Direct Energy Conversion from Flow Z-Pinch Exhaust Plume

Jeff Peachman

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

Master of Science

University of Washington

2024

Reading Committee:
Uri Shumlak
Justin Little

Program Authorized to Offer Degree:
Aeronautics and Astronautics

University of Washington

Abstract

Direct Energy Conversion from Flow Z-Pinch Exhaust Plume

Jeff Peachman

Chair of the Supervisory Committee:
Uri Shumlak
Aeronautics and Astronautics

Direct energy conversion by means of Venetian blind collectors is simulated by a custom Particle-in-Cell code named ZaP-DEC. The model predicts ion trajectories and energy conversion efficiency given user-specified collector and plasma properties. ZaP-DEC is verified against a benchmark case, then applied to study energy conversion from two different exhaust plumes generated by sheared-flow-stabilized Z pinches. The first case utilizes plasma properties measured on the ZaP-HD experiment, while the second case simulates a notional fusion power plant operating at $Q \gg 1$. A new experiment is proposed to test the model's validity, and a new vacuum chamber extension for ZaP-HD is designed to contain the experiment. The feasibility of alternative direct energy conversion approaches is also explored.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

First, I would like to acknowledge and thank my advisor Uri Shumlak for his guidance, insight, and patience. Half of this project started because I felt that I needed to build something, and the other half started because I wanted to do something really different. Together, we made both things work.

I'd also like to acknowledge my lab mates. I've spent many long days in the Flow Z-Pinch lab, and I couldn't have done it without good company. In particular, I'd like to thank Aqil Khairi for teaching me how to operate ZaP-HD, and for guidance about graduate school in general. I'd also like to thank Zhangsheng Lian, who was the first to suggest that I collect fractions of super-particles in my simulation. Also, Harry Furey-Soper deserves credit for continuous feedback on the Exhaust Chamber design, and suggesting features such as deployable optics tables.

Last, I need to acknowledge my wife, Erin. When I first expressed interest in going to graduate school, I conveyed a time commitment which was less than realistic. Despite that, she has supported me through it all, and ensures that I take care of myself. I couldn't do it without you.

DEDICATION

This work is dedicated to my wife Erin, my son Theo, and my dog Moose.

Chapter 1
INTRODUCTION

This work explores how the energy from a sheared-flow-stabilized (SFS) Z pinch may be captured by a Direct Energy Conversion (DEC) system. The Z pinch is inherently simple and compact compared to other fusion plasma configurations, while DEC systems have no moving parts, high efficiency, and low mass compared to a heat engine. In combination, these two technologies may enable space propulsion systems of unprecedented performance.

A proposed DEC system is simulated by a program created specifically for this research. Next, an experimental apparatus is designed to validate the model. This experiment does not fit in existing facilities, so an entire vacuum chamber is designed to host it, among other future experiments.

This thesis introduces the reader to the basics of fusion energy, flow stabilized Z pinches, and fusion propulsion. Next, a survey of DEC approaches is discussed. One of these approaches, called Venetian blinds, is selected for further study. Development of a custom simulation code is covered in detail, followed by the application of that code to various scenarios. Finally, the proposed experimental design is introduced.

1.1 Historical Background

One hundred years ago, we were just discovering that fusion powered the stars. In 1920, F.W. Aston[1] discovered that four hydrogen atoms were slightly heavier than a single helium. Sir Author Eddington applied Einstein's famous equation, $E = mc^2$, and understood that this mass deficit accounted for a huge quantity of energy. He argued that the sun must be powered by fusion reactions. Then, in her 1925 PhD thesis, Cecilia Payne-Gaposchkin[23]

showed that the sun is composed of hydrogen, and proposed that it is the most abundant element in the universe. By 1938, Hans Bethe[4] discovered the proton-proton chain reaction, explaining energy production in stars like our own sun. However, this reaction occurs at an incredibly low rate, and is too slow to be useful for energy on Earth. This is fortunate, as it means the sun will burn for billions of years.

Just before Bethe's discovery, Oliphant and Rutherford[20][19] produced several fusion reactions in the laboratory. Of these, the following became relevant to fusion power[7]:

$$\mathrm{D} + \mathrm{T} \longrightarrow \alpha + \mathrm{n} + 17.6\,\mathrm{MeV}, \tag{1.1}$$

$$\mathrm{D} + {}^3\mathrm{He} \longrightarrow \alpha + \mathrm{p} + 18.4\,\mathrm{MeV}, \tag{1.2}$$

$$\mathrm{D} + \mathrm{D} \longrightarrow \begin{cases} \mathrm{T} + \mathrm{p} + 4.03\,\mathrm{MeV}, \\ {}^3\mathrm{He} + \mathrm{n} + 3.27\,\mathrm{MeV}, \end{cases} \quad (50/50\text{ probability}) \tag{1.3}$$

$$\mathrm{p}^+ + {}^{11}\mathrm{B} \longrightarrow 3\,\alpha + 8.7\,\mathrm{MeV}. \tag{1.4}$$

The symbols above refer to deuterium (D = ^{2}H), tritium (T = ^{3}H), helium-4 (α = ^{4}He), helium-3 (^{3}He), protons (p), and neutrons (n).

These reactions have reactivities, $\langle \sigma v \rangle$, which are many orders of magnitude higher than stellar proton-proton fusion. The reaction rate is proportional to reactivity and the plasma density squared. These reactivities peak at temperatures which are achievable by present technology, and are high enough that useful power can be produced at reasonable plasma pressures. Therefore, it should be possible to utilize these reactions for production of energy on Earth or in space.

The D-T reaction Eq. (1.1) is the easiest to achieve, and is the fuel of choice for the majority of reactor designs in the field. Deuterium is abundant on earth, representing 1 out of every 6700 atoms of hydrogen. At our present rate of energy consumption, there is enough deuterium on Earth to last 2 billion years[11]. However, tritium is exceedingly rare due to its 12 year half-life. At present, it is produced in some nuclear fission reactors, but only in small amounts. This is a problem, because it takes an estimated 170 kg of tritium per

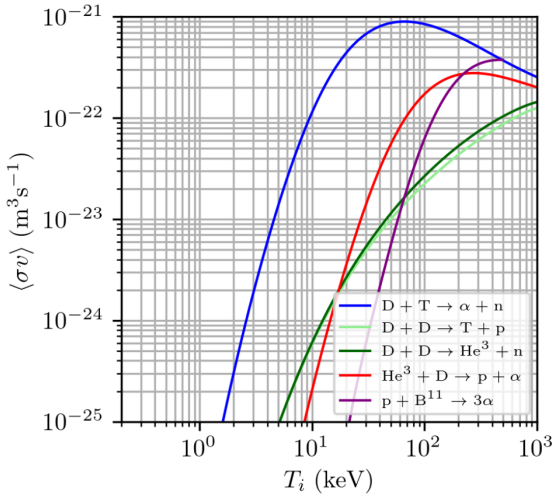


Figure 1.1: Fusion reactivities as a function of plasma temperature[35] for fuels relevant to fusion power. The reaction rate is proportional to reactivity times plasma density squared.

year[25] to power a 3 GW_{th} D-T Fusion Reactor (approx. 1 GW_e). D-T fusion reactors will only become practical if tritium is produced by the same fusion reactors that will consume it. This can be achieved by bombarding lithium with neutrons produced by the D-T reaction Eq. (1.1), which can yield additional energy depending on the isotope of lithium supplied,

$$^6\mathrm{Li} + \mathrm{n} \longrightarrow \alpha + \mathrm{T} + 4.78\,\mathrm{MeV}, \tag{1.5}$$

$$^7\mathrm{Li} + \mathrm{n} \longrightarrow \alpha + \mathrm{T} + \mathrm{n} - 2.47\,\mathrm{MeV}. \tag{1.6}$$

There is enough inexpensive ^{6}Li to last our civilization approximately 20,000 years[11].

