EXAMINATION OF MAGNETIC PLASMA EXPULSION

Ryan E. Phillips

Dissertation Prepared for the Degree of

DOCTOR OF PHILOSOPHY

UNIVERSITY OF NORTH TEXAS

May 2018

Phillips, Ryan E. *Examination of Magnetic Plasma Expulsion*. Doctor of Philosophy (Physics), May 2018, 114 pp., 24 figures, 59 numbered references.

Magnetic plasma expulsion uses a magnetic field distortion to redirect incident charged particles around a certain area for the purposes of shielding. Computational studies are carried out and for certain values of magnetic field, magnetic plasma expulsion is found to effectively shield a sizable area. There are however many plasma behaviors and interactions that must be considered. Applications to a new cryogenic antimatter trap design are discussed.

ACKNOWLEDGMENTS

This material is based upon work supported by the Department of Energy under Grant No. DE-FG02-06ER54883 and by the National Science Foundation under Grant No. PHY-1202428 and PHY-1500427.

Computational resources were provided by UNT's High Performance Computing Services, a division of the University Information Technology with additional support from UNT Office of Research and Economic Development.

The author would like to thank Dave Grote, Jean-Luc Vay, Ryan Lane, and Ryan Hedlof for helpful discussions. The author also gratefully acknowledges the motivation, support and encouragement of my wife Angela Phillips, our son Maxwell, my mother Peggy Phillips, my father Ken Phillips, my brother Ross Phillips, and my advisor Carlos A. Ordonez.

TABLE OF CONTENTS

LIST OF FIGURES

Page

CHAPTER 1

INTRODUCTION

Magnetic plasma expulsion seeks to prevent plasma from entering a given region by creating a magnetic field distortion in the area such that charged particles of either sign follow the distorted magnetic field lines away from the area of interest. Magnetic plasma expulsion techniques may allow instrumentation access to the interiors of plasmas and may also make alternative approaches for magnetic plasma confinement possible. Magnetic plasma expulsion has the potential to improve currently existing technologies in the areas of accelerator physics and plasma trapping. This work seeks to evaluate and model magnetic plasma expulsion with an eye towards these applications.

Chapter 2 examines magnetic plasma expulsion in the context of controlling a space-charge neutralized ion beam. Space-charge neutralized ion beam control is more complex than control of charged beams. While focusing techniques for charged beams have been widely studied, many of the tools available for controlling charged beams are not suitable for charge-neutralized beams. Presented here is a scheme for controlling charge-neutralized beams using a distortion of a magnetic guide field via the presence of two current carrying wires. The extent to which the beam can be controlled is evaluated using a classical trajectory Monte Carlo simulation.

Building on these results in the beam case, Chapter 3 uses a particle-in-cell simulation with the Warp code to study magnetic plasma expulsion in the presence of a trapped Maxwellian distributed plasma. The same distortion of a magnetic guide field with two current carrying wires is used. Some conditions for achieving magnetic plasma expulsion are reported. In this case the envisioned application is a plasma trap augmented by embedded magnetic field producing coils such as two coaxial solenoids or levitated rings. The extent to which a given area can be shielded from an impinging plasma is studied.

In Chapter 4, additional considerations of magnetic plasma expulsion are examined. An area in need of study with magnetic plasma expulsion is the difference in motion experienced by species of the plasma that have different masses. This will be studied in an electron/proton plasma. The mass of a proton is 42.85 times larger than the mass of an electron and as such, both the gyro radius and the thermal speed of the particles will be significantly different. This difference in speed leads to distinctly different time scales of interest which could potentially lead to the development of phenomena not seen in the case of equal mass opposite charged plasmas such as plasma sheathing around the cylindrical material surfaces or asymmetric losses between the two plasma species. For applications of magnetic plasma expulsion to be developed, an understanding of these phenomena must be gained. However, it is computationally expensive to simulate an electron-proton plasma as a result of the vastly different time scales involved. Computational methods are explored to circumvent this.

Previous work only reported loss rates as a fraction of the simulated plasma with no consideration to exactly where those lost particles came from. Characterization of the phase space that lost particles come from in the simulation could yield improvements to the existing design.

Depending on the methods used, a simulation in computational equilibrium may be drastically different than the same system in a physical experiment. Several permutations of a base case simulation are examined to determine what effects various computational boundaries have on the over all results of the simulation.

CHAPTER 2

MAGNETIC CONTROL OF A SPACE-CHARGE NEUTRALIZED ION BEAM [1]

2.1. Introduction

Charge neutralized beams have applications in a number of fields. Several studies have been carried out in the area of drift compression of a neutralized ion beam for the purpose of studying high energy density plasmas and fusion conditions [1]. Quality thin film deposition has been shown to depend on space charge neutralization as well [2]. Control of charge-neutralized ion beams must be handled differently than traditional set-ups used on charged ion beams. As such, novel complications and opportunities arise. For example, the work here uses a concept initially explored within a plasma confinement context [3] to exert control on a charge-neutralized ion beam. Deflection of a charge-neutralized ion beam is constrained by certain conditions however [4], and beam control can be difficult since standard electric field based techniques may not be effective. Ion beams can be space-charge neutralized when injected into a field free region with a source of electrons [5], and transport of space-charge neutralized, unmagnetized ion beams has been investigated using a spatially periodic field to confine a plasma which serves as the source of electrons that neutralize the ion beam in flight [6]. Studies have also been carried out using a spatially periodic field to confine ions that drag along electrons such that the two species system of particles can be considered a neutralized drifting plasma [7]. Certain issues surrounding pure charged ion beam transport specifically for intense beam applications can be solved using charge-neutralization. An example is the mutual repulsion caused by the space charge of the beam in flight. This "blow up" of the beam diameter is one of the primary limitations on transport and final spot size for a given beam, and charge-neutralization allows for the beam's space charge to be suppressed [8]. Many areas of physics, from heavy ion fusion

[1]This chapter is presented in its entirety from Physics Procedia, **66**, 2015, 148-155 with permission from Elsevier

studies [9, 10, 11, 12] to spectroscopy [13] to ion implantation [14, 15] and propulsion [16], benefit from a beam that is charge-neutralized.

2.2. Equations and Description of Simulation

Consider a charge-neutralized ion beam that travels parallel to a uniform magnetic field. Suppose magnetic field generating wires are immersed as shown in 2.1 in the beam's path in such a way as to create shielded regions which the beam deflects around. Targets could be moved vertically into place near the wires while the shielding magnetic field is active. Once in position, the wire current is shut off allowing a processing step to take place (e.g., ion implantation or thin film deposition). Automation of the target insertion/extraction could lead to significantly increased throughput of targets. Applications that alter only small surface areas would benefit most from this such as drill bit hardening or certain treatments for small medical equipment like scalpel blades etc. In the work presented here, the shielding field is assumed to be the field produced by two straight, parallel, infinitesimally thin wires. Thus, the total magnetic field is a superposition of a uniform field and that produced by the wires. Larger arrays of wires lead to a spatially periodic field [17], which may serve to provide a larger number of regions that can be simultaneously shielded.

The magnetic field near the region of interest can be described in two parts: A constant uniform field B_0 defined to be in the z direction, and a field generated by two infinitesimally thin, parallel, current carrying wires in the x,y plane

$$B(x,y,z) = B_m\beta(\frac{x}{S},\frac{y}{S},\frac{z}{S}) + B_0\hat{\boldsymbol{k}}. \tag{1}$$

Here,

$$\beta(x_n,y_n,z_n) = [-\frac{z_n}{(x_n-1)^2+z_n^2},0,\frac{(x_n-1)}{(x_n-1)^2+z_n^2}] - [-\frac{z_n}{(x_n+1)^2+z_n^2},0,\frac{(x_n+1)}{(x_n+1)^2+z_n^2}] \tag{2}$$

and $B_m = \frac{\mu_0 I}{2\pi S}$ is the standard expression for the magnetic field at a distance S from a single wire with I the magnitude of current, μ_0 the permeability of free space, and S half the distance between the wires. Note that the currents carried by the two wires must flow in opposite directions to generate a magnetic field as shown in 3.2.

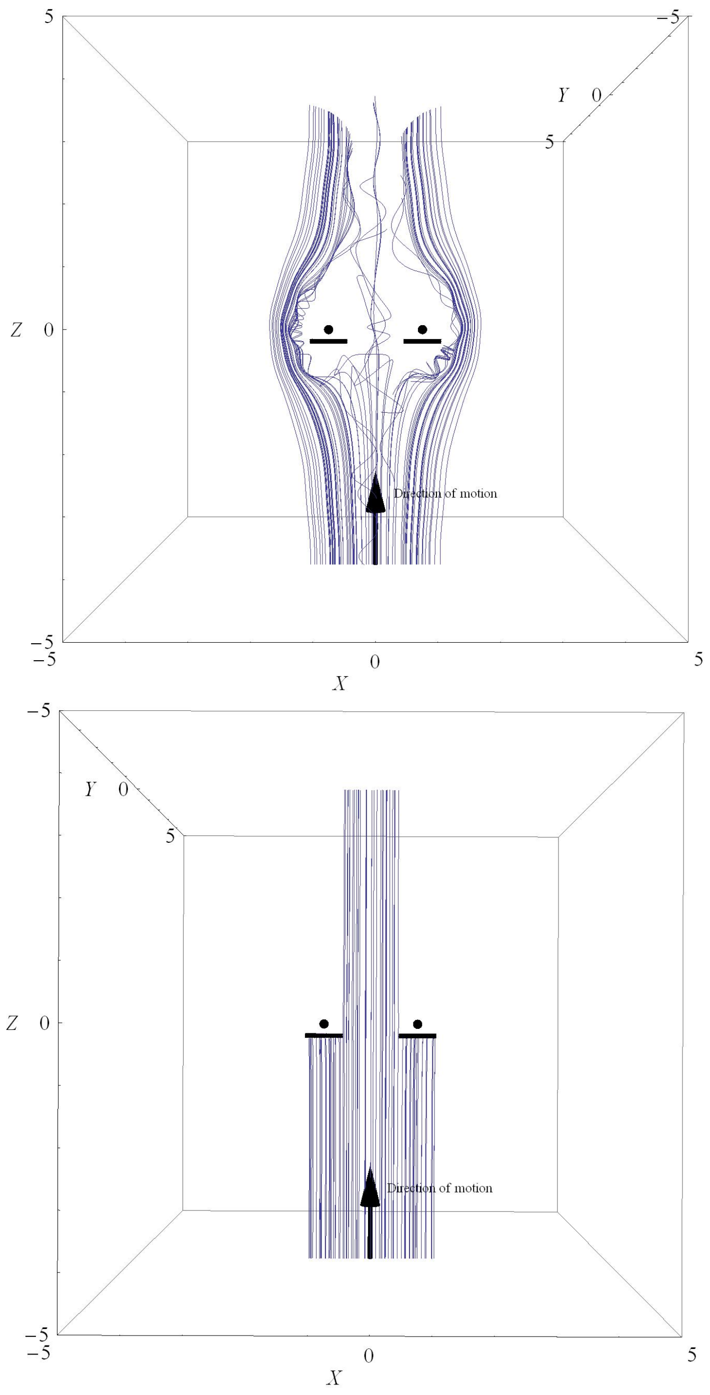

FIGURE 2.1. Two targets (horizontal lines) placed near two wires (solid dots) are shielded by the wires' magnetic fields. An ion beam travels along the z axis from negative to positive (bottom to top) as indicated by the arrow. In the left panel particles, represented by colored lines, are deflected around the area near the targets by the magnetic field generated by the shielding wires. When the wire currents are turned off, beam processing such as ion implantation or thin film deposition occurs.

Studying the trajectories of charged particles in this field is possible with a classical trajectory Monte Carlo simulation. To keep the results most generally applicable, all parameters are normalized [17]. Normalization is achieved by setting $m_n = q_n = K_n = S_n = 1$. Here m_n is the normalized mass, q_n is the normalized charge, K_n is the normalized kinetic energy, and S_n is the normalized length. Other parameters needed for the simulation are defined using their un-normalized counterparts. The position is normalized as $r_n = \frac{r}{S}$, the velocity as $v_n = v\sqrt{\frac{m}{K}}$, the acceleration as $a_n = \frac{amS}{K}$, time as $t_n = t(\frac{1}{S})\sqrt{\frac{K}{m}}$, and magnetic field as $B_n = \frac{BqS}{\sqrt{mK}}$. The un-normalized counterparts are then solved for and substituted into the Lorentz force law, $ma = qv \times B$. A normalized version of the Lorentz force law is then obtained:

$$a_n = v_n \times B_n. \tag{3}$$

In executing the Monte Carlo simulation, this vector equation is solved numerically. However, an expression for the combined normalized magnetic field is needed.

Returning to the uniform field B_0, B_0 can be used to define a parameter r_0,

$$B_0 = \sqrt{\frac{2mK}{q^2 r_0^2}}. \tag{4}$$

r_0 is the cyclotron radius of a particle trajectory in the uniform field B_0 with kinetic energy K associated with motion transverse to the magnetic field. Combining the expressions for the various magnetic fields into a single normalized expression yields

$$B_n(x_n, y_n, z_n) = \frac{sgn(q)\sqrt{2}}{r_{0n}}[\beta_0\beta(x_n, y_n, z_n) + \boldsymbol{\hat{k}}]. \tag{5}$$

Here, $sgn(q) = q/|q|$ is the sign of charge, $\beta_0 = B_m/B_0$ is the field strength ratio, and $r_{0n} = \frac{r_0}{S}$ is the normalized cyclotron radius. Revisiting Eq. (3) and splitting it into vector components yields the normalized equations of motion,

(6)

$$x_n''(t_n) = \frac{sgn(q)\sqrt{2}}{r_{0n}}\left(y_n'(t_n)(1 + \beta_0\beta_z[x_n(t_n), y_n(t_n), z_n(t_n)]) - z_n'(t_n)\beta_0\beta_y[x_n(t_n), y_n(t_n), z_n(t_n)]\right),$$

(7)
$$y_n''(t_n) = \frac{sgn(q)\sqrt{2}}{r_{0n}}\left(z_n'(t_n)\beta_0\beta_x[x_n(t_n), y_n(t_n), z_n(t_n)] - x_n'(t_n)(1 + \beta_0\beta_z[x_n(t_n), y_n(t_n), z_n(t_n)])\right),$$

(8)
$$z_n''(t_n) = \frac{sgn(q)\sqrt{2}}{r_{0n}}\left(x_n'(t_n)\beta_0\beta_y[x_n(t_n), y_n(t_n), z_n(t_n)] - y_n'(t_n)\beta_0\beta_x[x_n(t_n), y_n(t_n), z_n(t_n)]\right).$$

These equations are solved numerically via a classical trajectory Monte Carlo simulation for single particles. Note that all forces in the system are conservative. Thus, conservation of energy will hold for particles in the system. However, conservation of energy is not employed within the simulation, so evaluating the total kinetic energy difference between the initial and final points of a trajectory can serve as an indicator of numerical accuracy for the simulation.[17] The kinetic energy of the simulated trajectories varies typically by less than 0.01% in the present work.

Two cylindrical regions, one centered at each wire, are each defined with a normalized radius a_n inside of which no trajectories pass. The size of these regions is some function of the ratio of strengths between the two parts of the total magnetic field $\beta_0 = B_m/B_0$ and the normalized cyclotron radius r_{0n}. This parameter a_n is the parameter of interest and has values $0 < a_n < 1$. a_n cannot have larger values because at least a few trajectories pass between the wires relatively unimpeded (see 2.1). Each value of a_n is numerically determined. Evenly spaced test points are spread across the range of a_n at increments of 0.01, though any arbitrarily fine resolution could be used. For each value of a_n, every combination of values for $0 < r_{0n} < 5$ and $0 < \beta_0 < 5$ in increments of 0.1 is simulated to determine if any prevent all trajectories from crossing within the circle described by a_n by evaluating the position of the simulated particle in relation to the wires at each time step of the equation of motion solution. The maximum values used for r_{0n} and β_0 were determined by executing several successively finer grained simulations over very large ranges and finding that no behavior of interest appears to exist past the ranges stated here.

The initial conditions for particle trajectories are

(9)
$$x_n(0) = x_{n0} = R_x,$$

$$y_n(0) = y_{n0} = 0, \tag{10}$$

$$z_n(0) = z_{n0} = -5, \tag{11}$$

at time $t_n = 0$. Here, R_x has a range of

$$-2 < R_x < 2$$

and is a random number equally likely to have any value in the above range. This insures that particle trajectories are initially spread evenly across the entire region of the deformed magnetic field near the wires enabling comprehensive simulation of effects. Due to the kinetic energy being normalized to $1 = K_n = \frac{1}{2}v_{n0}^2$, each particle's initial normalized speed is $\sqrt{2}$. The simulated particles are considered to be a monoenergtic beam. Thus, the initial velocity components are as follows:

$$v_{nx}(0) = v_{nx0} = 0, \tag{12}$$

$$v_{ny}(0) = v_{ny0} = 0, \tag{13}$$

$$v_{nz}(0) = v_{nz0} = \sqrt{2}\ . \tag{14}$$

Equations $9 - 14$ are the six initial conditions needed to solve the equations of motion. Each trajectory is simulated for a maximum time $t_{n,max} = 20\frac{|z_{n0}|}{v_{nz0}} = \frac{100}{\sqrt{2}}$.

2.3. Evaluation of Simulation Results

2.3 and 2.4 illustrate separately the relationships a_n versus β_0 and a_n versus r_{0n}. The fits in 2.3 are

$$a_n(1.5, \beta_0) = 0.395 * \text{erfc}[0.829(1.733 - \beta_0)]\ , \tag{15}$$

$$a_n(3.0, \beta_0) = 0.339 * \text{erfc}[0.571(2.517 - \beta_0)]\ , \tag{16}$$

$$a_n(4.5, \beta_0) = 0.297 * \text{erfc}[0.492(3.080 - \beta_0)]\ , \tag{17}$$

where erfc[] is the complementary error function. The fits for both 2.3 and 2.4 were determined using the least squares method. In 2.4, the fits are

$$a_n(r_{0n}, 1.5) = -0.093 + 0.038 * r_{0n} + 1.093 * e^{-r_{0n}} \ , \tag{18}$$

$$a_n(r_{0n}, 3.0) = 0.666 - 0.093 * r_{0n} + 0.334 * e^{-r_{0n}} \ , \tag{19}$$

$$a_n(r_{0n}, 4.5) = 0.817 - 0.078 * r_{0n} + 0.182 * e^{-r_{0n}} \ . \tag{20}$$

In both cases, the fits match the data to within 6.4% determined by the largest residuals.

The shielded radius increases with increasing field ratio β_0, but decreases with increasing cyclotron radius r_{0n}. Note however, that the maximum radius of the shielded region is approximate because a finite number of trajectories is simulated (160 in this case). Simulations over larger sample sizes and finer resolutions can refine the edge of the shielded region to any arbitrary certainty desired given enough computation time.

The results obtained match the trend that would be expected from the system. As the field strength B_m of the wires becomes larger, the field ratio gets larger, and an increase in the radius of the shielded region is seen. Conversely, as the uniform field strength B_0 becomes significantly larger than the wire field, the effects of the wire fields are suppressed. Varying the cyclotron radius effectively gives the shielded radius as a function solely of the field strength B_0.

2.5 presents a surface plot of the relationship between all three parameters. As expected, larger values of a_n may be achieved with smaller values of r_{0n} and larger values of β_0. A two parameter fit expression for a_n is found by assuming independence (and therefore separability) between r_{0n} and β_0:

$$a_n(r_{0n}, \beta_0) = [0.367 - 0.044 * r_{0n} + 0.215 * e^{-r_{0n}}] * [\frac{1}{2} * 2.097 * \text{erfc}[\frac{1.896 - \beta_0}{\sqrt{2} * 1.896}]] \ . \tag{21}$$

Here, the fitting constants are determined via the least squares method. Equation (21) matches the simulated data in 2.5 to within 40.4%. By way of example, given a wire separation of 10cm such that $S = 0.05$m and a 5KeV singly charged Boron ion beam with $B_0 = B_m = 0.5$T such that $\beta_0 = 1$, $r_0 = 6.7$cm and $r_{0n} = 1.3$. This would make the unnormalized shielded radius $a = a_n * S = 1.2$cm.

2.4. Conclusion

Proposed here is a two wire control scheme for directing charge-neutralized ion beams and selectively shielding their targets. The regions shielded by the wire fields are quantified as cylindrical with a normalized radius a_n. A relationship between the shielded radius a_n, the ratio of strengths between the two parts of the magnetic fields β_0 and the normalized cyclotron radius r_{0n} is found via a classical trajectory Monte Carlo study. The resultant numerical values were fit using an exponential decay relationship between the shielded radius a_n and the normalized cyclotron radius r_{0n} and an error function relationship between the shielded radius and β_{0n}.

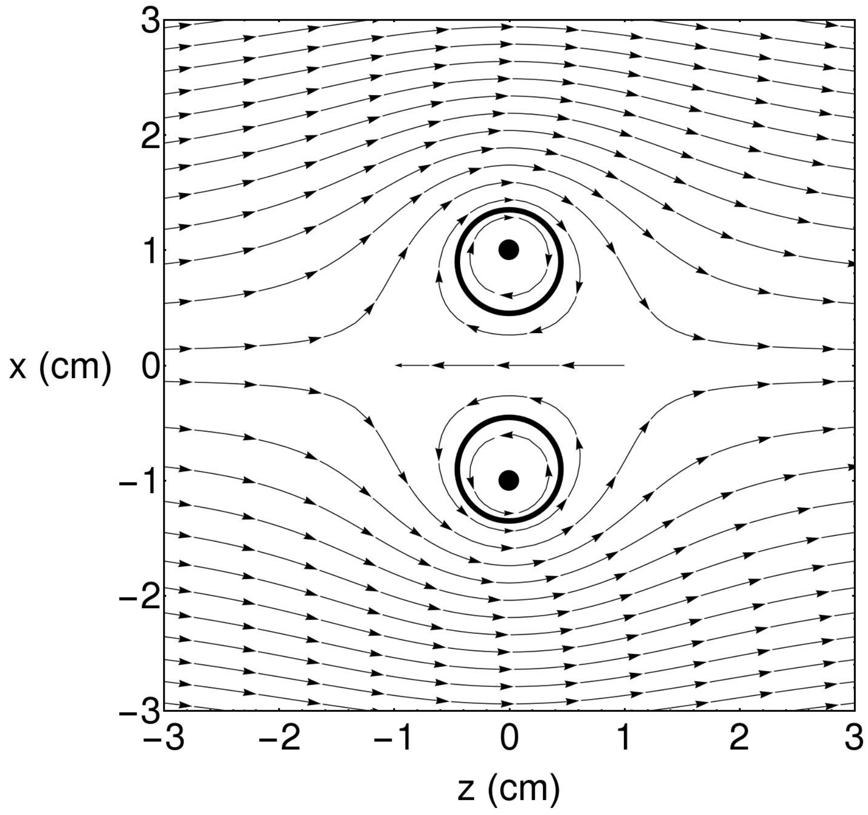

Figure 2.2. Illustration of the combined magnetic field in (x,z) plane. The solid dots represent the wires extending parallel to the y axis.

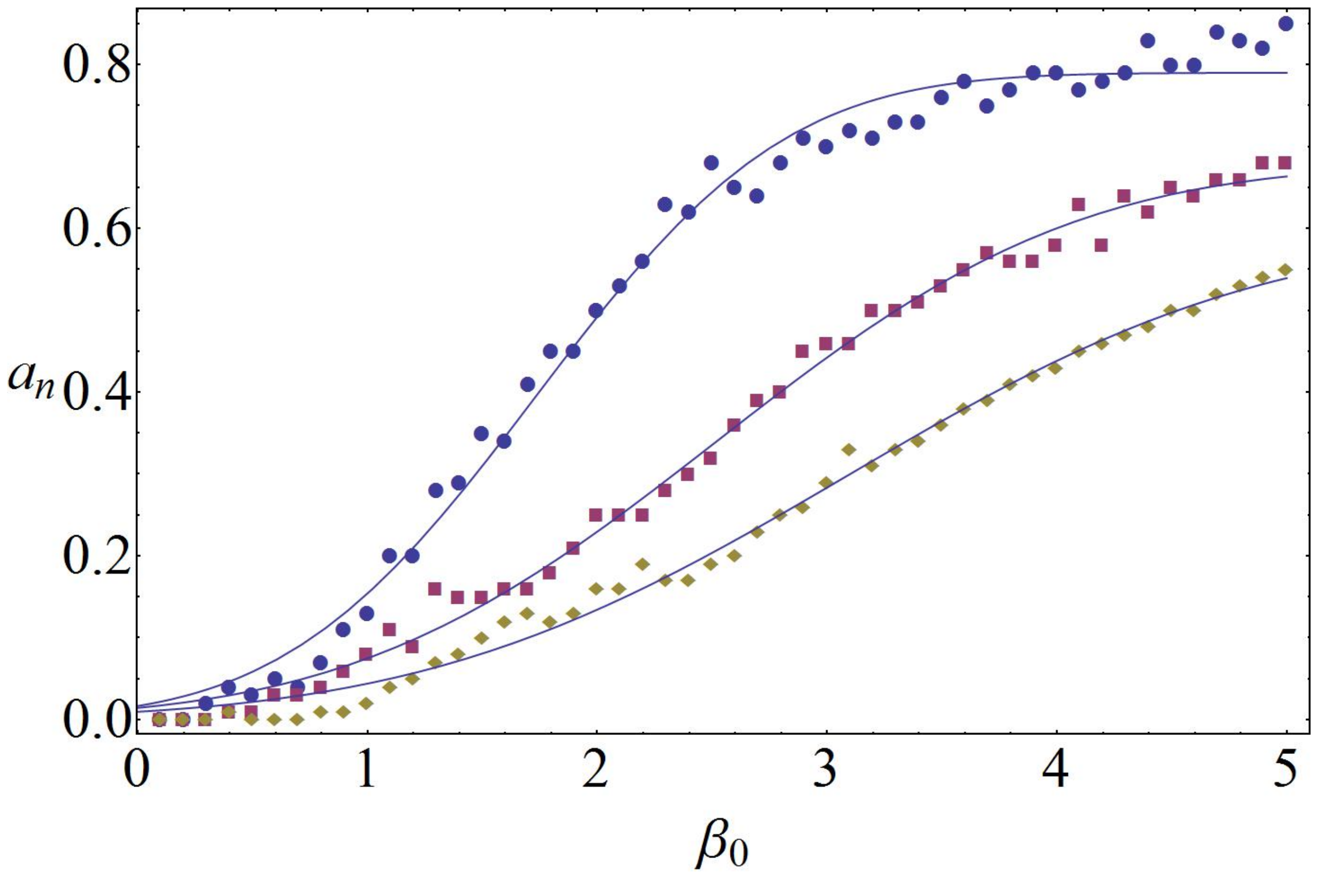

FIGURE 2.3. a_n versus β_0 for values $r_{0n} = 1.5$ (circles), $r_{0n} = 3.0$ (squares), $r_{0n} = 4.5$ (diamonds). The lines are Eqs. (15)-(17).

