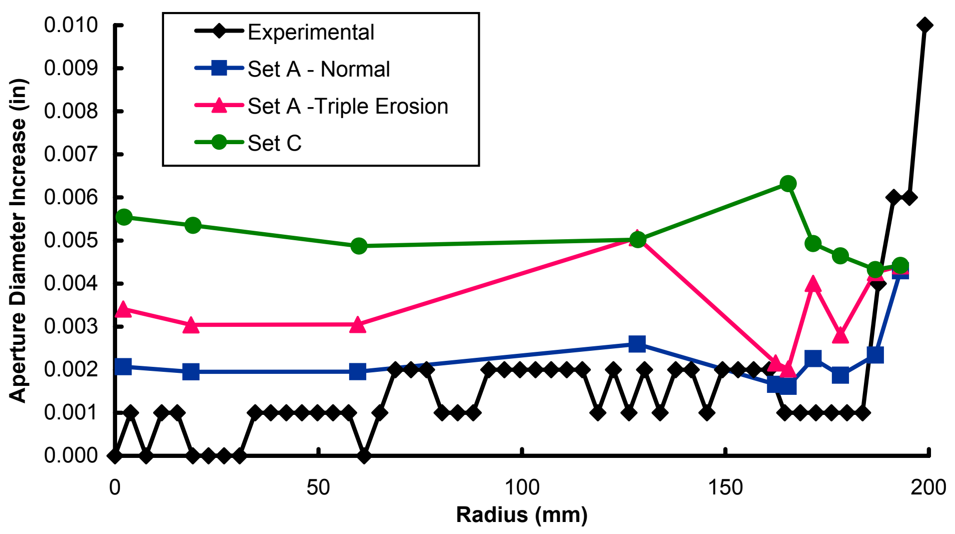


Figure 4.16: Aperture diameter increase due to erosion after 2000 hours of thruster operation. The pin-gauge measured experimental data and the Set A results are shown. Results for Set A with tripled erosion rates and Set C results are plotted as well. Cross-over of the beamlet in the low-current apertures at high radii causes the large amount of erosion seen in these apertures.

The resulting aperture diameters from the Set C simulations are much larger than seen experimentally, as Figure 4.16 shows. This is a result of higher erosion rates in the Set C simulations, caused by the increased amount of momentum-exchange collisions. As this behavior is clearly abnormal, the momentum exchange collision modeling in the simulation is most likely not being performed correctly. Until the source of the problem can be found, the Dalgarno MEX collision cross-sections will be used.

The normal Set A erosion matches the experimental data well, while the tripled erosion rate data is much higher than the experimentally measured values. This indicates that the barrel erosion rate is approximately correct, and thus the current collected on the barrel can be assumed to be correct as well. If the simulation is at fault for the deficient accelerator grid current, this current should be collected on the downstream face of the grid.

Consider the centerline aperture only. The experimental current collected from this aperture should be approximately 0.61 μA. The simulated barrel current for this aperture is 0.081 μA, leaving 0.53 μA of current to be collected on the downstream face. Assuming that all ions impact the grid surface at 230 eV and normal incidence, this amount of current will erode approximately 1.4 mg of grid material over 2000 hours. An estimate of the eroded mass seen in the 2000 hour wear test of the NEXT thruster may be made based on profilometer measurements given in *Kamhawi et al.* (2004). This estimate gives 0.47 mg of eroded grid material around a single aperture, one third the value calculated based on the measured current. The simulation gives 0.25 mg of eroded material after 2000 hours of erosion, or half the estimated value.

The calculated erosion given above, and the simulated erosion on the downstream face, do not take into account reduced erosion due to the pit and groove structures. As these form, eroded material is more likely to recombine on the wall of the pit or groove because the viewing angle of the sputtered material with respect to the grid walls is increased. Also, the angle of incidence of impacting ions will decrease as the walls of the pit or groove become steeper. However, it is not likely that this accounts for three times less erosion than the calculated amount for the 2000 hour wear test. The groove and pit structures are not deep after this amount of time, so any effect which might reduce erosion would be small. The simulation underestimates the amount of erosion somewhat, but there is still a large discrepancy between the calculated erosion and the measured erosion. It appears that some portion of the measured accelerator grid current originates from a location other than around the ion optics apertures.

One other possibility is that the simulated barrel current is in fact incorrect, rendering the above analysis void. The use of higher momentum-exchange collision cross-sections increases the erosion by a large amount, so it may be possible that any momentum-exchange collisions increase barrel erosion unrealistically. If this is the case, then the simulation might be able to model the correct amount of barrel erosion, while the current is lower than should be collected in reality. The momentum-exchange collision model must be examined in future work in order to determine if this is occurring.

A 3-dimensional simulation of the downstream accelerator grid face erosion is also required for further study, as the present model cannot reproduce the pit and groove structures. If the downstream face erosion in a 3-D simulation matches the experimental data, and the accelerator grid current is still deficient, then the experimental current is being collected at a point that is not intended to be simulated. 3-D simulations of downstream face erosion have been performed by *Farnell et al.* (2003) and *Wang et al.* (2003), but it is unknown if the accelerator grid current was deficient in these cases.

4.6 Electron Backstreaming Study

One of the primary failure modes of an ion thruster is loss of performance due to electron backstreaming. This occurs when the accelerator grid aperture diameter increases due to erosion, such that there is no longer a retarding potential keeping plume electrons from being accelerated into the discharge chamber. This form of engine failure may be mitigated by increasing the potential on the accelerator grid; however, this also has the effect of accelerating the erosion, as impacting ions will have a higher energy.

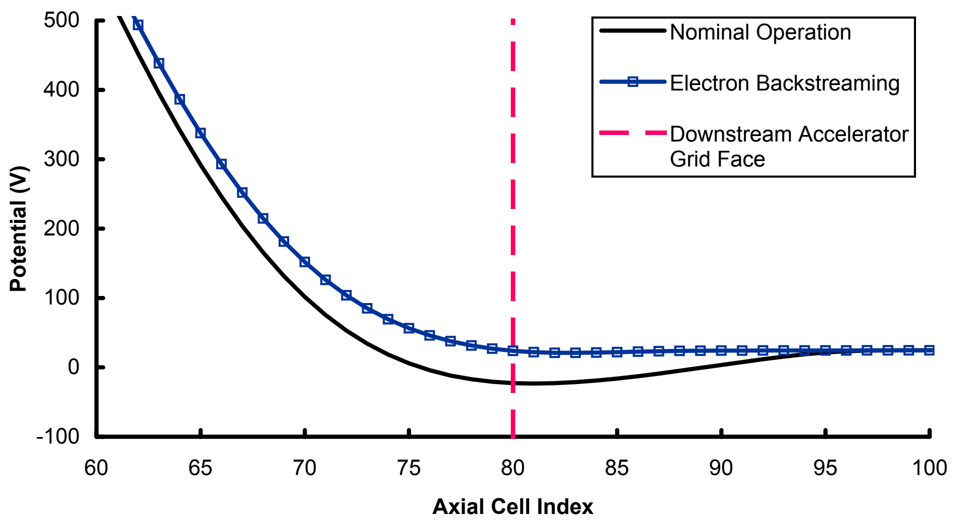


Figure 4.17: Centerline potential for nominal thruster operation and for operation with electron backstreaming. Electron backstreaming occurs when there is no significant drop in potential on the centerline near the accelerator grid.

The onset of electron backstreaming occurs when the minimum centerline potential in the ion optics rises to near the potential of the plume plasma. The centerline will always have the highest potential, as it is furthest from the accelerator grid, and also because the ion density is generally highest on the centerline. As Figure 4.17 shows, the accelerator grid creates a potential well that prevents electrons from being accelerated into the discharge chamber, but if that well is not deep enough, backstreaming will occur.

Experimental data for the electron backstreaming limit in the NEXT ion engine may be found in *Soulas et al.* (2002) and *Soulas et al.* (2004). The electron backstreaming limit for a given ion optics geometry is measured by increasing the accelerator grid potential until the measured beam current increases by 0.1 mA. This is approximately $3.36 \cdot 10^{-9}$ A of electron current for a single aperture. Apertures near the centerline of the thruster will generally produce backstreaming before the

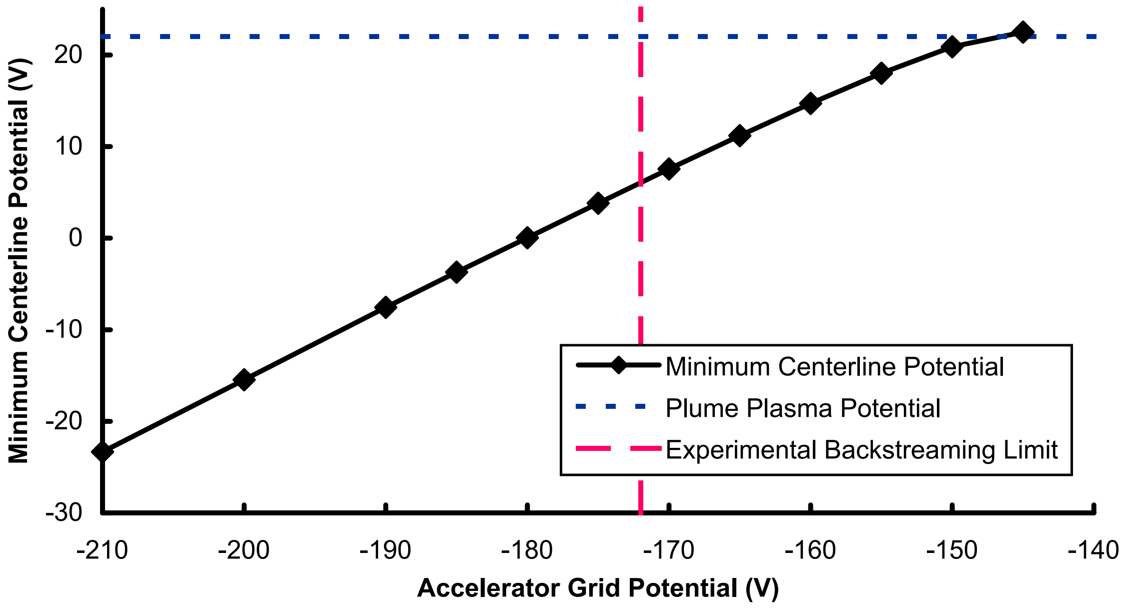


Figure 4.18: Minimum centerline potential as a function of accelerator grid potential. The downstream plasma potential and experimental backstreaming potential are also shown.

other apertures, because the higher ion density in these apertures results in a higher minimum centerline potential. These apertures also have the largest diameters, so backstreaming occurs more readily. For these reasons, the centerline aperture is simulated in the following cases.

4.6.1 Accelerator Grid Potential Variation

Figure 4.18 shows the simulated minimum centerline potential as a function of accelerator grid potential. Also shown is plume plasma potential of 22 V, and the experimentally measured backstreaming potential of -172 V. Backstreaming will occur when the minimum centerline potential nears the plume plasma potential, which is at an accelerator grid potential of about -150 V in the figure. This is a difference of 20 V between the simulation and the experiment. However, this may not be the appropriate comparison, as the electron backflow current is not taken into account.

To obtain a more accurate result, the simulated backflow current must be examined.

In the simulation, the electron population is determined by the Boltzmann relation (see Section 2.2.2), which gives the electron density at a point as a function of the potential, as well as a reference potential, temperature, and density. The average electron velocity can be estimated as the thermal velocity using the reference electron temperature. The electron backflow current can then be estimated by integrating radially across the domain at the minimum centerline potential point:

$$I_e = \int_0^{r_{max}} n_{e,0} \exp\left(\frac{\phi(r) - \phi_0}{T_{e,0}}\right) \frac{1}{4}\sqrt{\frac{8eT_{e,0}}{\pi m_e}} \pi r dr \ . \tag{4.5}$$

The simulation domain is discretized into cells, so this equation becomes a sum over a column of cells:

$$I_e = \pi n_{e,0} \frac{1}{4}\sqrt{\frac{8eT_{e,0}}{\pi m_e}} \Delta r \sum_{j=1}^{N_r} r_j \exp\left(\frac{\phi_j - \phi_0}{T_{e,0}}\right) \ . \tag{4.6}$$

In Figure 4.19, the simulated backflowing electron current as a function of the accelerator grid potential is shown for an electron temperature of 1 eV. The experimental current limit is also shown, as is the experimental backstreaming potential. Here, the comparison is somewhat better, as the simulation result is about 10 V higher than the experiment.

The method for computing the electron backflow current depends strongly on the electron temperature in the downstream region. Any changes in the electron temperature will likely have a significant effect on the current. Figure 4.20 verifies this, plotting the backflow current as a function of electron temperature for an accelerator grid potential of -170 V. Increasing from the base temperature of 1 eV to 2.2 eV is sufficient to pass the backstreaming limit.

Varying the electron temperature has no effect on the minimum centerline potential, as there are very few electrons at that point in the domain. The simulated

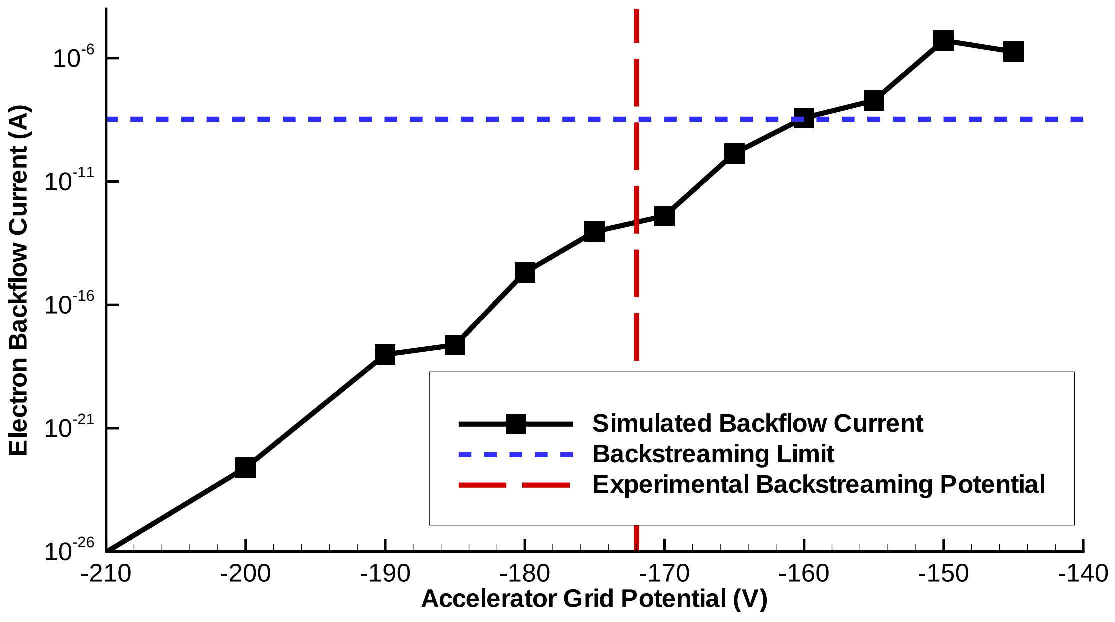


Figure 4.19: Electron backflow current as a function of accelerator grid potential. Also shown are the electron backstreaming limit and the experimental backstreaming potential. The electron temperature is 1 eV in these cases.

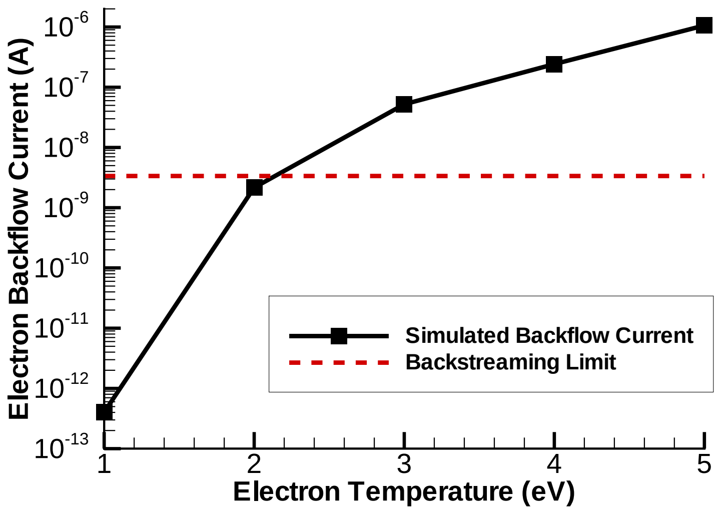


Figure 4.20: Electron backflow current as a function of electron temperature. The backstreaming limit current is also shown. The accelerator grid potential is set to -170 V for these cases.

minimum centerline potential is approximately 7.5 V at an accelerator grid potential of -170 V. This is the experimentally measured backstreaming potential, so it can be assumed that backstreaming will occur when the minimum centerline potential is higher than 7.5 V. This allows determination of backstreaming independent of the electron temperature used in the simulation.

4.6.2 Aperture Diameter Variation

Although the electron backstreaming limit is found experimentally by varying the accelerator grid potential, the thruster will actually fail when the accelerator grid aperture diameters increase enough such that backstreaming occurs. For simulations where the aperture is eroded until thruster failure, it is more useful to know at what aperture diameter electron backstreaming will occur.

In order to determine the necessary aperture diameter to allow electron backstreaming, several simulations are performed with a gradually increasing diameter while the grid potential is held constant at -210 V. In each case, one layer of cells is removed from the accelerator grid barrel. Figure 4.21 shows the resulting minimum centerline potential as a function of aperture diameter. The diameters here are normalized by the nominal diameter. Electron backstreaming occurs when the aperture enlarges by approximately 35%.

This method of aperture enlargement does not account for reduction in the thickness of the grid, and also assumes that the aperture diameter is increasing uniformly. A thinner grid will allow backstreaming to occur sooner, and a non-uniform erosion pattern may do this as well.

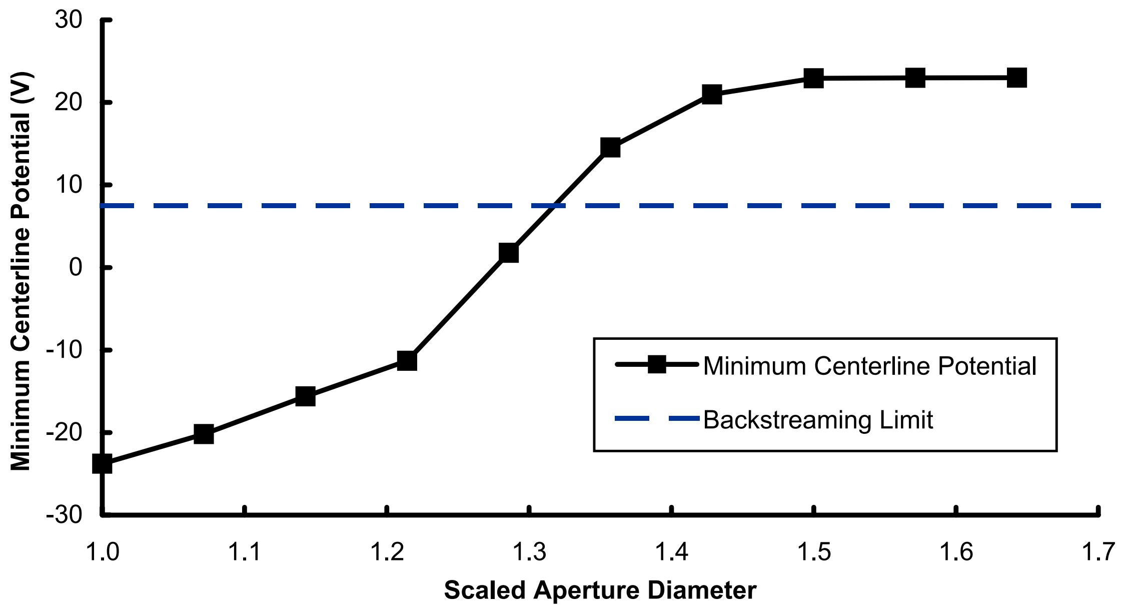


Figure 4.21: Minimum centerline potential as a function of accelerator grid aperture diameter. Also shown is the potential at which electron backstreaming will occur. The accelerator grid potential is -210 V for these cases.

4.7 Thruster Life Modeling Results

Thruster failure due to electron backstreaming can be estimated by the model in several ways. A single simulation gives results for the erosion rate in each accelerator grid cell. These erosion rates can be used to erode the ion optics until the aperture is large enough to allow electron backstreaming. Another method is to use the erosion rates from the nominal geometry to erode over a set amount of time. Then a new simulation is run with the eroded geometry, and the erosion rates from the second simulation are used to erode the grid further. This process is continued until electron backstreaming occurs. Finally, the dynamic erosion algorithm (see Section 2.1.3) may be used to erode the ion optics during a single simulation.

The axisymmetric domain of the model does not allow for accurate simulation of erosion on the downstream face of the ion optics. The hexagonal arrangement of the apertures in the optics creates a “pit-and-groove” pattern, where pits form between three adjacent apertures and grooves form between two adjacent apertures. The simulation is unable to model either of these structures. Thus, estimation of the life of the ion optics before they incur structural failure is not possible with the current model.

4.7.1 Multiple and Single Simulation Results

The thruster life is first estimated using multiple simulations with static geometry. In each case, the cusped NEXT ion optics wear test geometry is simulated at the 3.52 A, 1800 V operating point. The peak beamlet current of approximately 0.168 mA is modeled and the accelerator grid potential is fixed at -210 V. A total of five simulations are performed, with 10,000 hours of erosion being performed at each step. The initial geometry is simulated first, which gives erosion rates on the accelerator grid barrel. These erosion rates are applied to the geometry to erode it for 10,000 hours. The 10 k-hr erosion case is then simulated to give updated erosion rates, which are used to erode to 20 k-hr. This is done until electron backstreaming occurs at after approximately 40 k-hr of erosion. At each step, the entire flow field is updated, not just the grid geometry. This ensures that the evolution of the erosion pattern is as accurate as possible.

