Plasma Simulation

Thomas Beitel

April 25, 2021

Abstract

The purpose of this project was to create a visual fluid simulation of a plasma. A real time simulation of a simplified, fully ionized plasma was produced through a finite difference approximation of the linearized set of plasma equations. This simulation produced a fluid that propagated entirely on electromagnetic field effects that even produced visible waves. When run with random initial velocity conditions, the simulation converged on a web-like pattern that closely resembles interstellar nebulae.

1 Introduction

Plasma physics is the study of the plasma state of matter. More specifically, it involves the study of the interactions between the various neutral and charged particles and the resulting electromagnetic and acoustic waves caused by the interactions. Plasmas come in many forms from the sparse, weakly-ionized interstellar molecular clouds to the extremely-pressurized of fusion plasmas. This project will be focused on more sparse plasmas due to the ability to approximate them closely with a linear system. The study of these kinds of plasmas is important because sparse plasmas make up interstellar molecular clouds that are the kind of place where stars form. By studying the content of these clouds, one could learn a lot about the earliest stages in star formation including how the clouds of plasma fragment into star-forming masses.

The main difficulty with calculating the dynamics of plasmas is the fact that they are made up of charged particles. This causes a major electromagnetic

effect in addition to the effects standard for fluid dynamics. As the plasma moves, it generates a magnetic field which, in turn, affects the future movement of the plasma particles. This effect creates unique waves that cannot be found in any other kind of fluid. Through their interaction with the electromagnetic field, plasmas can propagate waves at the speed of light (itself a kind of electromagnetic wave) in addition to acoustic effects whereas non-charged fluids can only propagate through sound waves. The focus of this project is to create a fluid rendering of a plasma using a modification of conventional fluid rendering methods. [1] Because the purpose is to create a visual rather than a scientific simulation, a greater focus was put on computational speed rather than accuracy.

2 Related Work

Because this project straddles two scientific disciplines, there is not much work on the real-time simulation of plasmas. In physics research, there are many simulations of different kinds of plasmas from nuclear fusion experiments [2] to solar wind [3] [4]. However, these simulations are of an analytical nature and are not meant for efficient rendering. Although they are extremely parallelized, they are built to be as accurate as possible and are meant to be run on super computers and take hours to weeks to simulate. Needless to say, these algorithms would make poor real-time graphical simulations.

The more efficient algorithms come from computer science research into rendering fluids. [1] Using the finite difference approximations, these algorithms work a lot more efficiently. However, the main drawback is that the studied algorithms are almost entirely derived from the standard Navier-Stokes equation and would need to be significantly adjusted for electromagnetic effects. Nevertheless, this was the starting point for the project: the finite difference approximation of the Navier-Stokes equations.

3 Plasma Equations

A non-charged fluid is described by the Navier-Stokes equation (1a) and the continuity equation (1b). [1]

$$\rho \left[\frac{d}{dt}\right] \boldsymbol{v} = -\nabla P + \rho \boldsymbol{g} + \nabla^2 \nu \tag{1a}$$

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot (\rho \boldsymbol{v}) \tag{1b}$$

$$\left[\frac{d}{dt}\right] = \frac{\partial}{\partial t} + \boldsymbol{v} \cdot \nabla$$

$\boldsymbol{v}$ is the velocity, P is the pressure, $\boldsymbol{g}$ is the gravity, ν is the fluid viscosity, and ρ is the fluid density. The solution to this system of nonlinear partial differential equations is the density and velocity of the fluid as a function of position and time.

For a simple, fully ionized plasma (entirely made up of charged particles), there is a third variable to track of along with the density and velocity: the

magnetic field $\boldsymbol{B}$. Because the particles are charged and moving, the magnetic field at each point changes over time. Even for a sparse plasma like an interstellar plasma, the effects from the magnetic field are much more powerful than any other forces on the plasma to the point where the effects of pressure, viscosity, and even gravity can usually be ignored. More accurate measurements of interstellar plasmas do take self-gravity, pressure, and heating into effect, but for the purposes of this project, the simplified model will do. [5]

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot (\rho \boldsymbol{v}) \tag{2a}$$

$$\rho \left[\frac{d}{dt} \right] \boldsymbol{v} = \frac{(\nabla \times \boldsymbol{B}) \times \boldsymbol{B}}{4\pi} \tag{2b}$$

$$\frac{\partial \boldsymbol{B}}{\partial t} = \nabla \times (\boldsymbol{v} \times \boldsymbol{B}) \tag{2c}$$

$$\tag{2d}$$

The first equation (2a) is the same continuity equation found in the set for a standard fluid. The second equation (2b) is similar to the Navier-Stokes equation, but instead of the pressure, gravity, and viscosity terms, there is just one term for magnetic pressure. The third equation is derived from the Maxwell-Faraday equation and Lorentz's Force Law.