1.2 D-T Fusion Power

The D-T reaction releases a fast neutron which carries 80% of the reaction energy. Neutrons have no charge, so the only way to capture that energy is through many scattering collisions with a dense wall called a blanket. Using a working fluid, thermal energy is transferred from the blanket to a heat engine, identical to that of a traditional thermal power-plant. The remaining energy ($f_c = 20\%$) is carried by the alpha which can remain in the plasma

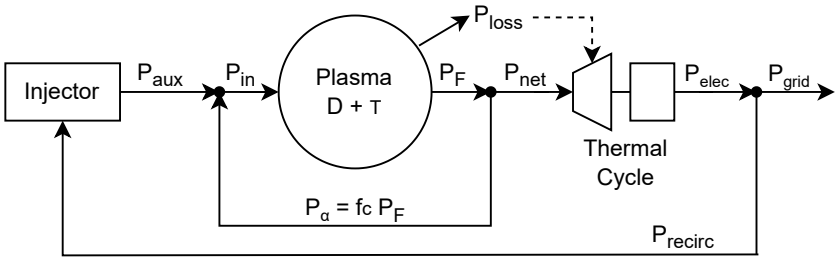


Figure 1.2: Power flow in a D-T fusion power-plant. Ignition is achieved when $P_\alpha = P_{loss}$.

if suitably confined. These hot alphas can be used to self-heat the plasma. If alpha-heating (P_α) exceeds energy losses, no energy input is required, and the reactor achieves *ignition*.

Ignition requires $P_\alpha > P_{loss}$. For a 50/50 mix of D-T fuel, alpha-heating power is,

$$P_\alpha = \frac{n^2}{4}\langle\sigma v\rangle_{DT} E_\alpha, \tag{1.7}$$

where $\langle\sigma v\rangle$ can be found in Figure 1.1 and $E_\alpha = f_c E_{DT} = 3.5$ MeV. There are numerous energy loss mechanisms due to radiation, conduction, and diffusion, but calculating these figures is beyond the scope of this work. Once P_{loss} is known, the characteristic energy confinement time is

$$\tau_E = \frac{3nT}{P_{loss}}. \tag{1.8}$$

Applying Eq. (1.7) and (1.8) to $P_\alpha > P_{loss}$ yields the Lawson criterion,

$$n\tau_E > \frac{12}{E_\alpha}\frac{T}{\langle\sigma v\rangle_{DT}}. \tag{1.9}$$

For D-T fusion, the Lawson criterion is minimized at temperatures of approximately 20 keV, requiring $n\tau > 1.32 \times 10^{20}$ sec/m^3. This has lead to a large variety of approaches. For example, inertial confinement fusion (ICF) achieves extremely high densities ($n \approx 10^{30}$ m^{-3}) for less than a nanosecond ($\tau \approx 10^{-10}$ s), while magnetic confinement fusion (MCF) achieves relatively low densities ($n \approx 10^{20}$ m^{-3}) for many seconds or sometimes minutes. This work focuses on an approach to magnetic confinement called the Z pinch.

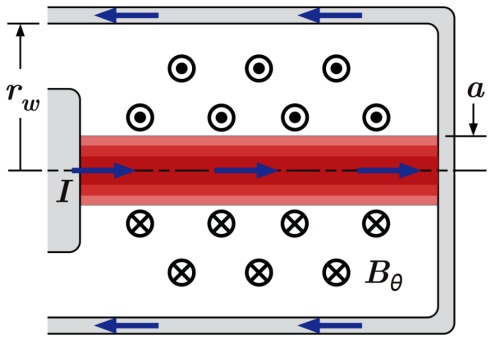


Figure 1.3: A schematic of a simple Z pinch[26]. A column of plasma exists between two electrodes. An axial electric current self-generates an azimuthal magnetic field, causing an inward radial Lorentz force ($\nabla p = -j_z B_\theta$) that compresses the plasma into a pinch.

1.3 Z-Pinch Fusion

The Z pinch is a simple plasma configuration in which a current is passed through a plasma to compress it, shown in Figure 1.3. As the plasma is compressed, its temperature increases. It continues to constrict until its thermal pressure balances the Lorentz force. The temperature at pressure equilibrium is described by the Bennett relation, a function of only the pinch current (I), the linear density (N_i), and the ion charge number (Z)[26].

$$\mu_0 I^2 = 8\pi(1+Z)N_i T. \tag{1.10}$$

This yields our first scaling relationship, $T \propto I^2$. Assuming adiabatic compression with an index $\gamma = 5/3$, scaling relationships for density (n) and pinch radius (a) can be derived, $n \propto I^3$, $a \propto I^{-3/2}$. Since the fusion power per unit volume is $P_F \propto n^2\langle\sigma v\rangle$, we can write $P_F \propto I^3\langle\sigma v\rangle$. The reactivity scales by approximately T^4 from 1-10 keV, meaning $P_F \propto I^{11}$. This is an extremely strong scaling law, making the Z pinch attractive. Previous work has estimated that scientific break-even ($Q_{sci} = P_F/P_{loss} > 1$) can be achieved at a pinch current of 0.6 to 0.7 MA[10].

This configuration has inherent system-level advantages. First, it has unity average beta,

defined as volume averaged thermal pressure over the magnetic pressure at the wall,

$$\beta = \frac{\langle p \rangle}{B_\theta^2(r_w)/(2\mu_0)}, \tag{1.11}$$
$$\text{where } \langle p \rangle = \frac{1}{\pi r_w} \int_0^{r_w} 2\pi r p dr.$$

This means that the magnetic field energy is being utilized for confinement as efficiently as possible. There are no external magnets, eliminating a major subsystem present in every other magnetic confinement scheme. The lack of axial magnetic fields means that axial heat conduction is extremely low, reducing P_{loss}. Since heating is achieved by compression, expensive and complex plasma heating subsystems are eliminated.

However, the Z pinch has a serious draw-back: extremely poor stability. In particular, Z pinches are highly susceptible to the sausage and kink instabilities shown in Figure 1.4. These instabilities grow exponentially on nanosecond timescales until current is disrupted or the pinch reaches the wall. Assuming that a Z Pinch can achieve a temperature of 5 keV and density of $n \approx 10^{25}$ m^{-3}[26], a confinement time of at least 44 μs is required to satisfy the Lawson criterion per Eq. (1.9). Z pinches must be stabilized before they can be useful for fusion power.