The erosion rates on the accelerator grid barrel are shown for each simulation in Figure 4.22. For each column of simulated optics cells, the erosion rate is applied to the cell with the lowest radius. Erosion will then occur in a strictly radial fashion, and downstream face erosion will not be accounted for. The erosion rates do not change an exceptional amount as geometry varies, although there are spikes at some points for the extremely eroded cases. At these geometries, parts of the optics are less

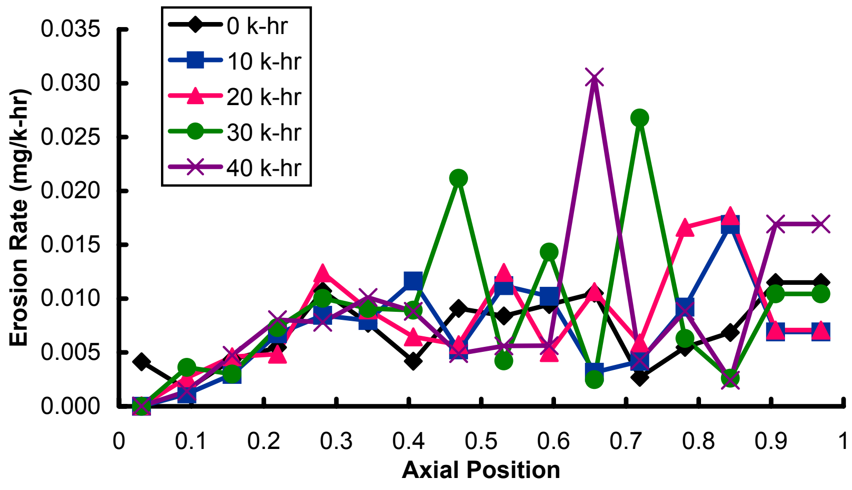


Figure 4.22: Accelerator grid barrel erosion rate evolution as the thruster is eroded. The peaks in erosion for late in the thruster life correspond to peaks in the geometry. These points are impacted by ions from the downstream region, increasing the erosion rate considerably.

eroded and thus are more likely to be impacted on by CEX ions, especially from the downstream region. Also, because the erosion measured on the downstream corner of the grid includes erosion of the downstream face, the erosion for that cell is always set to be the same as for the cell directly next to it.

The ion optics geometry at each simulation step is shown in Figure 4.23. Very little erosion occurs on the upstream surface of the grid, so these points do not vary greatly over the life of the thruster. The erosion rate then increases towards the center of the grid, creating a pit in front of the grid midpoint. This pit is caused by high-energy CEX ions created in the inter-grid region. Erosion in the center region is average, followed by another region of high erosion. The erosion is higher on the downstream half of the grid due to CEX ions created near the accelerator grid aperture and in the downstream region.

The minimum centerline potential as a function of erosion time is plotted in Figure 4.24. As the plot shows, the electron backstreaming limit is reached shortly

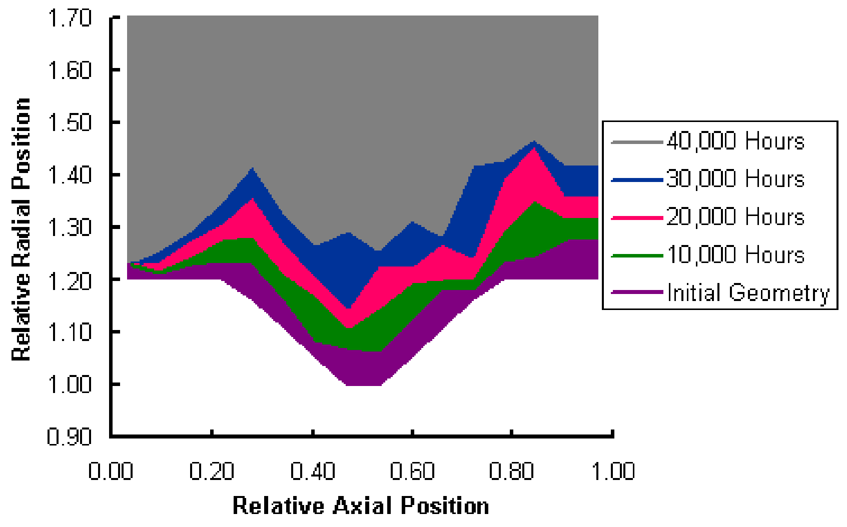


Figure 4.23: Geometry of the accelerator grid barrel as a function of erosion time. The axial position is normalized by the grid thickness, with its origin at the grid upstream surface. The radial position is normalized by the initial aperture radius— the origin is on the domain centerline.

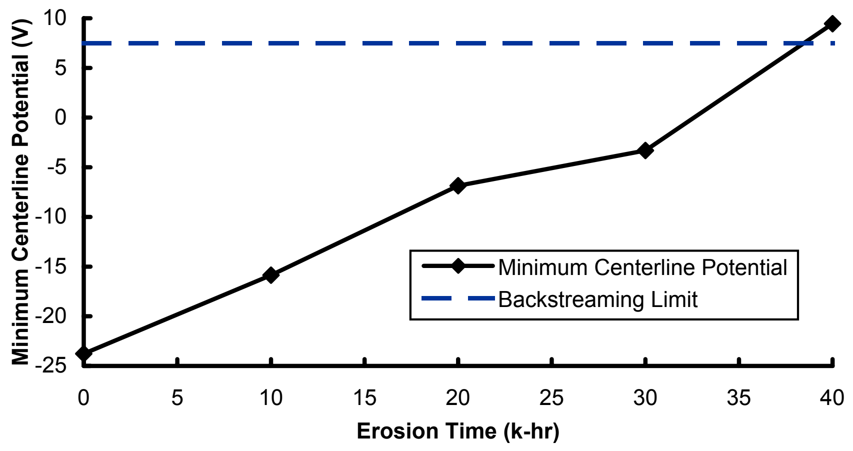


Figure 4.24: Minimum centerline potential as a function of erosion time. The backstreaming limit is also shown. The potential varies linearly with time, indicating that backstreaming may be predicted based solely on initial erosion rates.

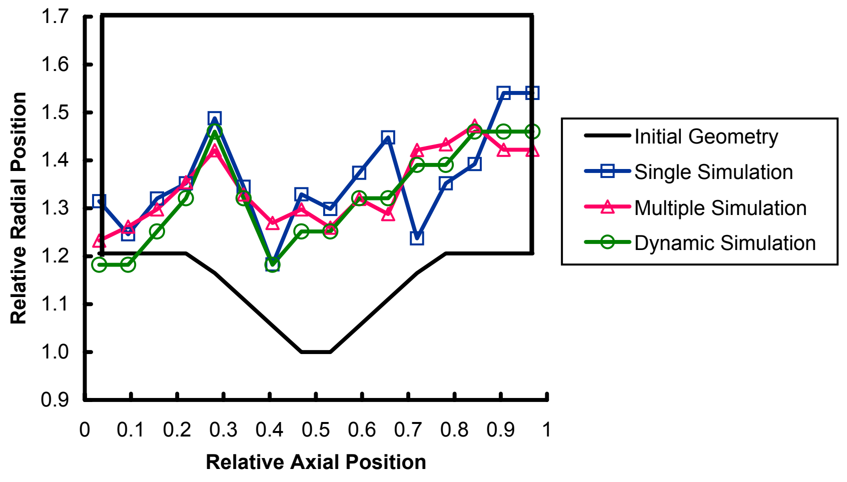


Figure 4.25: End-of-life accelerator grid barrel geometry for single, multiple, and dynamic simulations. The single and multiple simulation final geometries are both reached after approximately 40,000 hours of erosion. The multiple-simulation case is generally smoother because erosion is redistributed as different parts of the grid are eroded. The dynamic erosion case does not include downstream face erosion, and the final geometry is reached after 45,000 hours of erosion.

after 40 k-hr of erosion. The increase in potential is linear with time as well, which indicates that the backstreaming behavior can be predicted easily given initial erosion rates.

The life of the thruster may also be estimated by eroding the accelerator grid barrel using only erosion rates simulated at the initial geometry. The results plotted in Figure 4.24 indicate that this method will be effective. Indeed, electron backstreaming is found to occur in this case after approximately 40 k-hr of thruster operation, the same as seen in the multiple simulation case. Figure 4.25 plots the final geometries for both single-simulation and multiple-simulation results. The geometries are very similar, although the multiple-simulation result is smoother, due to the erosion being re-distributed as some parts of the grid are worn away.

4.7.2 Dynamic Erosion Results

Finally, erosion is modeled using dynamic erosion of the ion optics. In this case, the simulation begins with the initial geometry, and initializes the flow using this geometry. Once the flow is initialized, dynamic erosion is enabled. Whenever an ion impacts on an optics surface, the number of eroded molybdenum atoms is calculated. These atoms are removed from the optics cell, and if all of the atoms are removed from a cell, that cell is no longer considered an ion optics cell. Re-deposition of sputtered grid material is not simulated here.

The simulation time-step is on the order of $1{\cdot}10^{-10}$ seconds, and the dynamic erosion is performed for 100,000 iterations, giving a total simulated time on the order of 0.01 milliseconds. To allow simulation of thruster life, the number of atoms in an optics cell must be scaled. This scaling factor is simply the amount of time the simulation actually models divided by the thruster erosion time. In this simulation, the thruster erosion time is set to 50,000 hours, so the scaling factor is approximately $3{\cdot}10^{-13}$.

The thruster geometry and potential field is plotted in Figure 4.26 at approximately 10, 20, and 30 k-hr of erosion. After 30 k-hr of erosion, the electron backstreaming limit is reached. The barrel erosion in these cases is approximately the same as seen in the multiple-simulation case, but the downstream face erosion thins the grid such that backstreaming is allowed at a smaller aperture diameter. This thinning is not typically seen experimentally, as the downstream face erosion is largely contained within the "pits-and-grooves" pattern. Some erosion of the downstream edge of the aperture is expected, but thinning of the entire face is not realistic. However, the large amount of erosion does indicate that the thruster will likely fail due to structural failure of the accelerator grid before electron backstreaming occurs.

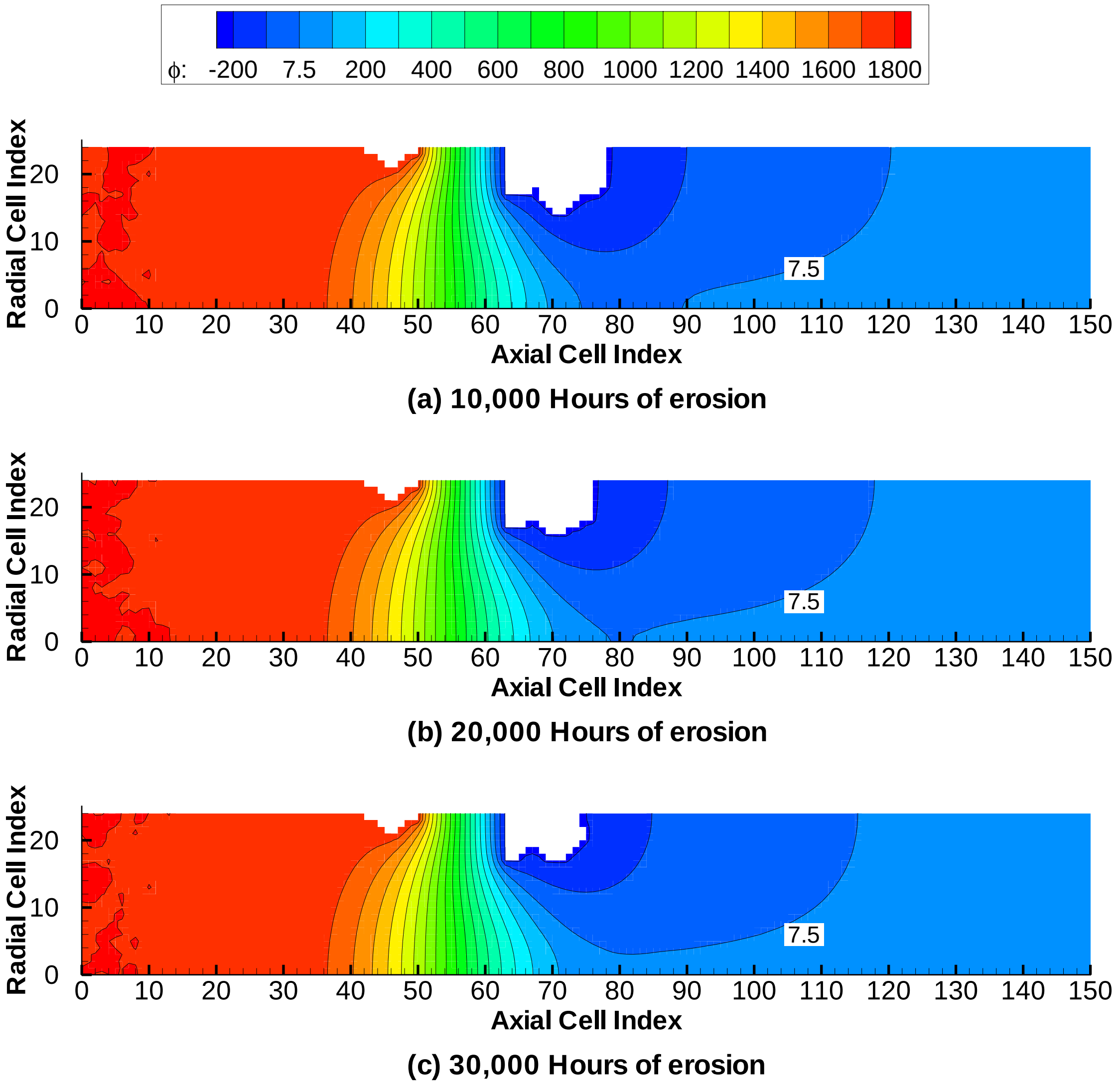


Figure 4.26: Potential field contours as the accelerator grid is eroded dynamically. Dimensions are in volts. The 7.5 V contour is labelled— when the downstream contour connects with the upstream contour, electron backstreaming will occur. After approximately 30,000 hours of erosion, backstreaming begins.

A better comparison to the multiple and single simulation results is to restrict the dynamic erosion algorithm, such that downstream face erosion does not occur or is mitigated. This is accomplished by not eroding material when an ion impacts on the downstream face of the grid. However, this still allows impacts on the downstream side of any eroded geometry or the cusps.

This method produces results very close to the multiple and single aperture simulation erosion. The erosion and potential field after 15, 30, and 45 k-hr of erosion is plotted in Figure 4.27. As before, electron backstreaming occurs when the 7.5 V contour is connected between the up and downstream regions. In this case, backstreaming occurs after 45,000 hours of erosion. Note that the downstream half of the grid is systematically chamfered by CEX ions from the downstream region. Although erosion is not allowed by these ions on the actual downstream face, they do contribute to erosion in other areas.

As shown by Figure 4.25, the dynamic erosion geometry at which backstreaming occurs is very similar to the previous results. One difference is that this result is reached after approximately 45,000 hours of erosion, rather than the 40,000 hours needed in the multiple and single simulation cases. This difference most likely occurs because the geometry simulated in the dynamic erosion case depends on the mesh of the domain. Thus, as the first two points in Figure 4.25 show, the dynamic erosion case must erode more material in some areas. In the other types of simulations, the true geometry is eroded using the simulated erosion rates. Also, the potential field for the dynamic erosion case is a snapshot- it is the potential for only one iteration. The other cases are able to average the potential over many iterations to find the minimum centerline potential. Thus, the potential in the dynamic case may fluctuate from the value measured.

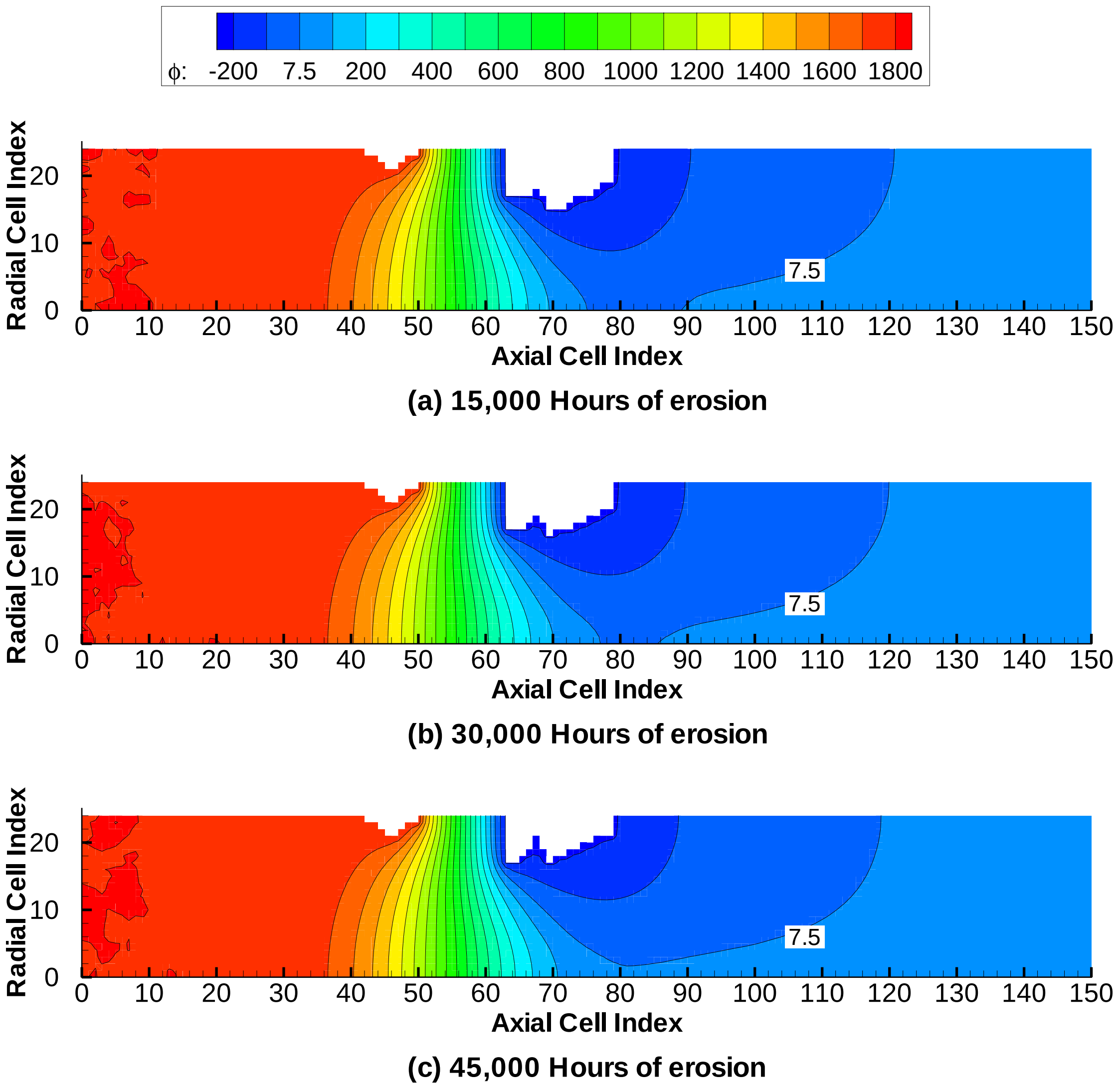


Figure 4.27: Potential field contours as the accelerator grid is eroded dynamically, without downstream face erosion. Dimensions are in volts. The 7.5 V contour is labelled— when the downstream contour connects with the upstream contour, electron backstreaming will occur. After approximately 45,000 hours of erosion, backstreaming begins.

4.7.3 Summary

The erosion predictions given here for the single and multiple simulation cases predict thruster failure after approximately 40,000 hours of thruster operation at the maximum operating point. This corresponds to 845 kg of propellent throughput. This is likely a very optimistic prediction given that downstream face erosion is ignored. However, it is apparent that erosion rates from a single simulation are sufficient to predict electron backstreaming behavior.

This model has been used previously to predict electron backstreaming. In *Emhoff and Boyd* (2003), the backstreaming limit was reached after approximately the same thruster operating time. Results for a 3-D model given in *Farnell et al.* (2003) predict thruster failure much sooner, after a propellant throughput of 625 kg. However, this estimate is for an accelerator grid potential of -250 V, and is due to structural failure, not electron backstreaming. These results define thruster failure due to structural failure as the point at which 50% of the grid material has been eroded. At an accelerator grid potential of -200 V, the estimate increases to 805 kg of throughput, much closer to the result given above, although end-of-life is again due to structural failure in this case. The rapid erosion of the downstream face in the dynamic erosion simulation supports this prediction. Predictions based on experimental data give failure after 750 kg of throughput (*Soulas et al.* (2004)), also due to structural failure. In this case, structural failure is defined as the point at which the grooves will erode through the grid.

CHAPTER V

Treecode Potential Solver Results

5.1 1-D Sheath Results

The one-dimensional sheath problem provides an ideal test domain for the use of the treecode. It is a simple domain with well-defined behavior, and as such the basic operation of the treecode may be easily verified. In this section, Particle-In-Cell results given by *Lieberman and Lichtenberg* (1994) are compared to the treecode results. Direct sum results are not given, as the treecode computes exact forces and potentials in 1-D. This is because the Green's function can be described exactly with only two Taylor expansion terms.

5.1.1 1-D Treecode Derivation

In a 1-dimensional domain, the Green's function is the following:

$$G(x_i, x_j) = 1 + \frac{1}{2}|x_i - x_j| \ . \tag{5.1}$$

This function only has one non-zero derivative:

$$\frac{dG}{dx} = \frac{(x_i - x_j)}{2|x_i - x_j|} \ . \tag{5.2}$$

As a result, the Taylor expansion does not need to be truncated and thus the treecode gives the exact result in a 1-D domain.