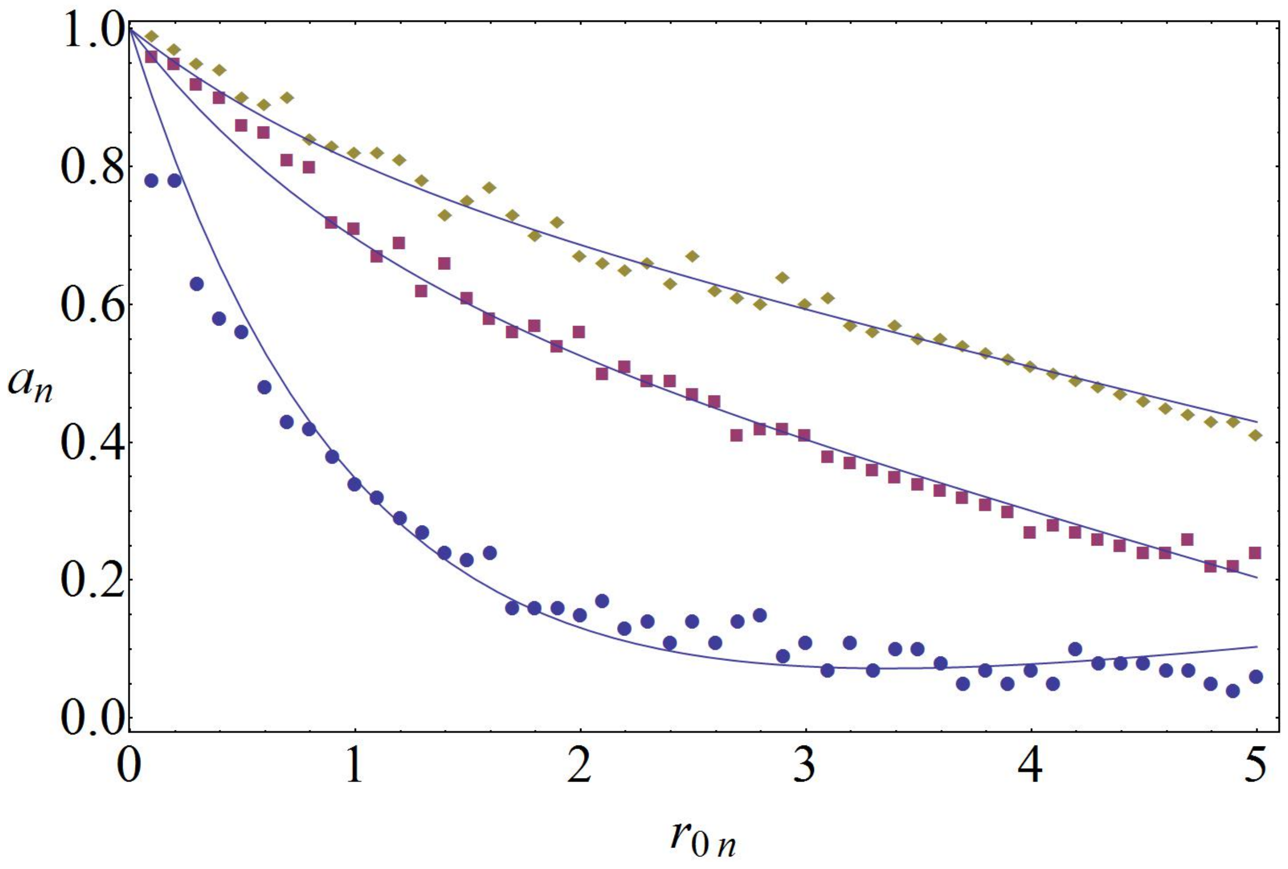

FIGURE 2.4. a_n versus r_{0n} for values $\beta_0 = 1.5$ (circles), $\beta_0 = 3.0$ (squares), and $\beta_0 = 4.5$ (diamonds). The lines are Eqs. (18)-(20).

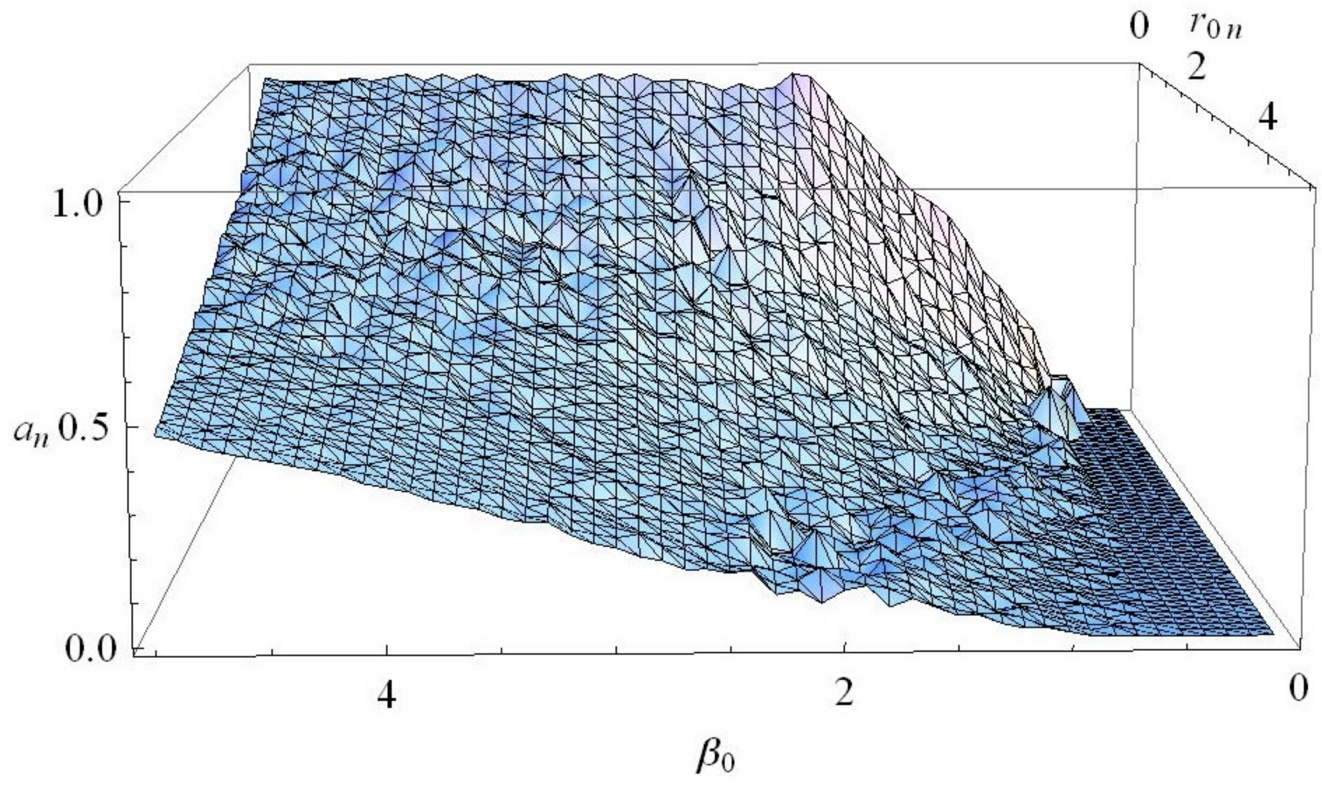

FIGURE 2.5. Surface plot of a_n versus β_0 versus r_{0n}.

CHAPTER 3

MAGNETIC PLASMA EXPULSION [1]

3.1. Introduction

The purpose of magnetic plasma expulsion is to keep plasma from entering a given boundary, in contrast to magnetic plasma confinement which seeks to contain plasma within a given boundary. The concept of magnetic plasma expulsion presented here is related to work recently reported for control of space charged neutralized ion beams,[18] though early studies were conducted in a fusion plasma context.[3] When used in conjunction with magnetic plasma confinement, magnetic plasma expulsion may allow the insertion of instrumentation into a magnetically confined plasma without adversely affecting the confinement properties or purity of the confined plasma. Further, plasma confinement may actually be enhanced via the insertion of field generating electrodes, particle emitters, or particle collectors that control space-charge effects. There exists a known correlation between the electric potential profile of the interior of certain plasmas and the formation of confinement-enhancing transport barriers.[19, 20, 21]

Tokamaks induce a toroidal current within the plasma, eliminating the need for plasma-embedded magnetic coils to achieve a rotational transform. Before the development of tokamaks, however, there were several confinement schemes studied that employed multiple current carrying magnetic coils embedded entirely within the plasma.[22, 23, 24] Although these alternative schemes had some highly desirable properties, plasma loss to the support structure of the embedded coils unacceptably deteriorated confinement. Magnetic plasma expulsion may provide an enabling technology for embedding multiple magnetic coils within a plasma, without significant particle loss at the connections to the embedded coils. Early research indicates that when losses to electrical connections and mechanical supports

[1]This chapter is presented in its entirety from Physics of Plasmas **25**, 012508 (2018) with permission from American Institute of Physics

are sufficiently small, classical plasma confinement via certain configurations of multiple plasma-immersed coils may occur.[22, 23, 24] It should also be noted that a Lockheed Martin team is investigating a magnetic fusion reactor concept, which uses multiple magnetic coils embedded within the plasma.[25] Additionally, it was recently reported that a net energy gain is theoretically possible in an inertial electrostatic confinement concept which also uses embedded magnetic coils.[26] This work may benefit both endeavors.

Collaborations such as ALPHA, ATRAP, ASACUSA, AEGIS, and GBAR are involved in studying antimatter in the form of antihydrogen.[27, 28, 29, 30, 31, 32, 33, 34] Current research is focused on comparisons of spectroscopic data between hydrogen and antihydrogen for CPT (charge conjugation, parity transformation, time reversal) symmetry violation studies and on gravity experiments.[29, 30, 35, 36, 37] Production of antihydrogen by ALPHA, ATRAP, and ASACUSA has been achieved via three-body recombination within plasmas confined by nested Penning traps. One of the difficulties limiting antihydrogen research is the quantity of low speed experimentally suitable antihydrogen that nested Penning traps can produce, due in part to particle drifts.[38, 39]

New alternatives to nested Penning traps are desirable that can generate low speed antihydrogen in much larger quantities than presently possible. Alternative approaches may also be desirable for achieving electron-positron plasma confinement. The study of such plasmas could enhance the understanding of new phenomena, such as the formation of magneto-bound states of positronium.[40] Magnetic plasma expulsion may provide one avenue for the advancement of alternative plasma confinement approaches that employ plasma-embedded magnetic coils.[26] Levitated multiple magnetic coil designs have been proposed,[41, 42] but physically supporting such coils and providing electrical connections might be accomplished with magnetic plasma expulsion techniques.

By way of example, an alternative plasma confinement concept based on finite-length coaxial solenoids is illustrated in 3.1. Using a pair of coaxial solenoids with oppositely directed magnetic fields, the magnetic mirror effect could be used to axially confine a plasma within the inner solenoid. In addition, the coaxial design would have “closed” magnetic field lines that do not cross any material structures at which plasma particles may be lost. Particles in the loss cone of the magnetic mirrors would recirculate back into the inner solenoid by following closed field lines. Such field lines are deflected around the cylindrical material surfaces of metal tubes located between the outer and inner solenoids by using a localized magnetic expulsion field as illustrated in 3.2. Mechanical supports and electrical connections could pass to the inner solenoid by being located inside of the cylindrical material surfaces.

Section 3.2 describes a particle-in-cell simulation used to study magnetic plasma expulsion. Section 3.3 describes a parametric study of magnetic plasma expulsion. A discussion and concluding remarks can be found in Sec. 3.4.

3.2. Simulation of Magnetic Plasma Expulsion

3.2.1. Magnetic Field Model

Suppose that one or more cylindrical objects are to be passed through a magnetized plasma in such a way as to minimize the effect that their presence has on the plasma's magnetic confinement. For example, cylindrical material surfaces could enclose electrical connections and/or mechanical supports. In the vicinity of the cylindrical material surfaces, assume that the magnetic plasma confinement approach produces an approximately uniform magnetic field. An additional field is to be added to keep the plasma expelled from the region near the cylindrical material surfaces. The total magnetic field is then a superposition of a uniform field and whatever additional field is used to achieve magnetic plasma expulsion. For a two-cylinder configuration, the expulsion field is approximated as the field produced by two straight, parallel, infinitesimally thin wires. The combined magnetic field is

$$B(x,y,z) = B_m\beta(\frac{x}{S},\frac{y}{S},\frac{z}{S}) + B_0\hat{\boldsymbol{k}}. \tag{22}$$

Here,

$$\beta(x_n,y_n,z_n) = \left[-\frac{z_n}{(x_n-1)^2+z_n^2}, 0, \frac{(x_n-1)}{(x_n-1)^2+z_n^2}\right] \tag{23}$$

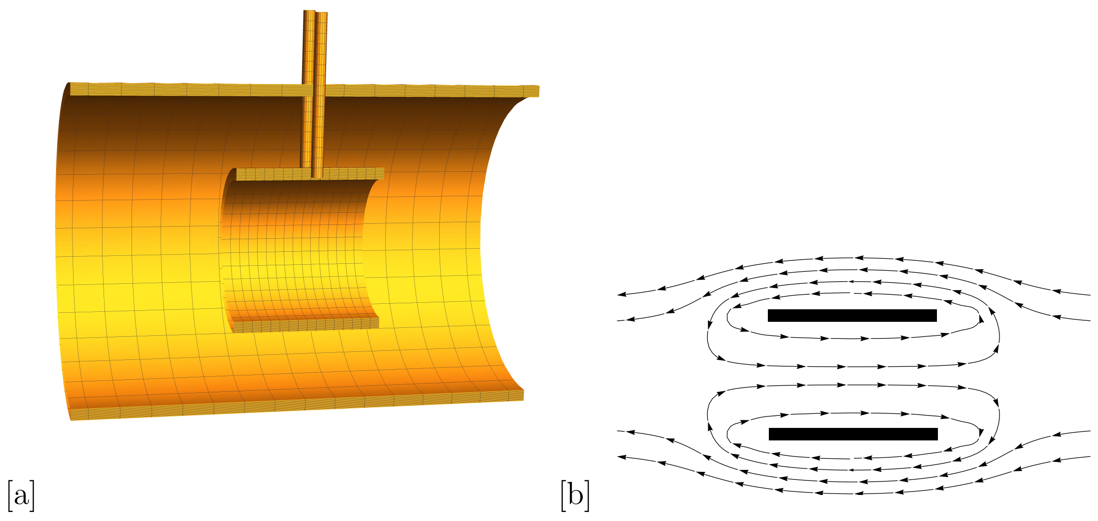

FIGURE 3.1. Illustration of an alternative plasma confinement approach consisting of two coaxial solenoids. (a) The cylindrical material surfaces of two metal tubes cross the space between the solenoids. These cylindrical material surfaces could enclose mechanical supports and electrical connections to the inner solenoid. (b) The magnetic field produced by the two coaxial solenoids. The outer solenoid is not shown. Plasma would be confined along closed magnetic field lines that enclose the inner solenoid. Two methods would be used to mitigate the loss of plasma particles to the cylindrical material surfaces. Magnetic mirrors would be formed from a field strength gradient along field lines between points within the interior solenoid, where the field is weaker, and points between the solenoids, where the field is stronger. Loss of plasma to the cylindrical material surfaces that cross the space between the two solenoids would also be inhibited by the use of magnetic plasma expulsion to redirect magnetic field lines around the cylindrical material surfaces.

$$-\left[-\frac{z_n}{(x_n+1)^2+z_n^2}, 0, \frac{(x_n+1)}{(x_n+1)^2+z_n^2}\right]$$

and $B_m = \mu_0 I/(2\pi S)$ is the standard expression for the magnetic field strength at a distance S from a single wire, with I the magnitude of current, and μ_0 the permeability of free space. (x, y, z) denotes Cartesian coordinates, while (x_n, y_n, z_n) denote Cartesian coordinates normalized by S. The currents carried by the two wires must flow in opposite directions to generate the field shown in 3.2. The current in these wires generates an expulsion field characterized by B_m, which is defined to be the magnitude of the magnetic field at the coordinate system origin generated by a single wire at a distance S. A uniform magnetic field approximating the confinement field is superimposed in the z direction parallel to the unit vector with magnitude B_0.

3.2.2. Computational Model

A study of plasma behavior is carried out with a Particle-in-Cell (PIC) simulation using the code Warp.[43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54] Warp has the underlying physics coded into the field solver and particle mover such that it solves for the potential from all particles at predetermined grid points, calculates the total electromagnetic fields experienced by each particle, and moves particles according to the Lorentz force law. Detailed explanations on PIC methods can be found in the literature.[55] A guiding center approximation is assumed for the particles during simulation by using an implicit "Drift-Lorentz" mover, which allows time steps larger than the gyro period to be taken. The positional uncertainty as a result of the implicit mover's inability to resolve the gyromotion is less than that introduced by the time step and considered negligible. A detailed description can be found in the literature.[56, 44]

With the exception of singularities at the wire locations $x = \pm S, z = 0$ in 3.2, the field can be discretized over a numerical mesh grid. A small offset is added to the field description during the discretization to ensure that the wire singularities parallel to the y

axis do not overlap the mesh grid. The discretized components of Eq. (1) are

$$B_x = (B_m S)\left[-\frac{k}{(i-S)^2+k^2+\delta}+\frac{k}{(i+S)^2+k^2-\delta}\right], \tag{24}$$

$$B_y = 0, \tag{25}$$

$$B_z = (B_m S)\left[\frac{i-S}{(i-S)^2+k^2+\delta}-\frac{i+S}{(i+S)^2+k^2-\delta}\right]+B_0. \tag{26}$$

Here, $i = (x_0 x_{step} - x_{shift})$, $k = (z_0 z_{step} - z_{shift})$, and $0 < x_0 < x_{grid}$, $0 < z_0 < z_{grid}$ in integer steps, where x_{grid} and z_{grid} are the number of grid points in their respective directions. x_{shift} and z_{shift} are added to center the coordinate origin of the simulation inside the simulation volume. x_{step} and z_{step} are the grid spacings for a given mesh resolution, and δ is a slight offset for the wire positions to ensure a field singularity doesn't land on a mesh point.

A simulated cubic volume with side L is created with particle absorbing boundary conditions on all faces. The space is divided up with a grid, and the external magnetic field described above is passed to Warp in component form on all the grid points. Dirichlet boundary conditions with potential set to zero are used on the boundaries of the simulation at $\pm L/2$ values of the x, y, and z coordinates.

Two cylindrical regions, one centered at each wire that produces the expulsion field, are each defined with a radius r and each represents a cylindrical material surface with a particle absorbing boundary.

3.2.3. Parameter Values and Base Results

Confinement of an electron-positron plasma is considered. Macroparticles of each species are introduced at starting points equally likely to be anywhere within two source volumes. The two source volumes are located at (-5 cm $< x_i <$ 5 cm, -5 cm $< y_i <$ 5 cm, -4.8 cm $< z_i < -4.75$ cm) and (-5 cm $< x_i <$ 5 cm, -5 cm $< y_i <$ 5 cm, 4.75 cm $< z_i <$ 4.8 cm) where x_i, y_i, z_i are the starting positions of the created macroparticles. Macroparticles are computational particles that represent a certain number of plasma particles determined by the macroparticle "weight." Since the Lorentz force law depends on the charge to mass ratio, and a macroparticle has the same charge to mass ratio as the plasma particles represented, a macroparticle will have the same trajectory as a single plasma particle. This allows for simulation of larger systems of plasma particles in a more computationally efficient manner. Both electron and positron macroparticles are introduced with fully Maxwellian velocity distributions in the x and y dimensions. Half Maxwellian velocity distributions are used in the z dimension, with either a positive or negative z velocity component, such that the macroparticles are initially directed into the simulated volume. The expulsion field generating wires are each displaced a distance $S = 1$ cm from the coordinate origin, and the magnetic plasma expulsion field parameter B_m is equal to the magnetic plasma confinement field strength B_0, with $B_m = B_0 = 1$ T. The $L = 10$ cm side cubic simulation volume is divided up with a $x_{grid} \times y_{grid} \times z_{grid} = 200 \times 200 \times 200$ grid, such that the grid spacing is $x_{step} = y_{step} = z_{step} = 0.5$ mm. $x_{shift} = z_{shift} = 5$ cm, and $\delta = 1 \times 10^{-50}$. The temperature of the electron-positron plasma is $T = 40$ K. 400 macroparticles are inserted into the simulated volume every time step, consisting of 100 electron macroparticles and 100 positron macroparticles created in each of the two source volumes.

Each simulation is run for a time $t_{max} = 90$ μs divided up into $90,000$ time steps of length $t_{step} = 1$ ns. The total run time is more than 22 times a 10 cm time-of-flight time based on the thermal speed of the plasma particles. The thermal speed is calculated as $v_{th} = (kT/m)^{1/2}$, where k is Boltzmann's constant, T is the temperature and m is the mass of a plasma particle. For a magnetic field strength of 1 T, the thermal gyroradius is $(mkT)^{1/2}/(eB) = 1.4 \times 10^{-5}$ cm where e is the elementary charge. The thermal gyroradius is three orders of magnitude smaller than the grid spacing, and the guiding center approximation is considered valid.

A line-averaged number density is used to determine a ratio Ω of the length $L = 10$ cm of the side of the cubic simulation volume to the column Debye length λ_D, $\Omega = L/\lambda_D$. Unless noted otherwise, all values of Ω are reported at time $t = t_{max}$. To calculate the column Debye length, the total number of macroparticles is determined in cell columns with dimensions $\Delta x = L/x_{grid}$, $\Delta y = L$, $\Delta z = L/z_{grid}$. The line-averaged number density is calculated as

$$n = \frac{wN_{positrons} + wN_{electrons}}{\Delta V}. \tag{27}$$

Here, $w = 10$ is the macroparticle weight equal to the number of plasma particles per macroparticle, $N_{positrons}$ is the number of positron macroparticles in the cell column with volume $\Delta V = \Delta x \Delta y \Delta z$, and $N_{electrons}$ is the number of electron macroparticles in the same cell column. Since the magnetic field has no y dependence, and the guiding center approximation is considered valid, line averaging effectively reduces the spatial distribution of the density to two dimensions. The line-averaged number density n is used to calculate the column Debye length in one cell column as $\lambda_D = [(\varepsilon_0 kT)/(nq^2)]^{1/2}$. Here, ε_0 is the permittivity of free space. Where $n = 0$, the column Debye length is set to $\lambda_D = 1 \times 10^{12}$ cm making Ω effectively zero. Values $\Omega > 1$ are where the column Debye length λ_D is smaller than the simulated length scale L. Values $\Omega > 200$ are where the column Debye length is smaller than the grid spacing, making the simulation susceptible to numerical instabilities. Values of Ω are used to generate the contour plot in 3.3, based on a simulation that had the parameter values defined thus far. 3.3 shows a density-type plot of Ω, with the cylindrical material surfaces moved to locations chosen for a base case. In the base case, the cylindrical material surfaces have a radius $r = 0.5$ cm each and are centered at $x = \pm S, z = 0$. The cylindrical material surfaces extend the entire length of the simulated y dimension. Small areas of low density can be seen at the four corners of 3.3. These artifacts stem from the magnetic field curvature in the simulation region as seen in 3.2. Magnetic field lines near the simulation borders curve into the boundaries of the simulation, which in turn cause artificial losses to particle populations following these field lines. The losses do not affect the plasma behavior near the cylindrical material surfaces. It is possible for macroparticles that are initially created near $x = 0$ and that have a sufficiently large velocity component in the z dimension to pass between the cylindrical material surfaces. Such behavior was more prevalent in a previously reported classical trajectory Monte Carlo study,[18] when a monoenergtic beam was considered to be incident on a similar magnetic field configuration.

To characterize how effective the expulsion field is, a normalized loss rate κ is defined. κ is defined as the rate of particle loss to the cylindrical material surfaces with $B_m > 0$ divided by the rate of particle loss to the cylindrical material surfaces when $B_m = 0$. Smaller values for κ are considered more desirable, indicating fewer macroparticles are able to cross into the expulsion region. To determine the normalized loss rate κ, two other parameters are defined. χ is defined (with $B_m > 0$) as the number of particles lost to the cylindrical material surfaces between times $t_{max}/2$ and t_{max} divided by the number of particles introduced during the same time period. χ_{max} is defined as the maximum fraction of particles that would be lost with $B_m = 0$, and χ_{max} is calculated analytically. In the case where no expulsion field is present, $B_m = 0$, the fraction of all simulated particles lost to the cylindrical material surfaces should approach the fractional area $\chi_{max} = A_{cms}/A_{sim}$ in the limit where t_{max} tends to infinity and the thermal gyro radius tends to zero. Here, $A_{cms} = 2(2rL)$ is the cross-sectional area of both volumes enclosed by the cylindrical material surfaces in the $z = 0$ plane, and $A_{sim} = L^2$ is the cross-sectional area of a source region. The values $\chi_{max} = A_{cms}/A_{sim} = 0.1, 0.2, 0.3, 0.4$ are used for values of $r = 0.25, 0.5, 0.75, 1.0$ cm, assuming that the guiding center approximation is valid, and that t_{max} is sufficiently large. The parameter κ is calculated as $\kappa = \chi/\chi_{max}$. For the base case, 14macroparticles are lost the the cylindrical material surfaces between $t_{max}/2$ and t_{max} while 1.8×10^7 macroparticles are injected in the same time period. With $\chi_{max} = 0.2$, the value of the normalized loss rate is $\kappa = 3.9 \times 10^{-6}$.