The reason why interstellar plasmas were chosen instead of other plasmas is because interstellar plasmas can be approximated with linear effects. This allows the plasma equations to be linearized which significantly simplifies the system which improves computational performance. As plasmas get denser, the higher order terms become significant preventing the linearized solution from being accurate. For the ultra-dense fusion plasmas, experimental effects don't completely line up with standard theoretical plasma models requiring methods of even higher complexity to accurately model. [6]

The linearization step invokes the assumption $f(\boldsymbol{x}, t) = f_0 + f_1(\boldsymbol{x}, t)$ for each of the unknown variables where f_0 is a constant and $f_0 >> f_1$.

$$\frac{\partial \rho}{\partial t} = -\rho_0 \nabla \cdot \boldsymbol{v} \tag{3a}$$

$$\frac{\partial \boldsymbol{v}}{\partial t} = \frac{1}{4\pi\rho_0} (\nabla \times \boldsymbol{B}) \times \boldsymbol{B}_0 \tag{3b}$$

$$\frac{\partial \boldsymbol{B}}{\partial t} = \nabla \times (\boldsymbol{v} \times \boldsymbol{B}_0) \tag{3c}$$

$$\tag{3d}$$

One common assumption made for plasma systems is that the initial magnetic field $\boldsymbol{B}_0$ is entirely in the $+\hat{z}$. For an interstellar cloud, the system is relatively symmetric and any coordinate system can be transformed into one where the initial magnetic field follows this condition. Because this condition greatly simplifies the system of equations, it is used. After expanding the vectors and simplifying the vector operators, the result is seven variables and seven linear partial differential equations.

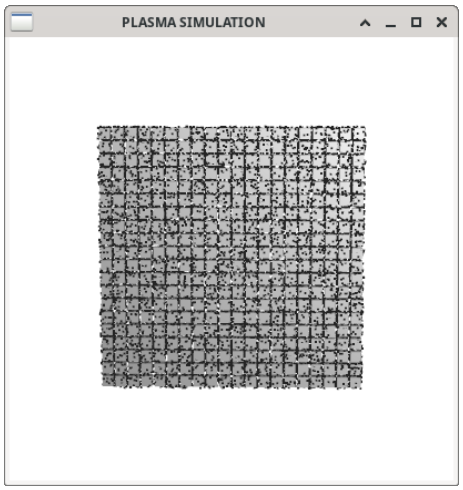

Figure 1: Visualization of the cells in the simulation. Each stores a value at its center and the motion of the particles is interpolated from nearby cells.

$$\frac{\partial \rho}{\partial t} = -\rho_0 \left(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z} \right) \tag{4a}$$

$$\frac{\partial v_x}{\partial t} = \frac{B_0}{4\pi\rho_0} \left(\frac{\partial B_x}{\partial z} - \frac{\partial B_z}{\partial x} \right) \tag{4b}$$

$$\frac{\partial v_y}{\partial t} = \frac{B_0}{4\pi\rho_0} \left(\frac{\partial B_y}{\partial z} - \frac{\partial B_z}{\partial y} \right) \tag{4c}$$

$$\frac{\partial v_z}{\partial t} = 0 \tag{4d}$$

$$\frac{\partial B_x}{\partial t} = B_0 \frac{\partial v_x}{\partial z} \tag{4e}$$

$$\frac{\partial B_y}{\partial t} = B_0 \frac{\partial v_y}{\partial z} \tag{4f}$$

$$\frac{\partial B_z}{\partial t} = -B_0 \left(\frac{\partial v_x}{\partial y} + \frac{\partial v_y}{\partial x} \right) \tag{4g}$$

4 Implementation

4.1 Finite Differences

Because the system of equations is made up entirely of first order derivatives, the system can be approximated by first order finite differences approximations.

$$\frac{\partial f(\boldsymbol{x}, t)}{\partial t} \approx \frac{f(\boldsymbol{x}, t + \delta t) - f(\boldsymbol{x}, t)}{\delta t} \tag{5}$$