5.1.2 1-D Sheath Domain

The 1-D sheath domain is bounded by two charged plates a distance L apart, each held at a potential of zero volts. The domain initially has a neutral plasma with electrons and ions at equal densities. In this simulation, electrons are modeled as particles while the ion density is held constant. The potential of the domain is then described by the following form of the Poisson equation:

$$\nabla^2\phi = \frac{e}{\varepsilon_0}\left[\sum_{i=1}^{N_e} w_i\delta(x-x_i) - n_i\right] . \tag{5.3}$$

x_i is the position and w_i is the weight of electron particle i, and N_e is the number of electron particles in the simulation. The potential can be split into contributions from the free-space electrons and from the bounded ion density:

$$\begin{aligned}
\phi &= \phi_e + \phi_i && (5.4)\\
\nabla^2\phi_e &= \frac{e}{\varepsilon_0}\sum_{i=1}^{N_e} w_i\delta(x-x_i) && (5.5)\\
\nabla^2\phi_i &= -\frac{e}{\varepsilon_0}n_i\ . && (5.6)
\end{aligned}$$

Integrating both contributions separately gives

$$\begin{aligned}
\phi_e(x) &= \frac{e}{\varepsilon_0}\sum_{i=1}^{N_e} w_i G(x,x_i) && (5.7)\\
\phi_i(x) &= -\frac{e}{\varepsilon_0}\left(\frac{1}{2}n_i x^2 + Ax + B\right)\ . && (5.8)
\end{aligned}$$

$G(x,x_i)$ is the 1-D Green's function as given in Eqn. 5.1. A and B are constants to be determined by the boundary conditions of the domain.

The boundary conditions are $\phi(0) = 0$ and $\phi(L) = 0$. This gives the equations:

$$\begin{aligned}
0 &= \frac{e}{\varepsilon_0}\left[\sum_{i=1}^{N_e} w_i\left(1 + \frac{1}{2}|0 - x_i|\right) - B\right] && (5.9)\\
0 &= \frac{e}{\varepsilon_0}\left[\sum_{i=1}^{N_e} w_i\left(1 + \frac{1}{2}|L - x_i|\right) - \frac{1}{2}n_i L^2 - AL - B\right]\ . && (5.10)
\end{aligned}$$

Solving these for the unknowns A and B gives:

$$B = \sum_{i=1}^{N_e} w_i + \frac{1}{2}\sum_{i=1}^{N_e} w_i x_i \tag{5.11}$$

$$A = \frac{1}{2}\sum_{i=1}^{N_e} w_i - \frac{1}{L}\sum_{i=1}^{N_e} w_i x_i - \frac{1}{2} n_i L \ . \tag{5.12}$$

Finally, the potential at any point x can be expressed as:

$$\phi(x) = \frac{e}{\varepsilon_0}\left[\sum_{i=1}^{N_e} w_i G(x, x_i) - \frac{1}{2} n_i x^2 - Ax - B\right] \ . \tag{5.13}$$

The simulation is initialized with a random distribution of electron particles with thermal velocities. Electrons will impact on the wall through thermal motion and be absorbed, reducing the electron density near the wall. This causes the formation of a plateau in potential in the center of the domain. This plateau contains the remaining electron particles, which are unable to overcome the potential gradient near the wall.

The treecode simulation results are compared against PIC results given by *Lieberman and Lichtenberg* (1994). The domain modeled is 0.1 m in length, the number density of each species is $1 \cdot 10^{13}\ m^{-3}$, and the electron temperature is 1 eV. 4000 electron particles are simulated initially. One difference between the PIC and the treecode is that the PIC simulation has a floating, or constant current, boundary condition on the right side of the domain. This does not affect the results a great deal however.

Figure 5.1 plots the initial electron phase space. Velocities are within the bounds of approximately $\pm 1 \cdot 10^6$ m/s. In Figure 5.2, the potential field is plotted as a function of position for both PIC and treecode. The PIC potential field is slightly higher than that of the treecode, but the sheath thickness is approximately the same at about 10 mm. The difference in potential field may be due to error in the PIC results— if the mesh size used was not small enough to accurately capture the field

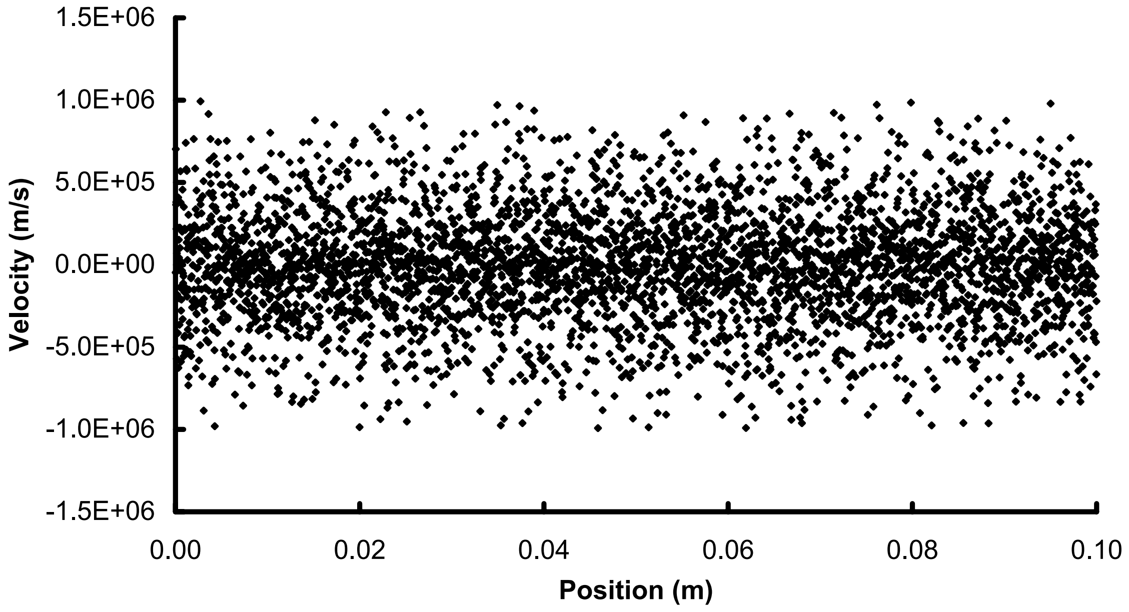


Figure 5.1: Phase space plot of the initial electron particle distribution. The electron temperature is 1 eV.

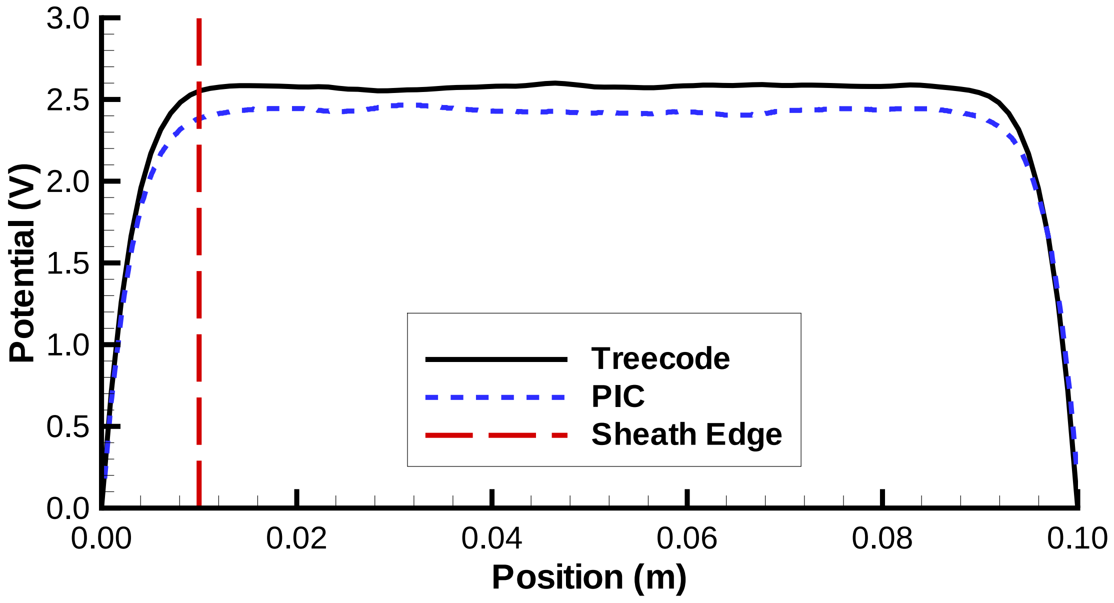


Figure 5.2: Potential field plot for both treecode and PIC results. The sheath edge is marked by the red line.

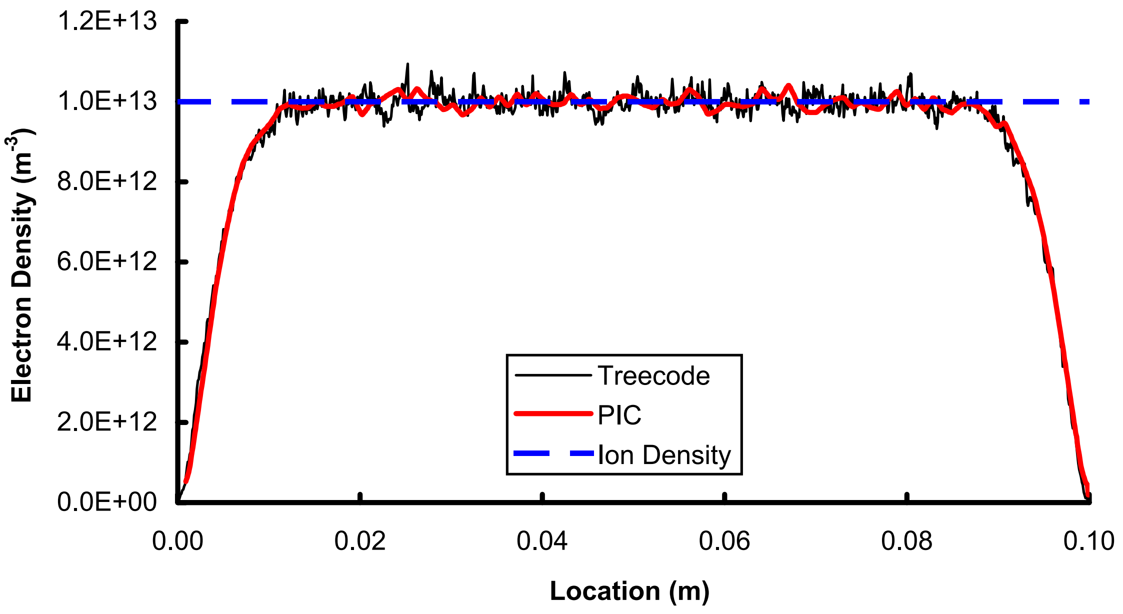


Figure 5.3: Electron density field for both PIC and the treecode. The ion density is also shown. The comparison between PIC and the treecode is excellent in this case.

gradient in the sheath region, then this would give some error in the potential field. The calculated matrix sheath thickness (*Roth* (2001)) for this domain is 5.4 mm, comparable to the computational result.

The electron and ion densities are shown in Figure 5.3 for both PIC and the treecode. Although the treecode electron density has more scatter than the PIC case, both are centered at $1 \cdot 10^{13}\ m^{-3}$, and the density in the sheath regions is matched exactly. The high amount of scatter in the treecode result is due to the use of a fine sampling mesh— a coarser mesh would smooth the results somewhat.

One major difference between the two simulations is the rate of particle depletion. In Figure 5.4, the number of simulated particles is plotted as a function of time. The PIC simulation converges nearly four times faster than the treecode simulation, although both eventually converge to approximately the same number of particles.

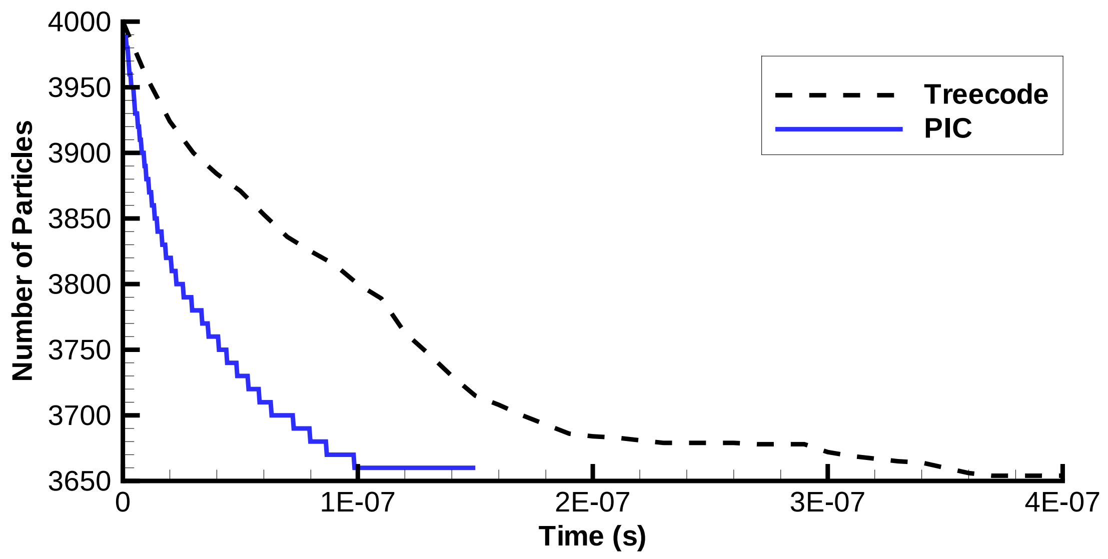


Figure 5.4: Number of simulated electron particles as a function of time for both PIC and the treecode. The PIC particle number drops much more rapidly than the treecode particle number, although both converge to about the same value.

The thermal flux to the wall gives an estimate of the loss rate early in the sheath formation process. This flux is about 134 particles per $1 \cdot 10^{-8}$ seconds. The PIC particle loss in the first $1 \cdot 10^{-8}$ seconds is 114 particles, while the treecode loses only 42 particles. So, it appears that the treecode is losing particles more slowly than it should. One possibility is that the treecode slows the loss rate more rapidly due to a more accurate force computation. Thus, the loss rate may be higher very early in the simulation, but the rapid formation of the sheath results in a lower loss rate than the thermal flux predicts.

In summary, the treecode results compare very well to the PIC results in this domain. The density is matched between the two methods, there is only a slight difference in potential, and the converged particle numbers are approximately the same, although the loss rate is higher in the PIC case.

5.2 2-D Box Domain Results

In a two-dimensional domain, the treecode is compared to the PIC and direct summation methods. The domain used is a simple 1.0 meter square box, where each of the four boundaries is either Dirichlet or Neumann. The boundary panel method described in Section 3.4 is used to represent the boundaries in this domain. Particles are placed randomly in the domain, however they are restricted to a 0.8 m square such that no particle can be closer than 0.1 m to a boundary. This is done to prevent interference in the particle force computation by the singular behavior of the panel method near the panel surface. This singular behavior is discussed further in Section 5.3.1.4.

5.2.1 Panel Method Results

Before examining the effect of the treecode on particle force computation, the boundary integral method is explored to determine what effect it has on the accuracy of the domain and particle forces. For these simulations, the left-hand boundary is set to a 100 V Dirichlet condition, the right-hand side is 200 V Dirichlet, and the upper and lower boundaries are 0 V/m Neumann conditions. For each case, the potential field is calculated on a 101 by 101 mesh in the domain, and the error is determined by differencing the result from the linear field value of $\phi = 100x + 100$. The potential at the boundaries is not included in this calculation. The computational time is defined as the time taken to multiply the inverted panel effects matrix with the source vector plus the time taken to compute the potential field.

Figure 5.5 plots the computation time required and the average error as a function of the number of panels per side. In these results the panel effects are integrated directly. The error decreases linearly with the number of panels— for double the

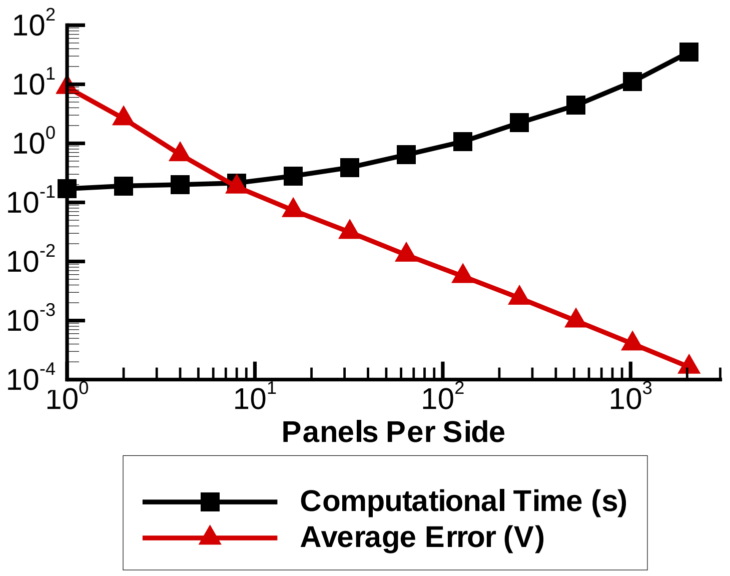


Figure 5.5: Computational time required and the average error as a function of the number of panels per side. The error decreases linearly with the number of panels per side, while the computational time increases more slowly.

number of panels, the error drops by half. The rate of increase in the timing rises with the number of panels, but for the highest number of panels tested, the timing still only increases by a factor of three for the factor of two increase in panel number. The efficiency of the simulation accuracy reaches a turning point at the 1024 panel per side point. Prior to this, doubling the number of panels halves the error for less than double the computation time. But beyond this point, halving the error costs more than double the computation time.

The increase in time is caused by the matrix-vector multiplication performed to give the panel strengths, as described in Section 3.4.1. This is an $O(N^2)$ process, where N is the number of panels, so as the number of panels becomes large, this multiplication will begin to dominate the computation time.

Gaussian Quadrature (GQ) may also be used in lieu of direct integration of the panel effects. This has the advantage of being more computationally efficient at the cost of accuracy. However, in the 2-D domain the savings in computation cost are not sufficient to warrant its use. For example, with 128 panels per side, direct integration requires 1.07 s to compute the potential field with an average error of 0.0056 V. The four point GQ integration requires 0.72 s and gives an error of 0.0069 V. Increasing the number of integration points only slightly increases the accuracy at this point, and requires as much time as direct integration.

5.2.2 Direct Summation Force Computation

Direct summation of particle forces gives exact answers, but has a computational cost of $O(N^2)$, where N is the number of particles. The purpose of the treecode is to decrease this computation time to $O(N \log(N))$ with a tolerable decrease in accuracy.

The computational time required to compute the forces on all particles as a function of the number of particles is plotted in Figure 5.6. The direct summation time and the treecode times are shown, in a domain with 0 V Dirichlet boundaries on all sides. The treecode parameters here are 4 Taylor expansion terms, 8 particles per cluster maximum, and an acceptance parameter of 0.16, giving an average relative error of about $2{\cdot}10^{-5}$. For each increase of a factor of approximately three in particle number, the direct sum timing increases by an order of magnitude, while the treecode time increases by a factor of about 4. Note that other combinations of treecode parameters are possible that may give the same accuracy for less computational work.

5.2.3 Treecode Parameter Variation

In this section, a study of the effects of the treecode parameters is conducted. The domain is set to 0 V Dirichlet on all sides, and the particle number is varied

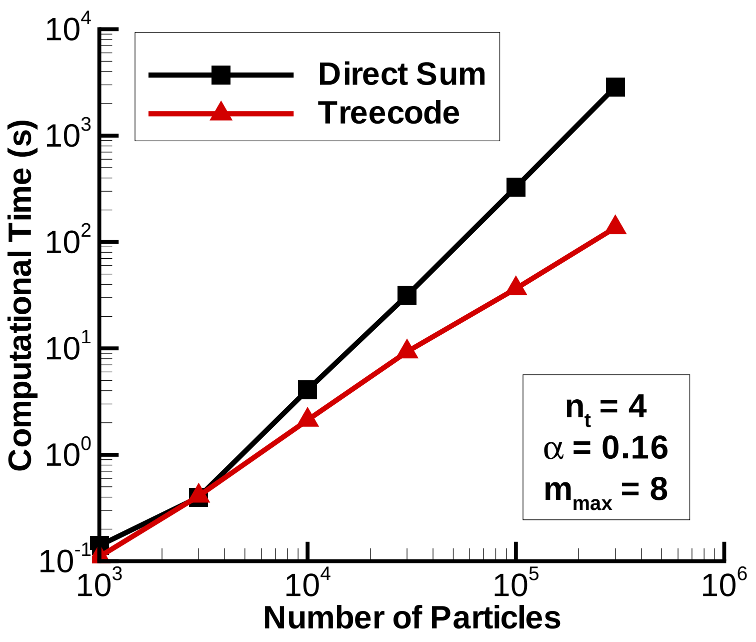


Figure 5.6: Computation time as a function of particle number, for both direct summation and the treecode. The direct sum timing is $O(N^2)$, as expected, while the treecode timing is approximately $O(N \log(N))$.

from 1,000 to 300,000 as each parameter is varied individually. The base treecode parameters are 4 Taylor expansion terms, a maximum of 8 particles per cluster, and an acceptance parameter of 0.16. One of these three parameters is varied while the other two are held constant at these values. In each case, the force is computed on each particle in both the X and Y directions. The absolute value of the difference between each force and the direct sum force is divided by the direct sum force to obtain the relative error. This is then averaged over all particles to obtain the average relative error for that operating point. The computational time required is defined as the time needed to compute the force on each particle.