3.3. Parametric Study of Magnetic Plasma Expulsion

3.3.1. Limits on Parameter Value Variations

There are some computational considerations regarding macroparticle weight and plasma temperature. Macroparticle weights that are too large can lead to numerical instabilities, while large plasma temperatures can lead to unacceptable positional uncertainty. A series of simulations with variations in macroparticle weight or plasma temperature were run to determine how each parameter individually affected the normalized loss rate κ with the magnetic expulsion field turned off, $B_m = 0$. If B_m is set to zero, the value of the normalized loss rate κ is expected to be near unity.

The results for the normalized loss rate κ versus macroparticle weight variation with $B_m = 0$ are found in 3.4. As the macroparticle weight increased past a critical value, the values of the normalized loss rate κ exhibit a decrease attributed to the gradual onset of a numerical instability, which caused some particles to be reflected in the z dimension back toward a source region. Values of the macroparticle weight less than $w = 100$ exhibited less than a 15% deviation from the expected value of unity. For $w = 100$, the Debye lengths calculated for a single cell, hereafter referred to as “cell Debye lengths,” can reach values smaller than the grid spacing in some cells. For $w = 10$, all cell Debye lengths are larger than the grid spacing. If the cell Debye length is smaller than the grid spacing, numerical instabilities can occur as a result of non-physical interactions.[57]

At sufficiently high plasma temperatures, macroparticles can cross appreciably large sections of the simulated volume in one time step. An erroneously low value for the normalized loss rate κ can be reported at high plasma temperatures, because a macroparticle can have a large enough speed to travel through sections of the cylindrical material surfaces in a single time step and not be counted as lost. A plasma temperature study was performed to determine the compatibility of the choice of time step with various plasma temperatures by varying the value of T with $40 \text{ K} \leq T \leq 4.0 \times 10^7 \text{ K}$, with $B_m = 0$, and with the macroparticle field solver turned off. 3.5 shows the resulting normalized loss rate κ versus plasma temperature. The field solver is turned off so as to observe particle motion that comes solely from thermal energy and interaction with the external magnetic field configuration. The blue circles are simulated data points, while the yellow squares are values predicted by a model. For the model, the guiding center approximation is assumed valid, such that all particles move only in the z direction along uniform magnetic field lines with infinitesimally small gyroradii. The model is defined with a coordinate system translation in the x dimension on the range $0 < x \leq r$, where $r = 0.5$ cm is the radius of a cylindrical material surface region and $x = 0$ is the cylindrical material surface center in the x dimension. Since $B_m = 0$, any particle starting within $0 < x \leq r$ will follow a uniform field line, intersect with the cylindrical material surface and should be counted as lost. However, determining if a particle meets the criterion for being lost only occurs at certain times separated by time steps. A spatial step $\delta z = v_z t_{step}$ is defined as the distance a particle moves during a single time step t_{step} traveling at speed, v_z. For the model, a normalized loss fraction is defined as $\kappa_{model} = \dfrac{\chi}{\chi_{max}} = P_{detect}$, where χ is written as $\chi = P_{detect}\chi_{max}$. P_{detect} is the probability that a particle is detected by the simulation as being incident on a cylindrical material surface and thus lost. Particles are equally likely to start anywhere between $0 < x \leq r$, such that $f_d(x) = 1/r$ is the probability density function describing the particle initial x coordinates. Particles have a half Maxwellian initial velocity distribution, such that a probability density function for particle velocity can be written as

$$f_v(v_z) = \left[\int_0^\infty e^{-mv_z^2/(2kT)}\, dv_z\right]^{-1} e^{-mv_z^2/(2kT)}. \tag{28}$$

With the two probability density functions, P_{detect} is

$$P_{detect}= \int_0^\infty \int_0^r P f_d f_v \, dx \, dv_z, \tag{29}$$

where P is the probability for a particle with initial coordinate (x, v_z) to be detected,

$$\text{P}= \begin{cases} \dfrac{c}{\delta z} & \delta z > c \\ 1 & \delta z \leq c \end{cases}$$

and $c = 2\sqrt{r^2 - d^2}$ is the chord length across the cylindrical material surface at coordinate x. Numerical integration yields the model in 3.5.

By taking $B_m = 0$, and with the macroparticle field solver turned off, the expected normalized loss rate approaches one, $\kappa \to 1.0$, as $t_{max} \to \infty$. A significant deviation from the expected normalized loss rate $\kappa = 1.0$ is attributed to the positional uncertainty inherent at higher plasma temperatures. As T increases, those temperatures that recover within 15% the expected normalized loss rate ($\kappa \geq 0.85$) are considered to be within acceptable limits for predicting particle loss to the cylindrical material surfaces. Those plasma temperatures that show a larger deviation from the expected loss rate ($\kappa < 0.85$) in 3.5 would require a smaller time step to more accurately simulate magnetic plasma expulsion. Temperatures below $T = 4.0 \times 10^6$ K are considered to be within acceptable limits (see 3.5).

3.3.2. Debye Length

A change in the column Debye length of the plasma can be accomplished by varying either the plasma temperature or the macroparticle weight. The effect of plasma temperature and macroparticle weight variation on the normalized loss rate κ is compared using a parameter Ω_p, which is defined to have the value of Ω at a point away from the distortion at $x = 4.0$ cm, $z = 4.0$ cm and is intended to represent the value of Ω in the undisturbed plasma. 3.6 shows the results. The maximum value of κ in 3.6 is $\kappa = 7.25 \times 10^{-5}$. For the value of $\Omega_p > 100$, κ is zero. The simulated data are taken using base case values, other than the parameter that is varied. For the plasma temperature variations, a macroparticle weight of $w = 10$ is used, while for the macroparticle weight variations, a plasma temperature of $T = 40$ K is used. In the base case, the value of Ω_p is $\Omega_p = 46.05$, which, for $T = 40$ K, corresponds to a line-averaged density of 4.04×10^{10} m^{-3}.

3.3.3. Magnetic Plasma Expulsion Field and Cylindrical Material Surface Radius

A parameter scan is performed over $0.5 \text{ T} \leq B_m \leq 4.0 \text{ T}$ in increments of 0.5 T, and over $0.25 \text{ cm} \leq r \leq 1.0 \text{ cm}$ in increments of 0.25 cm, with other parameters having base case values. 3.7 shows values of the normalized loss rate κ for the permutations of r and B_m. κ varies from a maximum value of $\kappa = 0.059$ for $r = 1.0$ cm and $B_m = 0.5$ T, to a minimum value of $\kappa = 1.11 \times 10^{-6}$ for $r = 0.25$ cm and $B_m = 2.5$ T. For a given value of r, κ is larger than 0.01 for B_m less than a threshold value. The value of κ is smaller than 10^{-5} for values of B_m greater than the threshold value. The threshold value of B_m increases with r.

3.3.4. Magnetic Plasma Confinement Field

3.8 shows the results of a variation of the uniform field strength B_0. Despite being an implicit particle mover, the drift-Lorentz mover in Warp retains information about the gyroradius. When the gyroradius sizes are sufficiently large, the values for the normalized loss rate κ can be affected. The gyroradius of a charged particle is $r_g = (mv_\perp)/(|q|B)$ where

m is the mass, $|q|$ is the absolute value of charge, B is the magnitude of the magnetic field, and $v_\perp$ is the particle's velocity component perpendicular to the magnetic field. Due to a non-negligible gyroradius, macroparticles that would otherwise follow magnetic field lines around the distortion can be lost to the cylindrical material surfaces causing an increase in the normalized loss rate. The increase in κ in 3.8 is attributed to the effect of a finite gyroradius size.

3.3.5. Charge Separation

A phenomenon that should be considered is that of charge separation. Charge separation may occur as plasma particles travel around the expulsion region due to a sign dependent cross-magnetic-field drift experienced by the plasma particles. The resulting non-neutrality causes an electric field to form. To assess the effect, a simulation is run with base case parameter values except for a variation of the uniform magnetic field strength, 5.0×10^{-6} T $\leq B_0 \leq 0.5$ T. Due to the symmetry of the simulation about the $x = 0$ plane, the y component of the electric field on only one side of the expulsion region, along 1.5 cm $\leq x \leq 5$ cm, $y = 0$, $z = 0$ is recorded. The maximum value of the electric field at $t = t_{max}$ is used to estimate the maximum $\mathbf{E} \times \mathbf{B}$ drift velocity of the particles as $v_{drift} = \dfrac{|\mathbf{E} \times \mathbf{B}|}{B^2}$. 3.9 shows the results of the study. For magnetic field values near the base case, the maximum drift velocity is several orders of magnitude below the thermal speed.

3.3.6. Particle Mass

An electron-positron plasma has been considered up to this point, with all particles having the same mass. The effect of changing the particle mass is evaluated by also considering a proton-antiproton plasma. Two simulations are compared. One simulation is the base case electron-positron plasma. The other simulation is a proton-antiproton plasma with base case parameter values, except that the time step is increased by a factor of the square root of the mass ratio, $t_{step} = 42.85$ ns. With such a change, equal numbers of particles are introduced in the two simulations, and the particles travel the same average distance during a time step.

For the electron-positron base-case simulation, 14 macroparticles are lost to the cylindrical material surfaces between times $t_{max}/2$ and t_{max}, 1.8×10^7 macroparticles are injected into the simulation volume during the same time period, $\chi_{max} = 0.2$, and the value of the normalized loss rate is $\kappa = 3.9 \times 10^{-6}$. For the proton-antiproton plasma simulation, 16 macroparticles are lost to the cylindrical material surfaces between times $t_{max}/2$ and t_{max}, 1.8×10^7 macroparticles are injected into the simulation volume during the same time period, $\chi_{max} = 0.2$, and the value of the normalized loss rate is $\kappa = 4.4 \times 10^{-6}$. Figure 3.10 shows a density plot of the proton-antiproton plasma. The magnetic plasma expulsion behavior is nearly the same as in Fig. 3.3 for an electron-positron plasma. However, there is a noticeable stream of particles traversing the gap between the two cylindrical material surfaces, indicating that plasma particles with greater mass may more readily pass between the cylindrical material surfaces.

Two more simulations are compared with the same parameter values as the preceding two, except with the value $r = 0.25$ cm. Three (3) macroparticles are lost to the cylindrical material surfaces in both the electron-positron plasma simulation and the proton-antiproton plasma simulation, corresponding to $\kappa = 1.6 \times 10^{-6}$ for each simulation.

3.4. Discussion and Concluding Remarks

Magnetic plasma confinement approaches have been proposed in the past that would require magnetic-field-producing coils to be embedded within the confined plasma. The development of magnetic plasma expulsion techniques may provide an enabling technology to shield mechanical supports and electrical connections attached to such embedded coils by redirecting the local magnetic field near such structures.

By way of example, a plasma confinement approach using magnetic plasma expulsion was described (see 3.1). This approach would use two coaxial solenoids, with a pair of metal tubes passing between the solenoids. Mechanical supports and electrical connections could be contained in the tubes. There would exist magnetic field lines that would pass through the inner solenoid and also the metal tubes were it not for the implementation of magnetic plasma expulsion. Due to the implementation of magnetic plasma expulsion, these field lines become closed and do not intersect with the metal tubes. Plasma particles following such field lines could then recirculate be into the inner solenoid where the magnetic field is weakest. Plasma within the inner solenoid may be confined by a magnetic mirror with closed field lines. As noted, plasma particles do pass between the tubes. It may be possible to utilize the magnetic null lines generated by the localized field distortion as a source of particles for injection into the device. Such injection schemes could help to control the neutrality, density and temperature of the confined plasma.

Investigation of the concept of magnetic plasma expulsion was carried out here using a Particle-in-Cell simulation. A system of magnetically confined electron-positron plasma near cylindrical material surfaces was modeled. Without magnetic plasma expulsion, plasma particles could follow field lines that pass intersect with the metal tubes and be lost. A figure of merit κ was defined as the ratio of losses to the metal tubes both with and without magnetic plasma expulsion present. Values for the figure of merit were found to be much less than one for a number of parameter values. It is concluded that magnetic plasma expulsion may reduce the effects associated with material objects passing through an electron-positron plasma and possibly other plasmas with similar parameter values. Augmenting current plasma trap designs with magnetic plasma expulsion concepts may provide a way to trap larger amounts of plasma for study. Magnetic plasma expulsion is also equally effective on plasma particles regardless of their charge sign which reduces the complexity of potential trap designs compared to electrostatic trap design equivalents.

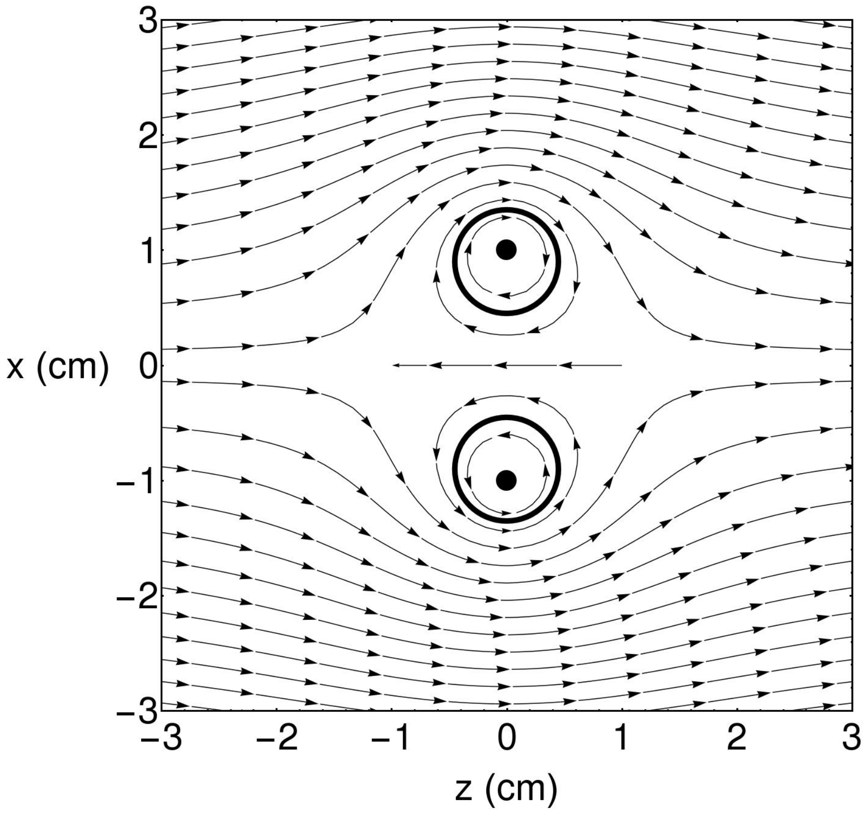

Figure 3.2. Illustration of the redirected magnetic field associated with magnetic plasma expulsion used in combination with magnetic plasma confinement. The plasma confinement field is approximated as being uniform near two cylindrical material surfaces (thick solid circles). The expulsion field is approximated as being produced by two straight wires (solid dots) located within the two cylindrical material surfaces. Loss of plasma to the cylindrical material surfaces would be inhibited by the use of magnetic plasma expulsion to redirect plasma along the modified magnetic field.

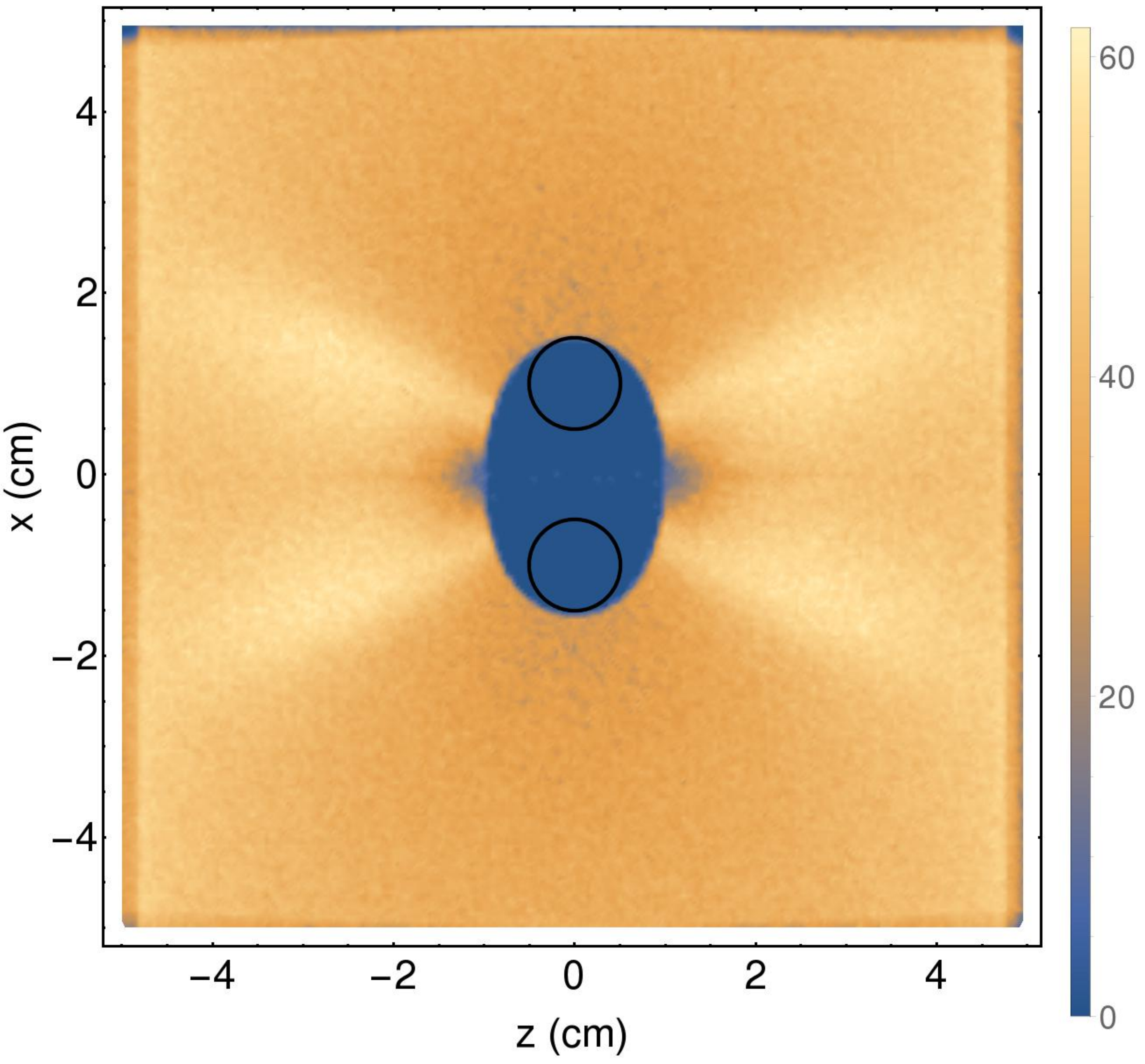

FIGURE 3.3. A density-type plot of Ω for the base case. The color at a given point represents the value of Ω at that point.

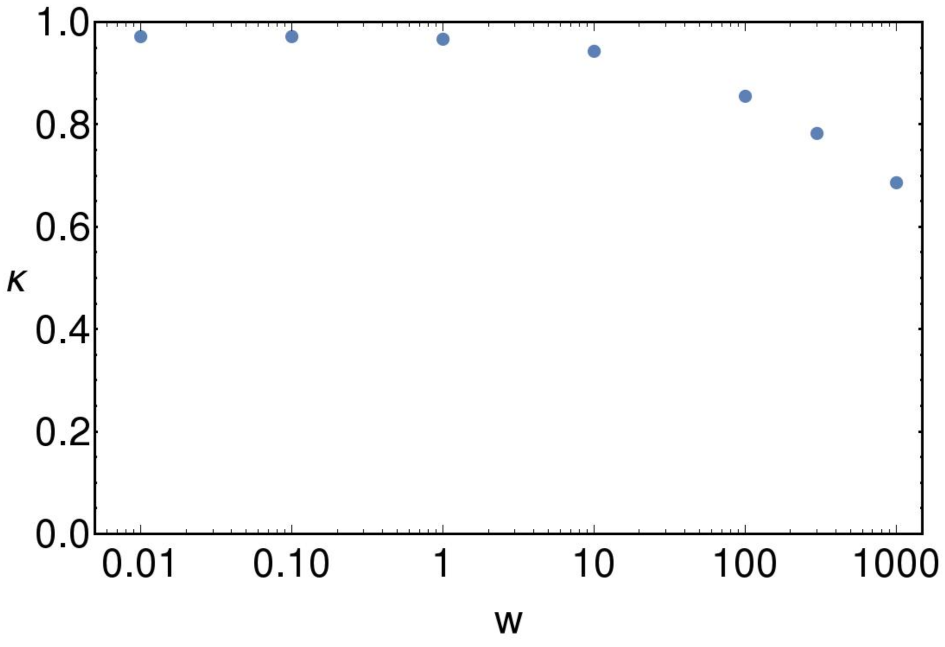

Figure 3.4. Normalized loss rate κ versus macroparticle weight w with base case parameter values, except with $B_m = 0$ and with different values for the macroparticle weight w.

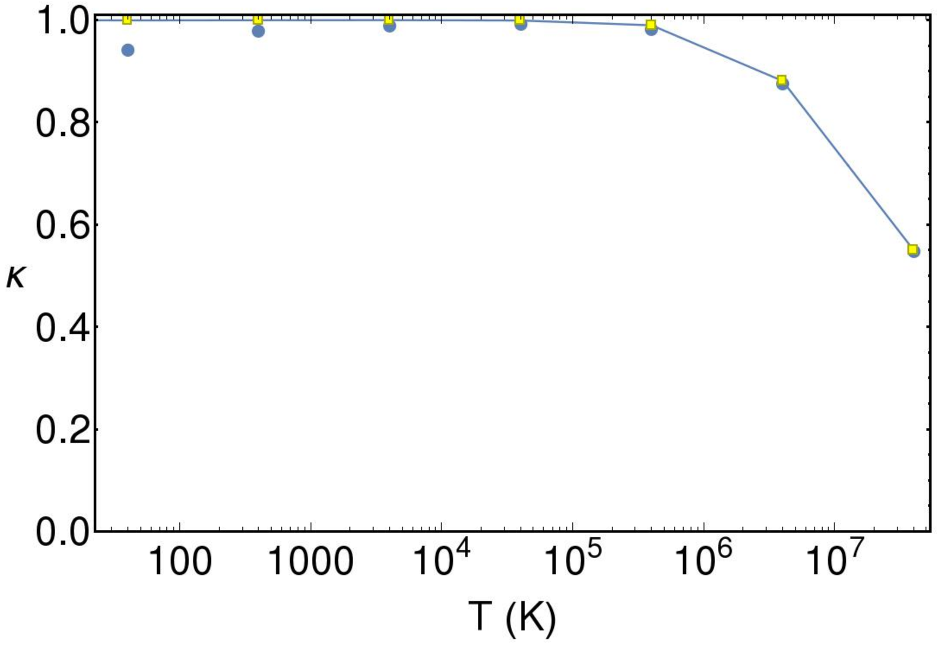

FIGURE 3.5. Normalized loss rate κ versus plasma temperature T. The simulations are run with base case parameter values, except with $B_m = 0$, the macroparticle field solver turned off, and the plasma temperature varied. Blue circles are simulated data, while yellow squares are values predicted by an analytical model.