The program first breaks up the fluid area into a number of discrete cells and calculates the fluid properties in the center of each cell. The particle velocities are tri-linearly interpolated between the nearest cells. When the finite difference approximation is applied to spacial terms, it results in a term similar to

$f(x, y+\delta y, z, t)$. Because of the discrete nature of the cellular fluid system, the spacial position can be translated to specific index of a cell $f(x, y, z, t) = f_{i,j,k}$. Thus a spacial shift is represented by incrementing the cell index by the respective amount; for example: $f(x, y+\delta y, z, t) = f_{i,j+1,k}$ where δx, δy, and δz would represent the distances between cells (center-to-center). This method was applied to the seven linearized plasma equations resulting in a discrete system of equations that increments by time steps.

$$\hat{\rho}_{i,j,k} = \rho_{i,j,k} - \rho_0 \delta t \left(\frac{u_{i+1,j,k} - u_{i,j,k}}{\delta x} + \frac{v_{i,j+1,k} - v_{i,j,k}}{\delta y} + \frac{w_{i,j,k+1} - w_{i,j,k}}{\delta z} \right) \tag{6a}$$

$$\hat{u}_{i,j,k} = u_{i,j,k} + \frac{B_0 \delta t}{4\pi\rho_0} \left(\frac{A_{i,j,k+1} - A_{i,j,k}}{\delta z} - \frac{C_{i+1,j,k} - C_{i,j,k}}{\delta x} \right) \tag{6b}$$

$$\hat{v}_{i,j,k} = v_{i,j,k} + \frac{B_0 \delta t}{4\pi\rho_0} \left(\frac{B_{i,j,k+1} - B_{i,j,k}}{\delta z} - \frac{C_{i,j+1,k} - C_{i,j,k}}{\delta y} \right) \tag{6c}$$

$$\hat{w}_{i,j,k} = w_{i,j,k} \tag{6d}$$

$$\hat{A}_{i,j,k} = A_{i,j,k} + B_0 \delta t \frac{u_{i,j,k+1} - u_{i,j,k}}{\delta z} \tag{6e}$$

$$\hat{B}_{i,j,k} = B_{i,j,k} + B_0 \delta t \frac{v_{i,j,k+1} - v_{i,j,k}}{\delta z} \tag{6f}$$

$$\hat{C}_{i,j,k} = C_{i,j,k} - B_0 \delta t \left(\frac{u_{i,j+1,k} - u_{i,j,k}}{\delta y} + \frac{v_{i+1,j,k} - v_{i,j,k}}{\delta x} \right) \tag{6g}$$

The representation of the time stepped value for each function is represented $\hat{f} = f(\boldsymbol{x}, t+\delta t)$. The vector variables are converted $\boldsymbol{v}(\boldsymbol{x}, t) = [u_{i,j,k}, v_{i,j,k}, w_{i,j,k}]$ and $\boldsymbol{B}(\boldsymbol{x}, t) = [A_{i,j,k}, B_{i,j,k}, C_{i,j,k}]$

4.2 Input Parameters and Data Flow

The program only takes one mandatory command line input: a path to a specific text file containing parameters for the program. This file contains additional initialization variables for the simulation. The exact parameter specifications for this file are listed in Appendix A.

Each cell stores its own values for density, velocity, and magnetic field which it updates on every time step. First the velocity is updated, then the density and magnetic field are updated using the new velocity terms where applicable. Because the linearization assumes relatively constant values for density and magnetic field, the program stores the initial values for density and magnetic field separately and uses them where applicable. Finally, the program calculates the velocity of each point through tri-linear interpolation and uses the Euler method to update the particles' positions.

5 Results

Before the application of the magnetic field, the program was first tested for the linearized Navier-Stokes equations without gravity or viscosity. This produced good, if uninteresting results. Without viscosity, the fluid kept moving, long after the initial state, but this is to be expected. Although unrealistic for a

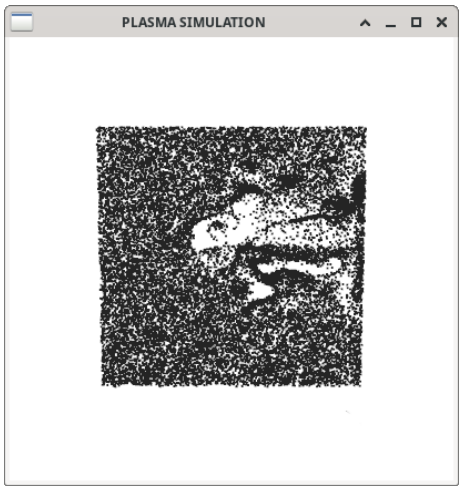

Figure 2: Non-charged fluid with cells in center given an initial velocity pointing to the right.