Early attempts to stabilize the Z pinch added axial magnetic fields, creating the screw pinch. However, these fields result in strong axial heat conduction, resulting in energy loss at the ends of the device. So, screw pinches were bent into a torus to connect the ends together, leading to modern tokamaks and stellarators which are the mainstream plasma configurations in the field. However, these stabilized devices lack many of the advantages of the Z pinch. Strong axial (or toroidal) fields are required for stability, and the pinch must compress these fields in addition to the plasma, resulting in low beta. The toroidal devices are extremely complex and have low power densities, so they are expensive to produce and operate.

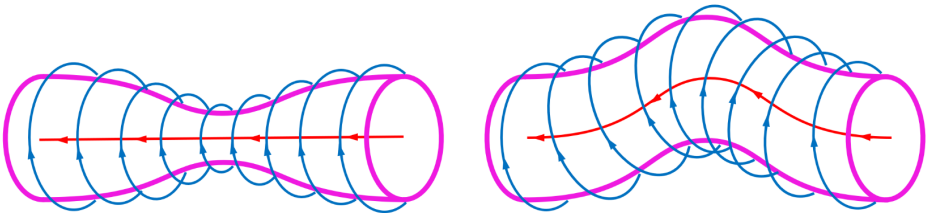


Figure 1.4: The sausage (left) and kink (right) instabilities present in a pure Z pinch. The magnetic field is represented in blue and the pinch current is red. The sausage instability occurs when a portion of the the pinch has radially contracted, resulting in a stronger local magnetic field that causes further contraction. The kink instability occurs due to a lateral displacement which concentrates the magnetic field on one side of the pinch and causes the displacement to grow.[14]

1.3.1 Sheared-Flow Stabilized Z Pinch

A Sheared-Flow Stabilized (SFS) Z pinch suppresses the sausage and kink instabilities without using applied magnetic fields. This preserves the advantages of a Z pinch without inheriting the disadvantages of a screw pinch or its toroidal decedents. This is achieved by establishing a radial shear of the axial flow velocity, dv_z/d_r.

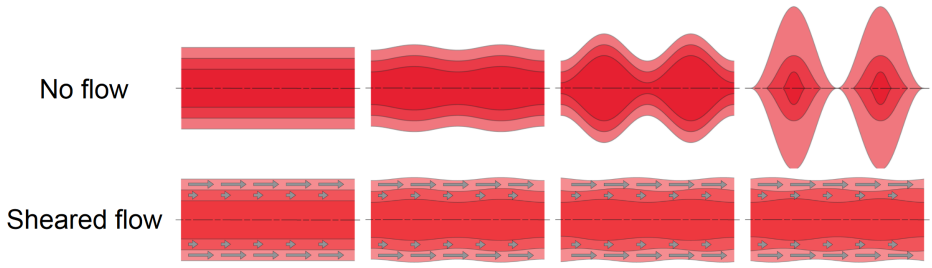


Figure 1.5: Illustration of the sausage instability (above) and it's stabilization using sheared axial flow (below). Flowing plasma fills in the radial displacement, limiting the amplitude of the instability.[28]

To stabilize the kink instability, radial shear must exceed a threshold[27]:

$$\frac{dv_z}{dr} \geq 0.1kV_a, \tag{1.12}$$

$$V_a = \frac{B_0}{\sqrt{\mu_0 \rho_0}}, \tag{1.13}$$

where k is the wavenumber, V_a is the Alfven speed, B_0 is the maximum magnetic field, and ρ_0 is the density.

1.3.2 The ZaP-HD Experiment

The ZaP-HD experiment produces SFS Z pinches at the University of Washington[29], seen in Figure 1.6. It operates by first injecting neutral gas into the chamber with gas puff valves. A capacitor bank discharges between the inner and middle electrode, which ionizes the gas creating a plasma. The discharge results in an azimuthal magnetic field around the inner electrode, and a radial current density $\vec{j}$ traveling from the middle electrode to the inner electrode. The current and magnetic field interact, and the $\vec{j} \times \vec{B}$ force pushes plasma axially down the length of the acceleration region. When the plasma reaches the end of the acceleration region, it attaches to the outer electrode, which is energized by a capacitor bank. While the outer edge of the plasma continues to travel down the length of the assembly region, some plasma is forced to remain attached to the nosecone on the inner electrode. This causes the plasma assemble into a Z pinch with a sheared-flow profile. ZaP-HD, and other devices like it, have improved the confinement time of Z pinches from nanoseconds to over 10 microseconds[10].

1.4 Fusion Propulsion

The SFS Z pinch is particularly attractive in the context of fusion propulsion. A propulsion configuration allows the plasma to be exhausted as efficiently as possible, shown conceptually in Figure 1.7. Compared to other fusion reactor designs, it is compact with high power density. It has few subsystems, reducing weight and complexity. It is an inherently linear device designed to accelerate plasma to high velocities. Operating with normal hydrogen, at

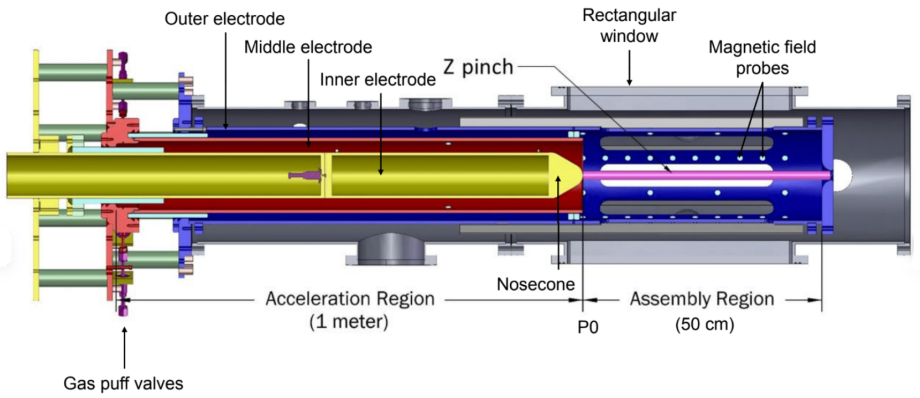


Figure 1.6: The ZaP-HD Experiment which produces Shear-Flow-Stabilized Z Pinches at the University of Washington. Plasma is produced in the acceleration region and accelerated by the Lorentz force. The inner electrode terminates at the end of the acceleration region, forcing the plasma to assemble into a pinch in the assembly region.

energies far too low for fusion reactions, ZaP-HD already produces flow velocities exceeding 100 km/sec[30], equivalent to a specific impulse of 10,000 sec. This is similar performance to modern gridded ion thrusters, so one may ask why a SFS Z pinch thruster is attractive compared to a traditional propulsion system.