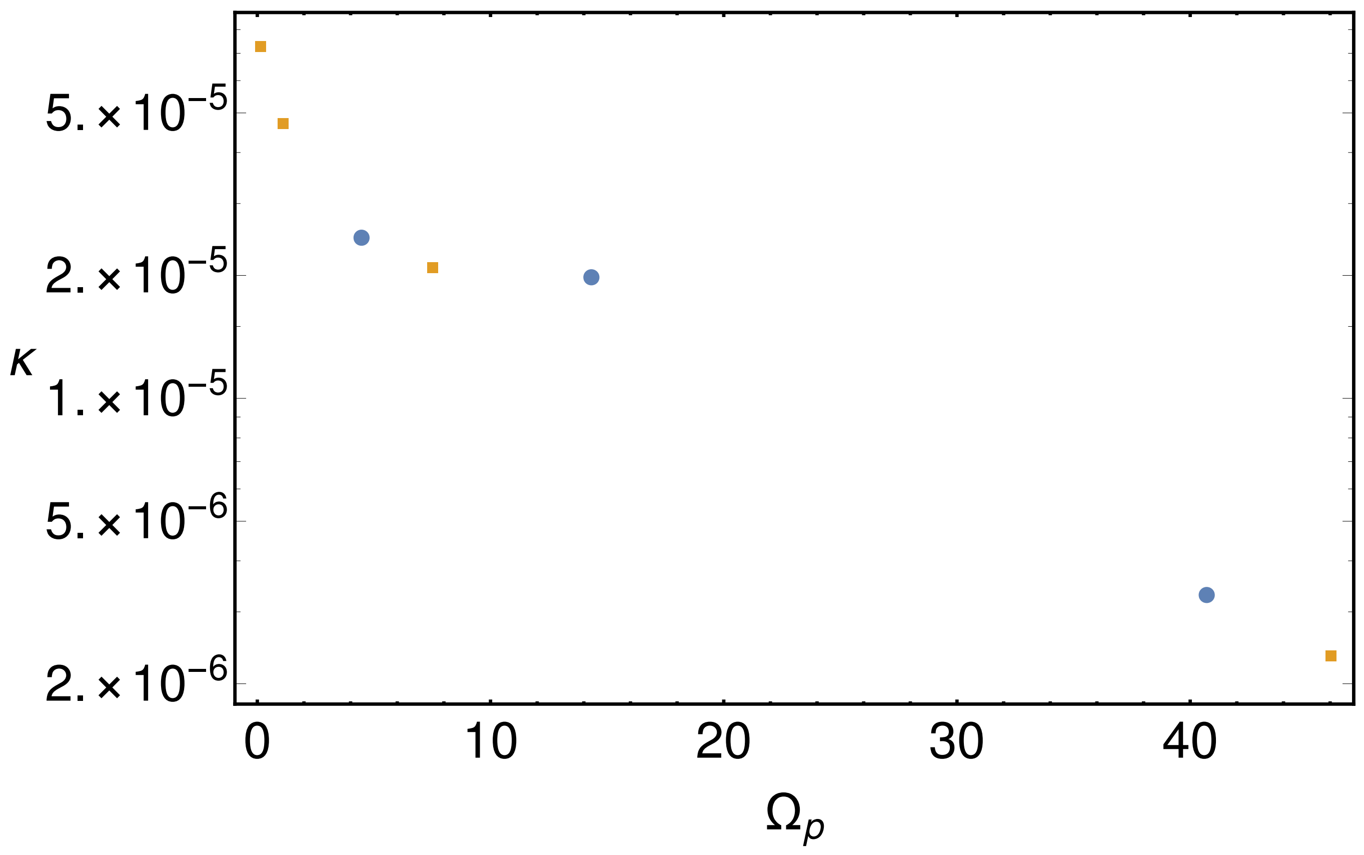

FIGURE 3.6. Normalized loss rate κ versus Ω_p. For variations of macroparticle weight (blue circular dots), the temperature is $T = 40$ K, and the values are $w = 0.1$ ($\Omega_p = 4.49$), $w = 1.0$ ($\Omega_p = 14.35$), $w = 10$ ($\Omega_p = 40.73$), $w = 100$ ($\Omega_p = 108.44$). For variations of plasma temperature (yellow squares), the macroparticle weight is $w = 10$, and the values are $T = 40$ K ($\Omega_p = 46.05$), $T = 400$ K ($\Omega_p = 7.53$), $T = 4,000$ K ($\Omega_p = 1.12$), $T = 40,000$ K ($\Omega_p = 0.14$). Otherwise, base case parameter values are used. The value of κ is zero for $\Omega_p = 108.44$ and the point is not shown.

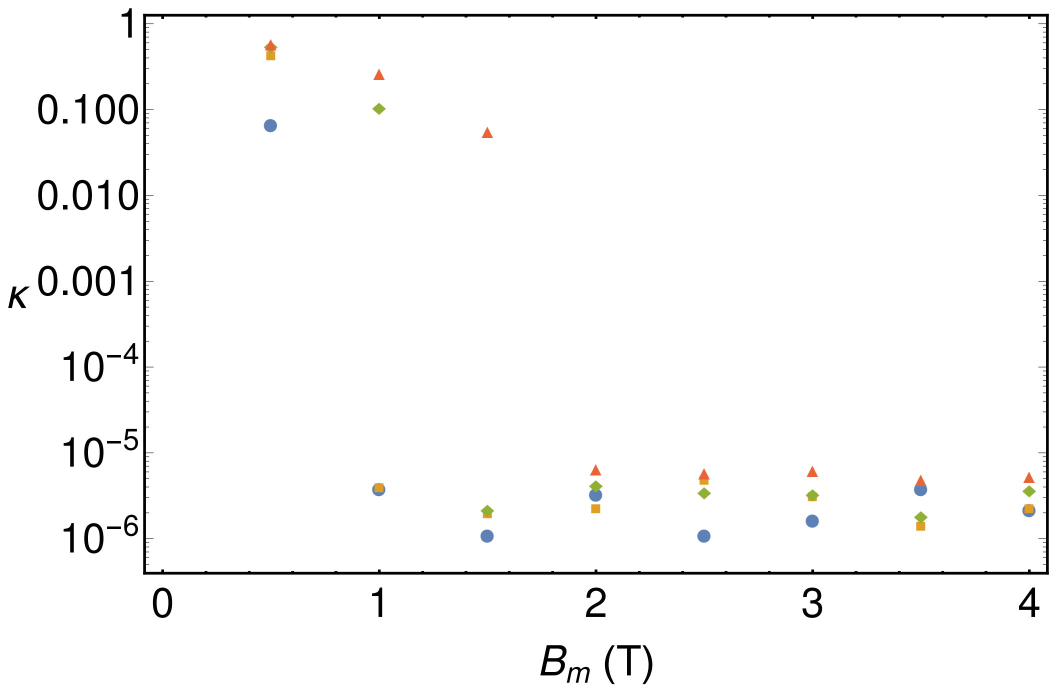

FIGURE 3.7. Normalized loss rate κ versus magnetic plasma expulsion field parameter B_m for $r = 0.25$ cm (circles), $r = 0.5$ cm (squares), $r = 0.75$ cm (diamonds) and $r = 1.0$ cm (triangles).

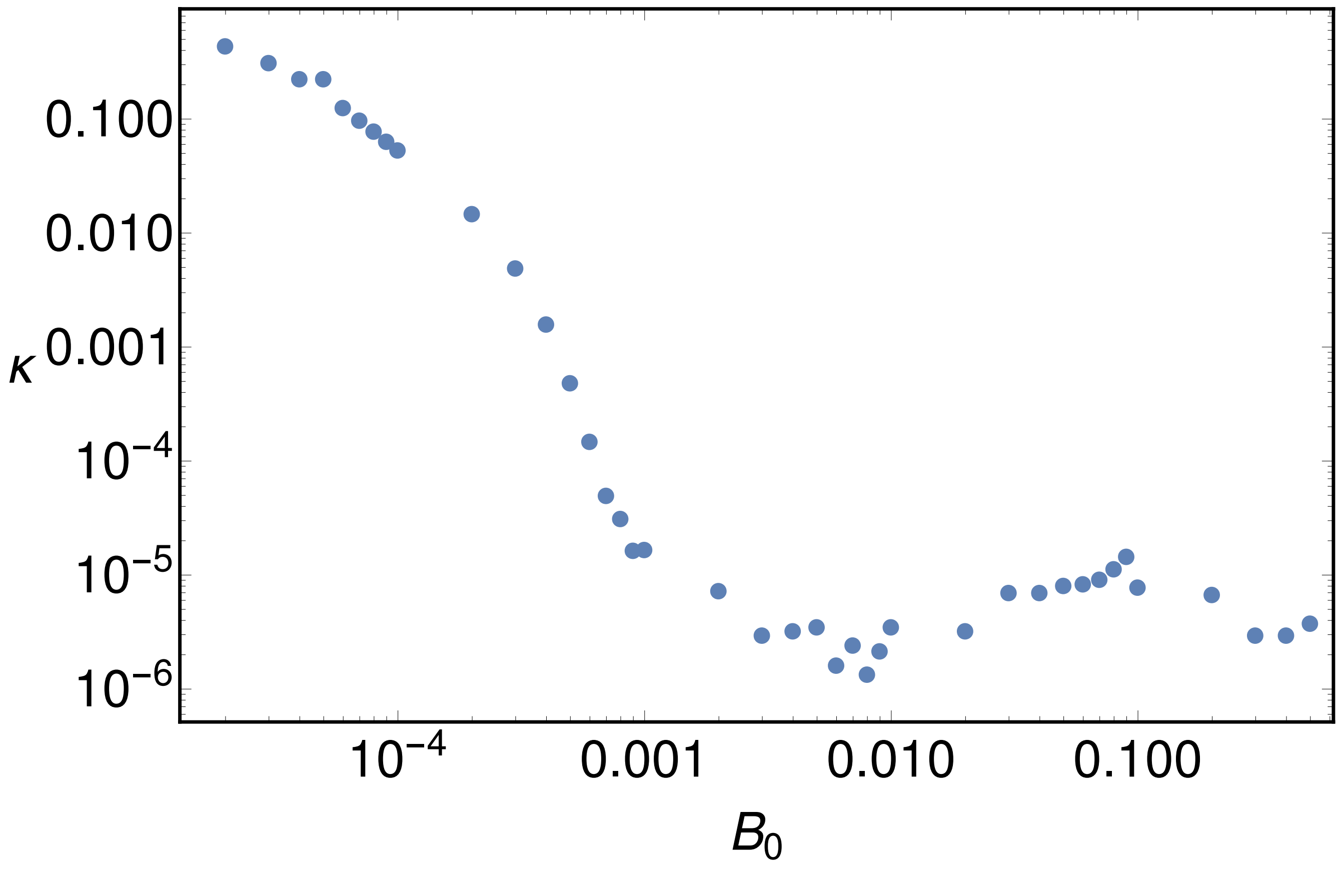

FIGURE 3.8. Normalized loss rate κ versus uniform magnetic field strength B_0 in Tesla, with base case values otherwise.

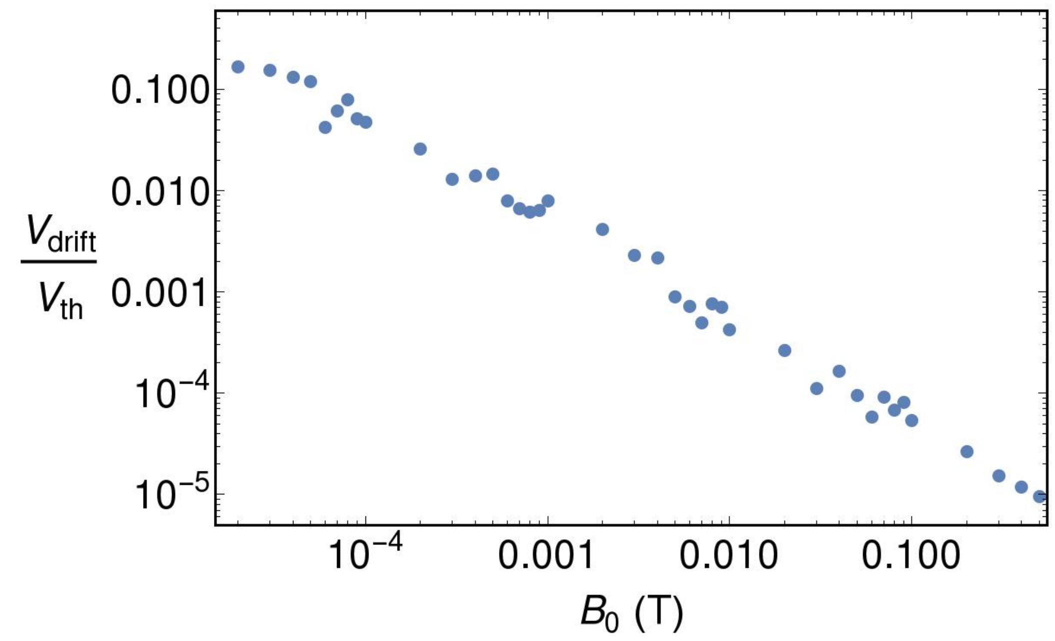

FIGURE 3.9. Maximum $\mathbf{E} \times \mathbf{B}$ drift speed nomalized by the thermal speed at $S + r \leq x \leq L/2$, $y = 0$, $z = 0$. All parameter values are the same as the base case, expect for the value of B_0.

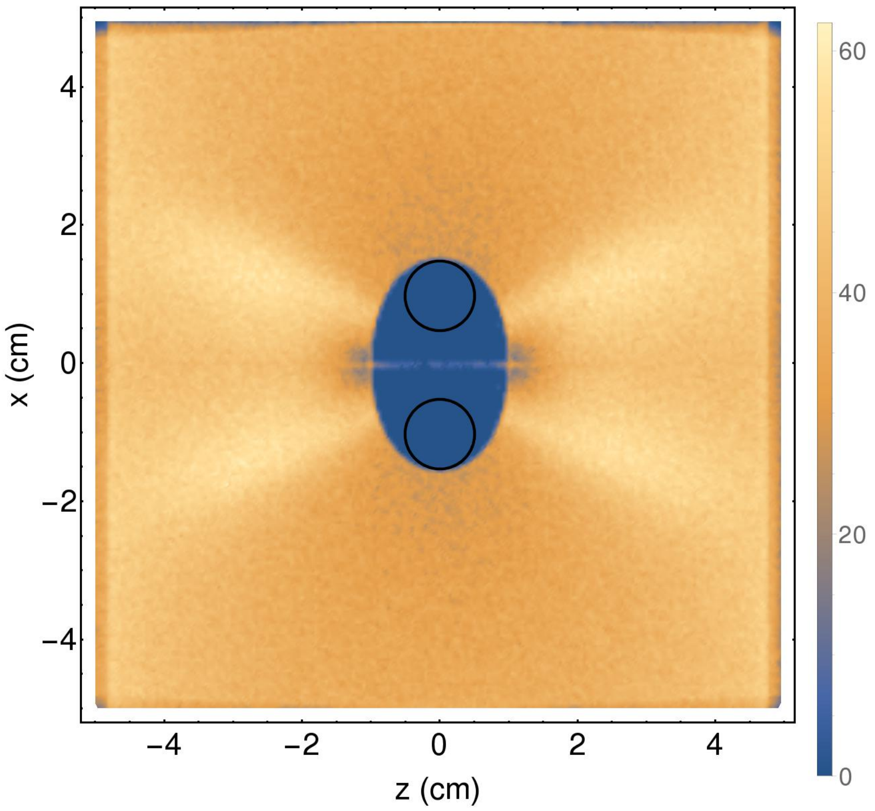

FIGURE 3.10. Same as 3.3, except for a proton-antiproton plasma, with an increased time step.

CHAPTER 4

ADDITIONAL CONSIDERATION FOR MAGNETIC PLASMA EXPULSION

4.1. Introduction

Magnetic plasma expulsion is defined as using a magnetic field to prevent plasma from entering a given region. Used in conjunction with traditional magnetic plasma confinement schemes, magnetic plasma expulsion may allow greater control over the motion of a confined plasma. Given that magnetic plasma expulsion relies solely on magnetic field distortions, magnetic plasma expulsion is predicted to be effective for both positive and negative charges and in the proposed configuration, does not have a time varying component.

An area in need of study with magnetic plasma expulsion is the effect mass differences between the positive and negative plasma species have on the magnetic plasma expulsion configuration. For example, due to a factor of 42.85 difference in the mass between an electron and a proton, both the gyro radius and the thermal speed of the particles will be significantly different. This difference in speed leads to distinctly different time scales of motion which could potentially lead to the development of phenomena not seen in the case of equal mass opposite charged plasmas such as plasma sheathing around the cylindrical material surfaces or asymmetric losses between the two plasma species. For applications of magnetic plasma expulsion to be developed, an understanding of how mass affects the plasma motion must be gained. Simulating the creation of equilibrium from first principles in an electron-proton plasma is computationally expensive however, as a result of the vastly different time scales involved in the plasma motion. Computational methods are explored to circumvent this using two same mass opposite charged pre-simulations to establish a preliminary equilibrium more quickly than from first principles.

Previous work only reported loss rates as a fraction of the simulated plasma with no consideration to exactly where those lost particles came from.[58] For practical configurations of magnetic plasma expulsion, all particles start on magnetic field lines that do not intersect with anything in the central region, but particles are still lost because of cross magnetic field drift. Characterization of the phase space that lost particles come from in the simulation could yield improvements to the existing design and insight into the cross magnetic field drift process that leads to particle losses.

Depending on the methods used, a simulation in computational equilibrium may be drastically different than the same system in a physical equilibrium.[59] One way to check the soundness of a computational equilibrium in absence of experimental data, is to alter the computational bounds of the simulation and examine what if any effect the alteration has on the resulting computational equilibrium. Several permutations of a base case simulation are examined to determine what effects various computational boundaries have on the over all results of the simulation.

4.2. Establishing a Base Case

Magnetic plasma expulsion relies on a localized time independent magnetic field distortion to redirect plasma away from a given area. The field consists of an element approximating the confining field, and an element approximating the distortion field. As was the case in the previous work, the confining magnetic field will be approximated as a uniform magnetic field in the z direction with a value B_0, while the distorting magnetic field will be approximated as due to two infinitesimally thin infinite wires that produce a field strength of B_m at a distance S. The wires are offset from the origin of the simulation by the distance S such that the value B_m is measured at the simulation origin.[58] The following external magnetic field is imposed onto the simulation volume.

$$B(x, y, z) = B_m \beta(\frac{x}{S}, \frac{y}{S}, \frac{z}{S}) + B_0 \boldsymbol{\hat{k}} \tag{30}$$

with

$$\begin{aligned}\beta(x_n, y_n, z_n) = &\left[-\frac{z_n}{(x_n - 1)^2 + z_n^2}, 0, \frac{(x_n - 1)}{(x_n - 1)^2 + z_n^2} \right] \\ &- \left[-\frac{z_n}{(x_n + 1)^2 + z_n^2}, 0, \frac{(x_n + 1)}{(x_n + 1)^2 + z_n^2} \right]\end{aligned} \tag{31}$$

and $B_m = \mu_0 I/(2\pi S)$ is the standard expression for the magnetic field strength at a distance $S = 1$ cm from a single wire, with I the magnitude of current, and μ_0 the permeability of free space. The introduced magnetic field represents two current carrying wires creating magnetic plasma expulsion within a confinement field. The current in these wires generates an expulsion field characterized by B_m, which is defined to be the magnitude of the magnetic field at the coordinate system origin generated by a single wire at a distance S. A uniform magnetic field approximating the confinement field is superimposed in the z direction having magnitude B_0. For these investigations, $B_0 = 1.0$ T, and $1.0 \text{ T} \leq B_m \leq 4.0$ T. To carry out these studies, a particle-in-cell simulation is constructed similar to the previous work but with a few notable changes.[58] The invariance of the magnetic field in the y direction is leveraged to create a simulation that is functionally the same as previous work, but is computationally more efficient.[58] The simulation volume has side lengths $X_L = Z_L = 10$ cm and $Y_L = 1$ cm. With the increased computational efficiency, the resolution of the particle in cell grid is able to be increased from a cell volume of 1.25×10^{-10} m^3 to a cell volume of 8.0×10^{-12} m^3 which allows for a more accurate solution to the particle motion. Motion in the y dimension still contributes to the overall behavior near the field distortion so the simulation can not be fully two dimensional.

Cylindrical material surfaces centered on the current carrying wires have a radius of r and represent structural connections or instrumentation that is being passed through a confined plasma and needs to be shielded. Any particles that contact these surfaces are removed from the simulation and lost. In the base case and in previous work, these surfaces are purely computational boundaries.[58]

Losses are defined in the same manner as in previous work[58]. A normalized loss rate κ is defined to characterize losses to the cylindrical material surfaces. κ is defined as the rate of particle loss to the cylindrical material surfaces with $B_m > 0, B_0 > 0$ divided by the rate of particle loss to the cylindrical material surfaces when $B_m = 0, B_0 > 0$. χ is defined (with $B_m > 0, B_0 > 0$) as the number of particles lost to the cylindrical material surfaces between times $t_{max}/2$ and t_{max} divided by the number of particles introduced during the same time period. χ_{max} is defined analytically as the maximum number of particles that would be lost with $B_m = 0, B_0 > 0$ between times $t_{max}/2$ and t_{max} divided by the number of particles introduced during the same time period. In the case where no expulsion field is present, $B_m = 0, B_0 > 0$, the fraction of all simulated particles lost to the cylindrical material surfaces should approach the fractional area $\chi_{max} = A_{cms}/A_{sim}$ in the limit where t_{max} tends to infinity. Here, $A_{cms} = 2(2rL)$ is the cross-sectional area of both volumes enclosed by the cylindrical material surfaces in the $z = 0$ plane, and $A_{sim} = X_L Y_L$ is the cross-sectional area of a source region where X_L and Y_L are the lengths of the x and y simulation dimensions respectively. The value $\chi_{max} = A_{cms}/A_{sim} = 0.2$ is used for a cylindrical material surface radius of $r = 0.5$ cm, assuming that the guiding center approximation is valid, and that t_{max} is sufficiently large. The parameter κ is calculated as $\kappa = \chi/\chi_{max}$.

The parameter Ω is used to examine how plasma is distributed in the simulation volume. $\Omega = L/\lambda_D$ is defined as the longest dimension in the simulation $L = 10$ cm, divided by the line-averaged Debye length. Unless otherwise noted, all values of Ω are reported at $t = t_{max}$. λ_D is calculated as the measurement of the Debye length in each cell volume of the simulation, averaged in the y dimension such that all cells with the same x and z coordinates are averaged together to give λ_D. Where the number density in a given cell is zero and thus the Debye length incalculable, the value of Ω is set to zero. Values of $\Omega \geq 1$ represent areas where the Debye length of the local plasma is smaller than the largest simulated dimension. Values of Ω greater than the number of grid points in the largest dimension represent areas where the Debye length of the local plasma is smaller than the grid spacing used in the simulation.

Hereafter, reference to the base case simulation refers to a simulation with the following parameters: $B_m = B_0 = 1.0$ T, $S = 1.0$ cm, $r = 0.5$ cm, $X_L = Z_L = 10$ cm and $Y_L = 1$ cm. 40 particles per time step are introduced into the simulation with 10 electrons and 10 positrons coming from each of the two simulated emission regions at -5.0 cm $\leq x \leq 5.0$ cm, -5.0 cm $\leq y \leq 5.0$ cm, -4.92 cm $\leq z \leq -4.9$, and 4.9 cm $\leq z \leq 4.92$. The temperature for the plasma is $T = 40$ K and the timestep is $t_{step} = 1.0 \times 10^{-9}$ s with the simulation being run for $90,000$ time steps. The particles have a Maxwellian velocity distribution in x and y, and a half Maxwellian velocity distribution in z such that all particles are initially directed towards the $z = 0$ midplane of the trap. The computational grid for the particle-in-cell solvers is $500 \times 50 \times 500$ corresponding to the x, y, and z dimensions respectively. The field solver is an implicit solution method developed by the Warp developers called the drift-Lorentz mover that allows for much larger time steps than would be possible with an explict particle in cell field solver.[56] Figure 4.1 shows a density type plot of Ω for base case parameter values. The effect of various alterations to the base case parameters will be compared against this base result.

4.3. Efficient Equilibrium with Differing Masses

Simulating an electron-proton plasma equilibrating in an empty simulation volume is computationally very expensive. Due to the significant difference in thermal speeds, electrons and protons created in a space charge neutralized configuration at the edges of the simulation volume as in the previous work rapidly experience a charge separation as the electrons reach the middle of the simulation volume long before the protons do. This motion is transitory and will eventually damp out, but a very large number of time steps is required to reach an equilibrium condition. Plasmas with charged species of different masses are both common and important to the field however so an examination of magnetic plasma expulsion in the context of differing mass species is necessary. To circumvent the thermal speed mismatch, simulations of differing mass species are run in two stages: A 'primer' stage and a 'seeded' stage for each given combination of parameters. The primer simulations are run with equal mass opposite charge species. This setup allows the simulated particle injection to remain space charge neutralized and equilibrium forms much faster because there is no need to wait for charge separation to damp out. The species chosen for the primer simulations are electron-positron, and proton-antiproton. These two-species plasmas will equilibrate into very similar equilibrium configurations. Once equilibrium is achieved, the phase space of all the particles is recorded. The recorded phase space of the electrons and protons is passed to the seeded electron-proton simulation and used to create an equilibrium starting point from which the seeded simulation advances using a time step that resolves the lighter species motion.

For the primer simulations, 40 macroparticles with a weight $w = 10$ and a temperature $T = 40$ K were introduced into the system each timestep. The species used are equal mass opposite charged species and have Maxwellian velocity distributions in the x and y dimensions and a half Maxwellian velocity distribution in the z dimension such that all created particles travel towards the $z = 0$ midplane of the simulated volume. 10 positive particles and 10 negative particles are introduced from two emission regions with dimensions (-5 cm $< x_i <$ 5 cm, -5 cm $< y_i <$ 5 cm, -4.9992 cm $< z_i < -4.999$ cm) and (-5 cm $< x_i <$ 5 cm, -5 cm $< y_i <$ 5 cm, 4.999 cm $< z_i <$ 4.9992 cm) where x_i, y_i, z_i are the starting positions of the created macroparticles. The simulation is run for $t_{max} = 90000$ timesteps. For the electron-positron simulation, $t_{step} = 1$ ns. For the proton-antiproton simulation, $t_{step} = 42.85$ ns to account for the mass difference in thermal motion and allow both simulations to have similar time of flights for their respective species. The grid for the primer simulations is $500 \times 50 \times 500$ in the x, y, and z dimensions respectively. The implicit drift-Lorentz mover is used in each two-species plasma.

For the seeded simulations, the electron and proton phase spaces from the corresponding primer simulations are used to establish an equilibrium particle population in the simulation volume. A time step of $t_{step} = 1$ ns is used to resolve electron time scale motion using the implicit drift-Lorentz field solver in Warp. Particles are injected into the simulation only when particles of the equilibrium population exit the simulation or are lost to the cylindrical material surface regions, and new particles are regenerated using the same parameters as in the primer simulations in order to maintain the established equilibrium. Figure 4.2 shows a density type plot of Ω for the electron-proton plasma equilibrium established in the seeded simulation. The plot is functionally identical to Fig. 4.1 indicating the equilibrium established in the primer sims is maintained in the seeded sim. Particle losses increased to 4 macroparticles lost between $t = t_{max}/2$ and $t = t_{max}$. Figure 4.3 shows the $z = 0$ midplane array of potential values at each grid point in the electron-proton plamsa. The blue areas indicate areas of negative potential as a result of the accumulation of negative space charge. Due to their smaller cyclotron radius, electrons are able to drift closer to the cylindrical material surface regions without being lost, and over time an electron-confining sheath appears to form near the cylindrical material surface regions. Further investigations are needed to determine the longer term effects of this sheath. Sheathing formation was not seen in the case of equal mass opposite charge species, indicating the effect is dependent on a mass difference between the two species.