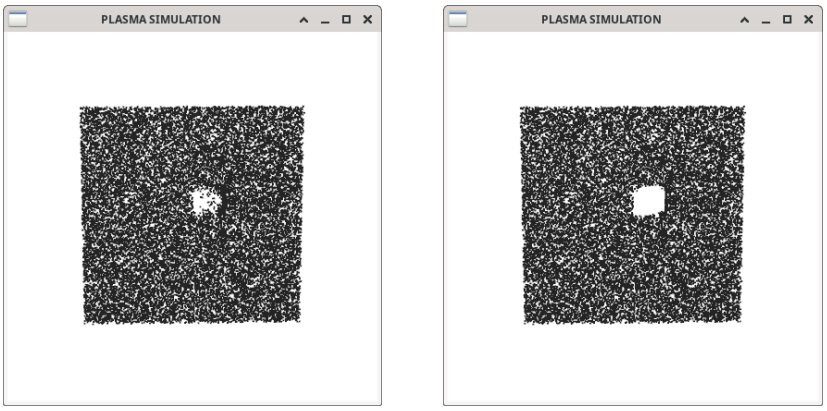

Figure 3: The cells that empty are the only ones given an initial velocity.

general fluid, viscosity would not be playing a large factor when a magnetic field is applied. Also as expected, the fluid remains static given zero initial conditions.

Interestingly, due to the fact that the magnetic field is in the $\hat{z}$ direction, the velocity in the $\hat{z}$ direction is constant allowing a representative sample of the plasma system to be made with an environment with a very thin depth.

However, after application of the magnetic field and the removal of the pressure term, the initial tests seemed to give poor results. Applying the same initial velocity in the center only resulted in that cell shifting without affecting other cells as can be seen in Figure 3. When the initial condition was applied to the whole simulation, the only result was all of the particles grouping on the right as can be seen in the bottom image of Figure 4.

However, when the magnetic field was greatly increased, the particles began to exhibit interesting effects. When applied to the simulation with a constant

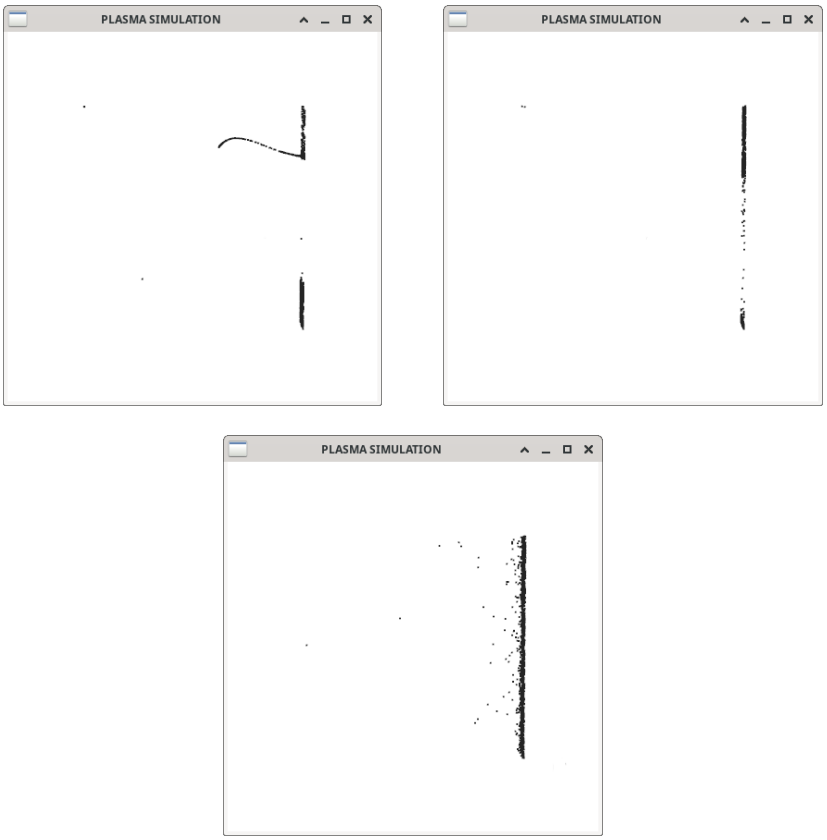

Figure 4: The particles move to the left, but when the magnetic field is high enough, the particles quickly shift in a rotation away from the center point as can be seen in the upper images.