Fundamentally, all propulsion systems are limited by conservation of energy and momentum. Thrust is the rate of change of momentum, $F = \dot{m}v_e$, where $\dot{m}$ is the propellant mass flow rate and v_e is the exhaust velocity. Accelerating propellants to v_e requires energy which can come from combustion of chemical propellants, from an electrical power supply, or directly from nuclear reactions. Jet power, P_{jet}, is the rate of energy consumption required to accelerate propellant with mass flow-rate $\dot{m}$ to v_e with a perfectly collimated exhaust. Thrust efficiency, η, accounts for all energy losses.

$$P_{in} = \eta P_{jet} = \eta \frac{1}{2}\dot{m}v_e^2 = \eta \frac{1}{2} T v_e \tag{1.14}$$

$$\frac{T}{P_{in}} = \frac{2\eta}{v_e} \tag{1.15}$$

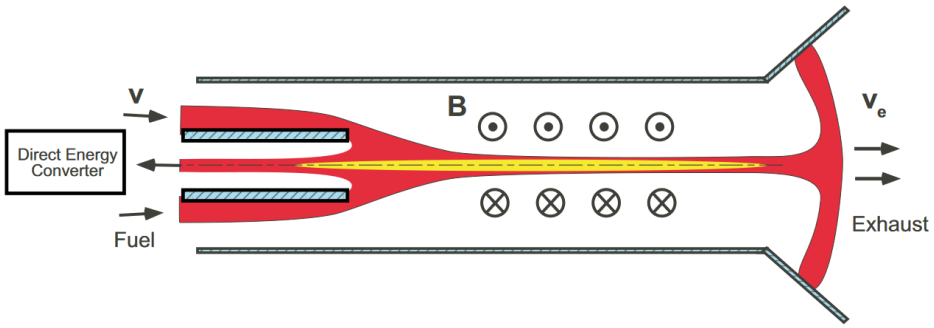


Figure 1.7: An early concept for a SFS Z-pinch Thruster featuring a diverging outer electrode, allowing plasma to cool before making contact. A notional DEC system is depicted upstream, but such systems may be designed for other locations.

In chemical propulsion, thermal power is produced in proportion to the fuel consumption rate, so P_{in} and $\dot{m}$ are inherently linked. Therefore, the maximum exhaust velocity is largely dependent on the choice of fuel. Electric propulsion (EP) systems allow these factors to be decoupled because electric power is provided by an external supply. However, if power is fixed, any increase in exhaust velocity requires a reduction in thrust due to conservation of energy. As a consequence, EP spacecraft are currently limited to low-thrust maneuvers because solar power has low power density.

Gridded ion thrusters already have thrust efficiencies of approximately 80%, and Hall thrusters achieve 60%[13]. While there is room for improvement in reliability, cost, weight, or choice of propellants, the performance of EP spacecraft will not improve dramatically due to this fundamental limitation. The only way to achieve high thrust and exhaust velocity simultaneously is to invent power sources with high power density, and to utilize that power efficiently.

Nuclear fuels (both fission and fusion) have much higher energy density than chemical propellants by a factor of 10^5 to 10^6. However, fission reactions release their energy in the form of energetic neutrons. To capture their kinetic energy, a large number of scattering collisions must occur. Neutrons can collide with propellants directly, which is done in nuclear

thermal rockets. However, maximum temperature of the propellant is limited by the materials used to make the rocket, limiting the exhaust velocities. To avoid this limitation, the propellant must be in the plasma state, and the walls may be protected by electromagnetic fields. At practical pressures, a plasma is not dense enough to capture neutrons directly. So, to utilize a fission energy source to produce high exhaust velocity, neutrons are used to heat a working fluid, and a thermal cycle is used to produce electricity. The electricity is used to power conventional EP thrusters, which is called Nuclear Electric Propulsion (NEP).

NEP certainly has promise, and has yet to fly in space. However, a brayton cycle generator operating in space has an efficiency of about 30%[8]. This heat must be rejected by large heat radiators which requires structure to support them. The power produced must be converted to a suitable voltage for EP thrusters, and every energy-conversion step results in a loss of efficiency.

By contrast, fusion reactions can heat the plasma directly. The D-^{3}He reaction Eq. (1.2) is aneutronic: it only produces high-energy charged particles which may be directed by a magnetic nozzle to produce thrust. In practice, D-D side reactions Eq. (1.3) will occur and produce neutrons, but this represents a small fraction of particles. As previously mentioned, deuterium is abundant, but ^{3}He is not. It is currently produced by the decay of tritium, making it extremely rare. There have been proposals to breed ^{3}He using the D-D reaction Eq. (1.3) which produces ^{3}He 50% of the time. There are also extraterrestrial sources of ^{3}He on the surface of the moon[9] and in the atmospheres of gas giants[22].

Fusion reactors do require significant power input, but that can be a small fraction of the total energy produced. Referring to Figure 1.8, if the recirculating power P_{rec} is small relative to the net power output P_{net}, most of the power produced is available as jet power. An energy conversion system is still required, and it can still be a heat engine, but it can be comparatively smaller.

Aneutronic fusion reactions have another important advantage, which is the central topic of this thesis: the energy can be captured directly, without use of a heat engine. This is called Direct Energy Conversion (DEC), and it may be achieved by means of electromagnetic

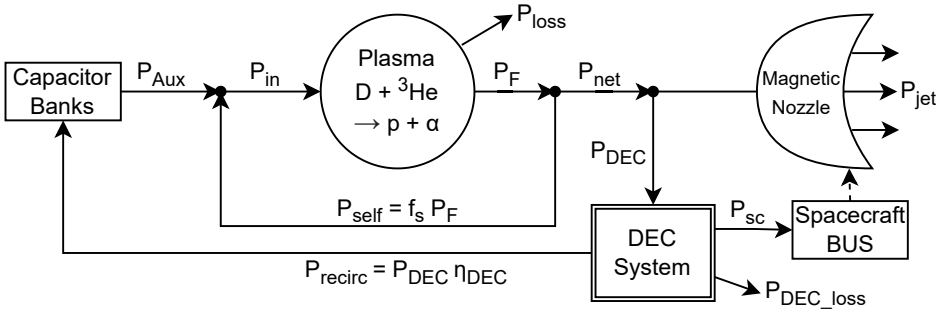


Figure 1.8: Fusion rocket power flow. If P_{recirc} is much smaller than P_{net}, the majority of P_{net} may be utilized directly for thrust. Recirculating power may be captured by a DEC system which is less massive and more efficient than a heat engine.

fields. Approaches to DEC are discussed in Chapter 2. For now, the system-level advantages are considered.

Due to thermodynamics, heat engines are constrained to operate at low efficiencies, typically about 30% or less. In space, waste heat must be rejected by heavy radiators. Heat engines also require moving parts: compressors, turbines, shafts, and pumps, with associated working fluids and plumbing. In contrast, DEC systems are electronic components with no moving parts or fluids, and may operate at higher efficiencies, saving radiator mass. These features are attractive for a spacecraft.

The advantage of higher operating efficiency η_{DEC} is nonlinear. In Figure 1.8, we observe that $P_{recirc} = P_{DEC}\eta_{DEC}$. As efficiency increases, P_{DEC} can decrease to maintain the same recirculating power. Also, P_{jet} can increase, yielding more thrust. This is fundamentally different than how a power-plant operates. As efficiency increases, the power input to the conversion system decreases, so there are two effects that reduce the waste heat produced. For example, one can imagine a reactor with an "effective gain" of 10 where $Q_{eff} = P_{net}/P_{recirc}$ (assuming $P_{recirc} = P_{aux}$). Next, one can see how P_{loss}/P_{jet} changes as the conversion efficiency is varied. Since radiator area scales linearly with P_{loss}, this metric can be used

to compare the radiator sizes of different spacecraft with fixed jet power. If efficiency is 30%, $P_{loss}/P_{jet} = 0.35$. If efficiency is 60%, $P_{loss}/P_{jet} = .08$. Doubling the efficiency reduces radiator area by a factor of 4.4. This simple analysis makes many assumptions, including a nozzle efficiency of 1 and no frozen flow losses. The same analysis may be applied to a fission NEP system with a thrust efficiency of $\eta = 1$. If the efficiency of a heat engine on an NEP spacecraft is 30%, $P_{loss}/P_{jet} = 0.7/0.3 = 2.3$. This suggests that an NEP spacecraft will require radiators which are 29 times larger than a fusion-propelled spacecraft with $Q_{eff} = 10$ and $\eta_{DEC} = 60\%$.