4.4. Characterization of Losses in the Base Case

In previous work, particles lost to the computational bounds of the cylindrical material surface regions were recorded in aggregate and reported as the value κ with no consideration for their originating phase space.[58] An analysis of the phase space lost particles originate from in the base case is conducted here to gain insight into the cross magnetic field motion necessary for particle to be lost to the cylindrical material surface regions. In a single base case simulation, so few particles are lost that an examination of the lost particle phase space would not be possible with proper statistics. To accumulate the proper statistics, the base case simulation is run repeatedly recording the birth phase space of each particle lost until an acceptable number of lost particles are accumulated. A population of 120 lost electrons and 120 lost positrons is accumulated using this method.

Figure 4.4 shows a histogram of observed x starting positions of lost particles in the base case. All particles lost in the base case were observed to originate within the central 1% of the x dimension directly in line with the magnetic null regions in the simulation volume. Particles that enter the simulation in the central 1% of the x dimension interact with the magnetic null regions that exist on the $x = 0$ plane as a result of the superposition of the two external magnetic fields. For base case simulation parameters, particle interaction with the null regions appears to be necessary for particles to be lost.

Figure 4.5 shows where on the cylindrical material surface regions the lost particles impacted. Cross referencing the particle loss locations with Fig. 3.3 reveals that the locations particles are lost at coincides with the area where plasma most closely approaches the cylindrical material surfaces, and the area where the local magnetic field has the strongest curvature.

To characterize what caused particles to be lost to the cylindrical material surface regions, a version of the base case simulation is run where particle-electric field interactions are turned off, and only the particles lost in previous simulations are introduced. Since achieving an equilibrium state is not a concern, the simulation is allowed to run for $t_{step} = 45,000$ time steps for computational expediency. Particles that are still lost in this case are considered lost due to cross magnetic field drift associated with passing near the null region. Particles that are not lost in this case are considered lost in the base case as a result of both passing near the null region and interacting with the surrounding plasma's electric field flucuations. Figure 4.6 shows a snapshot of the simulation with particle-electric field interactions turned off at step $t_{step} = 17,939$. Many of the particles previously lost in the base case simulation now exhibit stable closed loop orbits very near the cylindrical material surface regions instead of impacting onto the surfaces. This observation combined with the highly localized origination of lost particles seen in Fig. 4.4 supports the hypothesis that losses to the cylindrical material surface regions are primarily caused by particles interacting with the null regions, drifting onto closed loop magnetic field lines around the cylindrical material surface regions, and then particles transport into the cylindrical material surface regions by interactions with the plasma at their outer edges.

Some particles were lost in the case where particle-electric field interactions were turned off. Of the 120 electrons, 46 were lost. Of the 120 positrons, 35 were lost. These particles are considered lost as a result of null region induced cross magnetic field drift.

Another feature observed in Fig. 4.6 is the clear separation of charge species the cause of which currently remains an open question. The author speculates that the cause of this apparent separation of losses by charge sign is due to the sign dependence of the direction of gyro motion. Positive particles gyrate in one direction, while negative particles gyrate in the opposite direction. When the lost particles interact with the null region prior to their loss, this sign dependent directionality causes the particles to self sort.

4.5. Effect of Different Computational Boundaries

Computational equilibrium reached during simulations of magnetic plasma expulsion is examined. Magnetic plasma expulsion refers to using a distortion of a confining magnetic field to redirect a plasma around a given area. In the envisioned application, this area would be populated with instrumentation or even parts of the containment device and shielded from exposure to the confined plasma. These components could also potentially be biased, grounded, or insulated. A key feature to study in characterizing magnetic plasma expulsion is the equilibrium behavior of the plasma near the distortion field.

Interior conductors and Neumann boundary conditions are imposed on the simulation and compared to a standard simulation similar to the base case to examine the effect computationally imposed boundary conditions have on the equilibrium conditions of the simulated plasma and the recorded losses.

For this study, base case parameters are used except for the following: $r = 0.6$ cm for all three cases, Neumann boundary conditions are used on the simulation edges in one simulation, and conductors biased to zero volts are used instead of an artificial computational boundary for the cylindrical material surface regions in the third. An increase in the radius r ensures a sizable number of particles will be lost to the central regions which allows for a more accurate comparison of the effects of the various computational boundaries on the number of lost particles. 'Case 1' will refer to the simulation that has base case parameter values except $r = 0.6$ cm, 'Case 2' will refer to the simulation with $r = 0.6$ cm and electrically grounded conductors used as the cylindrical material surface regions, and 'Case 3' will refer to the simulation with $r = 0.6$ cm and Neumann boundary conditions at the simulation edges. For Case 1 the number of macroparticles lost between $t_{max}/2$ and t_{max} was $12,942$. For Case 2 the number of macroparticles lost between $t_{max}/2$ and t_{max} was $12,811$. For Case 3 the number of macroparticles lost between $t_{max}/2$ and t_{max} was $12,953$. The three loss measurements are almost within 1% of each other indicating that different computational boundaries have a negligible effect on the overall particle loss estimates.

Computational boundaries could potentially affect more than just the particle losses, such as the structure of the equilibrium plasma distribution. To compare the three cases, the distribution of Ω is used. Figures 4.7, 4.8, and 4.9 show the results. An examination of the equilibrium plasma distribution as indicated by Ω however reveals nearly identical equilibrium structures suggesting that the computational boundaries used have a negligible effect on the overall equilibrium as well as the particle losses.

4.6. Conclusion

Additional characterizations of magnetic plasma expulsion were conducted to expand on previous work.[58]

Using primer simulations with equal mass opposite charge species proved effective at accelerating the simulation time required to achieve a computational equilibrium in a two species plasma with differing masses. The equilibrium achieved in the seeded simulation does not appear to deviate substantially from the equilibrium previously established in the primer simulations. Sheathing formation was observed in the seeded electron-proton simulation. Further examination of sheathing formation is needed, as the phenomenon was not seen in the electron-positron plasma simulations.

Characterization of the initial phase space of particles lost to the cylindrical material surface regions revealed that for base case parameter values, all lost particles originate very near the $x = 0$ plane, while all losses occurred on the outside edges of the cylindrical material surface regions. This is consistent with lost particles drifting through the null region and onto magnetic field lines that take them very close to the cylindrical material surface regions. The fact that, in the absence of particle-electric field interactions, 66% of previously lost particles exhibited stable closed loop orbits around the cylindrical material surface regions indicates particle-electric field interactions as well as proximity to the magnetic null regions are significant factors when considering the effectiveness of magnetic plasma expulsion.

To investigate the effects the computational boundary conditions had on the established equilibrium, variations of the base case simulation were examined that had $r = 0.6$ cm, Neumann boundary conditions, and electrically grounded conductors in place of the cylindrical material surface regions. These simulations with different boundary conditions were compared to a simulation with base case parameter values except $r = 0.6$ cm. The boundary condition changes contributed to less than a 1% difference in particles lost to the cylindrical material surface regions, and did not significantly affect the values of Ω throughout the simulation volume.

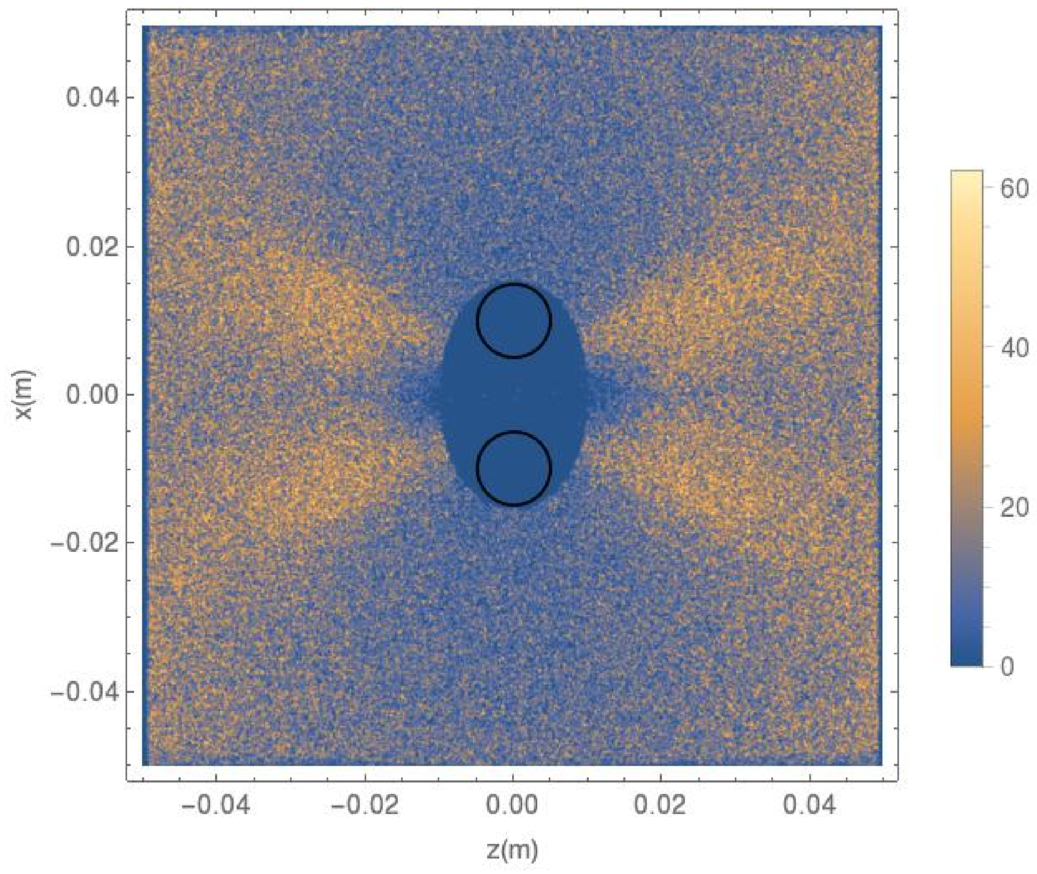

FIGURE 4.1. A density type plot of Ω for the electron positron plasma for base case parameter values. Structurally similar to previous work.

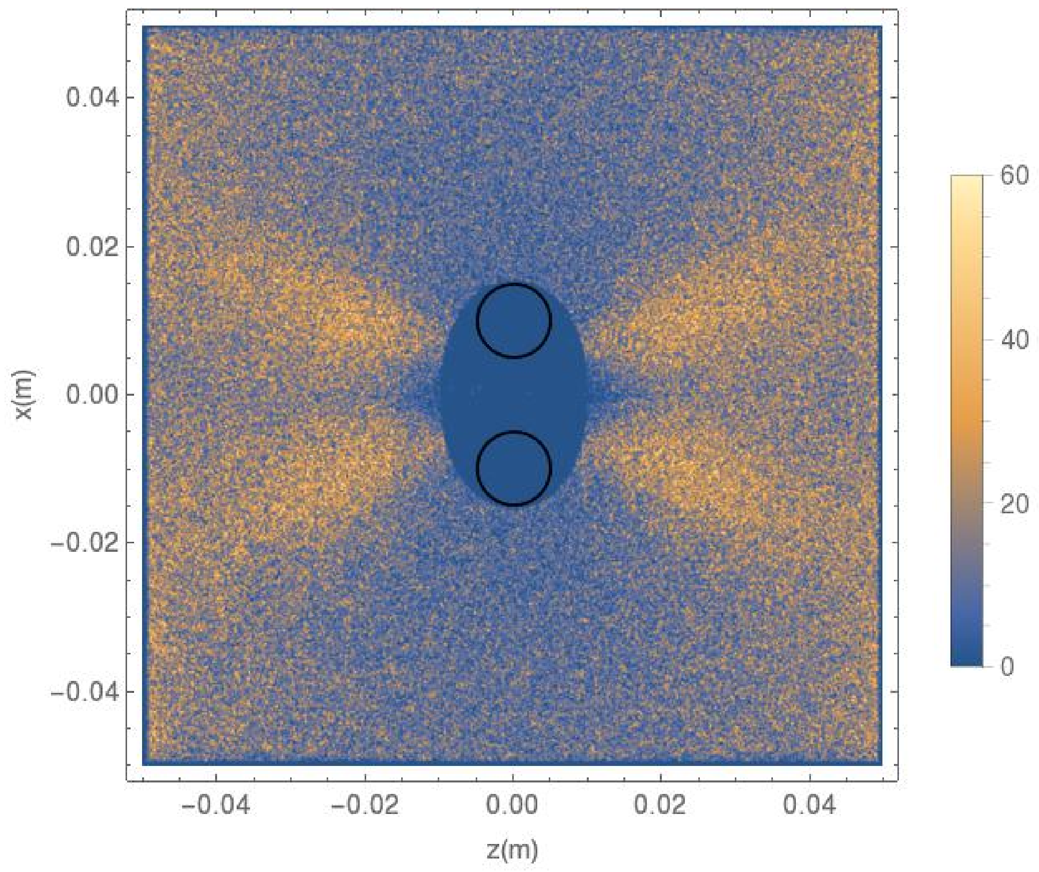

FIGURE 4.2. A density type plot of Ω for the electron-proton plasma for base case parameter values. Prominent equilibrium features appear unchanged suggesting the equilibrium established in the primer simulations carries over into the seeded simulation for the time scale simulated.

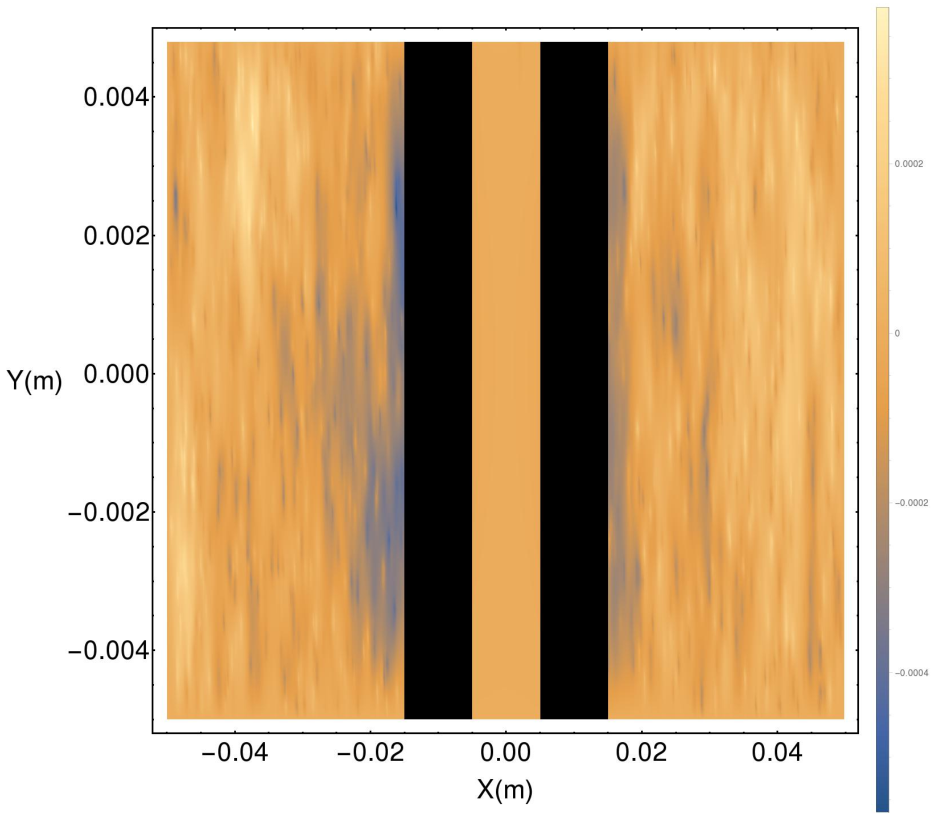

FIGURE 4.3. A density type plot of $z = 0$ midplane of the electron-proton plasma potential. The black areas are the cylindrical material surface regions. Negative potential (blue) near the cylindrical material surface regions indicates the formation of an electron-confining plasma sheath.

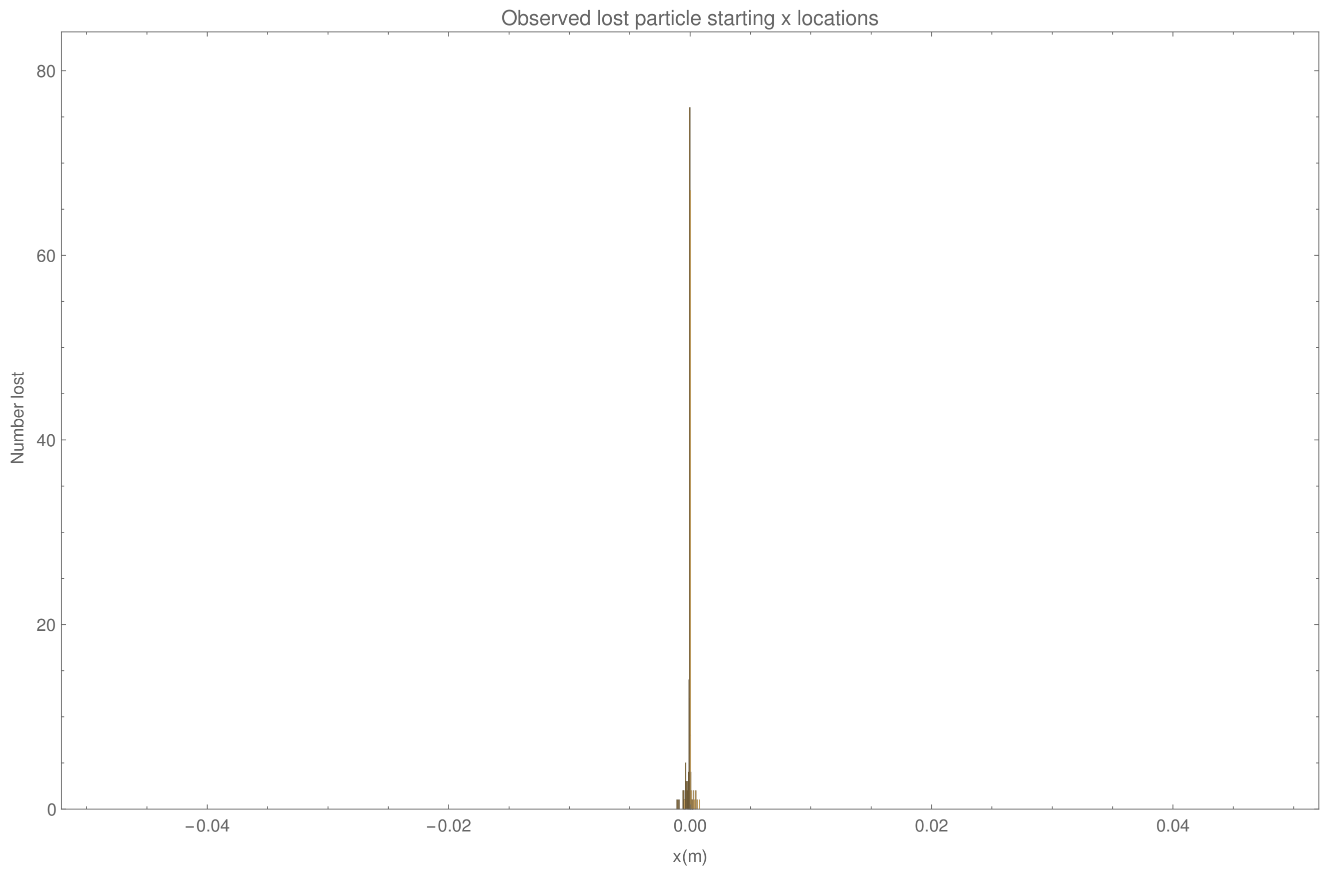

FIGURE 4.4. A histogram of observed x starting positions of particles lost in the base case. Magnetic null regions exist in the simulation volume on the $x = 0$ plane.

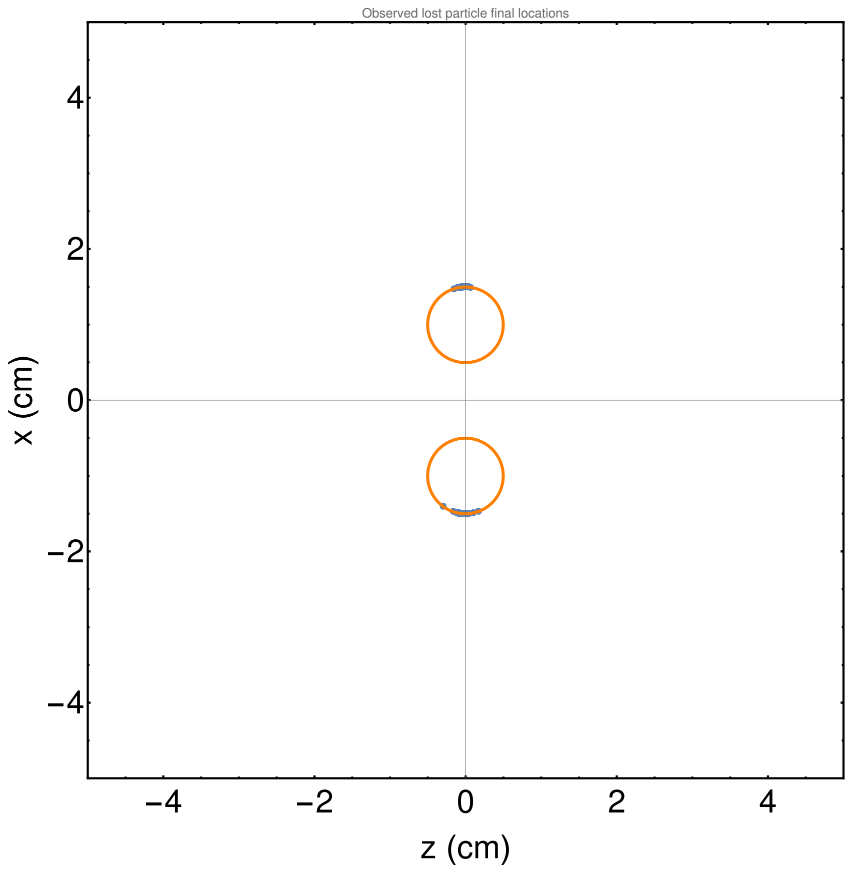

FIGURE 4.5. A plot of particle loss locations.

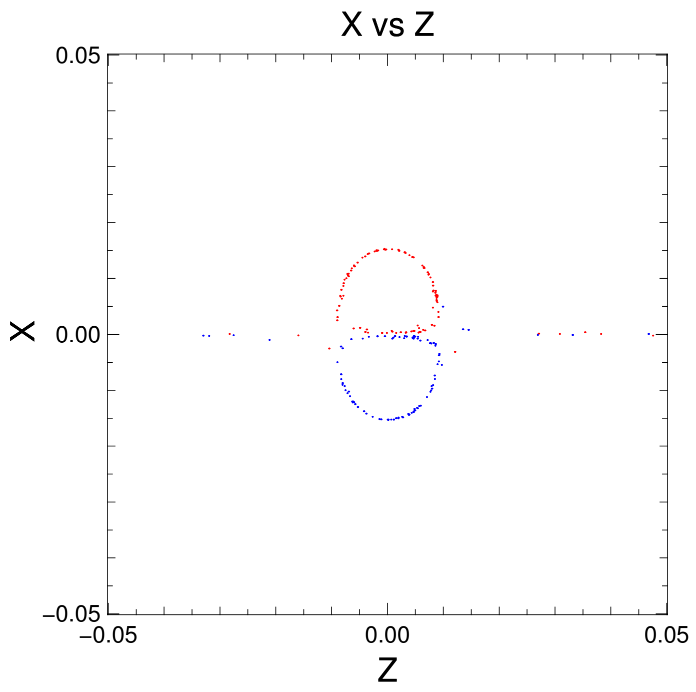

FIGURE 4.6. Snapshot at step 17, 939 of an examination of lost particle motion without particle-electric field interactions. Electrons, are represented as red dots, while positrons are represented as blue dots. A clear charge separation can be seen.

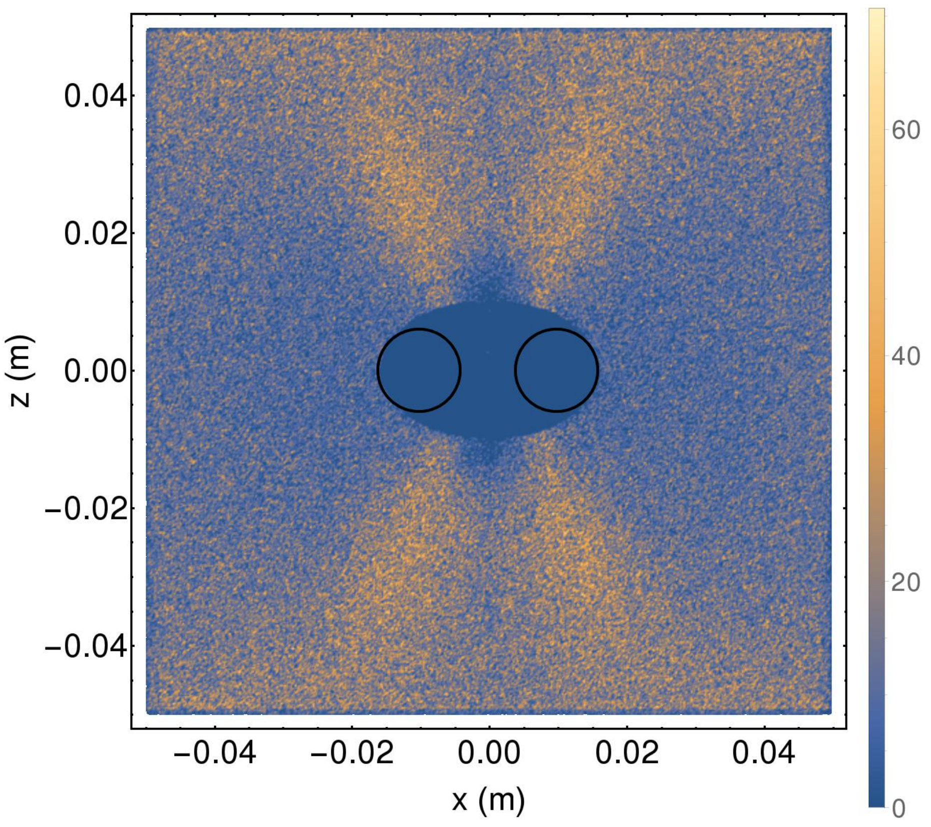

FIGURE 4.7. A density type plot of the averaged in y value of Ω in the standard case for comparison to variations of the computational bounds.