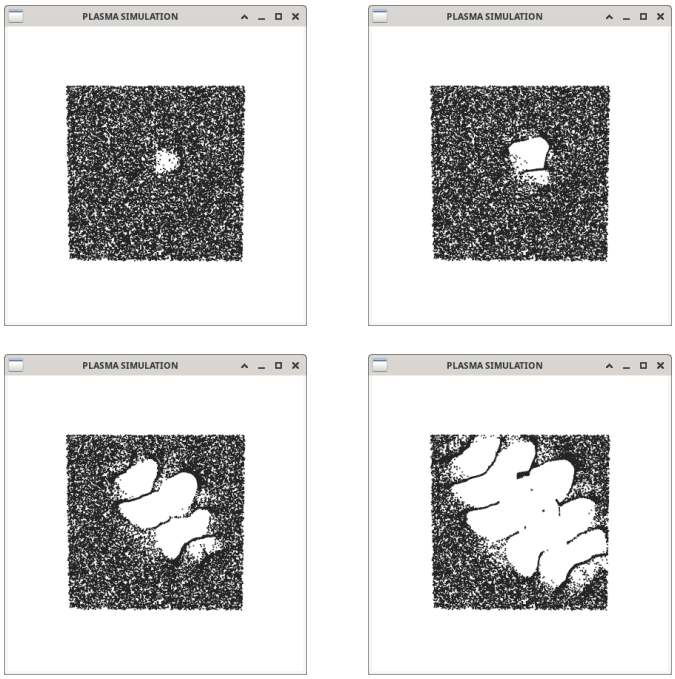

Figure 5: Although it starts much like Figure 3, the increased magnetic field quickly begins to propagate waves which continue to move through the simulation until they hit the boundaries.

initial velocity, it would eventually cause distortion in the particles grouped on the right causing them to shoot toward the corners or away from the edge as can be seen in the upper images of Figure 4.

The most interesting result comes from increasing the magnetic field for the simulation with the initial velocity in the center of the system (Figure 5). The result is a wave that propagates outward in a direction roughly orthogonal to the initial velocity. Given a larger simulation, these waves would doubtlessly continue throughout the plasma. This is significant because this wave is propagated entirely via the magnetic field, and this effect is a critical step to simulating more complex plasmas.

Another interesting case is when the simulation is given small, random initial velocities. Over time, the system converges to a web-like structure that closely resembles some kinds of nebulae (Figure 6). Because nebulae are clouds of plasma that come together over a long period of time, it is not particularly surprising that a plasma simulation might get a similar shape.

6 Future Work

The original intention of this project, to render a complete interstellar plasma as outlined by Mouschovias et al. [5], was unfortunately outside of the reasonable scope for this project.

One feature that did not get finished over the course of the project was the

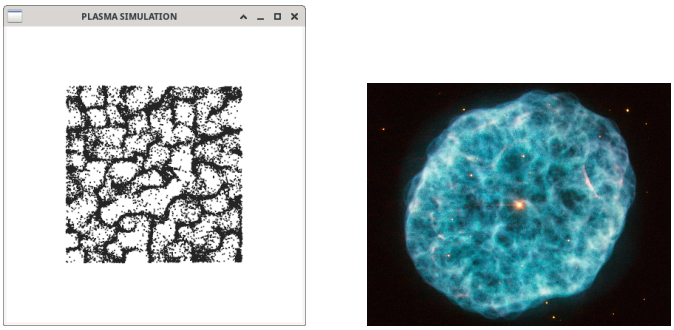

Figure 6: The random initial velocity case converges into a shape that closely resembles many real-world nebulae. (Image of Oyster Nebula courtesy of NASA https://www.nasa.gov/content/goddard/hubble-view-of-bubbly-nebula)

ability to set initial density in addition to initial velocity. In real plasma clouds, many kinds of events cause a sudden increase in density and how this disturbace propagates is an important point of study in the field of astrophysics. Being able to set an initial density will allow a researcher to simulate events like this.

Another possible expansion would be to adjust the algorithm so that it could handle several different kinds of plasma particles in the same system. Interstellar plasmas, for example, are made up of both ionized particles which are heavily influenced by electomagnetic forces (as shown above) and neutral particles which behave much like standard fluids except when they interact with the ionized particles. In addition, some plasmas have distinct positive and negative ions that each have their own waves.