1.5 Terrestrial Applications of DEC

DEC systems are also attractive for terrestrial applications. A D-T fusion reactor releases 20% of its fusion power as charged particles, and those which are not utilized to self-heat the plasma may be captured directly for energy. One such case is simulated in this work (see Section 4.3). An obvious possibility is to build a conventional thermal power-plant which utilizes both a primary heat engine and an auxiliary DEC system to increase the efficiency of the overall system.

Due to the complexity of a heat engine, it may be desirable to omit it entirely. For a D-T reactor, a market may exist for a reactor that supplies thermal energy for industrial process heat. In this case, a DEC system can supply recirculating power. Possible applications may include supporting chemical processes such as ammonia production, thermal cracking and refining in the petrochemical industry, or tempering and annealing processes in the metal processing industry.

There are also terrestrial applications for D-^{3}He fusion reactors, and these are good candidates for DEC systems. Since aneutronic reactors can be lightweight, they are attractive for mobile applications. It may be possible to deploy such reactors on a temporary basis where they are needed, such as for disaster relief, remote mining, or forward military bases. In such cases, the rarity and high cost of ^{3}He may be justified. Other possibilities include powering vehicles such as ocean vessels and aircraft.

Chapter 2
DIRECT ENERGY CONVERSION

Direct Energy Conversion refers to a set of technologies which convert the kinetic energy of charged particles into electricity without the use of a heat engine. There are two broad categories: Direct Collection and Electromagnetic Coupling[16][24].

2.1 Direct Collection

This method decelerates charged particles and ultimately collects them on high-voltage electrodes called collectors. When an ion strikes a collector, it almost always pulls an electron out of the material. A large number of such collisions drives a current at the grid voltage. Energy collected per collision is $E_c = qV$, and power collected is $P = IV = \frac{dq}{dt}V$. However, since protons and electrons have opposite charge, collection of a quasi-neutral plasma drives a net current of zero. Therefore, *ions and electrons must be separated*. Charge separation can be achieved using electrostatic or magnetic fields[16]:

1. Grid wires at negative potential can deflect electrons with relatively low energy. This condition ($T_e \ll T_i$) can be achieved with a magnetic expander (see Section 2.3.1).
2. A magnetic field can be applied whose flux density is large enough to magnetize electrons but not ions. Electrons are 3,672 times lighter than deuterons, and therefore have tighter gyro-radii about magnetic field lines. Electrons are trapped to the field lines, which guides them out of the plasma to be collected elsewhere. A common proposal is to utilize a magnetic expander whose field lines are sharply curved at the expander exit. The electrons turn with the field lines, the massive ions do not.
3. Magnetic fields can be used to produce a transverse drift velocity that causes ions and electrons to follow different trajectories, as in an $E \times B$ converter.

One obviously wants to use collectors with large voltage to collect as much energy as possible. The high positive voltage produces an electrostatic field which repels the ions, slowing them down, thus converting kinetic energy into electrostatic potential energy. If the voltage is too high, the particles reflect before ever being collected. If the voltage is too low, particles can impact the collector at high speed, and any remaining kinetic energy is converted into heat. So, a perfectly efficient collector should have voltage $V = W_{eV}/Z$, where W_{eV} is the particle's kinetic energy in electron volts and Z is the charge number. In a practical device, voltages can reach hundreds of kV, meaning direct collection cannot be used to recover fusion products with energies of several MV.

Plasmas typically don't exist as mono-energetic beams of charged particles. If the temperature exceeds zero, there must be a statistical distribution of particle velocities. Per the Boltzmann H-Theorem, a gas in local thermodynamic equilibrium has a Maxwellian velocity distribution. In 1-D, this is defined as,

$$f(v) = n\left(\frac{1}{2\pi v_t^2}\right)^{\frac{1}{2}} \exp\left(\frac{-(v-u)^2}{2v_t^2}\right), \quad v_t = \sqrt{\frac{kT}{m}}, \tag{2.1}$$

where $f(v)$ is the number of particles with velocity v, n is the plasma density, u is the mean fluid velocity, and v_t is the characteristic thermal velocity. A Maxwellian is a normal distribution whose standard deviation equals v_t.

Since the particles have a range of energies, an efficient collection system must have many collectors with a range of electric potentials. *Particles must be partitioned into several energy groups*, and each group must be directed to collectors of suitable voltages. This partitioning can be achieved in many different ways, a short survey of which is provided below.

2.1.1 Periodic Electrostatic Focusing

PEF collection was the first direct energy conversion method proposed for fusion energy in the early 1970s.[17] Collector plates are arranged along the length of a channel with gradually increasing potential (see Figure 2.1). Electrostatic lenses guide a beam of particles down the channel. As particles climb the potential well, they slow down until they become

over-focused. At that point, they are deflected into collectors. Early calculations suggested efficiencies as high 90% may be possible[18]. However, later work revised these estimates to between 40% and 70% depending on the input energy of the ion beam[3]. This approach is useful for energies between 400 keV and 900 keV, but is limited otherwise.

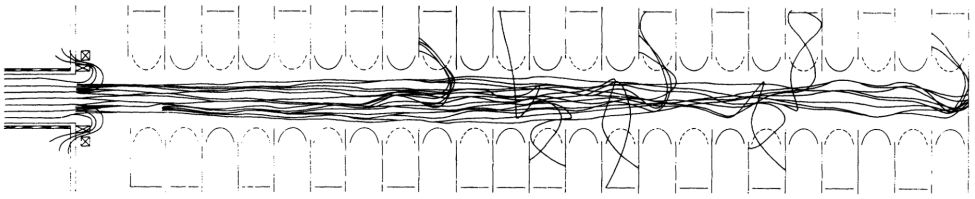


Figure 2.1: Simulated Ion Trajectories in a Periodic Electrostatic Focusing Collector [3]

2.1.2 Venetian Blind (VB) Collectors

VB collectors are arrays of ribbon-shaped electrodes[17]. The ribbons are thin and slanted at an angle, much like Venetian blinds (see Figure 2.2). The collectors exhibit angular-dependent opacity: if an ion approaches a ribbon edge on, the collector looks transparent. If an ion approaches perpendicular to the ribbon, the collector looks opaque. To exploit this angular dependence, collectors may be placed so that the electric field vector is at a small angle (α) relative to an incident beam of particles. This angle causes particles to follow parabolic trajectories: after deflection, particles do not re-trace their original path. So, the ribbons in each collector can be oriented such that they are edge-on to the incident beam, but they look opaque to deflected particles. The majority of particles should impact on the back of each collector.