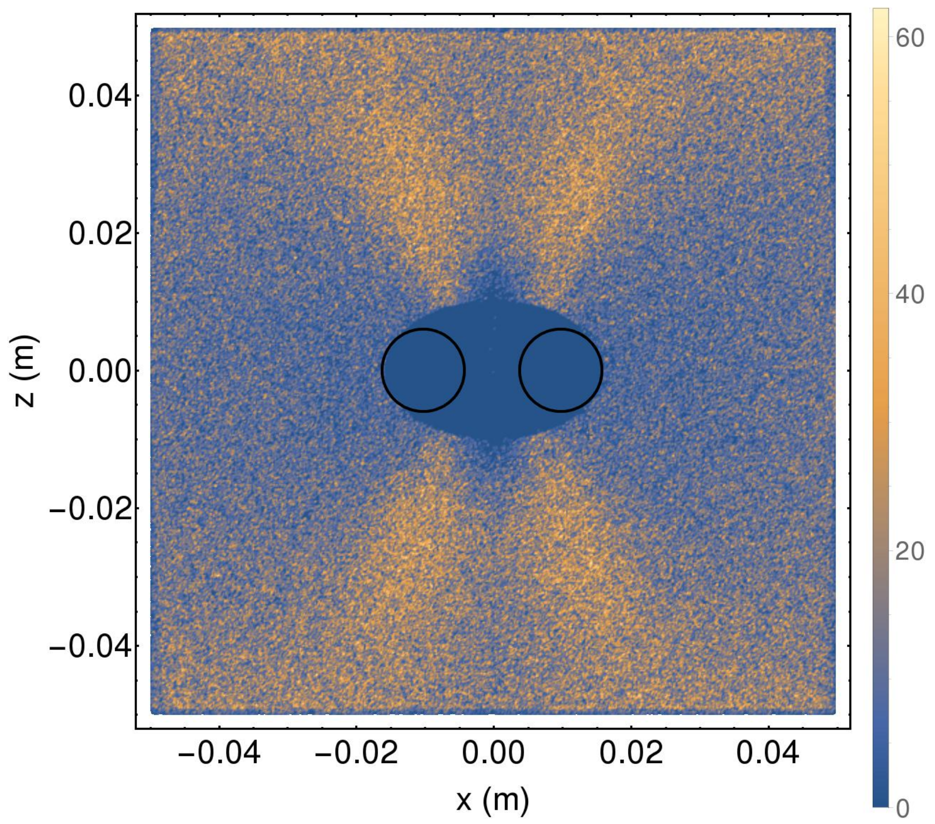

FIGURE 4.8. A density type plot of the averaged in y value of Ω. Comparison to Figs. 4.7 and 4.9 reveals no discernible differences.

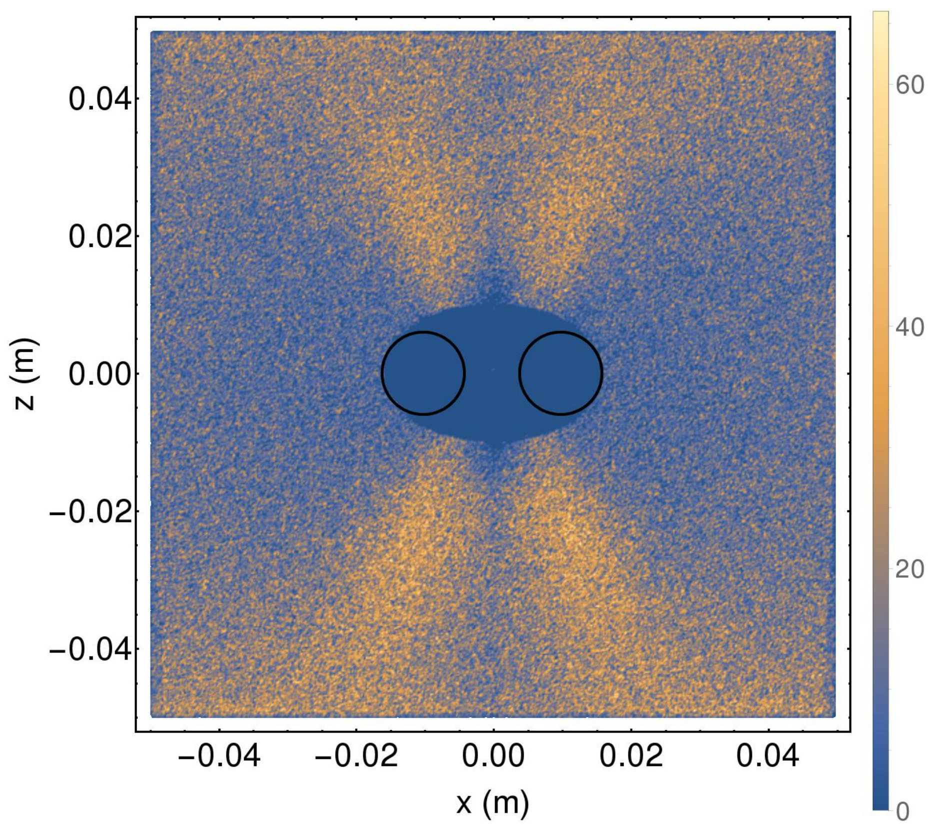

FIGURE 4.9. A density type plot of the averaged in y value of Ω. Comparison to Figs. 4.7 and 4.8 reveals no discernible differences.

CHAPTER 5

CONCLUSION

Magnetic plasma expulsion has been examined and characterized for potentially novel plasma trap applications. In Chapter 2, a relationship between the shielded radius a_n, the ratio of strengths between the two parts of the magnetic fields β_0 and the normalized cyclotron radius r_{0n} is found via a classical trajectory Monte Carlo study. The resultant numerical values were fit using an exponential decay relationship between the shielded radius a_n and the normalized cyclotron radius r_{0n} and an error function relationship between the shielded radius and β_{0n}.

Chapter 3 examined magnetic plasma expulsion using the Warp particle-in-cell code. Magnetic plasma confinement approaches have been proposed in the past that would require magnetic-field-producing coils to be embedded within the confined plasma. The development of magnetic plasma expulsion techniques may provide an enabling technology to shield mechanical supports and electrical connections attached to such embedded coils by redirecting the local magnetic field near such structures.

By way of example, a plasma confinement approach using magnetic plasma expulsion was described. This approach would use two coaxial solenoids, with a pair of metal tubes passing between the solenoids. Mechanical supports and electrical connections could be contained in the tubes. There would exist magnetic field lines that would pass through the inner solenoid and also the metal tubes were it not for the implementation of magnetic plasma expulsion. Due to the implementation of magnetic plasma expulsion, these field lines become closed and do not intersect with the metal tubes. Plasma particles following such field lines could then recirculate back into the inner solenoid where the magnetic field is weakest. Plasma within the inner solenoid may be confined by a magnetic mirror with closed field lines. As noted, plasma particles do pass between the tubes. It may be possible to utilize the magnetic null lines generated by the localized field distortion as a source of particles for injection into the device. Such injection schemes could help to control the neutrality, density and temperature of the confined plasma.

A system of magnetically confined electron-positron plasma near cylindrical material surfaces was modeled. Without magnetic plasma expulsion, plasma particles could follow field lines that intersect with the metal tubes and be lost. A figure of merit κ was defined as the ratio of losses to the metal tubes both with and without magnetic plasma expulsion present. Values for the figure of merit were found to be much less than one for a number of parameter values. It is concluded that magnetic plasma expulsion may reduce the effects associated with material objects passing through an electron-positron plasma and possibly other plasmas with similar parameter values. Augmenting current plasma trap designs with magnetic plasma expulsion concepts may provide a way to trap larger amounts of plasma for study. Magnetic plasma expulsion is also equally effective on plasma particles regardless of their charge sign which reduces the complexity of potential trap designs compared to electrostatic trap design equivalents.

Additional characterizations of magnetic plasma expulsion were conducted to expand on previous work in Chapter 4. Using primer simulations with equal mass opposite charge species proved effective at accelerating the simulation time required to achieve a computational equilibrium for a two species plasma with differing masses. The equilibrium achieved in the seeded simulation does not appear to deviate substantially from the equilibrium previously established in the primer simulations. Electron sheathing was observed in the seeded electron-proton simulation. Further examination of the sheathing is needed as the phenomenon was not seen in the electron-positron plasma simulations.

Characterization of the initial phase space of particle loss to the CMS regions revealed that for base case parameter values, all lost particles originate from very near the $x = 0$ line while all losses occur on the outside edges of the cylindrical material surfaces. This is consistent with lost particles drifting through the null region and onto magnetic field lines that take them very close to the cylindrical material surface regions. The fact that in the absence of particle-electric field interactions 66% of previously lost particles exhibited stable closed loop orbits around the cylindrical material surface regions indicates particle-electric field interactions as well as proximity to the magnetic null regions are significant factors when considering the effectiveness of magnetic plasma expulsion.

To investigate the effects the computational boundary conditions had on the established equilibrium, variations of the base case simulation were examined that had $r = 0.6$ cm, Neumann boundary conditions, and electrically grounded conductors in place of the cylindrical material surface regions. The boundary condition changes contributed to less than a 1% difference in the number of particles lost to the cylindrical material surface regions, and did not significantly affect the values of Ω throughout the simulation volume.

APPENDIX

COPY OF CODES USED

```
urs for 48 parts @ {a,.01,1,.01} {b,.01,5,.01} {r=1}*)
<< Utilities‘CleanSlate‘
ParallelEvaluate[CleanSlate[]];
Remove["Global‘*"];
numparts = 16;
numkerns = 16;

starta = .1;(*radius*)
startb = .1;(*ratio or strength*)
startr = .1;(*cyclotron*)

stopa = 1;
stopb = 1;
stopr = 1;

inca = .1;
incb = .1;
incr = .1;
runab := (

   (*Do loop interates over the parameter a.
   So whatever aspect of the sim you’d like to examine,
   make that parameter a.
   Works for both variable field ratios and variable forbidden zones.*)


   plottrajectories = False;
   graphtable = {};
   (*value of giveme comes from run notebook*)
   (*
   r0n=r0nfromrun;
   numparts=numpartsfromrun*)
   Do[
    Do[
      Do[

        trange = 100/Sqrt[2];(*full evaluation time frame.
        Almost all particles with have actual evaluation bounds \
shorter than this on the plot*)
        Strikes = 0;
        Trapped = 0;
        Spinner = 0;

        (*dimensionless wire mag field*)
        beta[x_, y_,
          z_] = {-z/((x - 1)^2 + z^2),
           0, (x - 1)/((x - 1)^2 + z^2)} - {-z/((x + 1)^2 + z^2),
           0, (x + 1)/((x + 1)^2 + z^2)};

        (*this is our dimensionless total magnetic field*)
        Bnorm[x_, y_, z_, Rnm_,
          b1_] = (Sqrt[2]/r0n)*(beta[x, y, z] + {0, 0, b1});

        (*gives our random initial particle position*)
        Rx[a_] := RandomReal[{-a - 1, a + 1}];
```

```
(*velocity definitions in DE form for solving*)
vx[t_] = Dt[x[t], {t, 1}];
vy[t_] = Dt[y[t], {t, 1}];
vz[t_] = Dt[z[t], {t, 1}];

eqnofmotion =
 Cross[{vx[t], vy[t], vz[t]}, Bnorm[x[t], y[t], z[t], r0n, b]];

Do[

 tau = Catch[
   NDSolve[
    {
     Dt[x[t], {t, 2}] == eqnofmotion[[1]],
     Dt[y[t], {t, 2}] == eqnofmotion[[2]],
     Dt[z[t], {t, 2}] == eqnofmotion[[3]],

     x[0] == Rx[a], y[0] == 0, z[0] == -5,
     vx[0] == 0, vy[0] == 0, vz[0] == Sqrt[2]
     },

    {x[t], y[t], z[t]},
    {t, 0, trange + 1.0001},

    MaxSteps -> 1000000,
    StepMonitor :> Which[
      Sqrt[(x[t] + 1)^2 + z[t]^2] <= a, Throw[{t, 1}];,
              Sqrt[(x[t] - 1)^2 + z[t]^2] <= a, Throw[{t, 1}];,
              z[t] >= 5, Throw[{t, 2}];,
              t > trange, Throw[{t, 3}];
              ]
            ]
           ];

         Which[
          tau[[2]] == 1, Strikes = Strikes + 1;,
          tau[[2]] == 2, Trapped = Trapped + 1;,
          tau[[2]] == 3, Trapped = Trapped + 1; Spinner = Spinner + 1;
          ];
         , {particlecount}]; (*Do loop that actually runs and solves \
for all the trajectories*)
        (*ERROR CODES:  -99  is no conditions were recorded   ;; -45 \
strikes and trapped are not equal to the total number of particles*)
        Which[
         Strikes == 0 && Trapped == 0 && Spinner == 0,
         AppendTo[
          graphtable, {a, b, r0n, Strikes, Trapped, Spinner, -99}],
         Strikes + Trapped != particlecount,
         AppendTo[
          graphtable, {a, b, r0n, Strikes, Trapped, Spinner, -45}],
         True,
         AppendTo[
          graphtable, {a, b, r0n, Strikes, Trapped, Spinner, 0}]];

        , {a, starta, stopa, inca}
        ];(*varies forbidden radius*),
      {b, startb, stopb, incb}(*varies field ratio,
      can switch places with r0n when you want to run a variable \
cyclotron radius graph*)
      ];
    , {r0n, startr, stopr, incr}];
   );
If[$KernelCount < numkerns, LaunchKernels[numkerns]]
particlecount = Quotient[numparts, numkerns];
Timing[DistributeDefinitions[runab, particlecount, combinedtable,
   sortscript];
  setitup =
   Table[ParallelSubmit[{i}, runab; graphtable], {i, 1, numkerns}];
  dataout = WaitAll[setitup]][[1]]

Timing[combinedtable =
   Table[{dataout[[1, dv, 1]], dataout[[1, dv, 2]],
     dataout[[1, dv, 3]], Total[dataout[[;; , dv, 4]]],
     Total[dataout[[;; , dv, 5]]], Total[dataout[[;; , dv, 6]]]}, {dv,
      1, Length[dataout[[1]]]}];]

ratioplot =
  Table[{dx,
    Max[First /@
      Cases[combinedtable, x_ /; x[[4]] == 0 && x[[2]] == dx]]}, {dx,
    startb, stopb, incb}];
(*creates b vs a table*)
radiusplot =
 Table[{dy,
   Max[First /@
     Cases[combinedtable, x_ /; x[[4]] == 0 && x[[3]] == dy]]}, {dy,
   startr, stopr, incr}];(*creates r vs a table*)
surfaceplot =
 Table[{dx, dy,
   Max[First /@
     Cases[combinedtable,
      x_ /; x[[4]] == 0 && x[[2]] == dx && x[[3]] == dy]]}, {dx,
   startb, stopb, incb}, {dy, startr, stopr,
   incr}];(*3d points b vs r vs a*)
Export["/home/rep0055/Talon_output/CAARI_14/radius_versus_ratio.csv",
  ratioplot, "CSV"];
Export["/home/rep0055/Talon_output/CAARI_14/radius_versus_radius.csv",
   radiusplot, "CSV"];
Export["/home/rep0055/Talon_output/CAARI_14/3d_plot_data.csv",
  surfaceplot, "CSV"];
Export["/home/rep0055/Talon_output/CAARI_14/combined_data_table.csv",
  combinedtable, "CSV"];
```

A.2. Chapter 3 Base Case: Warp

```
     # Load Warp and various script packages
from scipy.stats import *
from warp import *  # Warp code
import numpy as np
from mpi4py import MPI



setup()
```

```
########################################################################################

xsize = 0.05

ysize = 0.05

zsize = 0.05

xgrid = 200

ygrid = 200

zgrid = 200

xstep = (2 * xsize) / xgrid

ystep = (2 * ysize) / ygrid

zstep = (2 * zsize) / zgrid

xshift = -xsize

zshift = -zsize

smallestr = 1e10

stepval = 0

testvalue  = 0.000001

testevalue = 0.000001

testpvalue = 0.000001

EforbiddenX=[]

EforbiddenZ=[]

PforbiddenX=[]

PforbiddenZ=[]

phi=[]

EposX = []

EpoxZ = []

PposX = []

PposZ = []
```

```
delta = 1e-50

S = 0.01

beta = 1.0

r = 0.005

partslost = 0

stepcount = 0

ek = 0.5 * emass * (34821.01)**2

zlimit = 0.015

xlimit = 0.025

cellvolume=xstep*zstep*ystep

rank= MPI.COMM_WORLD.Get_rank()

uniformfieldstrength = 1.0

snapshot=90000


Bx = (np.fromfunction(lambda i, j, k:
 -((beta*uniformfieldstrength*S)*((k*zstep+zshift)/(((i*xstep+xshift)-S)**2+
 (k*zstep+zshift)**2 +
  delta))+((beta*uniformfieldstrength*S)*((k*zstep+zshift)/(((i*xstep+xshift)+S)**2+
  (k*zstep+zshift)**2 -
   delta))), (xgrid, ygrid, zgrid)))


Bz = (np.fromfunction(lambda i, j, k:
 ((beta*uniformfieldstrength*S)*((i*xstep+xshift-S)/(((i*xstep+xshift)-S)**2+
 (k*zstep+zshift)**2 +
  delta))-((beta*uniformfieldstrength*S)*((i*xstep+xshift+S)/(((i*xstep+xshift)+S)**2+
  (k*zstep+zshift)**2 -
   delta))), (xgrid, ygrid, zgrid))) +uniformfieldstrength
```

```
By = fzeros((xgrid, ygrid, zgrid))


weight = 10

#Cmax = 1

stepsize = 1

looptimes = 90000+1

#restartdump=100000/(stepsize+1)

#########################################################################################


# --- Set grid size


w3d.nx = xgrid


w3d.ny = ygrid


w3d.nz = zgrid


top.npmax = 400

top.ekin = (0.5 * emass * (34821.01)**2)/jperev

top.vbeam = 1e-50

top.dt = 1e-09

temp=40

filepath = "/home/rep0055/warpstuff/base_case/"

particle_density_file =

 str(top.npmax)+"parts_per_step_weight="+str(weight)+"_temp="+str(temp)+

 "_radius="+str(r)+"uni="+str(uniformfieldstrength)
```

```
w3d.xmmin = -xsize

w3d.xmmax = xsize

w3d.ymmin = -ysize

w3d.ymmax = ysize

w3d.zmmin = -zsize

w3d.zmmax = zsize

########################################################################################
# --- Build up magnetic fields
addnewbgrd(zs=-zsize, ze=zsize, xs=-xsize, dx=xstep,
 ys=-ysize, dy=ystep, bx=Bx, by=By, bz=Bz, nx=xgrid, ny=ygrid, nz=zgrid)

########################################################################################
# --- Build up particles in the sim
electron=Species(type=Electron,name="electron",color=blue,weight=weight,fselfb=None)
positron=Species(type=Positron,name="positron",color=red,weight=weight,fselfb=None)

vthermal = sqrt((1.38065e-23*temp)/(9.109e-31))

top.ssn = nextpid()

########################################################################################
xposmin = -xsize
```

```
xposmax = xsize
yposmin = -ysize
yposmax = ysize
zpposmin = zsize - 5*zstep
zpposmax = zsize - 4*zstep
zmposmin = -zsize + 4*zstep
zmposmax = -zsize + 5*zstep

def createelectronsupstream():
    electron.addparticles(x=np.random.uniform(xposmin, xposmax, size=top.npmax/4),
     y=np.random.uniform(yposmin, yposmax, size=top.npmax/4),
      z=np.random.uniform(zmposmin, zmposmax, size=top.npmax/4),
       vx=np.random.normal(loc=0.0, scale=vthermal, size=top.npmax / 4),
        vy=np.random.normal(loc=0.0, scale=vthermal, size=top.npmax / 4),
         vz=abs(np.random.normal(loc=0.0, scale=vthermal, size=top.npmax / 4)),
          lallindomain=false, lnewparticles=true)
installuserinjection(createelectronsupstream)



def createpositronsupstream():
    positron.addparticles(x=np.random.uniform(xposmin, xposmax, size=top.npmax/4),
     y=np.random.uniform(yposmin, yposmax, size=top.npmax/4),
      z=np.random.uniform(zmposmin, zmposmax, size=top.npmax/4),
       vx=np.random.normal(loc=0.0, scale=vthermal, size=top.npmax / 4),
        vy=np.random.normal(loc=0.0, scale=vthermal, size=top.npmax / 4),
         vz=abs(np.random.normal(loc=0.0, scale=vthermal, size=top.npmax / 4)),
          lallindomain=false, lnewparticles=true)
```

```
installuserinjection(createpositronsupstream)



def createelectronsdownstream():
    electron.addparticles(x=np.random.uniform(xposmin, xposmax, size=top.npmax/4),
     y=np.random.uniform(yposmin, yposmax, size=top.npmax/4),
      z=np.random.uniform(zpposmin, zpposmax, size=top.npmax/4),
       vx=np.random.normal(loc=0.0, scale=vthermal, size=top.npmax / 4),
        vy=np.random.normal(loc=0.0, scale=vthermal, size=top.npmax / 4),
         vz=-abs(np.random.normal(loc=0.0, scale=vthermal, size=top.npmax / 4)),
          lallindomain=false, lnewparticles=true)
installuserinjection(createelectronsdownstream)



def createpositronsdownstream():
    positron.addparticles(x=np.random.uniform(xposmin, xposmax, size=top.npmax/4),
     y=np.random.uniform(yposmin, yposmax, size=top.npmax/4),
      z=np.random.uniform(zpposmin, zpposmax, size=top.npmax/4),
       vx=np.random.normal(loc=0.0, scale=vthermal, size=top.npmax / 4),
        vy=np.random.normal(loc=0.0, scale=vthermal, size=top.npmax / 4),
         vz=-abs(np.random.normal(loc=0.0, scale=vthermal, size=top.npmax / 4)),
          lallindomain=false, lnewparticles=true)
installuserinjection(createpositronsdownstream)
########################################################################################

def partcounter():
```

```
EposX=electron.getx(zl=-zlimit,zu=zlimit,xl=-xlimit,xu=xlimit)
EposZ=electron.getz(zl=-zlimit,zu=zlimit,xl=-xlimit,xu=xlimit)
PposX=positron.getx(zl=-zlimit,zu=zlimit,xl=-xlimit,xu=xlimit)
PposZ=positron.getz(zl=-zlimit,zu=zlimit,xl=-xlimit,xu=xlimit)
global partslost
for j in range(1, len(EposX)):
    testvalueeplus = sqrt((EposX[j]+S)**2+EposZ[j]**2)
    testvalueeminus = sqrt((EposX[j]-S)**2+EposZ[j]**2)

    if testvalueeplus <= r:
        partslost = partslost + 1
    if testvalueeminus <= r:
        partslost = partslost + 1

for j in range(1, len(PposX)):
    testvaluepplus = sqrt((PposX[j]+S)**2+PposZ[j]**2)
    testvaluepminus = sqrt((PposX[j]-S)**2+PposZ[j]**2)

    if testvaluepplus <= r:
        partslost = partslost + 1
    if testvaluepminus <= r:
        partslost = partslost + 1
if rank == 0:
    with open(filepath+particle_density_file+"beta="+str(beta)+"lost.m", "a")
     as myfile:
        myfile.write("{%d, %d}," %(stepcount, (weight)*partslost))
EposX=[]
EposZ=[]
```

```
        PposX=[]
        PposZ=[]
installbeforescraper(partcounter)


def scrapebeam():
    rsqeplus = (electron.xp + S)**2 + electron.zp**2
    rsqpplus = (positron.xp + S)**2 + positron.zp**2
    rsqeminus = (electron.xp - S)**2 + electron.zp**2
    rsqpminus = (positron.xp - S)**2 + positron.zp**2
    electron.gaminv[rsqeplus <= (r)**2] = 0.
    electron.gaminv[rsqeminus <= (r)**2] = 0.
    positron.gaminv[rsqpplus <= (r)**2] = 0.
    positron.gaminv[rsqpminus <= (r)**2] = 0.
installparticlescraper(scrapebeam)


def totalcounter():
    EposX=electron.getx()
    PposX=positron.getx()
    electron_charge = len(EposX+1)
    positron_charge = len(PposX+1)
    if rank==0:
        with open(filepath+particle_density_file+"beta="+str(beta)+"totalcount.m", "a")
         as myfile:
            myfile.write("{%d, %d}," %(stepcount,
            (weight)*electron_charge+(weight)*positron_charge))
    positroncharge=[]
    electroncharge=[]
    EposX=[]
    PposX=[]
installafterscraper(totalcounter)


def xzdensitycounter():
    for i in np.linspace(-xsize, xsize, xgrid, endpoint=False):
        for j in np.linspace(-ysize, ysize, ygrid, endpoint=False):
            for k in np.linspace(-zsize, zsize, zgrid, endpoint=False):
                EposX=electron.getx(xl=i,xu=i+xstep,yl=j,yu=j+ystep,zl=k,zu=k+zstep)
                PposX=positron.getx(xl=i,xu=i+xstep,yl=j,yu=j+ystep,zl=k,zu=k+zstep)
                electron_charge = weight*len(EposX+1)
                positron_charge = weight*len(PposX+1)
                neutrality=(positron_charge-electron_charge)/cellvolume
                density=(electron_charge+positron_charge)/cellvolume
                phi=getphi(ix=i,iy=j,iz=k)
                if rank == 0:
                    with open(filepath+particle_density_file+"beta="+str(beta)+
                    "potential.m", "a")
                     as myfile:
                        myfile.write("{%d, %.8f, %.8f, %.8f, %.8f},"
                        %(stepcount, i, j, k, phi))
                if density == 0:
                    debye_length=1e10
                    neut=0
                else:
                    debye_length=sqrt((eps0*boltzmann*temp)/(density*(echarge)**2))
                    neut=neutrality/(density)
                if rank == 0:
                     with open(filepath+particle_density_file+"beta="+str(beta)+
                     "xzdebyelength.m", "a")
                      as myfile:
                         myfile.write("{%d, %.8f, %.8f, %.8f, %.8f},"
                          %(stepcount, i, j, k,(10*S)/(debye_length)))
                density=0
                debye_length=0
                neut=0
                phi=[]
                positron_charge=[]
                electron_charge=[]
                EposX=[]
                PposX=[]




def totalenergycounter():
    EvelXsq=numpy.array(electron.getvx())**2
    EvelYsq=numpy.array(electron.getvy())**2
    EvelZsq=numpy.array(electron.getvz())**2
    PvelXsq=numpy.array(positron.getvx())**2
    PvelYsq=numpy.array(positron.getvx())**2
    PvelZsq=numpy.array(positron.getvx())**2
    magEvelsq=EvelXsq+EvelYsq+EvelZsq
    magPvelsq=PvelXsq+PvelYsq+PvelZsq
    Ek= numpy.sum(magEvelsq) *((0.5*9.10938356e-31)/(1.602e-19))
    Pk= numpy.sum(magPvelsq) *((0.5*9.10938356e-31)/(1.602e-19))
    totalenergy=weight*(Ek+Pk)
    if rank==0:
        with open(filepath+particle_density_file+"beta="+str(beta)+"energy.m", "a")
         as myfile:
            myfile.write("{%d, %.8f}," %(stepcount, totalenergy))
    EvelXsq=[]
    EvelYsq=[]
    EvelZsq=[]
    PvelXsq=[]
    PvelYsq=[]
    PvelZsq=[]
    magEvelsq=[]
    magPvelsq=[]
    Ek=[]
    Pk=[]
    totalenergy=[]
installafterscraper(totalenergycounter)

def E_fields():
    for i in np.linspace(0, 70, 71):
        xfields= getselfe(comp='x', ix=i, iy=100, iz=100)
        yfields= getselfe(comp='y', ix=i, iy=100, iz=100)
        zfields= getselfe(comp='z', ix=i, iy=100, iz=100)
        if rank == 0:
            with open(filepath+particle_density_file+"beta="+str(beta)+"fields.m", "a")
            as myfile:
                myfile.write("{%d, %.8f, %.8f, %.8f, %.8f, %.8f},"
                 %(stepcount, Bx[i,100,100],Bz[i,100,100], xfields, yfields, zfields))
        xfields=0
        yfields=0
        zfields=0


####################################################################################################

w3d.interpdk[1] = 1
w3d.interpdk[0] = 1
w3d.igradb = 1

w3d.boundxy = dirichlet
top.pboundxy = absorb
w3d.boundnz = dirichlet
top.pboundnz = absorb
w3d.bound0 = dirichlet
top.pbound0 = absorb



#top.fstype = -1
top.lrelativ = 0
top.relativity = 0
solver = MultiGrid3D()
registersolver(solver)
if rank == 0:
    with open(filepath+particle_density_file+"beta="+str(beta)+"xzdebyelength.m", "a")
     as myfile:
            myfile.write("{")
    with open(filepath+particle_density_file+"beta="+str(beta)+"fields.m", "a")
     as myfile:
        myfile.write("{")
    with open(filepath+particle_density_file+"beta="+str(beta)+"xydebyelength.m", "a")
     as myfile:
        myfile.write("{")
    with open(filepath+particle_density_file+"beta="+str(beta)+"lost.m", "a")
     as myfile:
        myfile.write("{")
    with open(filepath+particle_density_file+"beta="+str(beta)+"energy.m", "a")
     as myfile:
        myfile.write("{")
    with open(filepath+particle_density_file+"beta="+str(beta)+"totalcount.m", "a")
     as myfile:
        myfile.write("{")
    with open(filepath+particle_density_file+"beta="+str(beta)+"xycenterplane.m", "a")
     as myfile:
        myfile.write("{")
    with open(filepath+particle_density_file+"beta="+str(beta)+"potential.m", "a")
     as myfile:
        myfile.write("{")

###########################################################################################


package("w3d")
generate()
fieldsolve()

from warp.lattice.loadgradb import setbsqgrad
```

```
setbsqgrad(xgrid, ygrid, zgrid, -xsize, xsize, -ysize, ysize, -zsize, zsize)



for i in xrange(1, looptimes):
    if stepcount >= (snapshot-1):
        installafterscraper(xzdensitycounter)

    step()
    if stepcount >= (snapshot-1):
        uninstallafterscraper(xzdensitycounter)

    chop_fraction = 10.e2/(float(len(EposX)+1)+float(len(PposX)+1))
#    electron.ppzx(particles=true, chopped=chop_fraction,
 color="red", msize=10000, lframe=true,
  pplimits=[w3d.xmmin,w3d.xmmax,w3d.zmmin,w3d.zmmax])
#    positron.ppzx(particles=true, chopped=chop_fraction,
 color="blue", msize=10000, lframe=true,
  pplimits=[w3d.xmmin,w3d.xmmax,w3d.zmmin,w3d.zmmax])

    fma()

    stepcount = stepcount + 1
    i = i + 1 #increments the for loops advancing the sim.


if rank == 0:
```

```
with open(filepath+particle_density_file+"beta="+str(beta)+"xzdebyelength.m", "a")
 as myfile:
        myfile.seek(-1, os.SEEK_END)
        myfile.truncate()
        myfile.write("}")

with open(filepath+particle_density_file+"beta="+str(beta)+"xydebyelength.m", "a")
 as myfile:
        myfile.seek(-1, os.SEEK_END)
        myfile.truncate()
        myfile.write("}")

with open(filepath+particle_density_file+"beta="+str(beta)+"lost.m", "a")
 as myfile:
        myfile.seek(-1, os.SEEK_END)
        myfile.truncate()
        myfile.write("}")

with open(filepath+particle_density_file+"beta="+str(beta)+"energy.m", "a")
 as myfile:
        myfile.seek(-1, os.SEEK_END)
        myfile.truncate()
        myfile.write("}")

with open(filepath+particle_density_file+"beta="+str(beta)+"totalcount.m", "a")
 as myfile:
        myfile.seek(-1, os.SEEK_END)
        myfile.truncate()
```

```
            myfile.write("}")


    with open(filepath+particle_density_file+"beta="+str(beta)+"xycenterplane.m", "a")
     as myfile:
            myfile.seek(-1, os.SEEK_END)
            myfile.truncate()
            myfile.write("}")


    with open(filepath+particle_density_file+"beta="+str(beta)+"fields.m", "a")
     as myfile:
            myfile.seek(-1, os.SEEK_END)
            myfile.truncate()
            myfile.write("}")


    with open(filepath+particle_density_file+"beta="+str(beta)+"potential.m", "a")
     as myfile:
            myfile.seek(-1, os.SEEK_END)
            myfile.truncate()
            myfile.write("}")
```