5.2.3.1 Number of Taylor Expansion Terms

Figures 5.7(a) and 5.7(b) show the relative error as a function of the number of terms, n_t, in the Taylor expansion, in the X and Y directions respectively. The force

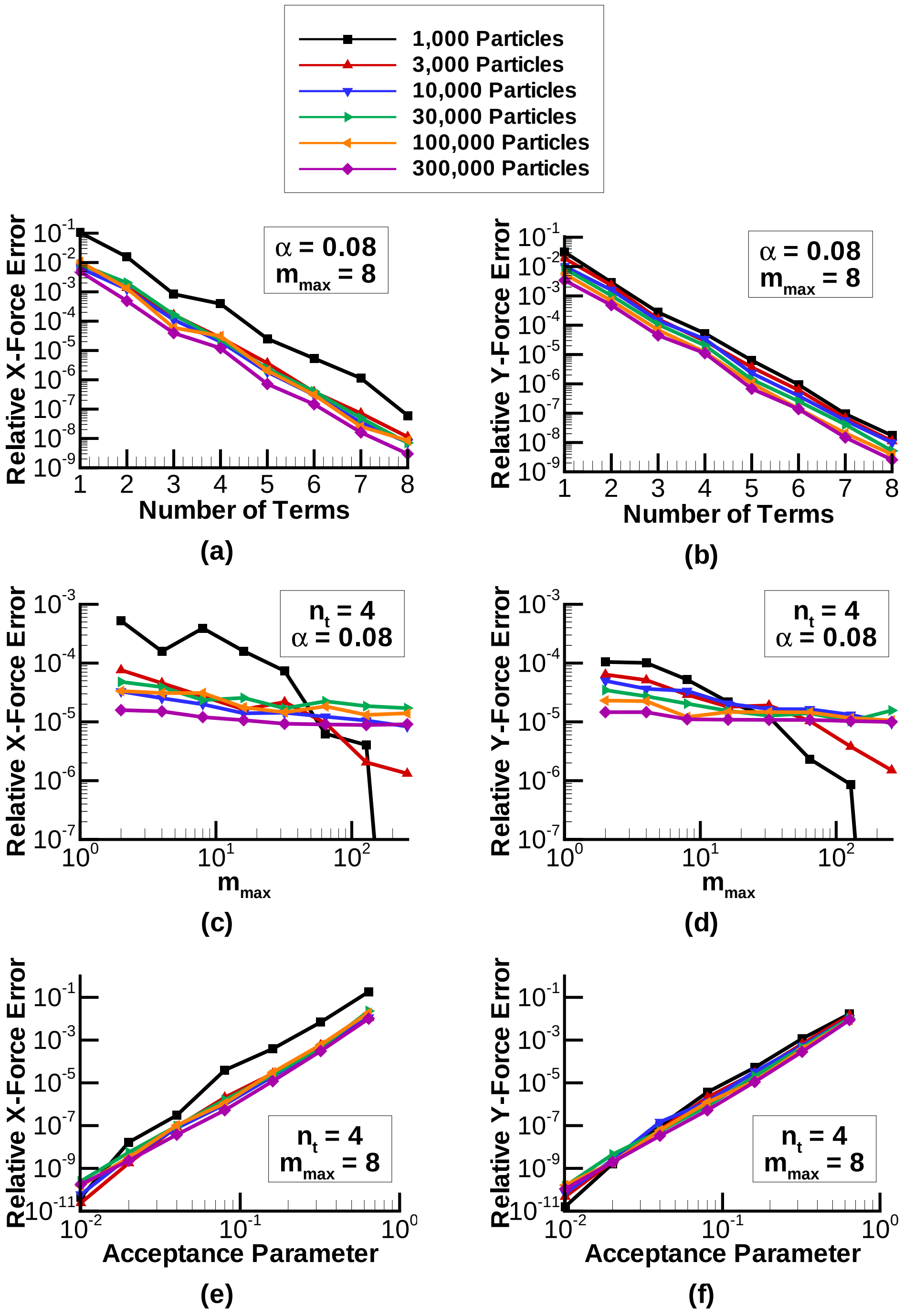


Figure 5.7: Average relative particle force error for varying treecode parameters. (a) and (b) show the X-force and Y-force error for a varying number of Taylor expansion terms. (c) and (d) plot the error against the maximum number of particles per cluster, and (e) and (f) plot the X and Y force error for varying acceptance parameter. The base values for each parameter are 4 Taylor terms, $m_{max} = 8$, and $\alpha = 0.16$.

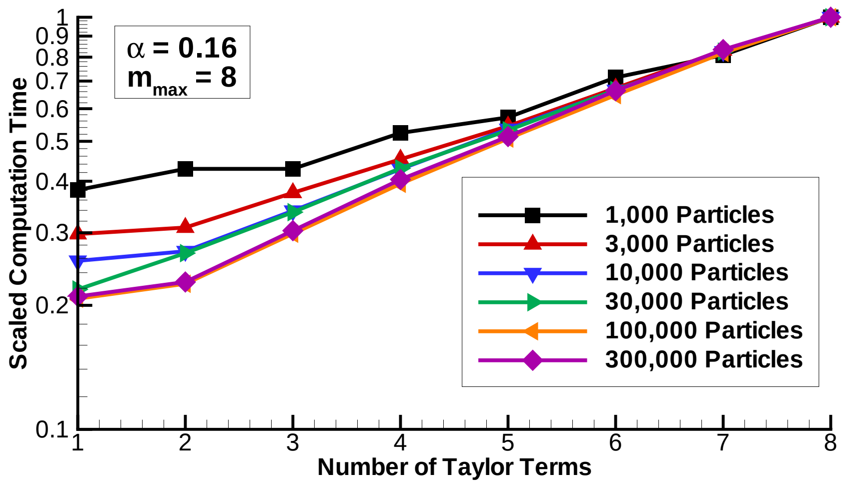


Figure 5.8: Computational time required as a function of the number of terms in the Taylor expansion and the number of particles simulated.

error decreases by an order of magnitude for each term in the expansion, as it should given the error in Taylor's expansion. The error is not exactly the same between the X and Y directions because the particles are distributed randomly. For only 1000 particles, there is a large difference between the X and Y force error, but for 300,000 particles, there is no noticeable difference.

Figure 5.8 shows the computational time required as a function of the number of terms in the Taylor expansion, with the values scaled by the 8-term time. The increase in computational cost with the number of terms is approximately linear for low numbers of terms, but the trend becomes steeper for higher numbers of terms. Even so, increasing the number of Taylor terms is a computationally efficient method for reducing the force error. Note that in 2-D, this computation will asymptotically approach $O(n_t^2)$, while in 3-D it approaches $O(n_t^3)$. Thus, in 3-D, the efficiency of adding Taylor terms is lower than in 2-D.

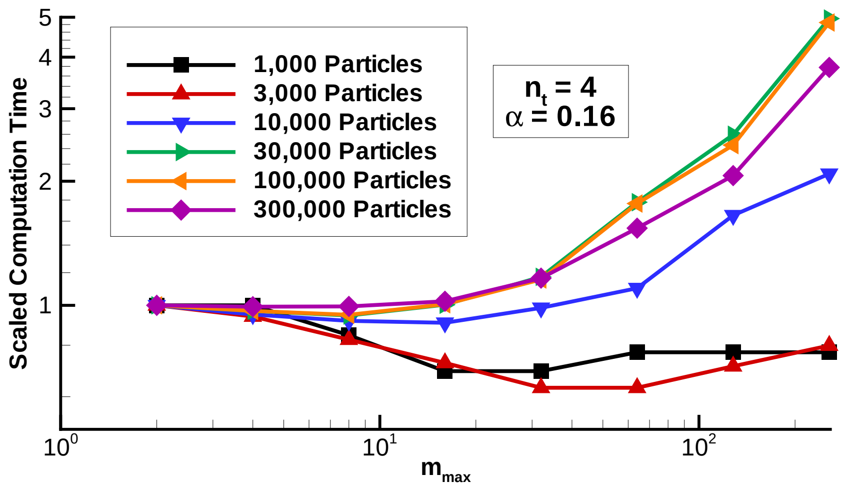


Figure 5.9: Computational time required as a function of m_{max} and the number of particles simulated. High values of m_{max} result in direct summation being over-used, giving poor time performance.

5.2.3.2 Maximum Number of Particles per Bottom-Level Cluster

The relative error as a function of m_{max}, the maximum number of particles per bottom-level cluster, is shown in Figures 5.7(c) and 5.7(d). As the plot shows, there is no advantage in terms of accuracy to larger values of this parameter. For higher numbers of particles, the accuracy is virtually flat, only decreasing slightly as the number of particles allowed in a cluster increases. For the lowest numbers of particles, the accuracy increases rapidly at some values of m_{max}, but this is because the simulation is performing direct summation on all clusters. This is seen in Figure 5.9, where the computation time as a function of m_{max} is plotted, scaled to the time required at $m_{max} = 2$. As m_{max} is increased from 2, the computation time decreases slightly before increasing a large amount as the parameter becomes large. For large numbers of particles, the optimal value for the parameter is about 8, while the optimal value is 16 to 32 for lower numbers of particles.

This behavior is caused by the balance between the effort spent on computation using the treecode and the effort spent on direct summation of particle forces. When m_{max} is low, then a very large number of clusters are constructed, and the treecode will compute the set number of terms in each cluster. For a cluster with only 2 or 4 particles, the use of 4 Taylor terms is excessive and leads to increased computational cost over simply using direct summation. Conversely, if $m_{max} = 128$, clusters will not be accepted very often and direct summation will have to be performed frequently.

5.2.3.3 Acceptance Parameter

Figures 5.7(e) and 5.7(f) plot the relative force error as a function of the acceptance parameter α. Each time the parameter is halved, the accuracy increases by more than an order of magnitude— generally the increase is by a factor of about 20. The scaled computation time is plotted in Figure 5.10, showing that the time increases by about a factor of 2 each time α is halved.

This is expected behavior, as the acceptance parameter essentially controls the amount of direct summation performed in the treecode. Lower values of α will result in more direct summation, so computation time and accuracy will both go up accordingly. The increase in time per increase in accuracy is much higher for α than it is for n_t however.

5.2.3.4 Summary

A strategy for parameter selection can be formed based on the above results. m_{max} should depend on the number of terms in the Taylor expansion, as well as the number of particles. However, this dependence is not strong, so a base value of $m_{max} = 8$ can be set without too much increase in computation time. The acceptance parameter gives very accurate results for lower values, but at a high computational cost. Instead, α should be set as high as possible, then the number of terms in the

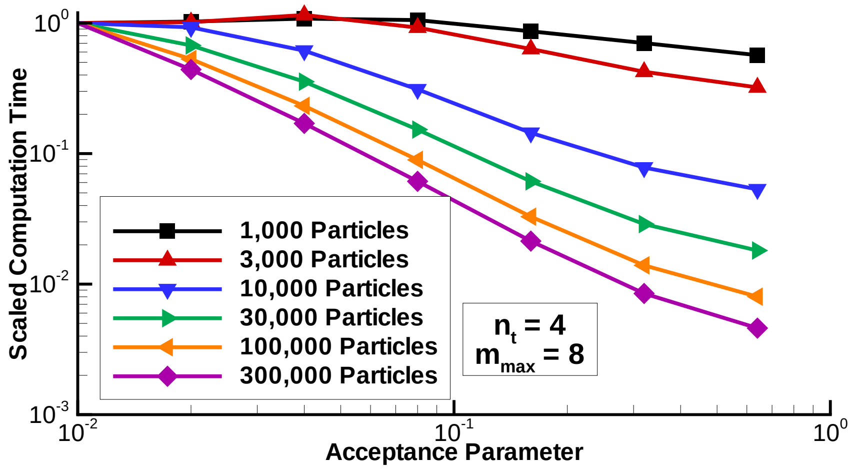


Figure 5.10: Scaled computation time as a function of the acceptance parameter. Computation time increases approximately linearly with α, with steeper increases as the particle number increases.

Taylor expansion can be increased to give more accurate results.

In general practice, a scheme is used in treecodes such that n_t is determined when a cluster is evaluated. For small clusters far away from the reference point, only a couple of terms are needed, while large clusters close to the reference point require many terms. This process is described further in Section 3.3.

5.2.4 Comparison to Particle-In-Cell

A simple Particle-In-Cell (PIC) potential field solver is used for comparison to direct sum and the treecode in the 2-D box domain. The PIC cells are square and uniform in all cases. An alternating-direction-implicit method solves the potential field, with a residual limit of $1 \cdot 10^{-9}$. In a domain with no particles, this limit gives the same average potential field error as the panel method with 128 panels per side.

The computation time of the PIC method does not depend strongly on the number of particles simulated. Instead, the number of cells simulated is the prime factor which determines the amount of computation time required. Figure 5.11 shows the

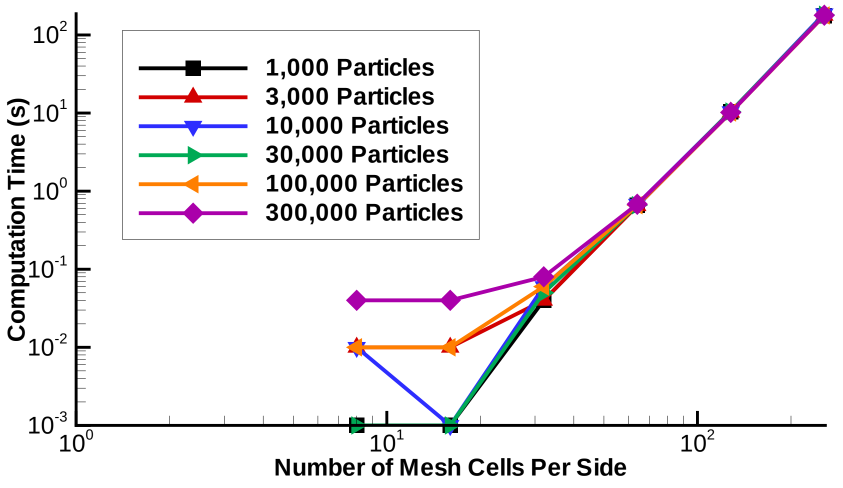


Figure 5.11: Computational time required for particle force calculation as a function of the number of mesh cells in each direction for the PIC solver. The domain solved is held at a 0 V Dirichlet condition on all sides.

time required to solve the potential field and compute particle forces as a function of the number of mesh cells in each direction. The total number of cells is the square of this value. The computational time increases at the same rate for larger cell numbers regardless of the particle number. Doubling the cell number from 128 cells to 256 cells increases the computation time by a factor of about 17. The independence from particle number means that comparing to the timing of the treecode is difficult. For 300,000 particles and 256 cells per side, the time required for PIC is on the same order of time as for the treecode. But PIC could compute the force on ten times as many particles without an increase in time, or the mesh size could be reduced to give an order of magnitude lower time.

However, even at the finest mesh, the most accurate force computed by the PIC method has an average relative force difference of about 8% from the direct sum result for this domain. This is much poorer accuracy than the treecode with $\alpha = 0.16$, $n_t = 4$, and $m_{max} = 8$, which gives a relative difference of about 0.002%. This inaccuracy is a result of PIC's inability to resolve inter-particle forces.

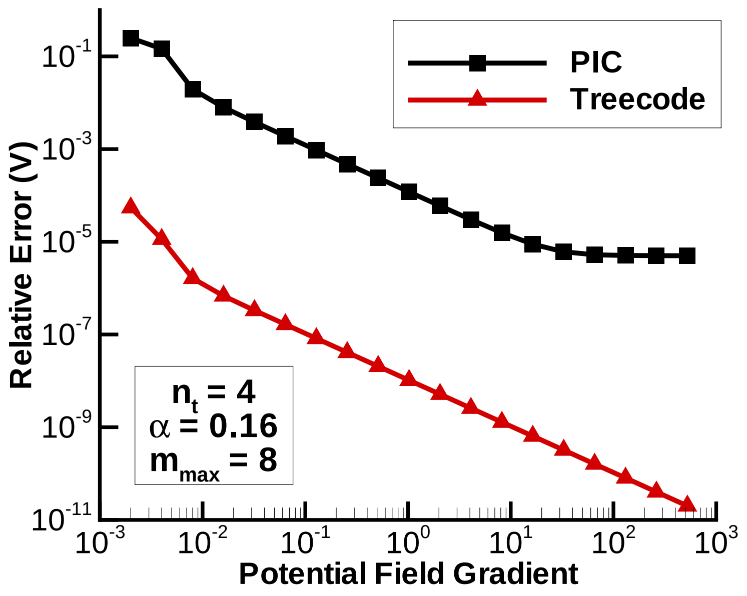


Figure 5.12: Average relative error in the X-force as a function of the imposed potential field gradient. Results for both PIC and the treecode are shown, with the direct summation result used as the exact force. The maximum space charge potential for the domain is approximately 0.002 V. The PIC error appears to level off for larger gradients, but this is actually due to error in the direct sum force computation.

If a potential gradient is imposed, then the relative PIC force error decreases rapidly as the imposed gradient becomes stronger than the space charge of the particles. Note that the error here is defined as the difference from the direct sum result with 128 panels per side, so there will be some error in the direct sum results themselves due to panel discretization error. This error can be reduced or eliminated by simply increasing the number of panels. Figure 5.12 shows this decrease in relative error as a function of potential field gradient in the X-direction, for both PIC and the treecode. Here, the left-hand boundary is held at 0 V, while the right hand boundary is held to the gradient potential. A linearly varying Dirichlet boundary condition is applied to the top and bottom of the domain for both PIC and the treecode. The PIC error decreases by a factor of two each time the potential gradient is doubled.

The error levels off for large potentials due to error in the panel method— increasing the number of panels decreases this error. In other words, the PIC result continues to have a lower relative error in the force computation as the field gradient increases, but the comparison to the direct sum result does not improve due to error in the direct sum itself.

The treecode error shows the same decrease in relative error as the PIC, although it begins at a much lower error level and so always has an accuracy four orders of magnitude higher. The treecode also does not level off for higher potential gradients because it also uses the panel method, and thus this error is not seen when comparing against direct sum. For both the PIC and the treecode, the average difference in forces compared to the direct sum does not change as the gradient changes. Since the forces are becoming larger due to the boundary conditions only, the inter-particle force error remains the same.

The results indicate that both the treecode and PIC are viable options depending on the domain of interest. For a plasma with weak imposed boundary conditions, the inter-particle force computation is important and the PIC method gives poor results compared to the treecode. But for a domain with a strong potential gradient such as ion optics, the PIC method will be more efficient than the treecode. In these domains, the inter-particle forces are not as important, whether because of Debye shielding or because of strong imposed boundary conditions.

5.3 2-Dimensional Axisymmetric Results

The axisymmetric domain presents several new problems for the treecode. One of the most serious is the lack of a known recursion relation for computing the derivatives of the Green's function. This seriously limits the efficiency of the treecode, as a limited number of terms in the Taylor expansion can only provide a limited amount of accuracy. Further reduction in error can only be obtained by decreasing the acceptance parameter, which is computationally expensive.

Other problems include an inability to directly integrate the boundary integrals and serious singularities in the panel method, as described in the following sections.

5.3.1 Axisymmetric Boundary Panels

Issues relating to the use of panels in the axisymmetric domain are discussed in this section. This includes the use of radial panel charge scaling, the Gaussian Quadrature numerical integration method, Chebyshev distribution of panels, and handling of the near-boundary singularity.

The domain used for all cases in this section is a cylinder 1.0 m high and 1.0 m in radius. The potential is set to 100 V on the left side of the domain, 200 V on the right side, and a 0 V/m Neumann condition is set on the upper boundary. The number of panels per side and Gaussian Quadrature points varies depending on the study being performed. Note that, because the domain is axisymmetric, there is no actual boundary on the centerline, and thus no panels are placed on that side of the domain.

5.3.1.1 Radial Scaling of Panel Charge

Consider a random distribution of many particles in an axisymmetric domain. If a constant charge density is desired in the field, then the total charge on each particle should decrease as a function of radius in order to maintain a constant density field. This applies to panels as well: the panel charge will decrease with r^2 towards the centerline for a constant potential domain. However, the use of constant charge on vertical panels means that integration points below the panel centroid will have a

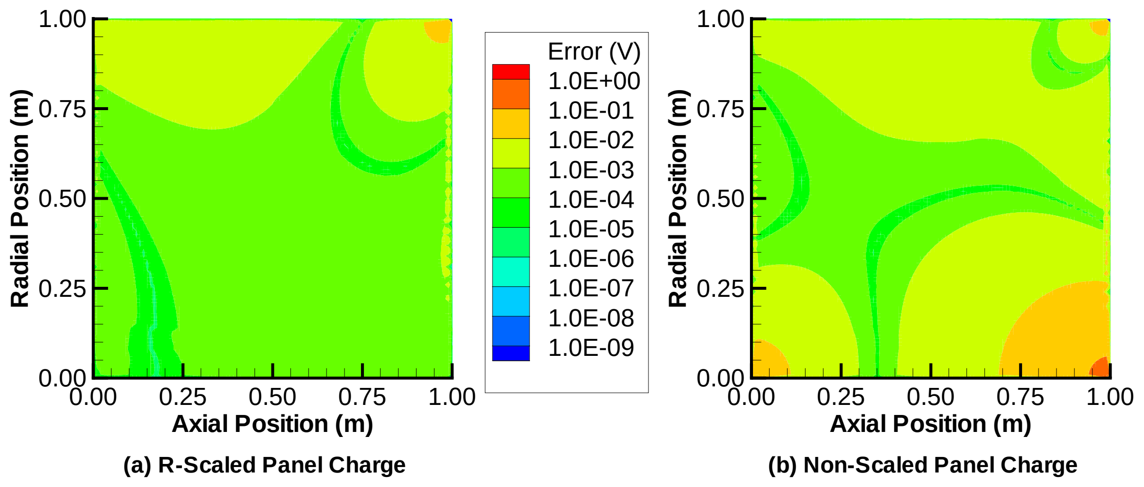


Figure 5.13: Potential field error with r-scaled panel charge and without. 100 panels per side were used here, and 16 Gaussian Quadrature points. The r-scaled panel charge greatly reduces the error near the panels on the centerline, although the error increases slightly at the upper domain corners.

charge that is too high, while points above the centroid will have too little charge. One way to approximate this effect is to use a linear charge on each panel, such that the charge varies smoothly with radius and between panels. This adds computational cost however, as twice as many parameters must be found for each panel.