From a preliminary engineering study of a Venetian Blind conversion system[2], efficiencies are estimated at 65% for four collectors operating at 150 keV injection energy. VB collectors are practical from 10-200 keV.

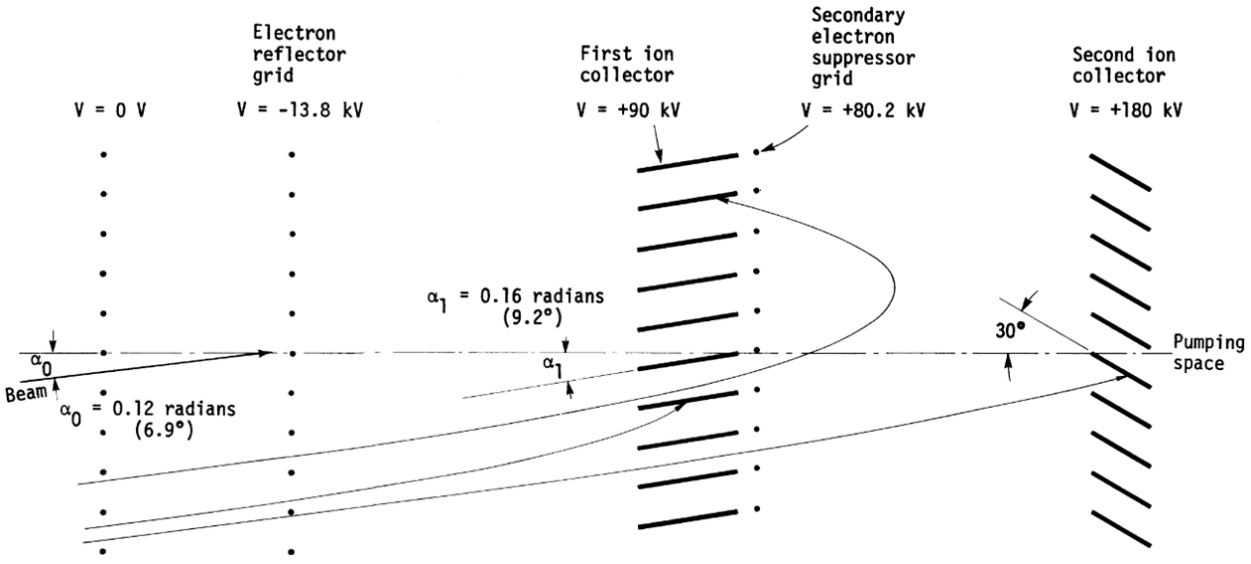


Figure 2.2: VB Collector Array. Wire grids at the left reflect the electrons. The Venetian blind appears transparent to the incoming ions but opaque to the ions reflected by the second collector. Modified from Barr, et al. [2]

2.1.3 Space Charge Expansion Converters

These devices work by removing the electrons from a beam of plasma, then allowing the ion beam to expand radially due to mutual repulsion of ions. This effect is called space charge, discussed in Section 2.3.2. Faster ions have less time to expand, so their trajectories are be deflected less, allowing them to be collected by appropriately charged grids. The space charge effect depends on plasma density, so these converters must operate over a range of beam densities. If the density is too low, space charge effects are small and particles are not properly separated. If density is too high, space charge effects modify the the potential field significantly and ions are directed to sub-optimal grids. Simulations have suggested peak efficiencies of 70%[12] at beam areal densities of 10^7 particles/m^2, with a sharp decrease in efficiency at higher densities.

2.1.4 $E \times B$ Converter

If a plasma is following a guiding magnetic field $\vec{B}_{\parallel}$, a transverse electric field $\vec{E}_{\perp}$ can be applied. This causes all charged particles to drift in a direction transverse to both fields. In Figure 2.3, particles drift upwards.

$$v_d = \frac{\vec{E}_{\perp} \times \vec{B}_{\parallel}}{B^2} \tag{2.2}$$

However, the longitudinal velocity (to the right) remains unchanged. If $T_e \approx T_i$, low mass electrons have high longitudinal velocity, so their overall trajectory is changed little. In contrast, heavy ions move slower, so the drift is a larger component of their total velocity, and their trajectory is deflected upwards. Thus, this collector has an advantage: charge separation occurs at the point of collection.

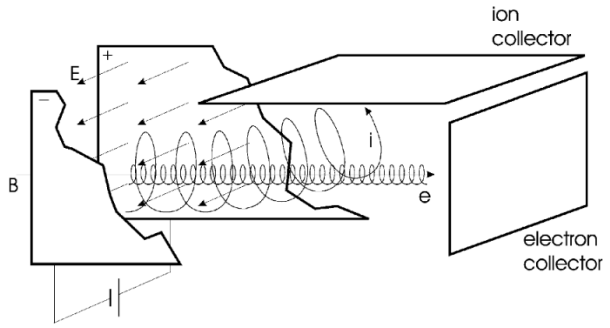


Figure 2.3: An E × B converter[15]. All particles experience transverse drift at the $E \times B$ drift velocity. The trajectory of faster particles is altered less than slower particles. This sorts ions by kinetic energy and removes electrons before collection.

This method also has complications. Particles should be hitting collection electrodes, not the electrodes which impose the transverse electric field. To do this, ions and electrons must be magnetized with a strong magnetic field, meaning their gyroradius should be be much smaller than the scale length, $r_g \ll L$.

$$r_g = \frac{mv_{\perp}}{|q|B} \tag{2.3}$$

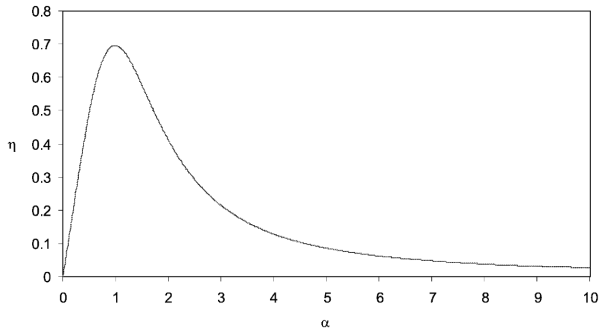


Figure 2.4: Ideal efficiency for an $\vec{E} \times \vec{B}$ converter as a function of α with $l/d \approx 5$[15].

The ideal efficiency of an $E \times B$ converter depends on the ratio of the drift to thermal speed, $\alpha = \vec{v}_d/\vec{v}_{th}$, shown in Figure 2.4. It peaks just above $\alpha = 1$. This yields competing requirements: a stronger magnetic field decreases gyroradius, but it also decreases v_d which is separating the particles.

As an example, consider an aperture of 1 meter. To ensure $r_g \ll L$, choose $r_g = 0.1$. For a deuteron with a transverse energy of 1 keV, Eq. (2.3) yields B = .06 Tesla. To achieve $\alpha = 1$, an electric field of 20 kV/meter is required, which seems possible. However, suppose one can only achieve 10 kV, so electrodes are placed 0.5 meters apart. That requires a smaller gyroradius, so the magnetic field strength must increase to 0.13 Tesla. That decreases v_d, and now an electric field of 40 kV/meter is required. It is clear that high energy particles require an expander to reduce $v_\perp$ significantly before extracting energy from fusion reactors operating at temperatures $>$ 10 keV.