A.3. Chapter 4 Base Case: Warp

```
# Load Warp and various script packages
from scipy.stats import *
from warp import *  # Warp code
import numpy as np
from mpi4py import MPI


setup()
```

```
####################################################################################################

xsize = 0.05

ysize = 0.005

zsize = 0.05

xgrid = 500

ygrid = 50

zgrid = 500

xstep = (2 * xsize) / xgrid

ystep = (2 * ysize) / ygrid

zstep = (2 * zsize) / zgrid

xshift = -xsize

zshift = -zsize

smallestr = 1e10

stepval = 0

testvalue  = 0.000001

testevalue = 0.000001

testpvalue = 0.000001

EforbiddenX=[]

EforbiddenZ=[]

PforbiddenX=[]

PforbiddenZ=[]

phi=[]

EposX = []

EpoxZ = []

PposX = []

PposZ = []

ltotalcount = 0
```

```
lpartrecord = 0

lfluxcount = 0

dxelectron = []

dyelectron = []

dzelectron = []

dxpositron = []

dypositron = []

dzpositron = []

electronxbirth = []

electronybirth = []

electronzbirth = []

positronxbirth = []

positronybirth = []

positronzbirth = []

electronxdeath = []

electronydeath = []

electronzdeath = []

positronxdeath = []

positronydeath = []

positronzdeath = []

dx = []

dy = []

dz = []

xbirth = []

ybirth = []

zbirth = []

xdeath = []

ydeath = []
```

```
zdeath = []
lostelectronxbirth=[]
lostelectronybirth=[]
lostelectronzbirth=[]
lostelectronxdeath=[]
lostelectronydeath=[]
lostelectronzdeath=[]
lostpositronxbirth=[]
lostpositronybirth=[]
lostpositronzbirth=[]
lostpositronxdeath=[]
lostpositronydeath=[]
lostpositronzdeath=[]
delta = 1e-50
S = 0.01
bm = 1.0
r = 0.005
partslost = 0
top.lsavelostpart= 1
stepcount = 0
ek = 0.5 * emass * (34821.01)**2
zlimit = 0.006
xlimit = 0.016
cellvolume=xstep*zstep*ystep
rank= MPI.COMM_WORLD.Get_rank()
uniformfieldstrength = 1.0
snapshot=90000
```

```
Bx = (np.fromfunction(lambda i, j, k:
 -((bm*uniformfieldstrength*S)*((k*zstep+zshift)/(((i*xstep+xshift)-S)**2+
 (k*zstep+zshift)**2 +
  delta))+((bm*uniformfieldstrength*S)*((k*zstep+zshift)/(((i*xstep+xshift)+S)**2+
  (k*zstep+zshift)**2 - delta))), (xgrid, ygrid, zgrid)))

Bz = (np.fromfunction(lambda i, j, k:
 ((bm*uniformfieldstrength*S)*((i*xstep+xshift-S)/(((i*xstep+xshift)-S)**2+
 (k*zstep+zshift)**2 +
  delta))-((bm*uniformfieldstrength*S)*((i*xstep+xshift+S)/(((i*xstep+xshift)+S)**2+
  (k*zstep+zshift)**2 -
  delta))), (xgrid, ygrid, zgrid))) +uniformfieldstrength

By = fzeros((xgrid, ygrid, zgrid))

weight = 10
#Cmax = 1
stepsize = 1
looptimes = 90000+1
##############################################################################################

# --- Set grid size

w3d.nx = xgrid

w3d.ny = ygrid
```

```
w3d.nz = zgrid


top.npmax = 40

top.ekin = (0.5 * emass * (34821.01)**2)/jperev

top.vbeam = 1e-50

top.dt = 1e-09

temp=40

filepath = "/home/rep0055/warpstuff/electron_proton/primer_sims/electron_positron/"

particle_density_file = "electron_positron"+str(top.npmax)+"parts_per_step_weight="+

str(weight)+"_temp="+str(temp)+"_radius="+str(r)+"uni="+str(uniformfieldstrength)



w3d.xmmin = -xsize


w3d.xmmax = xsize


w3d.ymmin = -ysize


w3d.ymmax = ysize


w3d.zmmin = -zsize


w3d.zmmax = zsize

##########################################################################################

# --- Build up magnetic fields

addnewbgrd(zs=-zsize, ze=zsize, xs=-xsize, dx=xstep,

 ys=-ysize, dy=ystep, bx=Bx, by=By, bz=Bz,
  nx=xgrid, ny=ygrid, nz=zgrid)



#############################################################################################
# --- Build up particles in the sim
electron=Species(type=Electron,name="electron",color=blue,weight=weight,fselfb=None)
positron=Species(type=Positron,name="positron",color=red,weight=weight,fselfb=None)

vthermal = sqrt((1.38065e-23*temp)/(9.109e-31))

top.xbirthpid = nextpid()
top.ybirthpid = nextpid()
top.zbirthpid = nextpid()
top.ssn = nextpid()

#############################################################################################

xposmin = -xsize
xposmax = xsize
yposmin = -ysize
yposmax = ysize
zpposmin = zsize - (5*zstep)
zpposmax = zsize - (4*zstep)
zmposmin = -zsize + (4*zstep)
zmposmax = -zsize + (5*zstep)
```

```
def createparticles():
        electron.addparticles(x=np.random.uniform(xposmin, xposmax, size=top.npmax/4),
         y=np.random.uniform(yposmin, yposmax, size=top.npmax/4),
          z=np.random.uniform(zmposmin, zmposmax, size=top.npmax/4),
           vx=np.random.normal(loc=0.0, scale=vthermal, size=top.npmax / 4),
            vy=np.random.normal(loc=0.0, scale=vthermal, size=top.npmax / 4),
             vz=abs(np.random.normal(loc=0.0, scale=vthermal, size=top.npmax / 4)),
              lallindomain=true, lnewparticles=true)

        positron.addparticles(x=np.random.uniform(xposmin, xposmax, size=top.npmax/4),
         y=np.random.uniform(yposmin, yposmax, size=top.npmax/4),
          z=np.random.uniform(zmposmin, zmposmax, size=top.npmax/4),
           vx=np.random.normal(loc=0.0, scale=vthermal, size=top.npmax / 4),
            vy=np.random.normal(loc=0.0, scale=vthermal, size=top.npmax / 4),
             vz=abs(np.random.normal(loc=0.0, scale=vthermal, size=top.npmax / 4)),
              lallindomain=true, lnewparticles=true)

        electron.addparticles(x=np.random.uniform(xposmin, xposmax, size=top.npmax/4),
         y=np.random.uniform(yposmin, yposmax, size=top.npmax/4),
          z=np.random.uniform(zpposmin, zpposmax, size=top.npmax/4),
           vx=np.random.normal(loc=0.0, scale=vthermal, size=top.npmax / 4),
            vy=np.random.normal(loc=0.0, scale=vthermal, size=top.npmax / 4),
             vz=-abs(np.random.normal(loc=0.0, scale=vthermal, size=top.npmax / 4)),
              lallindomain=true, lnewparticles=true)

        positron.addparticles(x=np.random.uniform(xposmin, xposmax, size=top.npmax/4),
         y=np.random.uniform(yposmin, yposmax, size=top.npmax/4),
          z=np.random.uniform(zpposmin, zpposmax, size=top.npmax/4),
          vx=np.random.normal(loc=0.0, scale=vthermal, size=top.npmax / 4),
           vy=np.random.normal(loc=0.0, scale=vthermal, size=top.npmax / 4),
            vz=-abs(np.random.normal(loc=0.0, scale=vthermal, size=top.npmax / 4)),
             lallindomain=true, lnewparticles=true)

if rank == 0:
    installuserinjection(createparticles)

 ###########################################################################################

def partcounter():
        global partslost, rank, lostelectronxbirth, lostelectronybirth,
         lostelectronzbirth, lostelectronxdeath, lostelectronydeath,
          lostelectronzdeath, lostpositronxbirth, lostpositronybirth,
           lostpositronzbirth, lostpositronxdeath, lostpositronydeath,
            lostpositronzdeath
        EposXbirth=electron.getxbirth(zl=-zlimit,zu=zlimit,xl=-xlimit,xu=xlimit)
        EposYbirth=electron.getybirth(zl=-zlimit,zu=zlimit,xl=-xlimit,xu=xlimit)
        EposZbirth=electron.getzbirth(zl=-zlimit,zu=zlimit,xl=-xlimit,xu=xlimit)
        PposXbirth=positron.getxbirth(zl=-zlimit,zu=zlimit,xl=-xlimit,xu=xlimit)
        PposYbirth=positron.getybirth(zl=-zlimit,zu=zlimit,xl=-xlimit,xu=xlimit)
        PposZbirth=positron.getzbirth(zl=-zlimit,zu=zlimit,xl=-xlimit,xu=xlimit)

        EposXdeath=electron.getx(zl=-zlimit,zu=zlimit,xl=-xlimit,xu=xlimit)
        EposYdeath=electron.gety(zl=-zlimit,zu=zlimit,xl=-xlimit,xu=xlimit)
        EposZdeath=electron.getz(zl=-zlimit,zu=zlimit,xl=-xlimit,xu=xlimit)
        PposXdeath=positron.getx(zl=-zlimit,zu=zlimit,xl=-xlimit,xu=xlimit)
        PposYdeath=positron.gety(zl=-zlimit,zu=zlimit,xl=-xlimit,xu=xlimit)
```

```
PposZdeath=positron.getz(zl=-zlimit,zu=zlimit,xl=-xlimit,xu=xlimit)
if rank == 0:
    for j in range(1, len(EposXdeath)):
        testvalueeplus = sqrt((EposXdeath[j]+S)**2+EposZdeath[j]**2)
        testvalueeminus = sqrt((EposXdeath[j]-S)**2+EposZdeath[j]**2)

        if testvalueeplus <= r:
            partslost = partslost + 1
            np.append(lostelectronxbirth,EposXbirth[j])
            np.append(lostelectronybirth,EposYbirth[j])
            np.append(lostelectronzbirth,EposZbirth[j])
            np.append(lostelectronxdeath,EposXdeath[j])
            np.append(lostelectronydeath,EposYdeath[j])
            np.append(lostelectronzdeath,EposZdeath[j])
        if testvalueeminus <= r:
            partslost = partslost + 1
            np.append(lostelectronxbirth,EposXbirth[j])
            np.append(lostelectronybirth,EposYbirth[j])
            np.append(lostelectronzbirth,EposZbirth[j])
            np.append(lostelectronxdeath,EposXdeath[j])
            np.append(lostelectronydeath,EposYdeath[j])
            np.append(lostelectronzdeath,EposZdeath[j])

    for j in range(1, len(PposXdeath)):
        testvaluepplus = sqrt((PposXdeath[j]+S)**2+PposZdeath[j]**2)
        testvaluepminus = sqrt((PposXdeath[j]-S)**2+PposZdeath[j]**2)

        if testvaluepplus <= r:
```

```
                    partslost = partslost + 1
                    np.append(lostpositronxbirth,PposXbirth[j])
                    np.append(lostpositronybirth,PposYbirth[j])
                    np.append(lostpositronzbirth,PposZbirth[j])
                    np.append(lostpositronxdeath,PposXdeath[j])
                    np.append(lostpositronydeath,PposYdeath[j])
                    np.append(lostpositronzdeath,PposZdeath[j])
                if testvaluepminus <= r:
                    partslost = partslost + 1
                    np.append(lostpositronxbirth,PposXbirth[j])
                    np.append(lostpositronybirth,PposYbirth[j])
                    np.append(lostpositronzbirth,PposZbirth[j])
                    np.append(lostpositronxdeath,PposXdeath[j])
                    np.append(lostpositronydeath,PposYdeath[j])
                    np.append(lostpositronzdeath,PposZdeath[j])

            with open(filepath+particle_density_file+"bm="+str(bm)+"lost.m", "a")
             as myfile:
                myfile.write("{%d, %d}," %(stepcount, (weight)*partslost))

installbeforescraper(partcounter)


driftlimitpos=zsize-3*zstep
driftlimitneg=-zsize+3*zstep


def scrapebeam():
        rsqeplus = (electron.xp + S)**2 + electron.zp**2
        rsqpplus = (positron.xp + S)**2 + positron.zp**2
        rsqeminus = (electron.xp - S)**2 + electron.zp**2

        rsqpminus = (positron.xp - S)**2 + positron.zp**2

        zelectron = electron.zp

        zpositron = positron.zp

        electron.gaminv[rsqeplus <= (r)**2] = 0.

        electron.gaminv[rsqeminus <= (r)**2] = 0.

        positron.gaminv[rsqpplus <= (r)**2] = 0.

        positron.gaminv[rsqpminus <= (r)**2] = 0.

        electron.gaminv[electron.zp < driftlimitneg ] = 0.

        electron.gaminv[electron.zp > driftlimitpos ] = 0.

        positron.gaminv[positron.zp < driftlimitneg ] = 0.

        positron.gaminv[positron.zp > driftlimitpos ] = 0.

installparticlescraper(scrapebeam)



def totalcounter():

        global ltotalcount, rank

        if ltotalcount == 1000 :

            ltotalcount = 0

            totE=electron.getn()

            totP=positron.getn()

            if rank ==0:

                with open(filepath+particle_density_file+"bm="+str(bm)+"totalcount.m"

                , "a") as myfile:

                    myfile.write("{%d, %d},"

                    %(stepcount, (weight)*totE+(weight)*totP))

        else:

            ltotalcount=ltotalcount+1
```

```
installafterscraper(totalcounter)



def drift_delta():

        global dx, dy, dz, xbirth, ybirth, zbirth,
        xdeath, ydeath, zdeath, dxelectron, dxpositron,
         dyelectron, dypositron, dzelectron, dzpositron,
          electronxbirth, electronybirth, electronzbirth,
           positronxbirth, positronybirth, positronzbirth,
            electronxdeath, electronydeath, electronzdeath,
             positronxdeath, positronydeath, positronzdeath, rank

        dxelectron=np.append(np.subtract(electron.getx(zl=driftlimitpos, zu=zsize),
        electron.getxbirth(zl=driftlimitpos, zu=zsize)),
        np.subtract(electron.getx(zl=-zsize, zu=driftlimitneg),
        electron.getxbirth(zl=-zsize, zu=driftlimitneg)))

        dxpositron=np.append(np.subtract(positron.getx(zl=driftlimitpos, zu=zsize),
        positron.getxbirth(zl=driftlimitpos, zu=zsize)),
        np.subtract(positron.getx(zl=-zsize, zu=driftlimitneg),
        positron.getxbirth(zl=-zsize, zu=driftlimitneg)))

        transitiondx=np.append(dxelectron,dxpositron)

        dx=np.append(dx,transitiondx)

        dyelectron=np.append(np.subtract(electron.gety(zl=driftlimitpos, zu=zsize),
        electron.getybirth(zl=driftlimitpos, zu=zsize)),
        np.subtract(electron.gety(zl=-zsize, zu=driftlimitneg),
        electron.getybirth(zl=-zsize, zu=driftlimitneg)))

        dypositron=np.append(np.subtract(positron.gety(zl=driftlimitpos, zu=zsize),
        positron.getybirth(zl=driftlimitpos, zu=zsize)),
        np.subtract(positron.gety(zl=-zsize, zu=driftlimitneg),
        positron.getybirth(zl=-zsize, zu=driftlimitneg)))

        transitiondy=np.append(dyelectron,dypositron)

        dy=np.append(dy,transitiondy)

        dzelectron=np.append(np.subtract(electron.getz(zl=driftlimitpos, zu=zsize),
        electron.getzbirth(zl=driftlimitpos, zu=zsize)),
        np.subtract(electron.getz(zl=-zsize, zu=driftlimitneg),
        electron.getzbirth(zl=-zsize, zu=driftlimitneg)))

        dzpositron=np.append(np.subtract(positron.getz(zl=driftlimitpos, zu=zsize),
        positron.getzbirth(zl=driftlimitpos, zu=zsize)),
        np.subtract(positron.getz(zl=-zsize, zu=driftlimitneg),
        positron.getzbirth(zl=-zsize, zu=driftlimitneg)))

        transitiondz=np.append(dzelectron,dzpositron)

        dz=np.append(dz,transitiondz)
```

```
electronxbirth=np.append(electron.getxbirth(zl=driftlimitpos, zu=zsize),
 electron.getxbirth(zl=-zsize, zu=driftlimitneg))
electronybirth=np.append(electron.getybirth(zl=driftlimitpos, zu=zsize),
 electron.getybirth(zl=-zsize, zu=driftlimitneg))
electronzbirth=np.append(electron.getzbirth(zl=driftlimitpos, zu=zsize),
 electron.getzbirth(zl=-zsize, zu=driftlimitneg))
positronxbirth=np.append(positron.getxbirth(zl=driftlimitpos, zu=zsize),
 positron.getxbirth(zl=-zsize, zu=driftlimitneg))
positronybirth=np.append(positron.getybirth(zl=driftlimitpos, zu=zsize),
 positron.getybirth(zl=-zsize, zu=driftlimitneg))
positronzbirth=np.append(positron.getzbirth(zl=driftlimitpos, zu=zsize),
 positron.getzbirth(zl=-zsize, zu=driftlimitneg))
transitionxbirth=np.append(electronxbirth,positronxbirth)
transitionybirth=np.append(electronybirth,positronybirth)
transitionzbirth=np.append(electronzbirth,positronzbirth)
xbirth = np.append(xbirth,transitionxbirth)
ybirth = np.append(ybirth,transitionybirth)
zbirth = np.append(zbirth,transitionzbirth)
electronxdeath=np.append(electron.getx(zl=driftlimitpos, zu=zsize),
 electron.getx(zl=-zsize, zu=driftlimitneg))
electronydeath=np.append(electron.gety(zl=driftlimitpos, zu=zsize),
 electron.gety(zl=-zsize, zu=driftlimitneg))
electronzdeath=np.append(electron.getz(zl=driftlimitpos, zu=zsize),
 electron.getz(zl=-zsize, zu=driftlimitneg))
positronxdeath=np.append(positron.getx(zl=driftlimitpos, zu=zsize),
 positron.getx(zl=-zsize, zu=driftlimitneg))
positronydeath=np.append(positron.gety(zl=driftlimitpos, zu=zsize),
         positron.gety(zl=-zsize, zu=driftlimitneg))
        positronzdeath=np.append(positron.getz(zl=driftlimitpos, zu=zsize),
         positron.getz(zl=-zsize, zu=driftlimitneg))
        transitionxdeath=np.append(electronxdeath,positronxdeath)
        transitionydeath=np.append(electronydeath,positronydeath)
        transitionzdeath=np.append(electronzdeath,positronzdeath)
        xdeath = np.append(xdeath,transitionxdeath)
        ydeath = np.append(ydeath,transitionydeath)
        zdeath = np.append(zdeath,transitionzdeath)



############################################################################################

w3d.interpdk[1] = 1
w3d.interpdk[0] = 1
w3d.igradb = 1

w3d.boundxy = dirichlet
top.pboundxy = absorb
w3d.boundnz = dirichlet
top.pboundnz = absorb
w3d.bound0 = dirichlet
top.pbound0 = absorb


#top.fstype = -1
top.lrelativ = 0
top.relativity = 0

solver = MultiGrid3D()

registersolver(solver)

if rank == 0:

    with open(filepath+particle_density_file+"bm="+str(bm)+"fields.m", "a")

     as myfile:

            myfile.write("{")

    with open(filepath+particle_density_file+"bm="+str(bm)+"lost.m", "a")

     as myfile:

            myfile.write("{")

    with open(filepath+particle_density_file+"bm="+str(bm)+"totalcount.m", "a")

     as myfile:

            myfile.write("{")

    with open(filepath+particle_density_file+"bm="+str(bm)+"potential.m", "a")

     as myfile:

            myfile.write("{")

#########################################################################################

# ---Run Pic


package("w3d")

generate()

fieldsolve()




from warp.lattice.loadgradb import setbsqgrad
```

```
setbsqgrad(xgrid, ygrid, zgrid, -xsize, xsize, -ysize, ysize, -zsize, zsize)



for i in xrange(1, looptimes):
    if stepcount>= ((snapshot-1)/2) and lpartrecord == 0:
        lpartrecord = 1 #only want this code to fire once.
        installbeforescraper(drift_delta)
    step()




    stepcount = stepcount + 1
    i = i + 1 #increments the for loops advancing the sim.




EX=numpy.array(electron.getx())
EY=numpy.array(electron.gety())
EZ=numpy.array(electron.getz())
PX=numpy.array(positron.getx())
PY=numpy.array(positron.gety())
PZ=numpy.array(positron.getz())
EvX=numpy.array(electron.getvx())
EvY=numpy.array(electron.getvy())
EvZ=numpy.array(electron.getvz())
PvX=numpy.array(positron.getvx())
PvY=numpy.array(positron.getvy())
```

```
PvZ=numpy.array(positron.getvz())

if rank==0:

    np.savetxt(filepath+particle_density_file+"bm="+str(bm)+"EX.txt",EX, delimiter=',')
    np.savetxt(filepath+particle_density_file+"bm="+str(bm)+"EY.txt",EY, delimiter=',')
    np.savetxt(filepath+particle_density_file+"bm="+str(bm)+"EZ.txt",EZ, delimiter=',')
    np.savetxt(filepath+particle_density_file+"bm="+str(bm)+"PX.txt",PX, delimiter=',')
    np.savetxt(filepath+particle_density_file+"bm="+str(bm)+"PY.txt",PY, delimiter=',')
    np.savetxt(filepath+particle_density_file+"bm="+str(bm)+"PZ.txt",PZ, delimiter=',')
    np.savetxt(filepath+particle_density_file+"bm="+str(bm)+"EvX.txt",EvX,delimiter=',')
    np.savetxt(filepath+particle_density_file+"bm="+str(bm)+"EvY.txt",EvY,delimiter=',')
    np.savetxt(filepath+particle_density_file+"bm="+str(bm)+"EvZ.txt",EvZ,delimiter=',')
    np.savetxt(filepath+particle_density_file+"bm="+str(bm)+"PvX.txt",PvX,delimiter=',')
    np.savetxt(filepath+particle_density_file+"bm="+str(bm)+"PvY.txt",PvY,delimiter=',')
    np.savetxt(filepath+particle_density_file+"bm="+str(bm)+"Pvz.txt",PvZ,delimiter=',')



EposX=[]
PposX=[]
charge=getrho()
if rank==0:
    for i in range(w3d.nx):
        for j in range(w3d.ny):
            for k in range(w3d.nz):
                with open(filepath+particle_density_file+"bm="+str(bm)+"rho.m", "a")
                 as myfile:
                    myfile.write(str((i-(xgrid/2))*xstep))
                    myfile.write(str(' '))
                    myfile.write(str((j-(ygrid/2))*ystep))
                    myfile.write(str(' '))
                    myfile.write(str((k-(zgrid/2))*zstep))
                    myfile.write(str(' '))
                    myfile.write(str(charge[i][j][k]))
                    myfile.write("\n")

electroncharge=electron.get_density()
positroncharge=positron.get_density()
if rank==1:
    for i in range(w3d.nx):
        for j in range(w3d.ny):
            for k in range(w3d.nz):
                with open(filepath+particle_density_file+"bm="+str(bm)+"debyelengths.m",
                 as myfile:
                    myfile.write(str((i-(xgrid/2))*xstep))
                    myfile.write(str(' '))
                    myfile.write(str((j-(ygrid/2))*ystep))
                    myfile.write(str(' '))
                    myfile.write(str((k-(zgrid/2))*zstep))
                    myfile.write(str(' '))
                    if electroncharge[i][j][k]+
                    positroncharge[i][j][k]==0:
                        myfile.write(str(0))
                        myfile.write("\n")
                    else:
```

```
                              chargedensity=float(electroncharge[i][j][k]
                              +positroncharge[i][j][k])
                              debye_length=sqrt((eps0*boltzmann*temp)/
                              (chargedensity*(echarge)**2))
                              myfile.write(str((10*S)/debye_length))
                              myfile.write("\n")


phi=getphi()
if rank == 2:
    for i in range(w3d.nx):
        for j in range(w3d.ny):
            for k in range(w3d.nz):
                with open(filepath+particle_density_file+"bm="+str(bm)+"potential.m", "a
                 as myfile:
                    myfile.write(str((i-(xgrid/2))*xstep))
                    myfile.write(str(' '))
                    myfile.write(str((j-(ygrid/2))*ystep))
                    myfile.write(str(' '))
                    myfile.write(str((k-(zgrid/2))*zstep))
                    myfile.write(str(' '))
                    myfile.write(str(phi[i][j][k]))
                    myfile.write("\n")



tote=[]
totp=[]
for i in range(w3d.nx):
    for k in range(w3d.nz):
```

```
ylineaverage = (np.sum(electroncharge[i,:,k])+
np.sum(positroncharge[i,:,k]))/(ygrid)
if rank==2: #trying to get the lineaveraged omega values
    with open(filepath+particle_density_file+
    "bm="+str(bm)+"ylineaverage.m", "a")
     as myfile:
        myfile.write(str((i-(xgrid/2))*xstep))
        myfile.write(str(' '))
        myfile.write(str((k-(zgrid/2))*zstep))
        myfile.write(str(' '))
        if ylineaverage==0:
            myfile.write(str(0))
            myfile.write("\n")
        else:
            debye_length=sqrt((eps0*boltzmann*temp)/
            (ylineaverage*(echarge)**2))
            myfile.write(str((10*S)/debye_length))
            myfile.write("\n")


    if i==0.04/xstep and k==0.04/zstep:
        with open(filepath+particle_density_file+"bm="+
        str(bm)+"numberdensity.m", "a")
         as myfile:
            myfile.write("%.8f," %((10*S)/debye_length))
    elif i==-0.04/xstep and k==0.04/zstep:
        with open(filepath+particle_density_file+"bm="+
        str(bm)+"numberdensity.m", "a")
         as myfile:
```

```
                    myfile.write("%.8f," %((10*S)/debye_length))
            elif i==-0.04/xstep and k==-0.04/zstep:
                with open(filepath+particle_density_file+"bm="+
                str(bm)+"numberdensity.m", "a")
                 as myfile:
                    myfile.write("%.8f," %((10*S)/debye_length))
            elif i==0.04/xstep and k==-0.04/zstep:
                with open(filepath+particle_density_file+"bm="+
                str(bm)+"numberdensity.m", "a")
                 as myfile:
                    myfile.write("%.8f," %((10*S)/debye_length))

if rank == 0:
        np.savetxt(filepath+particle_density_file+"bm="+str(bm)+"xdeltas.m",
        np.c_[dx,xbirth,xdeath], delimiter=' ')

if rank == 1:
        np.savetxt(filepath+particle_density_file+"bm="+str(bm)+"ydeltas.m",
        np.c_[dy,ybirth,ydeath], delimiter=' ')

if rank == 2:
        np.savetxt(filepath+particle_density_file+"bm="+str(bm)+"zdeltas.m",
        np.c_[dz,zbirth,zdeath], delimiter=' ')

if rank == 0:

    with open(filepath+particle_density_file+"bm="+str(bm)+"lost.m", "a")
     as myfile:
            myfile.seek(-1, os.SEEK_END)
            myfile.truncate()
            myfile.write("}")

    with open(filepath+particle_density_file+"bm="+str(bm)+"totalcount.m", "a")
     as myfile:
            myfile.seek(-1, os.SEEK_END)
            myfile.truncate()
            myfile.write("}")

    with open(filepath+particle_density_file+"bm="+str(bm)+"fields.m", "a")
     as myfile:
            myfile.seek(-1, os.SEEK_END)
            myfile.truncate()
            myfile.write("}")

    with open(filepath+particle_density_file+"bm="+str(bm)+"potential.m", "a")
     as myfile:
            myfile.seek(-1, os.SEEK_END)
            myfile.truncate()
            myfile.write("}")
```