A simpler method is used instead— a linear charge is still assumed, but the slope of the line is fixed on each panel as the inverse of the panel midpoint. Thus, the charge on each integration point is simply multiplied by the radius of the point divided by the panel's midpoint. Below the midpoint the charge will decrease, at the midpoint the charge will be the same, and above the midpoint the charge is increased. Figure 5.13 compares a potential field with this r-scaling against a field with a constant charge on each panel. The error on the right side of the domain is higher because the potential on this side of the domain is 200 V, as opposed to the 100 V potential on the left side. Although the error near the upper corners is

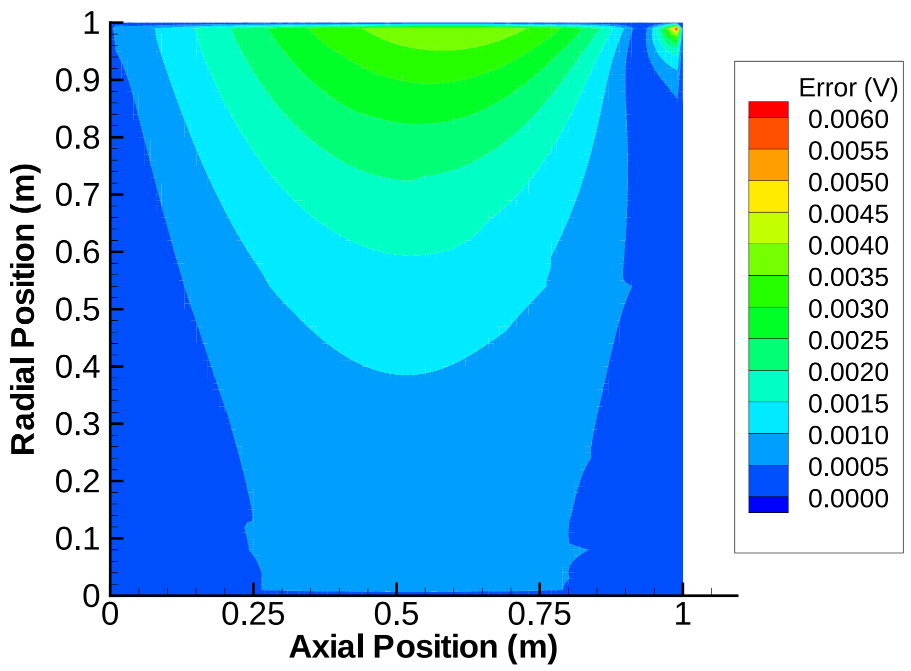


Figure 5.14: Potential field error for 8 Gaussian Quadrature points and 256 panels per side. The error in the upper-right-hand corner is caused by panel discretization error, while the error in most of the domain is due to integration error.

slightly higher in the scaled cases, the error at the centerline is up to three orders of magnitude lower. This is a clear advantage over the constant charge distribution, and so the r-scaling method is used in all subsequent results.

5.3.1.2 Gaussian Quadrature

The boundary panel method in an axisymmetric domain requires numerical computation of the boundary integrals. This is performed using Gaussian Quadrature (GQ), which gives accurate results for a small number of integration points. Ideally, the accuracy of the integration should match the accuracy of the panel discretization. Panel discretization error is the error caused by the use of finite length panels with constant charge density. The two types of error can be seen in Figure 5.14. Integration error causes the large area of error at the top center of the domain, while panel discretization error causes the small, high-error, region near the upper-right-hand corner. Again, the error in this corner is higher because the potential is at 200 V on the right side of the domain.

At the upper corners of the domain, a horizontal Neumann panel is close to a vertical Dirichlet panel. Thus, there is a discontinuity in panel strength at the corner, resulting in a region of high error. The discontinuity is made slightly worse by the r-scaling, because the integration point charge is increased above the panel midpoint, and the discontinuity is then larger than that without r-scaling. The integration error appears everywhere in the domain because it is directly caused by inaccuracy of the numerical integration performed over each panel. Thus, computing the potential at any point in the domain has this error associated with each panel potential contribution calculation.

Figure 5.15 plots the average and maximum error in potential as a function of the number of Gaussian Quadrature integration points on each panel. The maximum error in the domain does not decrease for more than 8 GQ points. This is because the panel discretization error dominates when more accurate integrals are performed. The average error continues to drop as the solution becomes more accurate in most parts of the domain.

5.3.1.3 Panel Distribution

The panels on each boundary do not need to have uniform lengths— in fact more accurate results may be obtained if the panel lengths are varied in a specific way. Chebyshev polynomials are used to give near-minimax approximations to functions (*Atkinson* (1993)). Such approximations seek to reduce the maximum error while maintaining or reducing the average error. The Chebyshev points are distributed in

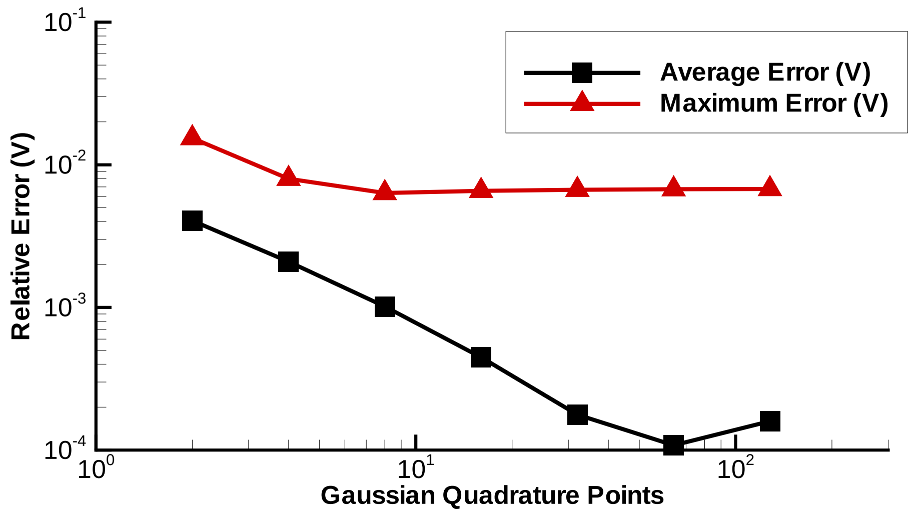


Figure 5.15: Average and maximum potential field error as a function of the number of Gaussian Quadrature points. The maximum error does not decrease for more than 8 GQ points because panel discretization error dominates. 256 panels per side are used in all cases.

the range [-1,1], by the equation

$$x_i = \cos\left(\frac{2i+1}{2n+2}\pi\right), \qquad i = 0, 1, \ldots, n \tag{5.14}$$

where n is the number of intervals in the distribution. The distribution spaces points evenly along the circumference of a unit circle, as shown in Figure 5.16. The x-location of each point on the unit circle is used as a function approximation point. This results in points being closer to each other near the ends of the interval.

These points can be used to distribute the panels along a given boundary. The Chebyshev points, along with the endpoints of the domain, give the endpoints of the panels. Thus, n will be equal to the number of panels on the boundary minus 2. To convert to the boundary coordinates, a linear transformation is performed:

$$\hat{x}_i = \frac{L}{2}\left[1-\cos\left(\frac{2j-1}{2N_p-2}\pi\right)\right], \qquad j = 1, 2, \ldots, N_p - 1 \tag{5.15}$$

$$\hat{x}_0 = 0 \tag{5.16}$$

$$\hat{x}_{N_p} = L\,. \tag{5.17}$$

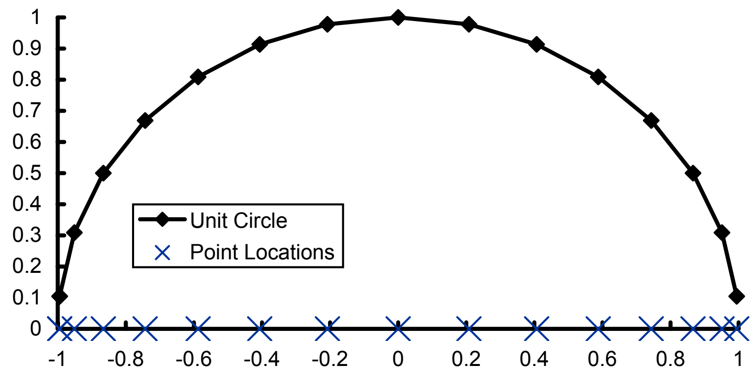


Figure 5.16: 16-point Chebyshev distribution along the unit circle and on the axis.

Here L is the length of the boundary and N_p is the number of panels on that boundary. In this system, the j^{th} panel begins at $\hat{x}_{j-1}$ and ends at $\hat{x}_j$.

The maximum and average error as a function of the number of panels per side is plotted in Figure 5.17. Values are shown for both uniform panel lengths and Chebyshev distributed panels. 64 Gaussian Quadrature points are used for both data sets, and the boundary conditions are the same as in Section 5.3.1.2. The Chebyshev distributed panels consistently have a lower maximum error than the uniformly distributed panels. The uniform distribution gives a lower average error for cases with fewer panels, although the Chebyshev distribution has lower average error when more panels are used. Although it is preferable to always have both the average and maximum errors reduced by the Chebyshev distribution, the reduction in maximum error is significant enough to make use of Chebyshev distributed panels exclusively.

A contour plot comparison of the error between the two methods is shown in Figure 5.18. The larger error in the upper-right-hand corner is characteristic of the uniform distribution. The Chebyshev distribution results in a higher density of panels towards the corners of the domain, where the gradients in panel charge are

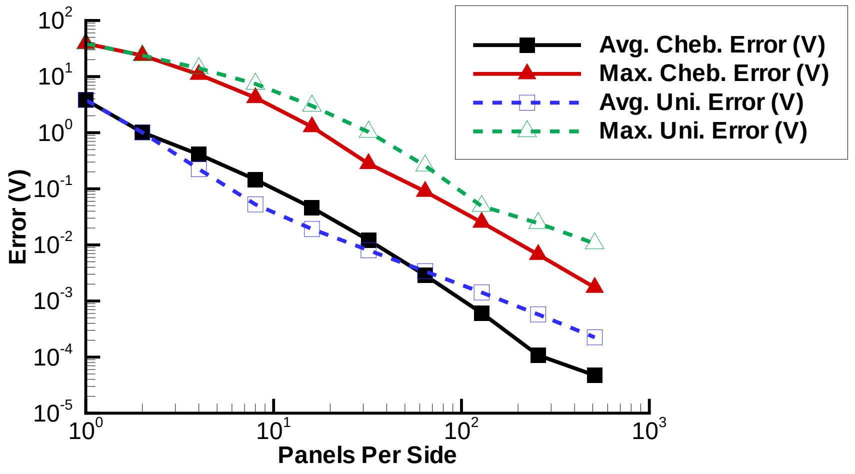


Figure 5.17: Error in the potential field calculation for both Chebyshev and uniformly distributed panels. Both maximum and average error are shown, with 64 GQ integration points.

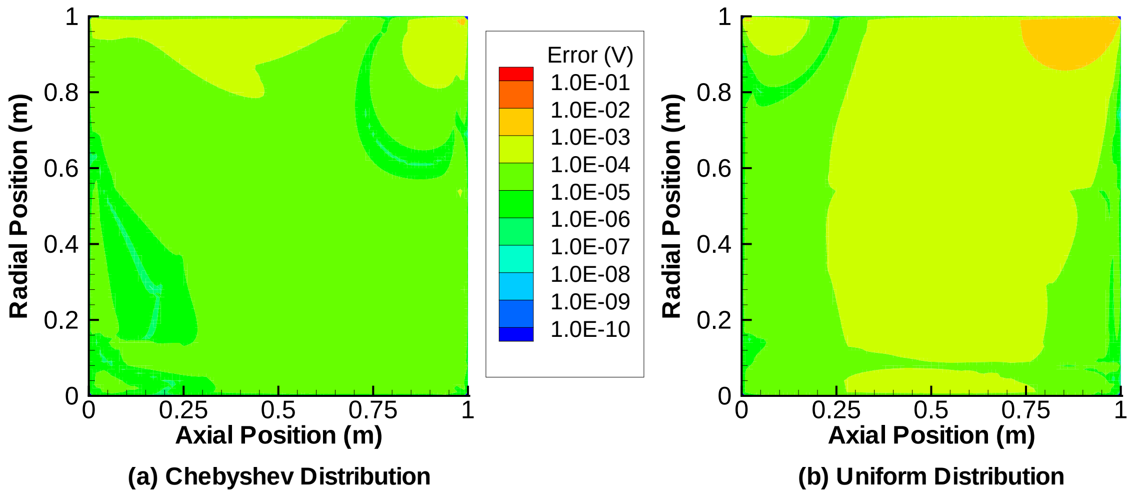


Figure 5.18: Error field comparison between (a) Chebyshev and (b) uniformly distributed panels. 512 panels per side are used here, and 64 GQ points.

the highest. This is what causes the reduction in the error in the corner regions, although the error in the center regions may be slightly higher.

One additional note is that the use of the Chebyshev distribution means that the number of panels that may be used on a boundary is limited, depending on the number of Gaussian Quadrature points. When computing a panel's self-effect integral, the parameter m defined in Section 3.6.1 must be less than one, or else the Green's function is infinite. For computing the self-effect on a vertical panel, m may be re-written like so:

$$m = 1 - \frac{\Delta r^2}{(2r_m + \Delta r)^2} \ . \tag{5.18}$$

Here, r_m is the panel midpoint radius and Δr is the distance between the midpoint and an integration point. If the second term is smaller than the machine precision, then m will be one and the integral will be singular. If there is a large number of panels on a boundary, then the shortest Chebyshev panel will have Gaussian Quadrature points that are very close to the panel midpoint, resulting in a very small Δr^2.

The uniform distribution has panels of constant length, so a very large number of them may be placed on a boundary before a problem occurs. For example, on a 1.0 m boundary with 64 GQ points, the most Chebyshev panels that may be used is 710. For this number of panels, the smallest panel length is approximately $1.23 \cdot 10^{-6} m$. The distance between the nearest GQ point and the midpoint is $1.494 \cdot 10^{-8} m$, so the square of this distance will be very close to the machine precision. For uniform panels, about $8.13{\cdot}10^5$ panels are required to reach this same limit. In general practice, large numbers of panels are not required or desired, so this problem does not occur often.

5.3.1.4 Near-Boundary Singularity Handling

The Green's function in any domain becomes singular as a target approaches a source. In the case of the panel method in the axisymmetric domain, the target is a charged particle in the domain, and the source is a Gaussian Quadrature point on a panel. If the particle is actually on the boundary, the numerical integration over the panel may be performed in a way that takes account of the singularity. For particles that are close to the boundary however, the problem is nearly singular, and is thus more difficult to address. The use of numerical integration means that the exact nature of the singularity is not captured, and the efficiency of methods such as Gaussian Quadrature decreases.

This problem is a well-known one in the boundary integral method, and several schemes can be found for mitigating error. One possibility is to more accurately perform the boundary integral using a quadrature method designed for the Green's function (*Sidi and Israeli* (1988) and *Nitsche* (1999)). Another method is to offset the boundary panels such that they are further from the physical boundary, and thus particles are not able to approach the nearly-singular region (*Liu et al.* (1993)). The singularity may also be regularized and corrected to obtain a more accurate solution (*Beale and Lai* (2001)). One other method is to increase the number of integration points on a panel as a particle approaches that panel. This was implemented previously, and did give satisfactory results in a 2-D domain.

In the interest of simplicity, a much more basic solution is used in the current model. For domains such as ion optics, the near-boundary forces are often either insignificant or unimportant. The Neumann condition on much of the boundary sets the force to zero in one direction, for example. Also, particles rarely approach the ion optics. The method used for near-boundary force computation is to simply offset the particle's position from the boundary for the purposes of calculating the boundary interaction. The particle is not actually moved, and the forces from other particles are calculated normally. If the particle is less than a distance δ from the boundary, then the position is offset to be equal to δ.

The distance δ is set according to the average panel length for each boundary. An offsetting factor β determines the fraction of the average panel length to use as δ. The optimal value of β will decrease the error due to the singularity as much as possible, but also will not increase the error by offsetting too far from the panel.

A line of potential is computed by varying the axial position in the domain with constant radial position. The radius of the Gaussian quadrature point closest to the midpoint of the panel closest to the domain midpoint gives the position of the potential line. This ensures that the line of potential will have the worst behavior near the wall, while not including effects from the upper boundary. An example plot of the error, with and without an offset, is shown in Figure 5.19, for a domain with 32 panels per side, and 8 GQ points. The error peaks and decreases sharply, especially near the wall, due to the integration error of the panels. The effect of the offset parameter is also seen, as the error peak is two to three orders of magnitude lower near the walls in the offset case.

The line of potential is calculated for varying β, and each line is evaluated for its maximum and average errors. In Figure 5.20, the maximum error in the potential is plotted as a function of the offset parameter and the number of Gaussian Quadrature points. The number of panels used here is 32 per side. The minimum point for the maximum error changes with the number of Gaussian Quadrature points, decreasing as the number of points increases. The minimum point for each case is a cusp where the error from the singularity has dropped to the same level as the error being

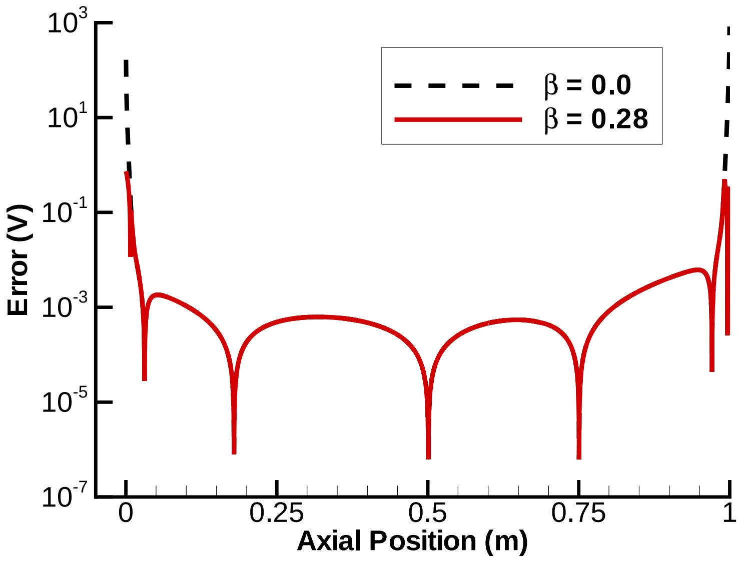


Figure 5.19: Line plot of the potential field error as a function of axial position. Error with and without the offset parameter is shown. This case has 32 panels per side, 8 GQ points, and $\beta = 0.28$ for the offset data.

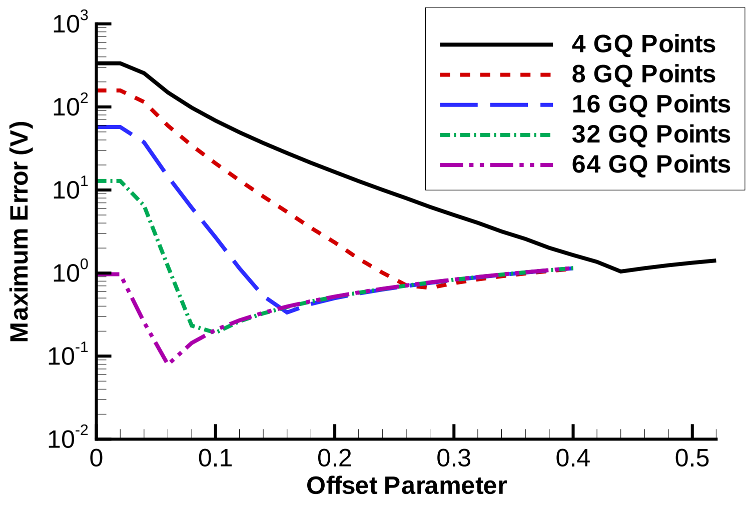


Figure 5.20: Maximum error in potential as a function of offset parameter and number of Gaussian Quadrature points. There is a cusp minimum point for all numbers of GQ points, and this minimum's location approaches the boundary as more GQ points are used.

caused by the offset factor. Further increasing the offset past this point increases the error. The offset factor which produces the minimum error can be approximated by a simple power law function of the number of GQ points. However, this function changes slightly with the number of panels.

5.3.2 Application of the Treecode to Axisymmetric Panels

The treecode may be used to evaluate the panel contributions to forces and potentials, as described in Section 3.4.2. This is important for fast evaluation of the panel forces in any domain, especially given the higher number of panels and points required in order to mitigate the error introduced by the panel method. The parameter variation cases below use 512 panels per side and 16 GQ points, in the 100 V-200 V gradient domain used previously. Base values for the treecode parameters are as follows: 3 Taylor expansion terms, $\alpha = 0.08$, and $m_{max} = 8$.

Figure 5.21 plots the average and maximum potential field error as a function of the number of terms in the Taylor expansion. The average error drops in a stair-step fashion due to the moments of the tree clusters for panels. All panel integration points on a vertical boundary will have the same axial position, so the z-moments of clusters for this boundary will always be zero. Thus, any Taylor derivative involving a differentiation with respect to z will be multiplied by zero. Also, the panels on any boundary are distributed symmetrically, so the cluster moments of odd power will be zero or close to zero in the r-direction for vertical boundaries. The same is true for z-direction moments on horizontal boundaries. The end result is that an improvement in accuracy of two orders of magnitude can be expected for every two Taylor expansion terms, rather than one order of magnitude improvement for one term.

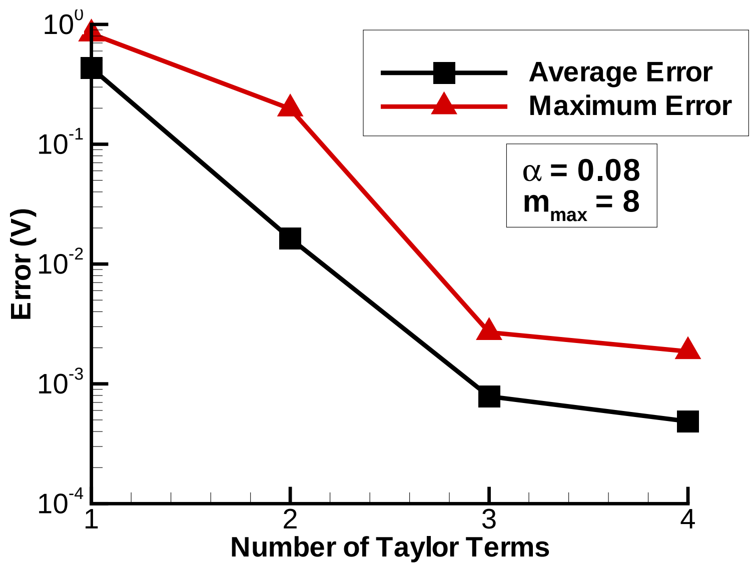


Figure 5.21: Average and maximum error in potential as a function of n_t. The accuracy decreases approximately two orders of magnitude for every two terms due to the structure of the panel tree clusters.