Researchers have also noted[15] that particle collisions have a large effect on efficiency, limiting the plasma density. As ions are collected, space charge increases, complicating the design. Some researchers have suggested that a large number of collectors, required to approach ideal efficiency, is not achievable[16].

2.1.5 Cyclotron-Resonance Collectors

In this design, electrons are first separated from ions. The ions are confined to a guiding magnetic field, and gyrate at the ion cyclotron frequency $\omega_i = qB/m_i$. A constant electric field, parallel to the guiding magnetic field, slows the particles electrostatically. Superimposed over this system are transverse fields which are spatially periodic, like a sine wave. These fields may be electric, magnetic, or both, and they can cause forces that oscillate at the cyclotron frequency. This occurs when ions are decelerated to a velocity of $v_{\|} = \omega_i \lambda / 2\pi$, where λ is the wavelength of the periodic fields. The result is cyclotron resonance: the ion gyroradius increases until ions escape confinement and reach collectors.

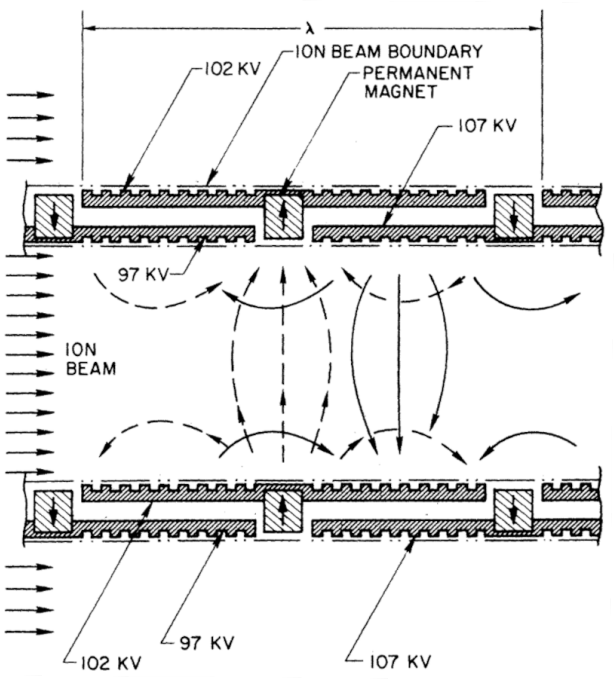


Figure 2.5: Cyclotron-resonance collector. Spatially periodic fields (solid lines: electric; dashed: magnetic) are superimposed over an axial electric field (to the left, not shown) and an axial magnetic guide field (not shown). Moving ions experience periodic forces in the transverse direction. At a particular velocity, cyclotron resonance causes the gyroradius to increase until collection occurs.

2.2 Electromagnetic Coupling

In electromagnetic coupling, a current is generated in a wire by means of magnetic induction. Thus, many of these methods do not require electrodes to be in contact with the plasma. An additional advantage is that charge separation is not necessary, so these systems are not limited by space charge effects.

2.2.1 Traveling Wave Direct Energy Conversion (TWDEC)

TWDEC operates like a linear accelerator in reverse.[32]. First, a beam of charged particles is directed towards a region called a modulator. A series of antennas excites an electromagnetic wave within the beam, and the phase velocity of the wave matches the velocity of the particles. This causes particles to concentrate into bunches between the peaks and valleys of the wave. Next, the beam enters the converter, where another series of antennas gradually reduces the phase velocity to slow the particles. The particles transfer energy to the waves, which is then transferred to the antennas by induction. This is particularly attractive for D-He fusion reactors. Expanding upon Eq. (1.2), this reaction is

$$\mathrm{D} + {}^{3}\mathrm{He} \longrightarrow {}^{4}\mathrm{He}\,(3.6\,\mathrm{MeV}) + p\,(14.7\,\mathrm{MeV}). \tag{2.4}$$

The energy of the 14.7 MeV proton is particularly difficult to recover using direct electrostatic collection because it requires an electrode charged to 14.7 MV. The TWDC avoids such issues. In Figure 2.6, a cusp field is used to separate electrons and ions. A TWDC converter is used to capture 14.7 MeV protons, while direct collectors are used for lower energy particles.

Theoretical efficiencies of 80% are predicted for the TWDEC, and an overall efficiency of 90% is predicted if traditional electrostatic collection occurs downstream from the TWDEC[34]. However, experimental work has demonstrated an efficiency of only 50% so far. An important limitation of TWDEC devices is that they require a limited spread of ion energies. Thus, they are well suited for recovering fusion reaction products which have not yet thermalized.

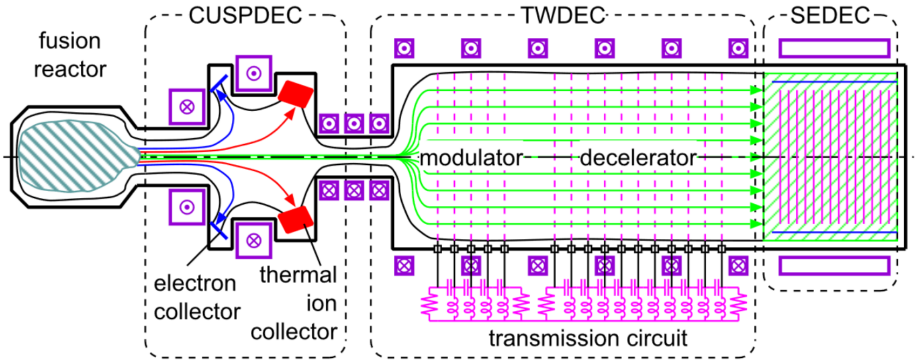


Figure 2.6: A DEC system which includes TWDEC and direct collection[33]

Due to this limitation, some protons are accelerated by a TWDEC. This may not be an issue if the application is propulsion.

2.2.2 Magnetic Compression and Expansion

This method employs a thermodynamic cycle using the plasma itself as working fluid[21][24]. In the 1970s, this was studied in the context of a theta pinch reactor, shown in Figure 2.7.

A system using the "Type-S" Cycle, similar to the Brayton cycle, follows these steps:

1. Slow-adiabatic compression from r_1 to r_2, reaching max magnetic field strength.
2. Fusion burn at constant pressure, increasing temperature. Radius increases to r_3.
3. Slow-adiabatic expansion back to r_1 and low magnetic field strength.
4. Radiative cooling of remaining thermal energy that cannot be recovered, returning the plasma to its original state.