REFERENCES

[1] P.K. Roy, S.S. Yu, E. Henestroza, A. Anders, F.M. Bieniosek, J. Coleman, S. Eylon, W.G. Greenway, M. Leitner, B.G. Logan, W.L. Waldron, D.R. Welch, C. Thoma, A.B. Sefkow, E.P. Gilson, P.C. Efthimion, and R.C. Davidson. Drift compression of an intense neutralized ion beam. *Phys. Rev. Lett.*, 95:234801, Nov 2005.

[2] W. Halverson and D.T. Quinto. Effects of charge neutralization on ionbeamdeposited boron nitride films. *Journal of Vacuum Science Technology A*, 3(6):2141–2146, 1985.

[3] B Lehnert. Plasma confinement in presence of magnetically shielded supports. *Plasma Physics*, 17(7-8):501, 1975.

[4] H. Ishizuka and S. Robertson. Propagation of an intense chargeneutralized ion beam transverse to a magnetic field. *Physics of Fluids*, 25(12):2353, 1982.

[5] J. Maenchen, L. Wiley, S. Humphries, E. Peleg, R.N. Sudan, and D.A. Hammer. Magnetic focusing of intense ion beams. *Physics of Fluids*, 22(3):555, 1979.

[6] J.L. Pacheco, C.A. Ordonez, and D.L. Weathers. Spatially periodic electromagnetic force field for plasma confinement and control. *AIP Conference Proceedings*, 1525:88, apr 2013.

[7] C.A. Ordonez. Drifting plasma confinement with a spatially periodic field. *Plasma Science, IEEE Transactions on*, 38(3):388–392, March 2010.

[8] S. Humphries. *Charged Particle Beams*, chapter 11. John Wiley & Sons, Inc., 1990.

[9] T. Schenkel, S.M. Lidia, C.D. Weis, W.L. Waldron, J. Schwartz, A.M. Minor, P. Hosemann, and J.W. Kwan. Towards pumpprobe experiments of defect dynamics with short ion beam pulses. *Nuclear Instruments and Methods in Physics Research B*, 315(0):350–355, 2013.

[10] D.P. Grote, A. Friedman, and W.M. Sharp. Simulations of ion beams for ndcx-ii. *Nuclear Instruments and Methods in Physics Research A*, 733(0):134–140, 2014.

[11] P.C. Efthimion, E.P. Gilson, R.C. Davidson, L. Grisham, B.G. Logan, P.A. Seidl,

W. Waldron, and S.S. Yu. Ferroelectric plasma source for heavy ion beam space charge neutralization. *Nuclear Instruments and Methods in Physics Research A*, 577(12):203–206, 2007.

[12] P.C. Efthimion, E.P. Gilson, R.C. Davidson, L. Grisham, B.G. Logan, P.A. Seidl, and W. Waldron. Meter-long plasma source for heavy ion beam space charge neutralization. In *Particle Accelerator Conference, 2007. IEEE, PAC07, THPAS 082*, page 3672, June 2007.

[13] P.E. Larson and M.A. Kelly. Surface charge neutralization of insulating samples in x-ray photoemission spectroscopy. *Journal of Vacuum Science and Technology A*, 16(6):3483, 1998.

[14] K.N. Leung, K.C. Gordon, W.B. Kunkel, C.M. McKenna, S.R. Walther, and M.D. Williams. An electron-beam charge neutralization system for ion implanters. *Nuclear Instruments and Methods in Physics Research B*, 55(14):94–96, 1991.

[15] D.L. Smatlak, M.E. Mack, and S. Mehta. Charge neutralization in ion implanters. *Nuclear Instruments and Methods in Physics Research B*, 96(12):22–29, 1995.

[16] G.C. Theodoridis and E.P. Gyftopoulos. Cesium ion-beam neutralization with energetic electrons. *AIAA Journal*, 3:1414, 1965.

[17] C.A. Ordonez. Charged particle reflection from an artificially structured boundary that produces a spatially periodic magnetostatic field. *Journal of Applied Physics*, 106:024905, 2009.

[18] R.E. Phillips and C.A. Ordonez. Magnetic control of a charge-neutralized ion beam. *Physics Procedia*, 66:148–155, 2015. The 23rd International Conference on the Application of Accelerators in Research and Industry - CAARI 2014.

[19] K.H. Burrell. Effects of eb velocity shear and magnetic shear on turbulence and transport in magnetic confinement devices. *Physics of Plasmas*, 4(5):1499–1518, 1997.

[20] E.J. Synakowski. Formation and structure of internal and edge transport barriers. *Plasma Physics and Controlled Fusion*, 40(5):581, 1998.

[21] E.J. Doyle, W.A. Houlberg, Y. Kamada, V. Mukhovatov, T.H. Osborne, A. Polevoi,

G. Bateman, J.W. Connor, J.G. Cordey, T. Fujita, X. Garbet, T.S. Hahm, L.D. Horton, A.E. Hubbard, F. Imbeaux, F. Jenko, J.E. Kinsey, Y. Kishimoto, J. Li, T.C. Luce, Y. Martin, M. Ossipenko, V. Parail, A. Peeters, T.L. Rhodes, J.E. Rice, C.M. Roach, V. Rozhansky, F. Ryter, G. Saibene, R. Sartori, A.C.C. Sips, J.A. Snipes, M. Sugihara, E.J. Synakowski, H. Takenaga, T. Takizuka, K. Thomsen, M.R. Wade, H.R. Wilson, ITPA Transport Physics Topical Group, ITPA Confinement Database, Modelling Topical Group, ITPA Pedestal, and Edge Topical Group. Chapter 2: Plasma confinement and transport. *Nuclear Fusion*, 47(6):S18, 2007.

[22] S. Yoshikawa. Experiments on plasma confinement in internal-ring devices. *Nuclear Fusion*, 13(3):433, 1973.

[23] J.C. Sprott and S.C. Prager. Multipole and tokamak research at the university of wisconsin. *Nuclear Fusion*, 25(9):1179, 1985.

[24] M.W. Maisel, T. Ohkawa, K.H. Burrell, R.L. Freeman, F.J. Helton, T.H. Jensen, R.J. La Haye, D.O. Overskei, R. Prater, J.M. Rawls, and T. Tamano. The fusion power research programme at ga technologies inc. *Nuclear Fusion*, 25(9):1113, 1985.

[25] News picks : Lockheed martin looking for compact fusion reactor partners, October 2014.

[26] J. Hedditch, R. Bowden-Reid, and J. Khachan. Fusion energy in an inertial electrostatic confinement device using a magnetically shielded grid. *Physics of Plasmas*, 22(10):102705, 2015.

[27] C. Amole, G.B. Andresen, M.D. Ashkezari, M. Baquero-Ruiz, W. Bertsche, P.D. Bowe, E. Butler, A. Capra, P.T. Carpenter, C.L. Cesar, S. Chapman, M. Charlton, A. Deller, S. Eriksson, J. Escallier, J. Fajans, T. Friesen, M.C. Fujiwara, D.R. Gill, A. Gutierrez, J.S. Hangst, W.N. Hardy, R.S. Hayano, M.E. Hayden, A.J. Humphries, J.L. Hurt, R. Hydomako, C.A. Isaac, M.J. Jenkins, S. Jonsell, L.V. Jrgensen, S.J. Kerrigan, L. Kurchaninov, N. Madsen, A. Marone, J.T.K. McKenna, S. Menary, P. Nolan, K. Olchanski, A. Olin, B. Parker, A. Povilus, P. Pusa, F. Robicheaux, E. Sarid, D. Seddon, S. Seif El Nasr, D.M. Silveira, C. So, J.W. Storey, R.I. Thompson, J. Thornhill, D. Wells, D.P.

van der Werf, J.S. Wurtele, and Y. Yamazaki. The alpha antihydrogen trapping apparatus. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 735(0):319–340, 2014.

[28] N. Kuroda, H. A. Torii, K.Y. Franzen, Z. Wang, S. Yoneda, M. Inoue, M. Hori, B. Juhász, D. Horváth, H. Higaki, A. Mohri, J. Eades, K. Komaki, and Y. Yamazaki. Confinement of a large number of antiprotons and production of an ultraslow antiproton beam. *Phys. Rev. Lett.*, 94:023401, Jan 2005.

[29] A.I. Zhmoginov, A.E. Charman, R. Shalloo, J. Fajans, and J.S. Wurtele. Nonlinear dynamics of anti-hydrogen in magnetostatic traps: implications for gravitational measurements. *Classical and Quantum Gravity*, 30(20):205014, 2013.

[30] G. Gabrielse, R. Kalra, W.S. Kolthammer, R. McConnell, P. Richerme, D. Grzonka, W. Oelert, T. Sefzick, M. Zielinski, D.W. Fitzakerley, M.C. George, E.A. Hessels, C.H. Storry, M. Weel, A. Müllers, and J. Walz. Trapped antihydrogen in its ground state. *Phys. Rev. Lett.*, 108:113002, Mar 2012.

[31] H. Higaki, Y. Enomoto, N. Kuroda, K. Michishio, D.J. Murtagh, S. Ulmer, S. Van Gorp, C.H. Kim, Y. Nagata, Y. Kanai, H.A. Torii, M. Corradini, M. Leali, E. Lodi-Rizzini, V. Mascagna, L. Venturelli, N. Zurlo, K. Fujii, M. Otsuka, K. Tanaka, H. Imao, Y. Nagashima, Y. Matsuda, B. Juhsz, A. Mohri, and Y. Yamazaki. Towards the production of anti-hydrogen beams. *AIP Conference Proceedings*, 1521(1):134–143, 2013.

[32] P. Scampoli and J. Storey. The aegis experiment at cern for the measurement of antihydrogen gravity acceleration. *Modern Physics Letters A*, 29(17):1430017, 2014.

[33] P. Comini, P.A. Hervieux, and F. Biraben. Hbar production from collisions between positronium and kev antiprotons for gbar. *Hyperfine Interactions*, 228(1-3):159–165, 2014.

[34] The ALPHA Collaboration. Confinement of antihydrogen for 1000 seconds. *Nature Physics*, 7:558–564, 2011.

[35] C. Amole, M.D. Ashkezari, M. Baquero-Ruiz, W. Bertsche, P.D. Bowe, E. Butler, A. Capra, C.L. Cesar, M. Charlton, A. Deller, P.H. Donnan, S. Eriksson, J. Fajans,

T. Friesen, M.C.Fujiwara, D.R. Gill, A. Gutierrez, J.S. Hangst, W.N. Hardy, M.E. Hayden, A.J. Humphries, C.A. Isaac, S. Jonsell, L. Kurchaninov, A. Little, N. Madsen, J.T.K. McKenna, S. Menary, S.C. Napoli, P. Nolan, K. Olchanski, A. Olin, P. Pusa, C.O. Rasmussen, F. Robicheaux, E. Sarid, C.R. Shields, D.M. Silveira, S. Stracka, C. So, R.I. Thompson, D.P. van der Werf, and J.S. Wurtele. Resonant quantum transitions in trapped antihydrogen atoms. *Nature*, 483(7390):439–443, Mar 2012.

[36] C. Amole, M.D. Ashkezari, M. Baquero-Ruiz, W. Bertsche, E. Butler, A. Capra, C.L. Cesar, M. Charlton, S. Eriksson, J. Fajans, T. Friesen, M.C. Fujiwara, D.R. Gill, A. Gutierrez, J.S. Hangst, W.N. Hardy, M.E. Hayden, C.A. Isaac, S. Jonsell, L. Kurchaninov, A. Little, N. Madsen, J.T.K. McKenna, S. Menary, S.C. Napoli, P. Nolan, K. Olchanski, A. Olin, A. Povilus, P. Pusa, C.Ø. Rasmussen, F. Robicheaux, E. Sarid, D.M. Silveira, C. So, T.D. Tharp, R.I. Thompson, D.P. van der Werf, Z. Vendeiro, J.S. Wurtele, A.I. Zhmoginov, and A.E. Charman. An experimental limit on the charge of antihydrogen. *Nature Communications*, 5:3955 EP –, Jun 2014.

[37] A.E. Charman. Description and first application of a new technique to measure the gravitational mass of antihydrogen. *Nature Communications*, 4:1785 EP –, Apr 2013.

[38] C.A. Ordonez and R.M. Hedlof. Simulation of an aperture-based antihydrogen gravity experiment. *AIP Advances*, 2(1):012176, 2012.

[39] R.M. Hedlof and C.A. Ordonez. Simulation of an antihydrogen gravity experiment utilizing multiple apertures. *AIP Conference Proceedings*, 1525(1):102–105, 2013.

[40] J.R. Correa and C.A. Ordonez. Magnetobound positronium and protonium. *Physics of Plasmas*, 21(8):082115, 2014.

[41] J.D. Wofford and C.A. Ordonez. Dual levitated coils for antihydrogen production. In F.D. McDaniel, B.L. Doyle, G.A. Glass, and Y. Wang, editors, *American Institute of Physics Conference Series*, volume 1525 of *American Institute of Physics Conference Series*, pages 106–110, apr 2013.

[42] R.A. Lane and C.A. Ordonez. Classical trajectory monte carlo simulations of particle confinement using dual levitated coils. *AIP Advances*, 4(7):–, 2014.

[43] A. Friedman, R.H. Cohen, D.P. Grote, S.M. Lund, W.M. Sharp, J.L. Vay, I. Haber, and R. A. Kishek. Computational methods in the warp code framework for kinetic simulations of particle beams and plasmas. *IEEE Transactions on Plasma Science*, 42(5):1321–1334, May 2014.

[44] J.L. Vay, D.P. Grote, R.H. Cohen, and A. Friedman. Novel methods in the particle-in-cell accelerator code-framework warp. *Computational Science Discovery*, 5(1):014019, 2012.

[45] J.L. Vay, C.G.R. Geddes, E. Cormier-Michel, and D.P. Grote. Numerical methods for instability mitigation in the modeling of laser wakefield accelerators in a lorentz-boosted frame. *Journal of Computational Physics*, 230(15):5908–5929, 2011.

[46] S.M. Lund, T. Kikuchi, and R.C. Davidson. Generation of initial kinetic distributions for simulation of long-pulse charged particle beams with high space-charge intensity. *Phys. Rev. ST Accel. Beams*, 12:114801, Nov 2009.

[47] J.L. Vay. Noninvariance of space- and time-scale ranges under a lorentz transformation and the implications for the study of relativistic interactions. *Phys. Rev. Lett.*, 98:130405, Mar 2007.

[48] S.M. Lund, S.H. Chilton, and E.P. Lee. Efficient computation of matched solutions of the kapchinskij-vladimirskij envelope equations for periodic focusing lattices. *Phys. Rev. ST Accel. Beams*, 9:064201, Jun 2006.

[49] J.L. Vay, P. Colella, J.W. Kwan, P. McCorquodale, D.B. Serafini, A. Friedman, D.P. Grote, G. Westenskow, J.C. Adam, A. Hron, and I. Haber. Application of adaptive mesh refinement to particle-in-cell simulations of plasmas and beams. *Physics of Plasmas*, 11(5):2928–2934, 2004.

[50] D.P. Grote, A. Friedman, J.L. Vay, and I. Haber. The warp code: Modeling high intensity ion beams. *AIP Conference Proceedings*, 749(1):55–58, 2005.

[51] A. Friedman, D.P. Grote, and I. Haber. Threedimensional particle simulation of heavyion fusion beams. *Physics of Fluids B: Plasma Physics*, 4(7):2203–2210, 1992.

[52] K. Gomberoff, J. Fajans, A. Friedman, D.P. Grote, J.L. Vay, and J.S. Wurtele. Simula-

tions of plasma confinement in an antihydrogen trap. *Physics of Plasmas*, 14(10):102111, 2007.

[53] K. Gomberoff, J. Fajans, J.S. Wurtele, A. Friedman, D.P. Grote, R.H. Cohen, and J.L. Vay. Simulation studies of non-neutral plasma equilibria in an electrostatic trap with a magnetic mirror. *Physics of Plasmas*, 14(5):052107, 2007.

[54] A. Narimannezhad, C.J. Baker, M.H. Weber, J. Jennings, and K.G. Lynn. Simulation studies of the behavior of positrons in a microtrap with long aspect ratio. *The European Physical Journal D*, 68(11):351, 2014.

[55] C.K. Birdsall. Particle-in-cell charged-particle simulations, plus monte carlo collisions with neutral atoms, pic-mcc. *Plasma Science, IEEE Transactions on*, 19(2):65–85, Apr 1991.

[56] R.H. Cohen, A. Friedman, D.P. Grote, and J.L. Vay. Large-timestep mover for particle simulations of arbitrarily magnetized species. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 577(12):52–57, 2007. Proceedings of the 16th International Symposium on Heavy Ion Inertial FusionHIF 06.

[57] L. Garrigues, G. Fubiani, and J.P. Boeuf. Appropriate use of the particle-in-cell method in low temperature plasmas: Application to the simulation of negative ion extraction. *Journal of Applied Physics*, 120(21):213303, 2016.

[58] R. E. Phillips and C. A. Ordonez. Magnetic plasma expulsion. *Physics of Plasmas*, 25(1):012508, 2018.

[59] K. Gomberoff, J. Wurtele, A. Friedman, D.P. Grote, and J.-L. Vay. A method for obtaining three-dimensional computational equilibrium of non-neutral plasmas using warp. *Journal of Computational Physics*, 225(2):1736 – 1752, 2007.