The variation in error as a function of acceptance parameter is shown in Figure 5.22. The error drops roughly by an order of magnitude each time the parameter is halved for $\alpha > 0.08$. For $\alpha < 0.08$, the panel method error is reached, limiting the possible accuracy. An acceptance parameter of 0.08 then gives the most accuracy possible for these domain and panel parameters.

The potential field error due to the boundary treecode does not vary with the parameter m_{max}, matching the result seen previously in 2-D. The computation time is then the only consideration when choosing what value of the parameter to use. Figure 5.23 plots the computational time as a function of m_{max}. As in the 2-D case, there is a minimum point for each set of results. The range of m_{max} values producing the minima is approximately the same as well— approximately 8 to 16. For $\alpha = 0.01$, the minimum is at $m_{max} = 4$, because fewer clusters are accepted at a low α. It is more important that the clusters be small enough for some to be accepted.

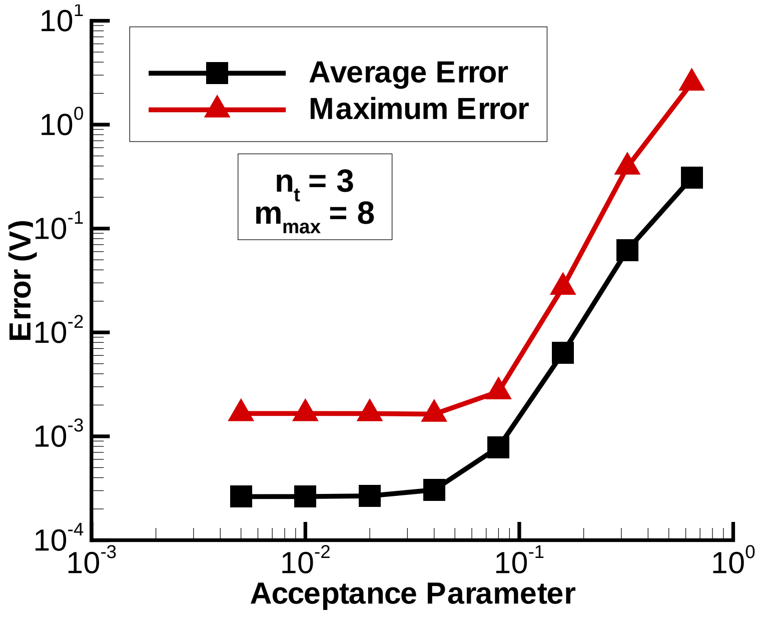


Figure 5.22: Average and maximum error in potential as a function of α. At an acceptance parameter of 0.08, the remaining error is due to the boundary panel approximation.

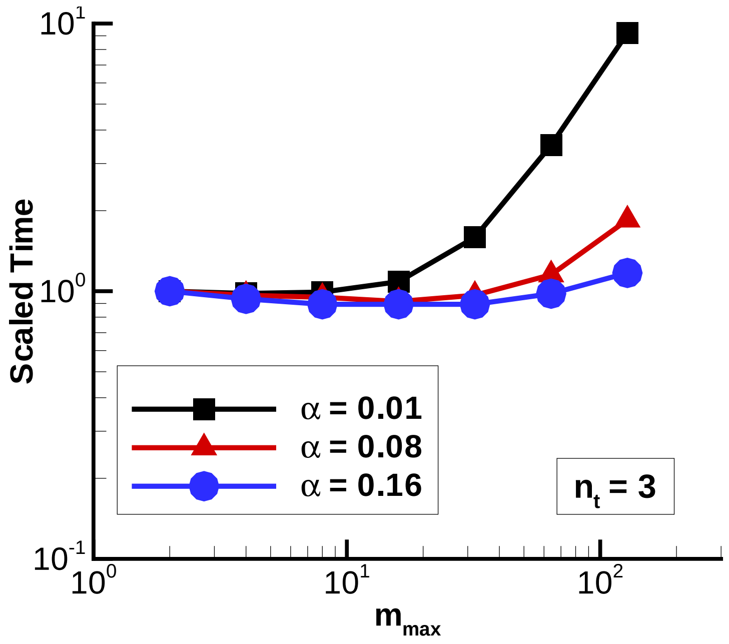


Figure 5.23: Scaled variation in computational time as a function of m_{max} and α. Each value of α has a value of m_{max} that produces the minimum time.

The results indicate that the best overall strategy is to use 4 Taylor terms to give the greatest accuracy possible, set α equal to about 0.08, and use 8 as the value of m_{max}.

5.3.3 Axisymmetric Particle Results

The treecode is next applied to particles in an axisymmetric domain. The cylinder domain used previously is once again applied, although panel forces are not included in the force computation. Particles are distributed randomly in the domain, with a varying weight depending on the particle's radius. The particle weight is varied such that the density of the domain is constant at $1 \cdot 10^{10}\ m^{-3}$. The base treecode parameters are $n_t = 3$, $m_{max} = 8$, and $\alpha = 0.08$.

The change in error as a function of each parameter is plotted in Figure 5.24. The results are very similar to the 2D domain in almost every case, with the sole exception of the radial force error as a function of the number of Taylor expansion terms. In this case, the stair-stepped behavior seen for the panel treecode appears once again. The reason for this behavior in the particle treecode is unknown. In the panel case, it may be attributed to cancelling moment terms, but here the particle distribution is random, so there is no apparent reason why the error should not decrease with every new term.

5.4 Comparison to PIC Ion Optics Results

The two-dimensional and axisymmetric version of the treecode algorithm may be applied to the ion optics domain. In this domain, the treecode is applied to a more complex problem than in the previous test domains. The treecode and direct summation results can be compared to PIC results for this domain.

The particles simulated are not truly particles, but are macro-particles repre-

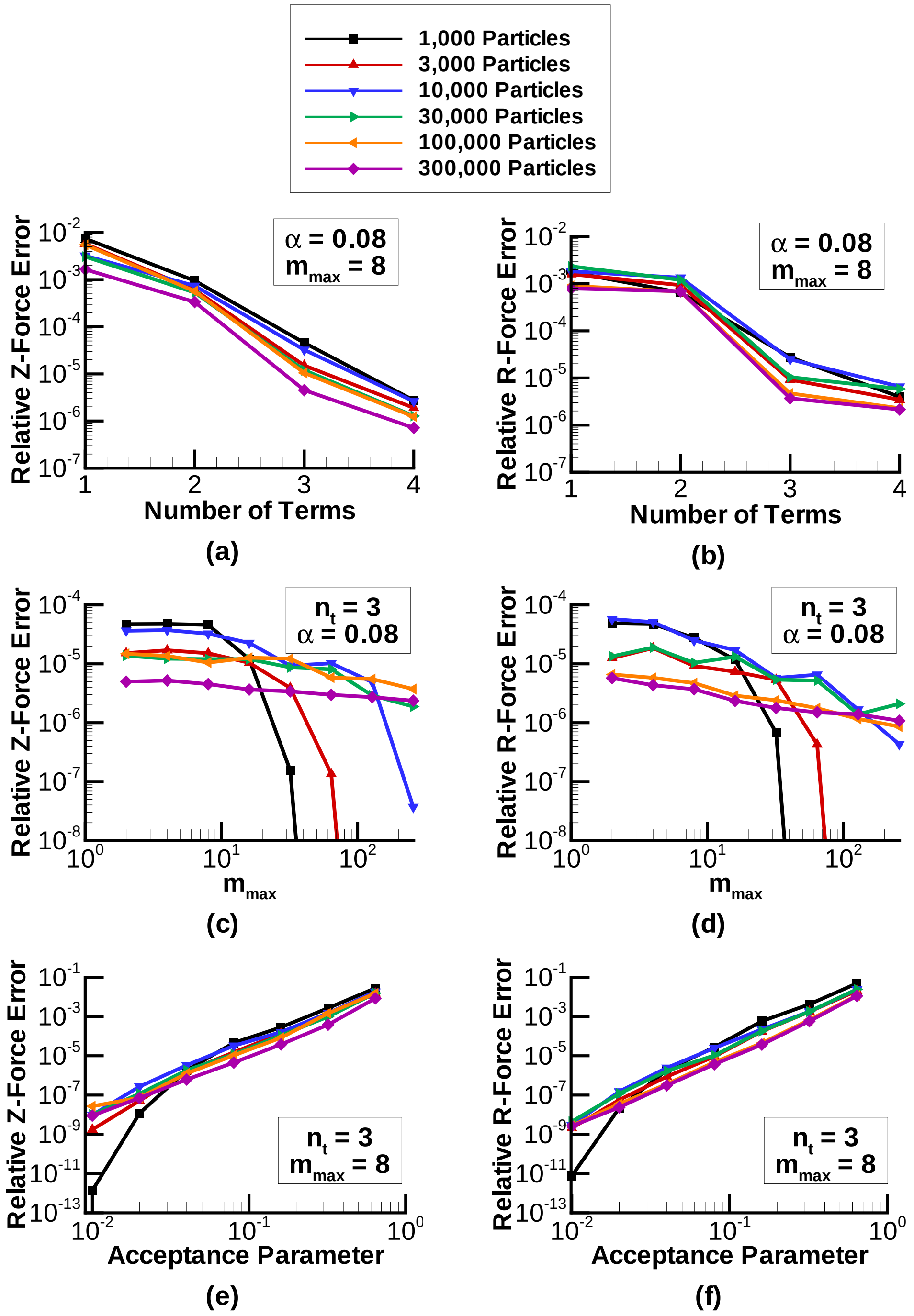


Figure 5.24: Average relative particle force error for varying treecode parameters. (a) and (b) show the Z-force and R-force error for a varying number of Taylor expansion terms. (c) and (d) plot the error against the maximum number of particles per cluster, and (e) and (f) plot the Z and R force error for varying acceptance parameter. The base values for each parameter are $n_t = 3$, $m_{max} = 8$, and $\alpha = 0.08$.

senting a large number of true particles. The macro-particles are more accurately represented as a cloud of charged particles, rather than as the point charges considered to this point. If instead the macro-particles are assumed to be clouds of charged particles, the force is damped. In order to do this, a smoothing parameter is required in the treecode and direct sum. This parameter is a constant addition to the distance between macro-particles, such that even when two particles are very close, the force computation will see a larger distance, and decrease the resulting force accordingly. The axisymmetric Green's function becomes

$$
\begin{aligned}
G(\vec{x}_0, \vec{x}_s) &= \frac{K(m)}{\pi\sqrt{L}} && (5.19)\\
L &= (r_0 + r_s)^2 + (z_0 - z_s)^2 + \delta^2 && (5.20)\\
m &= \frac{4 r_0 r_s}{L}\ . && (5.21)
\end{aligned}
$$

For a finite smoothing parameter δ, m cannot go to 1, even when $\vec{x}_0 = \vec{x}_s$. This prevents the singular evaluation of $K(1)$. Note that this parameter is not used for particle-panel interactions, which would cause instability in the panel strength computation.

One problem that arises is the inclusion of an electron population for the direct summation and treecode. The PIC potential solver models a Boltzmann electron fluid at the mesh nodes, but the treecode has no mesh for such a simulation. Modeling the electrons as particles is computationally intractable, as this requires both a much smaller time-step and a larger number of particles. For the purpose of comparing treecode and PIC results, the electron density field generated by the PIC solution is used by the treecode to compute the potential field.

For these reasons, the focus is placed on differences between the treecode and PIC results, rather than the absolute accuracy of either. In this study, the error will be considered as the difference between the treecode or direct sum and the PIC. The axisymmetric treecode is computationally inefficient due to complex derivatives of the Green's function and a lack of a recursion relation for those derivatives. The computation time for the treecode is at least an order of magnitude higher than the PIC computation time in this domain. Also, the PIC potential computation is performed in order to obtain the electron contribution used by the treecode. As such, no timing comparisons are made between PIC and the treecode.

The domain simulated is the same as is used for the mesh refinement study in Section 4.2. In this domain, the boundaries are simulated exactly by PIC, which is required in order to obtain a valid comparison to the treecode. The PIC simulation beamlet current is approximately 0.104 mA, with a discharge ion density of $1.32 \cdot 10^{17}\ m^{-3}$. For comparison to the treecode and direct sum, particle positions after the final iteration of the PIC simulation are used for computing the forces and fields of the domain. Approximately 77,000 ion particles are in the domain— these particles are not advanced in time in this study. The PIC potential field solver has a refinement of 8x, giving a potential mesh size of $6.25 \cdot 10^{-6}$ m, with 1280 by 160 cells.

The electron number density may be calculated in the center of each cell on this mesh, based on the PIC potential field calculation. This is done by averaging the potential at the four cell nodes, then applying the Boltzmann relation to give the density at the cell center. The treecode simulates this electron density as a population of static electron particles. The particle weight is determined by multiplying the density by the cell volume, and the position of each particle is simply the cell center. This method places approximately 200,000 electron particles in the domain. The PIC-calculated electron number density field is shown in Figure 5.25. Forces are not

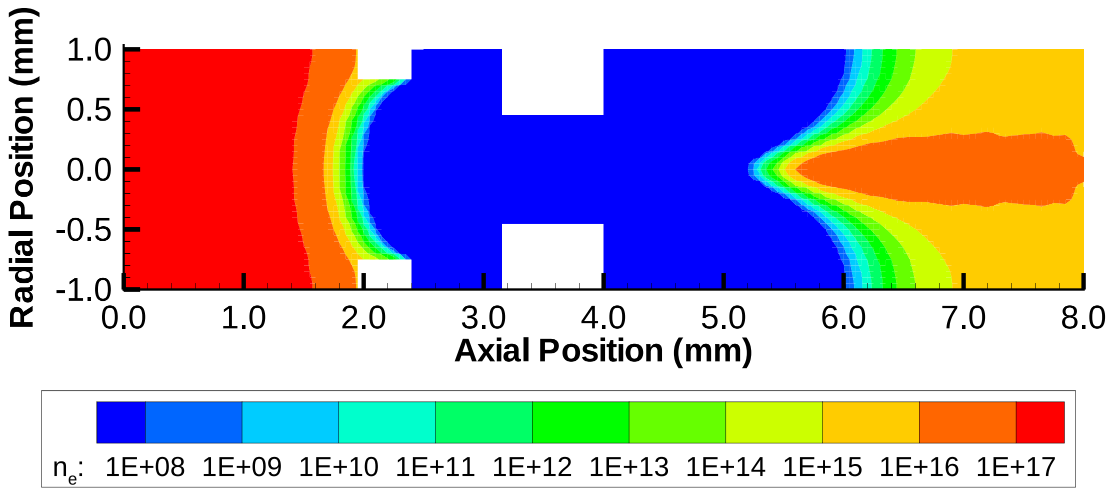


Figure 5.25: Electron number density field used in the PIC-treecode ion optics comparison cases. The density is in m^{-3}. This field is computed from the potential given by the PIC potential field solver.

computed on the electron particles in the treecode.

The base treecode and direct sum domain parameters are set as follows: 50 panels on each boundary, 64 Gaussian Quadrature points per panel, $\delta = 2 \cdot 10^{-5}$ m, and $\beta = 1.0$. The treecode parameters for panel integration points are set to $n_t = 3$, $\alpha = 0.08$, and $m_{max} = 8$. For the particle treecode, the parameters are $n_t = 4$, $\alpha = 0.16$, and $m_{max} = 8$. These values are assumed in each case, unless specified otherwise.

5.4.1 Direct Summation Results

It is important to compare direct summation results to PIC in order to determine a baseline for treecode comparison. The smoothing parameter is equal to 4 $\cdot 10^{-5}$ m in this case. In Figure 5.26, the relative error in the particle force is plotted in the domain. The discharge chamber and downstream region of the domain have much higher errors than the central region. In the central region, the imposed electric

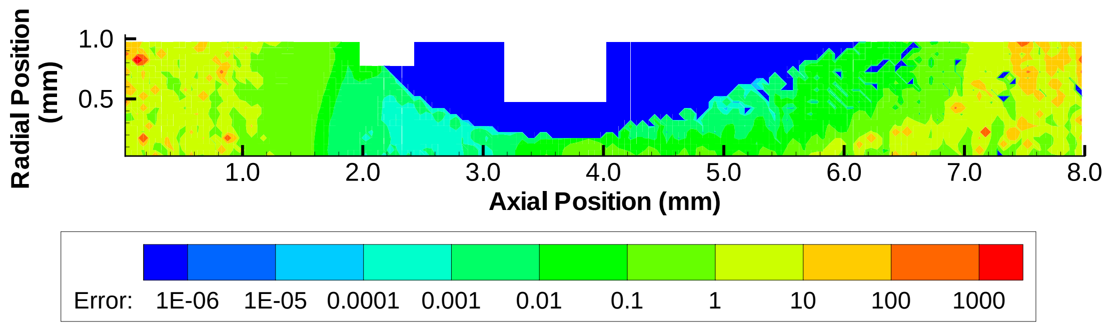


Figure 5.26: Relative z-force error between PIC and direct summation. The relative error is defined as the difference between the PIC and direct summation force, divided by the PIC force. The smoothing parameter is set to $4 \cdot 10^{-5}$ m in this case.

field dominates, giving a small relative error. In the neutral regions of the domain, PIC computes very small forces, while direct summation still sees large inter-particle forces. The average PIC force magnitude in the upstream region is on the order of $1 \cdot 10^{-17}$ N, while the average direct sum force is approximately $1 \cdot 10^{-16}$ N. As such, similar particle forces between direct sum and PIC cannot be expected in these regions. One contribution to this difference is likely the electron population. In the PIC method, the electron population is a smooth background density, whereas in the treecode and direct sum, electrons are represented as point charges. This will result in incomplete or unbalanced Debye shielding of inter-particle forces.

The difference in potential field between PIC and direct sum is plotted in Figure 5.27. The region of error on the centerline near the accelerator grid is caused by the smoothing parameter. The space charge of the ion population is very high in this region, so the potential is affected more significantly by δ. Note that, since this higher potential is not seen by the PIC, the treecode will predict a higher beamlet divergence, and also will predict electron backstreaming earlier. The large region of error along the upper boundary downstream of the optics, and the very high error

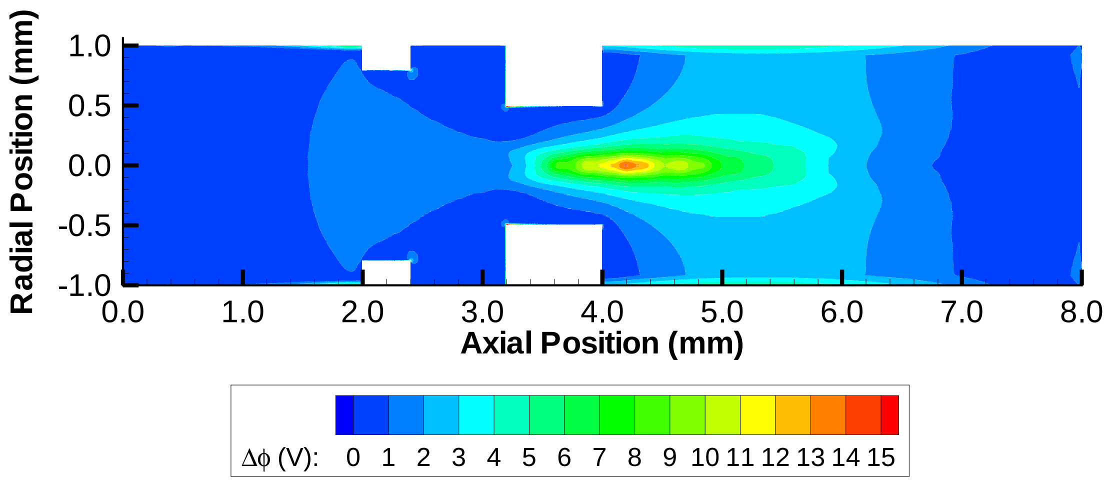


Figure 5.27: Difference in potential field calculation between PIC and direct summation. The region on the centerline is caused by the smoothing parameter, while the boundary offset causes the differences seen near the upper boundary and on the ion optics surfaces.

right on the grid boundaries is caused by the offset parameter β. Particles are not affected by this error, as they only approach the optics in this region due to CEX collisions, which are not simulated here. Elsewhere in the domain, the direct sum matches the PIC within one or two volts.

5.4.2 Smoothing Parameter Variation

The smoothing parameter δ can have a strong effect on the particle force computation. If the value is too small, then particles near each other will experience extremely large forces. If the value is too large, the particle force computation will not be accurate. The upstream and downstream regions of the domain are neutral, so δ should ideally be large enough to treat the electron particles as a density field. The average force error as a function of δ is plotted in Figure 5.28. The error has a minimum at $\delta = 2 \cdot 10^{-5}$ m, or approximately triple the distance between electron particles. This minimum is also seen in all other result comparisons, so this point is

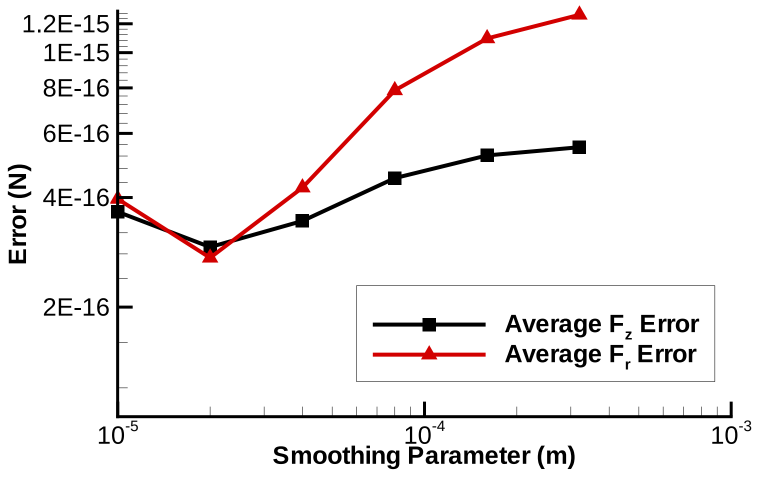


Figure 5.28: Average force difference between PIC and the treecode for varying smoothing parameter. The minimum is at $\delta = 2 \cdot 10^{-5}$ m for both the z-error and r-error.

used as the base value for δ.