It would also be great to find a better way to visualize plasma waves. For the larger simulations that involve variation in the z direction, it would be hard to see what is going on within the plasma. Perhaps the fluid density could be modeled visually using transparency. An explicit wave finder would also be useful, though it would likely require specialized analysis to pick up different waves.

Although the program works well for smaller systems, it quickly slows down as the system size increases. Setting up a parallelization scheme to update the variables for each cell would greatly increase the performance and allow for much larger renderings.

Because the algorithm used is a computational approximation, it should be possible to solve the system of plasma equations without linearizing them. This method was passed over for this project due to increased complexity, but a future work should definitely look into what kinds of adjustments would need to be made in order to solve a nonlinear system. This kind of calculation would be vital to accurately rendering more complex plasmas.

References

[1] N. Foster and D. Metaxas, "Realistic animation of liquids," *Graphical Models and Image Processing*, vol. 58, no. 5, pp. 471–483, 1996.

[2] H. Sugama *et al.*, "Recent progress in the numerical simulation reactor research project," *Plasma and Fusion Research*, vol. 14, pp. 3503059–3503059, 2019.

[3] N. V. Pogorelov, S. N. Borovikov, M. C. Bedford, J. Heerikhuisen, T. K. Kim, I. A. Kryukov, and G. P. Zank, "Modeling Solar Wind Flow with the Multi-Scale Fluid-Kinetic Simulation Suite," in *Numerical Modeling of Space Plasma Flows (ASTRONUM2012)* (N. V. Pogorelov, E. Audit, and G. P. Zank, eds.), vol. 474 of *Astronomical Society of the Pacific Conference Series*, p. 165, Apr. 2013.

[4] J. Heerikhuisen, V. Florinski, G. P. Zank, and N. V. Pogorelov, "MHD-Boltzmann Simulations of the Solar Wind-Interstellar Medium Interaction," in *Numerical Modeling of Space Plasma Flows* (G. P. Zank and N. V. Pogorelov, eds.), vol. 359 of *Astronomical Society of the Pacific Conference Series*, p. 251, Dec. 2006.

[5] T. C. Mouschovias, G. E. Ciolek, and S. A. Morton, "Hydromagnetic waves in weakly-ionized media – I. Basic theory, and application to interstellar molecular clouds," *Monthly Notices of the Royal Astronomical Society*, vol. 415, pp. 1751–1782, 07 2011.

[6] F. F. Chen, *Introduction to Plasma Physics and Controlled Fusion*, vol. 1. 233 Spring Street, New York, N.Y. 10013: Plenum Press, 2nd ed., 1984.

A Configuration File

The program accepts the parameters in the following order:

1. `grid` *int int int*: The number of cells along each dimension of the boundaries.
2. `cell_dimensions` *float float float*: The dimensions of each cell.
3. `timestep` *float*: Determines the amount of time between each variable recalculation. Reduction increases computation time and accuracy.
4. `xy_boundary` *string*: Either *free_slip* or *no_slip*. Determines the edge characteristics on the xy boundary faces. If *free_slip*, the particle has no friction with the boundary. If *no_slip*, the particle experiences friction at the boundary.
5. `yz_boundary` *string*: See `xy_boundary`. Instead affects yz boundary faces.
6. `zx_boundary` *string*: See `xy_boundary`. Instead affects zx boundary faces.
7. `initial_particles` *string*: Either *uniform* or *random*. Determines the way the particles are distributed throughout the simulation. If *uniform*, the particles will be spaced out so there is roughly a uniform distance between each particle and its neighbors. If *random*, the particles will be distributed randomly throughout the simulation.
8. `density` *float*: Determines the initial density for each cell.

9. `initial_B` *float*: Determines the initial magnetic field for each cell. This initial field is restricted to the $+\hat{z}$ direction.

10. `initial_velocity` *string*: Either *zero* or *random*. Determines the kind of initial velocity that will be given to each cell. If *zero*, all of the cell velocities will be initialized as zero. If *random*, all of the cell velocities will be initialized as a small random value.

11. *string int int int float*: First value is either `u`, `v`, or `w`. This parameter can be repeated on the end. This initializes one of the velocity components for a cell at the index of the first three integer parameters (x, y, and z respectively) to the value of the final float term. `u`, `v`, and `w` represent v_x, v_y, and v_z respectively.