5.4.3 Offset Parameter

In Section 5.3.1.4, the offset parameter β is discussed for a simple cylindrical domain. Results for this domain give an optimal value of about 0.06 for 64 Gaussian Quadrature points. However, in the ion optics domain, a higher offset parameter than this gives better results. The PIC method computes very low or zero electric fields near the Neumann boundaries, so it is important that the treecode does not give singular forces at these locations. It is preferable in this case to have inaccuracy due to particle offsetting than to have a nearly-singular force.

The variation in average particle force error as a function of β is shown in Figure 5.29. Here, the lower values of β give a very high error, which decreases steadily as the parameter increases. A minimum is reached at $\beta = 1.0$, after which the error increases again, especially for the z-force. The same behavior is seen in the maximum force error, although the maximum does not increase for $\beta > 1.0$.

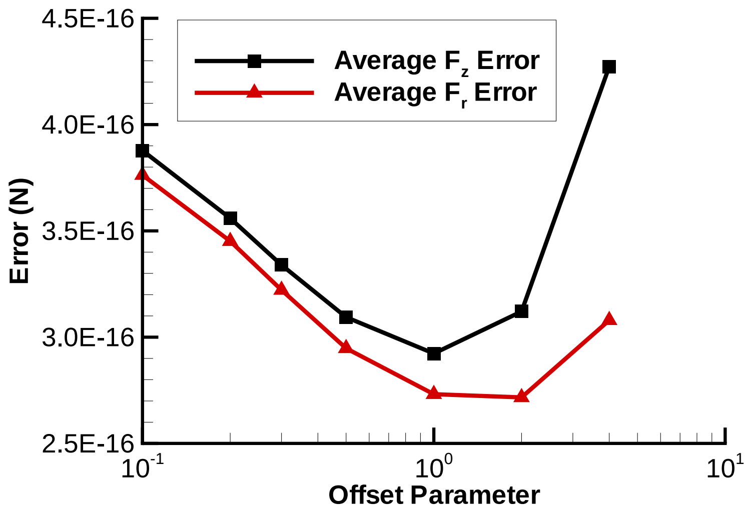


Figure 5.29: Average difference in PIC and treecode force for varying offset parameter. The minimum z-force difference occurs at a value of 1.0. This is also nearly a minimum for the r-force.

The disadvantage to this large offset is that the potential field calculation has a high error near the ion optics, as seen in Figure 5.27. However, ions rarely reach these region of high error, so the effect on the particle force computation is small.

5.4.4 Number of Panels

The treecode and direct summation results will become more accurate as the number of boundary panels increases. In the ion optics domain, the results are compared to PIC data, so a plateau in the differences should be reached once the treecode is at the same accuracy as the PIC. Figure 5.30 plots the average difference in particle forces as a function of the number of panels per boundary. The error drops rapidly as the panel number increases, reaching a plateau at 50 panels per side.

5.4.5 Discussion

The primary difference between the PIC result and the treecode is the treatment of inter-particle forces. The lack of such forces in PIC result in lower forces and

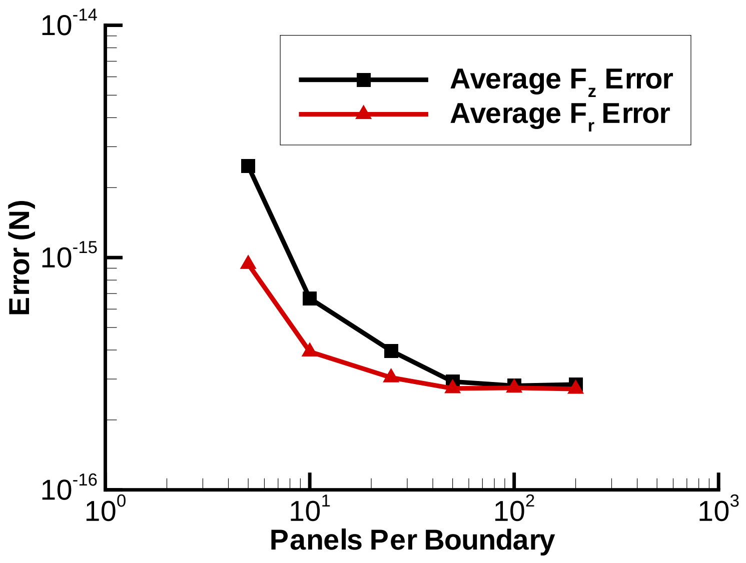


Figure 5.30: Average force difference between PIC and the treecode as the number of panels used on each boundary is varied. For higher than 50 panels per side, there is no decrease in the difference between the two methods.

potentials in regions of high ion density or neutral plasma. Boundary singularities also play a role in adding error to the treecode computation.

At this time, it is not feasible to apply a treecode to the ion optics domain. The primary reason for this is the much lower computational efficiency of the treecode compared to PIC. To compute the forces on the particles as performed above, the treecode requires about 90 seconds with the standard parameter values, whereas PIC computes the forces in 0.5 seconds. This is two orders of magnitude of difference in computational time, making even the simplest axisymmetric ion optics treecode simulation computationally intractable.

Although the treecode is able to more accurately simulate boundaries, singularities that accompany those boundaries further complicate the computation. The lack of ability to simulate the electron population accurately and easily also hinders the application of the treecode to the ion optics problem. Also, the ion optics simulation converges to a steady-state potential field. The PIC simulation takes advantage of this by only updating a previously calculated potential field, leading to lower computation cost as the field converges. The treecode does not do this— at each iteration the force is completely recalculated. This also adds to the computational cost for the treecode, such that even with a higher-order Taylor expansion the treecode would likely still require much more computational time than the PIC.

The conclusion is that the treecode is not suitable for axisymmetric ion optics simulation at this stage. Although more accurate inter-particle forces are computed with the treecode, these forces are not important in the ion optics domain, and any benefit in accuracy is more than offset by the poor efficiency of the treecode. The treecode's inability to easily model electrons as a fluid also detracts from its usefulness in ion optics simulation. However, in a 3-D domain, the treecode will likely perform much better, and it may be possible to produce results more accurately and for less computational work than PIC. Also, even in the axisymmetric domain, the treecode may provide insight into the accuracy of PIC's approximation of the ion optics geometry.

CHAPTER VI

Summary and Future Work

Ion optics simulations are presented in this thesis, with the goal of accurately modeling performance and erosion in the NEXT ion engine. The development and use of a treecode potential field solver is also explored as a possible method for improving the simulation of ion optics. The primary results of the preceding chapters are summarized here, and possibilities for future work are given.

6.1 Conclusions

This work shows that the use of an accurate and fast potential field solver is required in order to obtain useful Particle-In-Cell ion optics simulation results. The potential solver mesh must be refined enough to resolve the field gradients of the domain. Also, the domain itself must have the proper lengths in order to accurately model both the upstream sheath region and the downstream charge-exchange creation region.

PIC simulation of several ion optics apertures at different radii on the thruster surface provides a more accurate picture of thruster performance than simply using results from a single simulation. The current model is capable of accurately simulating performance of the NEXT ion engine. Accelerator grid barrel erosion is also modeled correctly, but discrepancies remain for current collected and downstream face erosion of the grid.

Simulation of electron backstreaming in the NEXT thruster via PIC is strongly dependent on the electron temperature in the downstream region. The model can predict the onset of electron backstreaming to within an acceptable amount of error. Three methods can be used to estimate the life of the thruster before the onset of electron backstreaming due to aperture enlargement. The use of a single simulation, multiple simulations, and dynamic erosion all predict that the NEXT ion engine will encounter electron backstreaming after approximately 40,000 hours of operation.

A treecode is applied in this work to bounded plasmas in several domains, which has been performed in the past by only *Christlieb et al.* (2004). A treecode potential field solver gives excellent comparison to Particle-In-Cell results for a 1-D sheath domain. In a 2-D box domain, the treecode can achieve optimal efficiency by using a large number of Taylor expansion terms, with a large acceptance parameter. The PIC method can still give faster results depending on the mesh size used, but the treecode will give more accurate particle forces.

This work describes the initial development of the treecode for application to an axisymmetric domain and plasma. The axisymmetric domain is problematic for the treecode, due to a lack of a complete recursion relation for derivatives of the Green's function. Also, the boundary element method encounters problems with charge distribution and singularities near boundaries. In an ion optics domain, the treecode does not currently simulate electrons, limiting its ability to compute forces in the domain accurately. These factors combine to indicate that the treecode is not currently appropriate for use in axisymmetric ion optics modeling.

6.2 Future Work

Several possibilities for areas of future research arise from the results presented in this thesis.

6.2.1 3-Dimensional Ion Optics Simulation

For both the PIC and treecode methods, a fully three-dimensional simulation is needed to provide a better ion optics model. Downstream face erosion of the accelerator grid cannot be modeled accurately without a 3-D domain, nor can other effects such as grid misalignment or dishing. The treecode will also be more efficient in a 3-D domain, as a full recursion relation exists for the Green's function.

The boundary element method may also provide more accurate results due to more easily addressed singularities. However, the 3-D domain means that the boundary elements will need to be planar rather than linear, introducing a large amount of computation into the boundary calculation.

6.2.2 Electron Fluid or Particle Modeling

The current electron fluid model used by the PIC method assumes that the electron fluid is thermalized, which may affect the formation of the ion extraction surface. Solution of mass, momentum, and energy conservation equations for the electron fluid would give a more accurate picture of the electron behavior and may improve the simulation significantly. This would also serve as an excellent test of the assumption of a Boltzmann electron fluid, which is used in many ion optics models.

An alternative is to use smoothed particle hydrodynamics (SPH) to model electrons. In this method, particles are simulated that represent a large cloud of electrons. This cloud has a velocity and density distribution that can vary as the particle moves through the domain. The advantage to this method is that the particle motion of the electrons can be captured without reducing the time-step, and the resulting density and velocity fields are much smoother than a normal particle distribution. Also, SPH is compatible with the treecode potential field solver. This may allow a fast and accurate treecode simulation in a 3-D domain.

6.2.3 Momentum-Exchange Collision Investigation

It is shown in Section 4.5.7 that, when the correct MEX collision cross-sections are used in the simulation, a large amount of abnormal erosion occurs. The root of this problem is most likely in the method used to process momentum-exchange collisions and the resulting particle velocities. It is not expected that MEX has a large effect on the simulation in general, but it is important to find the cause of the problem and resolve it in order to make the simulation as accurate as possible.

6.2.4 Non-Uniform Potential Solver Mesh

The mesh currently used in the PIC potential field solver is rectangular and uniform. However, it was shown in Section 4.2 that the mesh needs to be small only in the region near the ion optics. To provide a more efficient simulation, the mesh should only be refined in this part of the domain. Also, a coarser mesh can be used in the very long downstream region, as the potential field does not vary in this region. The potential solving step adds a significant amount of computation time to the simulation, and taking these steps could considerably shorten the running time without affecting accuracy.

6.2.5 Treecode Development

The treecode can be further developed in several areas. One possibility is the use of a more advanced cluster acceptance procedure, as detailed in Section 3.3. This would give better computational efficiency than the method currently used.

Another area of development would be the use of Gaussian Quadrature points designed for integration of Green's functions, as mentioned in Section 5.3.1.4. This would give more accurate results than the current use of an offset parameter.

The axisymmetric computation could also be made more computationally efficient by approximating the elliptic integrals via Bessel functions. This may lead to a complete recursion relation for the axisymmetric domain as well, further reducing computational cost.

APPENDICES

APPENDIX A

Nomenclature

Variables and physical constants that are used in the thesis are defined here.

Variables

z	$\cdots$	Axial Position $[m]$
r	$\cdots$	Distance from Centerline $[m]$
θ	$\cdots$	Out-of-Plane Angle $[radians]$
n_i	$\cdots$	Ion Plasma Density $\left[\frac{\#}{m^3}\right]$
n_e	$\cdots$	Electron Plasma Density $\left[\frac{\#}{m^3}\right]$
n_{Xe}	$\cdots$	Neutral Density $\left[\frac{\#}{m^3}\right]$
n_0	$\cdots$	Reference Plasma Density $\left[\frac{\#}{m^3}\right]$
ρ	$\cdots$	Charge Density $\left[\frac{C}{m^3}\right]$
T_e	$\cdots$	Electron Temperature $[eV]$
$T_{e,0}$	$\cdots$	Reference Electron Temperature $[eV]$
j	$\cdots$	Current Density $\left[\frac{A}{m^2}\right]$
$\omega_{p,i0}$	$\cdots$	Reference Plasma Frequency $\left[\frac{1}{s}\right]$
$\lambda_{D,0}$	$\cdots$	Reference Debye Length $[m]$

m_i	$\cdots$	Ion Mass $[kg]$
q	$\cdots$	Particle Charge $[C]$
ϕ	$\cdots$	Electrostatic Potential $[V]$
$\vec{E}$	$\cdots$	Electric Field $\left[\frac{V}{m}\right]$
Δt	$\cdots$	Time-step $[s]$
$\vec{u_i}$	$\cdots$	Ion Velocity $\left[\frac{m}{s}\right]$
$\dot{m}$	$\cdots$	Propellant Mass Flow Rate $\left[\frac{kg}{s}\right]$
I_{sp}	$\cdots$	Specific Impulse $[s]$
I_{beam}	$\cdots$	Beam Current $[A]$
$\sigma_{Xe^+,Xe}$	$\cdots$	$Xe^+ - Xe$ Charge-Exchange Collision Cross-Section $[m^2]$
$\sigma_{Xe^{++},Xe}$	$\cdots$	$Xe^{++} - Xe$ Charge-Exchange Collision Cross-Section $[m^2]$
$\vec{x}_0$	$\cdots$	Force and Potential Computation Reference Point
$\vec{x}_s$	$\cdots$	Force and Potential Computation Source Point
$K(m)$	$\cdots$	Complete Elliptic Integral of the First Kind
$E(m)$	$\cdots$	Complete Elliptic Integral of the Second Kind
n_t	$\cdots$	Number of terms in the treecode Taylor Expansion
α	$\cdots$	Treecode acceptance parameter
m_{max}	$\cdots$	Maximum number of particles per lowest-level cluster

Constants

$e = 1.602 \cdot 10^{-19}\,[C]$	$\cdots$	Electron Charge
$m_{Xe} = 2.18 \cdot 10^{-25}\,[kg]$	$\cdots$	Xenon Mass
$m_e = 9.11 \cdot 10^{-31}\,[kg]$	$\cdots$	Electron Mass
$\varepsilon_0 = 8.85 \cdot 10^{-12}\,\left[\frac{F}{m}\right]$	$\cdots$	Permittivity of Free Space

APPENDIX B

Axisymmetric Derivatives of the Green's Function

Presented in the following are the derivatives of the axisymmetric Green's function used in the tree-code Taylor expansion. The derivatives are shown with respect to $z = z_1 - z_2$, r_1 and r_2, where the point $\vec{x}_1$ is the reference point and $\vec{x}_2$ is the source point. The derivative with respect to z_2 can be obtained from the z derivatives by multiplying by $(-1)^n$, where n is the order of the z_2 derivative desired. For example, $\partial^3 G/\partial z_2^2 \partial z_1 = (-1)^2 \partial^3 G/\partial z^3$.

Define the following:

$$z = z_1 - z_2 \tag{B.1}$$

$$r = r_1 + r_2 \tag{B.2}$$

$$L = r^2 + z^2 \tag{B.3}$$

$$m = \frac{4r_1 r_2}{L} \tag{B.4}$$

$$w = 1 - m \tag{B.5}$$

$$K(m) = \int_0^{\pi/2} [1 - m \sin^2\theta]^{-1/2} d\theta \tag{B.6}$$

$$E(m) = \int_0^{\pi/2} [1 - m \sin^2\theta]^{1/2} d\theta \tag{B.7}$$

$$D(m) = \frac{E(m)}{w} \tag{B.8}$$

$K(m)$ and $E(m)$ are the complete elliptic integrals of the first and second kinds, respectively.

The 2-D axisymmetric free-space Green's function is

$$G = \frac{K(m)}{\sqrt{L}} \tag{B.9}$$

First Order Derivatives:

$$\frac{\partial G}{\partial z} = -\frac{zD(m)}{L^{3/2}} \tag{B.10}$$

$$\frac{\partial G}{\partial r_1} = \frac{(L - 2rr_1)D(m) - LK(m)}{2r_1 L^{3/2}} \tag{B.11}$$

$$\frac{\partial G}{\partial r_2} = \frac{(L - 2rr_2)D(m) - LK(m)}{2r_2 L^{3/2}} \tag{B.12}$$

Second Order Derivatives:

$$\frac{\partial^2 G}{\partial z^2} = -\frac{D(m)[Lw - 2z^2(2-m)] + z^2 K(m)}{wL^{5/2}} \tag{B.13}$$

$$\begin{aligned}\frac{\partial^2 G}{\partial z \partial r_2} = \frac{-z}{mwL^{5/2}}\Big[&2D(m)\left[r_1(1+m) - rm(2-m)\right] - \\ &K(m)(2r_1 - rm)\Big]\end{aligned} \tag{B.14}$$

$$\begin{aligned}\frac{\partial^2 G}{\partial z \partial r_1} = \frac{-z}{mwL^{5/2}}\Big[&2D(m)\left[r_2(1+m) - rm(2-m)\right] - \\ &K(m)(2r_2 - rm)\Big]\end{aligned} \tag{B.15}$$

$$\begin{aligned}\frac{\partial^2 G}{\partial r_1 \partial r_2} = \frac{-1}{mwL^{5/2}}\Big[&D(m)[2r^2(1-mw) - Lm(1+m)] + \\ &K(m)[(mL - r^2(2-m)]\Big]\end{aligned} \tag{B.16}$$

Third Order Derivatives:

$$\frac{\partial^3 G}{\partial z^3} = \frac{-z}{L^{7/2}w^2}\Big[D(m)\big[L(12-18m)-23z^2w+m^2(6L-8z^2)\big]+ K(m)\left[4z^2(2-m)-3Lw\right]\Big] \tag{B.17}$$

$$\frac{\partial^3 G}{\partial z^2\partial r_1} = \frac{1}{L^{7/2}w^2}\Bigg[\frac{2r_2\left(D(m)-K(m)\right)\left(2z^2-r^2\right)}{m}+ D(m)\Big[2r^3w(2-m)-z^2r(19-17m+6m^2)+2r_2(z^2(7-m)+r^2m)\Big]+ K(m)\Big[rz^2(7-3m)-2r_2(r^2+2z^2)-r^3w\Big]\Bigg] \tag{B.18}$$

$$\frac{\partial^3 G}{\partial z^2\partial r_2} = \frac{1}{L^{7/2}w^2}\Bigg[\frac{2r_1\left(D(m)-K(m)\right)\left(2z^2-r^2\right)}{m}+ D(m)\Big[2r^3w(2-m)-z^2r(19-17m+6m^2)+2r_1(z^2(7-m)+r^2m)\Big]+ K(m)\Big[rz^2(7-3m)-2r_1(r^2+2z^2)-r^3w\Big]\Bigg] \tag{B.19}$$

$$\frac{\partial^3 G}{\partial z\partial r_1\partial r_2} = \frac{-z}{L^{7/2}w^2}\Bigg[\frac{6r^2(K(m)-D(m))}{m}+ D(m)\Big[L(3+7m-2m^2)-r^2(9-19m+8m^2)\Big]- K(m)\Big[L(3+m)+2r^2(3-2m)\Big]\Bigg] \tag{B.20}$$

$$\frac{\partial^3 G}{\partial z\partial r_2^2} = \frac{-z}{4L^{7/2}w^2r_2^2}\Bigg[D(m)\Big[2L^2(-1+2m+m^2)+16Lr_2^2w^2- 4r_2^2[4(3-m)(r^2-2r_2^2)+z^2w(11-8m)]\Big]+ K(m)\Big[L^2(2-3m)-8Lr_2^2w+8r_2^2[r^2-2r_2^2+(3-2m)z^2]\Big]\Bigg] \tag{B.21}$$

$$
\begin{aligned}
\frac{\partial^3 G}{\partial r_1 \partial r_2^2} \quad = \quad & \frac{1}{2L^{7/2}w^2 r_2}\Bigg[\frac{4(D(m)-K(m))(Lr(r_1-r_2)+3r_2^4)}{m} - \\
& D(m)\Big[L\big[Lm(7m-4)+2m^2 r r_1 + r_1^2(8-m)\big] - r_2^2 L(8-29m) - \\
& \quad r_2[10Lm^2 r - 2r^3(9-19m+8m^2) + Lr_1(9+4m)]\Big] - \\
& K(m)\Big[m[5L^2+8r^3 r_2 - Lr(2r_1+11r_2)] - \\
& \quad 12rr_2^3 - L(6r_1^2+7r_1 r_2 - 6r_2^2)\Big]\Bigg] \qquad \text{(B.22)}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial^3 G}{\partial r_2^3} \quad = \quad & \frac{-1}{2L^{7/2}w^2 r_2^3}\Bigg[\\
& D(m)\Big[L^3(2-m(5-4m)) - L^2 r_1 r_2 m(1+m) - \\
& \quad r_2^2 L[Lm(3-m) + r_1^2(1+5m+2m^2)] + \\
& \quad 2r_2^3[r^3(1+15m-8m^2) + 2Lr_1(3-m)(1-3m)] + \\
& \quad r_2^4 L(33-39m+14m^2) - 16r_2^5 r(3-m)\Big] - \\
& K(m)\Big[L^3(2-4m) + r_2 L^2 m(10r_1 - r_2) - 8r_2^3 r^3 w - 8r_2^5 r - \\
& \quad r_2^2 L[r^2 w - 4r_2^2 + 2r(3r_1 - 6r_2 + 4mr_2)]\Big]\Bigg] \qquad \text{(B.23)}
\end{aligned}
$$

Fourth Order Derivatives:

$$\frac{\partial^4 G}{\partial z^4} = \frac{-1}{L^{9/2}w^3}\Big[2D(m)\Big[-z^4(25-9m+11m^2-3m^4) - 3r^4w^2(2-m) + 3z^2r^2w(19-17m+6m^2)\Big] + K(m)\Big[3r^4w^2 - 6z^2r^2w(7-3m) + z^4(26-5m+3m^2)\Big]\Big] \tag{B.24}$$

$$\frac{\partial^4 G}{\partial z^3 \partial r_1} = \frac{-z}{L^{9/2}w^3}\Big[\frac{6r_2(K(m)-D(m))(3r^2-2z^2)}{m} - D(m)\Big[6r^2r_2(4-m)(1-2m) - 3r^3w(23w+8m^2) + rz^2(107-126m+91m^2-24m^3) - 4r_2z^2(23-3m+m^2)\Big] + K(m)\Big[z^2[r(47-35m+12m^2) - 2r_2(19-m)] - 6r^2\big[2rw(2-m) + r_2(2+m)\big]\Big]\Big] \tag{B.25}$$

$$\frac{\partial^4 G}{\partial z^3 \partial r_2} = \frac{z}{L^{9/2}w^3}\Big[\frac{6r_1(D(m)-K(m))(3r^2-2z^2)}{m} + D(m)\Big[2r^2r_1(2-11m+4m^2) - r^3w(31-35m+12m^2) + 4r_1[L(5-8m+m^2) - z^2(28-11m+2m^2)] + rz^2(145-198m+137m^2-36m^3) - 2rLw(19-17m+6m^2)\Big] - K(m)\Big[-2r(1-m)(L(7-3m)+(5-3m)r^2) - 2r_1[4Lm+(6-m)r^2] + z^2\big[r(61-55m+18m^2) - 2r_1(19-5m)\big]\Big]\Big] \tag{B.26}$$

$$\begin{aligned}
\frac{\partial^4 G}{\partial z^2 \partial r_1 \partial r_2} &= \frac{1}{L^{9/2} w^3}\Bigg[\frac{6r^2(D(m)-K(m))(r^2-4z^2)}{m} - \\
&\quad D(m)\Big[r^2\big[L(15-28m+27m^2-8m^3)-6z^2(10-33m+28m^2-8m^3)\big]+ \\
&\qquad L^2(3+4m-9m^2+2m^3)-Lz^2(15+58m-33m^2+8m^3)\Big]+ \\
&\quad K(m)\Big[r^2[2L(6-5m+2m^2)-3z^2(15-21m+8m^2)]+ \\
&\qquad L^2w(3+m)-Lz^2(15+13m-4m^2)\Big]\Bigg] \qquad \text{(B.27)}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial^4 G}{\partial z^2 \partial r_2^2} &= \frac{-1}{4L^{9/2} r_2^2 w^3}\Bigg[\\
&\quad D(m)\Big[4Lr_2^2w[4(m-3)(r^2-2r_2^2)+5z^2w(8m-11)]- \\
&\qquad 2L^3(1-3m+m^2+m^3)+ \\
&\qquad L^2[16r_2^2w^3+z^2(6-15m-19m^2+4m^3)]+ \\
&\qquad 8r_2^2z^2\big[(47-31m+8m^2)(r^2-2r_2^2)+z^2w(41-60m+24m^2)\big]\Big]+ \\
&\quad K(m)\Big[Lw(L^2(2-3m)-8r_2^2(Lw-r^2)-16r_2^4)+ \\
&\qquad 2z^2\big[20Lr_2^2w(3-2m)-L^2(3-6m-m^2)+16r_2^2(m-3)(r^2-2r_2^2)\big]- \\
&\qquad 4r_2^2z^4(47-63m+24m^2)\Big]\Bigg] \qquad \text{(B.28)}
\end{aligned}$$

$$
\begin{aligned}
\frac{\partial^4 G}{\partial z \partial r_1 \partial r_2^2} &= \frac{-z}{2L^{9/2} r_2 w^3}\Bigg[\frac{12(D(m)-K(m))[Lr(r_1-r_2)+5r_2^4]}{m} + \\
&\quad D(m)\Big[L^2 m(34-69m+105m^2-71m^3+18m^4) - \\
&\quad 2Lr_1^2(15+16m-39m^2+38m^3-12m^4) + \\
&\quad 2Lr_2^2(15-160m+225m^2-146m^3+36m^4) - \\
&\quad 12r_2^4(10-33m+28m^2-8m^3) + Lr_1r_2(1+m)^2\Big] + \\
&\quad K(m)\Big[L^2 m(-19+29m-27m^2+9m^3) + 3Lr_1^2(8+5m-9m^2+4m^3) - \\
&\quad r_1r_2(L(1+m+2m^2)-2wr_1^2) + 2r_1r_2^3 w + 6r_2^4(15-21m+8m^2) + \\
&\quad 2r_2^2(r_1^2 w - 3L(4-21m+19m^2-6m^3))\Big]\Bigg] \qquad \text{(B.29)}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial^4 G}{\partial z \partial r_2^3} &= \frac{-z}{2L^{9/2} r_2^3 w^3}\Bigg[\\
&\quad D(m)\Big[8r^3r_2^3(3+35m-38m^2+12m^3) - 8rr_2^5(47-31m+8m^2) + \\
&\quad 2L^3w^3 + L^2r_2\big[2r_1m^3(1+m) + r_2m(1-73m+48m^2-12m^3)\big] + \\
&\quad Lr_2^2\big[w^2r^2 - 2(3+5m^2)r_1^2 + 2rw(3r_1-6r_2+4mr_2) + \\
&\quad\quad 2r_2^2(97-154m+109m^2-28m^3)\big]\Big] + \\
&\quad K(m)\Big[L^3(-2+5m-6m^2) + 2r^3r_2^3(23-55m+24m^2) + \\
&\quad 32rr_2^5(3-m) + L^2mr_2(6r+r_1+2mr_1+r_2(4+9m-3m^2)) + \\
&\quad Lr_2^2\big[2r^2w(2-m) + r_1^2(1+m+2m^2) - 16rr_2(3w+m^2) - \\
&\quad\quad r_2^2(49-39m+14m^2)\big]\Big]\Bigg] \qquad \text{(B.30)}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial^4 G}{\partial r_1 \partial r_2^3} \quad = \quad & \frac{1}{2L^{9/2} r_2^2 w^3}\Bigg[\frac{-2(D(m)+K(m))r1^4 r2^2}{m} + \\
& \frac{1}{m}\Big[(K(m)-D(m))[4L^2 r_1^2 + 30 r_2^6 - 2L r_2^2(r_1^2 + 9r_2^2)]\Big] - \\
& D(m)\Big[L^2\big[Lm^2(1-13m+17m^2-12m^3+3m^4) - r_1^2(3w+m^2)(4-3m^3)\big] - \\
& \quad Lr_1r_2\big[L(1-2m^3(1+m)) + r1^2 m^2(1-2m)\big] + \\
& \quad Lr_2^4(45-260m+345m^2-208m^3+48m^4) - \\
& \quad 6r_2^6(10-33m+28m^2-8m^3) + \\
& \quad r_2^2\big[9L^2 m(1-wm)(7-7m+2m^2) - \\
& \qquad Lr_1^2(1-4m-11m^2+8m^3) - 2r_1^4(2+m)\big]\Big] + \\
& K(m)\Big[L^2\big[Lm^2(1-6m+5m^2-m^3) - r_1^2(10-8m+m^2-3m^3+m^4)\big] - \\
& \quad Lr_1r_2\big[L(1+m-4m^2+2m^3+2m^4) + r_1^2 m(3+2m^2)\big] + \\
& \quad r_2^2\big[L^2 m(30-44m+33m^2-9m^3) - Lr_1^2 m(1-3m) - r_1^4(1+m)\big] - \\
& \quad 3Lr_1r_2^3 + r_2^4\big[L(36-101m+83m^2-24m^3) + 2r_1^2(1-3m)\big] - \\
& \quad 3r_2^6(15-21m+8m^2)\Big]\Bigg]
\end{aligned} \tag{B.31}
$$

BIBLIOGRAPHY

BIBLIOGRAPHY

Milton Abramowitz and Irene A. Stegun. *Handbook of Mathematical Functions.* Dover Publications, Incorporated, 1974.

J. M. Alimi, A. Serna, C. Pastor, and G. Bernabeu. Smooth Particle Hydrodynamics: Importance of Correction Terms in Adaptive Resolution Algorithms. *Journal of Computational Physics*, 192(1):157–174, 2003.

John R. Anderson, Ira Katz, and Dan Goebel. Numerical Simulation of Two-Grid Ion Optics Using a 3D Code. AIAA 2004-3782, Fort Lauderdale, FL, July 2004. 40th AIAA/ASME/SAE/ASEE Joint Propulsion Conference.

Kendall Atkinson. *Elementary Numerical Analysis.* John Wiley & Sons, Inc., 1993.

Prasanta K. Banerjee. *The Boundary Element Methods in Engineering.* McGraw-Hill Book Company, 1994.

Josh Barnes and Piet Hut. A Hierarchical O(N log N) Force-Calculation Algorithm. *Nature*, 324(6096):446–449, 1986.

J. Thomas Beale and Ming-Chih Lai. A Method for Computing Nearly Singular Integrals. *SIAM Journal on Numerical Analysis*, 38(6):1902–1925, 2001.

G. A. Bird. *Molecular Gas Dynamics and the Direct Simulation of Gas Flows.* Oxford University Press, 1994.

C. K. Birdsall and A. B. Langdon. *Plasma Physics Via Computer Simulation.* Adam Hilger Press, 1991.

Iain D. Boyd and R. A. Dressler. Far-field Modeling of the Plasma Plume of a Hall Thruster. *Journal of Applied Physics*, 92:1764–1774, 2002.

Iain D. Boyd. Conservative Species Weighting Scheme for the Direct Simulation Monte Carlo Method. *Journal of Thermophysics and Heat Transfer*, 10:579–585, 1996.

William L. Briggs, Van Emden Henson, and S. F. McCormick. *A Multigrid Tutorial.* Society for Industrial and Applied Mathematics, 2000.

John R. Brophy, David E. Brinza, James E. Polk, Michael D. Henry, and Anita Sengupta. The DS1 Hyper-Extended Mission. AIAA 2002-3673, Indianapolis, IN, July 2002. 38th AIAA/ASME/SAE/ASEE Joint Propulsion Conference.

Edgar Y. Choueiri. Fundamental Difference between the Two Variants of Hall Thrusters: SPT and TAL. AIAA 2001-3504, Salt Lake City, UT, July 2001. 37th AIAA/ASME/SAE/ASEE Joint Propulsion Conference.

Edgar Y. Choueiri. A Critical History of Electric Propulsion: The First 50 Years (1906-1956). *Journal of Propulsion and Power*, 20(2):193–203, March-April 2004.

Andrew J. Christlieb, Robert Krasny, and John P. Verboncoeur. Efficient Particle Simulation of a Virtual Cathode Using a Grid-Free Treecode Poisson Solver. *IEEE Transactions on Plasma Science-Special Issue*, 32(2):384–389, 2004.

Mark W. Crofton and Iain D. Boyd. The Origins of Accelerator Grid Current: Analysis of T5-Grid Test Results. AIAA 99-2443, Los Angeles, CA, June 1999. 35th AIAA/ASME/SAE/ASEE Joint Propulsion Conference.

A. Dalgarno, M. R. C. McDowell, and A. Williams. The Mobilities of Ions in Unlike Gases. *Royal Society of London Philosophical Transactions Series A*, 250:411–425, 1958.

Jerold W. Emhoff and Iain D. Boyd. Grid Erosion Modeling of the NEXT Ion Thruster Optics. AIAA 2003-4869, Huntsville, AL, July 2003. 39th AIAA/ASME/SAE/ASEE Joint Propulsion Conference.

Cody C. Farnell, John D. Williams, and Paul J. Wilbur. NEXT Ion Optics Simulation Via ffx. AIAA 2003-4869, Huntsville, AL, July 2003. 39th AIAA/ASME/SAE/ASEE Joint Propulsion Conference.

John E. Foster, George C. Soulas, and Michael J. Patterson. Plume and Discharge Plasma Measurements of an NSTAR-type Ion Thruster. AIAA 2000-3812, Huntsville, AL, July 2000. 36th AIAA/ASME/SAE/ASEE Joint Propulsion Conference.

John E. Foster, Tom Haag, Hani Kamhawi, Michael Patterson, Shane Malone, Fred Elliot, George J. Williams Jr., James S. Sovey, and Christian Carpenter. The High Power Electric Propulsion (HiPEP) Ion Thruster. AIAA 2004-3812, Fort Lauderdale, FL, July 2004. 40th AIAA/ASME/SAE/ASEE Joint Propulsion Conference.

Keith D. Goodfellow, Gani B. Ganapathi, and John F. Stocky. An Experimental and Theoretical Analysis of the Grid Clearing Capability of the NSTAR Ion Propulsion System. AIAA 99-2859, Los Angeles, CA, June 1999. 35th AIAA/ASME/SAE/ASEE Joint Propulsion Conference.

Leslie Greengard and Vladimir Rokhlin. A Fast Algorithm for Particle Simulations. *Journal of Computational Physics*, 73(2):325–348, 1987.

Daniel A. Herman and Alec D. Gallimore. Discharge Chamber Plasma Structure of a 30-cm NSTAR-type Ion Engine. AIAA 2004-3794, Fort Lauderdale, FL, July 2004. 40th AIAA/ASME/SAE/ASEE Joint Propulsion Conference.

Charles Hirsch. *Numerical Computation of Internal and External Flows*. John Wiley & Sons, Inc., 1997.

Roger W. Hockney and James W. Eastwood. *Computer Simulation Using Particles*. McGraw-Hill Inc., 1981.

R. Kafafy and Joseph Wang. Whole Ion Optics Simulations of a Subscale Gridlet Using a Hybrid-Grid IFE-PIC Code. AIAA 2004-3783, Fort Lauderdale, FL, July 2004. 40th AIAA/ASME/SAE/ASEE Joint Propulsion Conference.

Hani Kamhawi, George C. Soulas, Michael J. Patterson, and Michael M. Frandina. NEXT Ion Engine 2000 Hour Wear Test Plume and Erosion Results. AIAA 2004-3792, Fort Lauderdale, FL, July 2004. 40th AIAA/ASME/SAE/ASEE Joint Propulsion Conference.

John T. Katsikadelis. *Boundary Elements: Theory and Applications*. Elsevier Science Limited, 2002.

Harold R. Kaufman. One-Dimensional Analysis of Ion Rockets. *NASA TN D-261*, March 1960.

Rainer Killinger, Ralf Kukies, Michael Surauer, Giorgio Saccoccia, and Howard Gray. Final Report on the ARTEMIS Salvage Mission using Electric Propulsion. AIAA 2003-4546, Huntsville, AL, July 2003. 39th AIAA/ASME/SAE/ASEE Joint Propulsion Conference.

Hitoshi Kuninaka, Kazutaka Nishiyama, Yukio Shimizu, and Kyoichiro Toki. Flight Status of Cathode-Less Microwave Discharge Ion Engines Onboard HAYABUSA Asteroid Explorer. AIAA 2004-3438, Fort Lauderdale, FL, July 2004. 40th AIAA/ASME/SAE/ASEE Joint Propulsion Conference.

Barbara F. Lasinski, A. Bruce Langdon, Stephen P. Hatchett, Michael H. Key, and Max Tabak. Particle-in-Cell Simulations of Ultra Intense Laser Pulses Propagating Through Overdense Plasma for Fast-Ignitor and Radiography Applications. *Physics of Plasmas*, 6(5):2041–2047, May 1999.

Michael A. Lieberman and Allan J. Lichtenberg. *Principles of Plasma Discharges and Materials Processing*. Wiley-Interscience, 1994.

Keith Lindsay and Robert Krasny. A Particle Method and Adaptive Treecode for Vortex Sheet Motion in Three-Dimensional Flow. *Journal of Computational Physics*, 172(2):879–907, 2001.

Yijun Liu, Daming Zhang, and Frank J. Rizzo. Nearly Singular and Hypersingular Integrals in the Boundary Element Method. In C.A. Brebbia and J.J. Rencis, editors, *Boundary Elements XV*. Southampton and Elsevier Applied Science, 1993.

Shane P. Malone and George C. Soulas. Computational Ion Optics Design Evaluations. AIAA 2004-3784, Fort Lauderdale, FL, July 2004. 40th AIAA/ASME/SAE/ASEE Joint Propulsion Conference.

J. S. Miller, S. H. Pullins, D. J. Levandier, Y. Chiu, and R. A. Dressler. Xenon Charge Exchange Cross Sections for Electrostatic Thruster Models. *Journal of Applied Physics*, 91(3):984–991, 2002.

Monika Nitsche. Axisymmetric Vortex Sheet Motion: Accurate Evaluation of the Principal Value Integral. *SIAM Journal on Scientific Computing*, 21(3):1066–1084, 1999.

Yasushi Okawa, Yukio Hayakawa, and Shoji Kitamura. Three-Dimensional Divergence Characteristics of Ion Beamlets in an Ion Thruster. AIAA 2004-3785, Fort Lauderdale, FL, July 2004. 40th AIAA/ASME/SAE/ASEE Joint Propulsion Conference.

James E. Polk, N. R. Moore, L. E. Newlin, John R. Brophy, and D. H. Ebbeler. Probabilistic Analysis of Ion Engine Accelerator Grid Life. IEPC-93-176, Seattle, WA, September 1993. 23rd International Electric Propulsion Conference.

James E. Polk. Personal Communication, September 2002.

Ji Qiang, Robert D. Ryne, Salman Habib, and Viktor Decyk. An Object-Oriented Parallel Particle-In-Cell Code for Beam Dynamics Simulation in Linear Accelerators. *Journal of Computational Physics*, 163(2):434–451, September 2000.

Thomas M. Randolph and James E. Polk. An Overview of the Nuclear Electric Xenon Ion System (NEXIS) Activity. AIAA 2004-3450, Fort Lauderdale, FL, July 2004. 40th AIAA/ASME/SAE/ASEE Joint Propulsion Conference.

Karl-Ulrich Riemann. The Bohm Criterion and Sheath Formation. *Journal of Physics D: Applied Physics*, 24(4):493–518, 1991.

Vladimir Rokhlin. Rapid Solution of Integral Equations of Scattering Theory in Two Dimensions. *Journal of Computational Physics*, 86(2):413–449, 1990.

J. Reece Roth. *Industrial Plasma Engineering: Volume 1: Principles.* Institute of Physics Publishing, 2001.

Robie I. Samanta Roy. *Numerical Simulation of Ion Thruster Plume Backflow for Spacecraft Contamination Assessment.* PhD thesis, Massachusetts Institute of Technology, 1995.

Wilhelmus M. Ruyten. Density-Conserving Shape Factors for Particle Simulations in Cylindrical and Spherical Coordinates. *Journal of Computational Physics*, 105:224–232, 1993.

Anita Sengupta, John R. Brophy, and Keith D. Goodfellow. Status of the Extended Life Test of the Deep Space 1 Flight Spare Ion Engine After 30,352 Hours of Operation. AIAA 2003-4558, Huntsville, AL, July 2003. 39th AIAA/ASME/SAE/ASEE Joint Propulsion Conference.

L. Shure and Bjorn Bonnevier. MATLAB Function ellipke. MATLAB source code, April 2001.

Avram Sidi and Moshe Israeli. Quadrature Methods for Periodic Singular and Weakly Singular Fredholm Integral Equations. *Journal of Scientific Computing*, 3(2):201–231, 1988.

George C. Soulas, Thomas. W. Haag, and Michael J. Patterson. Performance Evaluation of 40 cm Ion Optics for the NEXT Ion Engine. AIAA 2002-3834, Indianapolis, IN, July 2002. 38th AIAA/ASME/SAE/ASEE Joint Propulsion Conference.

George C. Soulas, Matthew T. Domonkos, Hani Kamhawi, and Michael J. Patterson. Status of the NEXT Ion Engine Wear Test. AIAA 2003-4863, Huntsville, AL, July 2003. 39th AIAA/ASME/SAE/ASEE Joint Propulsion Conference.

George C. Soulas, Hani Kamhawi, Michael J. Patterson, Melissa A. Briton, and Michael M. Frandina. NEXT Ion Engine 2000 Hour Wear Test Results. AIAA 2004-3791, Fort Lauderdale, FL, July 2004. 40th AIAA/ASME/SAE/ASEE Joint Propulsion Conference.

James S. Sovey, Vince K. Rawlin, and Michael J. Patterson. Ion Propulsion Development Projects in U. S.: Space Electric Rocket Test 1 to Deep Space 1. *AIAA Journal of Propulsion and Power*, 17, 2001.

James H. Strickland and Donald E. Amos. Fast Solver for Systems of Axisymmetric Ring Vortices. *AIAA Journal*, 30(3):737–746, 1992.

Deborah Sulsky, Shi-Jian Zhou, and Howard L. Schreyer. Application of a Particle-In-Cell Method to Solid Mechanics. *Computer Physics Communications*, 87(1-2):236–252, 1995.

Michael Tartz, E. Hartmann, and H. Neumann. Evolution of Extraction Grid Erosion With Operation Time. AIAA 2004-3787, Fort Lauderdale, FL, July 2004. 40th AIAA/ASME/SAE/ASEE Joint Propulsion Conference.

Lloyd N. Trefethen and David Bau, III. *Numerical Linear Algebra.* Society for Industrial and Applied Mathematics, 1997.

Joseph Wang, James Polk, John Brophy, and Ira Katz. Three-Dimensional Particle Simulations of Ion-Optics Plasma Flow and Grid Erosion. *AIAA Journal of Propulsion and Power*, 19(6):1192–1199, November-December 2003.

V. V. Zhurin, H. R. Kaufman, and R. S. Robinson. Physics of closed drift thrusters. *Plasma Sources Science and Technology*, 8:R1–R20, 1999.