During Step 1, magnetic pressure compresses the plasma, performing PdV compression work and requiring energy input. The plasma becomes hot, and magnetic flux becomes frozen in to the plasma. In steps 2 and 3, the plasma expands, changing the magnetic flux in the region between the plasma and the coil. PdV expansion work is performed on the

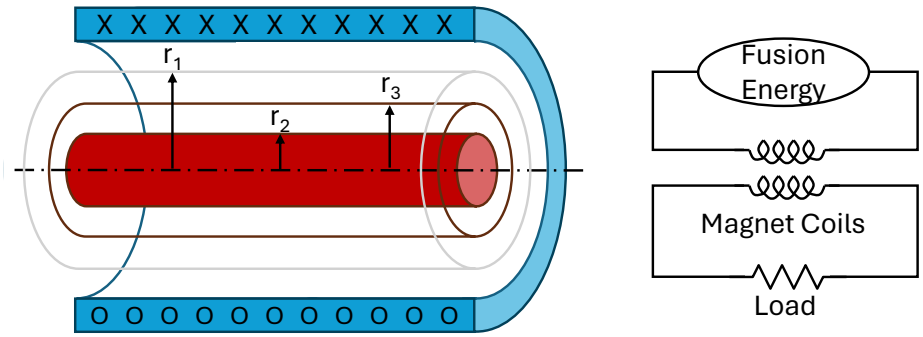


Figure 2.7: Magnetic coupling via compression-expansion of a cylindrical plasma in a θ pinch. Plasma is compressed to fusion conditions, then fusion reactions release thermal energy. This causes the plasma to expand, performing PdV work on the magnetic field. The energy is recovered by induction. Reproduced from Miley[16].

magnetic field, which appears as electrical energy in the magnetic field coils due to magnetic induction.

A simplified analysis of step 2 assumes a constant confining field $B_v = \mu_0 IN/L$, produced by a solenoid of length L, with N turns of current I. The power induced is

$$P = IV = -nI\frac{d\Psi}{dt} - I^2 R_c \tag{2.5}$$

where $\frac{d\Psi}{dt}$ is the rate of change in magnetic flux caused by expanding plasma. The second term represents ohmic losses in the coil, which is neglected from this point on. The total change in energy is

$$E = -n\int_2^3 I d\Psi = -nI(\Psi_3 - \Psi_2), \tag{2.6}$$

$$= \frac{B_v L}{\mu_0}(\Psi_3 - \Psi_2). \tag{2.7}$$

Since flux outside the plasma is $\Psi = B_v(R^2 - r^2)$, and plasma volume is $V = \pi r^2 L$,

$$E = \frac{B_v^2}{\mu_0}(V_3 - V_2). \tag{2.8}$$

This method has some obvious advantages. There is no charge separation, so there are no space charge effects. There is no known theoretical limit on plasma density. Energy which cannot be recovered from the plasma remains as thermal energy in the plasma, which is beneficial for a propulsion system. A magnetic nozzle is essentially a divergent theta pinch, so the compression/expansion DEC approach may synergize well with propulsion systems employing a magnetic nozzle. Conceptually, one can imagine a long cylinder of theta pinch coils where energy is recovered, with a magnetic nozzle at the end, and a single power supply for the whole system.

The efficiency of such a system is highly dependent on the compression ratio. For example, for the Type-S cycle, a compression ratio of 2 results in an ideal cycle efficiency of 38%, while a compression ratio of 10 yields 61%, and a ratio of 100 yields 63.7%[16]. These are ideal efficiencies, neglecting many loss terms which must be considered. For theta pinch reactor designs in the 1970s, achieving a compression ratio of 5 was considered difficult. However, if this method can be adapted to a SFS Z pinch, high compression ratios may be achievable. A SFS Z pinch operating at Q=1 can have a pinch diameter of approx 0.7 mm, and can expand to an outer radius of $>$ 100 mm, yielding a compression ratio of $>$ 20,000[26].

However, using this method to recover energy from a SFS Z pinch is difficult. If an axial field exists before the pinch is assembled, a screw pinch is created. A screw pinch has high axial heat conduction, which greatly increases the energy loss rate, making break-even impossible.

A speculative idea is that it may be possible to surround the Z pinch with axial magnetic fields after the pinch is assembled. If the pinch is sufficiently hot, the axial fields do not have time to diffuse through the pinch, which may prevent the axial heat conduction. When the axial current is disrupted, the plasma expands, doing PdV work on the axial magnetic field around it. This concept has not been studied rigorously, but may be an opportunity for future research.

2.3 Method Selection (Venetian Blinds)

The author is motivated to study a collection method which is within his capabilities to simulate while also being relevant to a notional fusion propulsion system. The only published method which satisfies both criteria is direct collection by means of Venetian Blinds.

From coursework, the author has experience designing a magnetostatic PIC code which simulates 1 spatial dimension and 2 velocity dimensions (1D2V). However, if the collection methods listed in this chapter are assessed, this is insufficient to simulate particle trajectories and collection efficiency. Periodic Electrostatic Focusing, Space Charge Expansion, $E \times B$, and Cyclotron-Resonance converters are all inherently 2D. Particle trajectories in a Traveling Wave DEC system are 1D, but energy recovery depends on a time-varying electromagnetic field; it is not magnetostatic. Magnetic compression and expansion is more suitable to analysis by MHD than by PIC.

A PIC code which simulates two velocity components can calculate the angle of incidence with a Venetian blind. While particles in these systems do have 2D trajectories, the applied electric fields only exist in one dimension. This means that a 1D PIC code can compute accelerations in the axial (z) direction while perpendicular acceleration is always zero. It is therefore trivial to evolve the particle's position in the perpendicular dimension. Such a code can be considered 'pseudo-2D' because the second spatial dimension exists for visualization purposes only.

As discussed in Section 1.4, a D-^{3}He propulsion system's plasma exhaust contains fast protons and alphas in the MeV range, but MV voltages are difficult to achieve with a practical collection system. However, the plasma exhaust also contains thermalized ions with energies of 60-200 keV[35][32]. According to the published literature, Venetian Blinds can be practical at energies from 10-200 keV, so they can be useful for this application.

However, VB collectors present several problems for an experimental plasma physicist. The first problem is obvious: they are practical at energies as low as 10 keV, but that energy still difficult to achieve. For reference, particles in the exhaust of the ZaP-HD device have

mean energies of < 60 eV (See Section 4.2.1). The 10 keV limit is caused by space charge effects. Another practical problem is the need for a magnetic expander to direct particles axially and to reduce the energy of electrons. Both problems are discussed below.

2.3.1 Magnetic Expander

Previous work on Venetian Blind collection system included a magnetic expander, as shown in Figure 2.8. The expander operates like a magnetic mirror in reverse. There are two conserved quantities: particle kinetic energy (W_{tot}) and the magnetic moment (μ),

$$\mu = \frac{W_\perp}{B}, \tag{2.9}$$

$$W_{tot} = W_z + W_\perp, \tag{2.10}$$

$$\frac{1}{2}mv_z^2 = W_{tot} - \mu B. \tag{2.11}$$

As B decreases, v_z must increase, causing an energy exchange from $W_\perp$ to W_z, lowering the plasma temperature in the the radial direction while increasing the mean plasma velocity in the axial direction. This is important because Venetian Blind collectors are most efficient when particle velocities vectors are oriented axially. The electrodes create a retarding electric field in a single direction, so only W_z may be recovered while $W_\perp$ is lost. Additionally, VB collectors depend on predictable trajectories. A large spread of velocities in perpendicular directions can result in premature collection.

Electrons are also be accelerated due to this effect. Due to their small mass, they accelerate to high speeds. However, this results in a violation of quasi-neutrality along the length of the expander, creating an ambipolar electric field. The electrons are impeded by this field, decelerating them, while the ions are accelerated by it. This results in an exchange of energy from electrons to ions. This is fortunate, because it means the electrons may be easily deflected by small electric fields, and only the ions need to be collected.

In the context of a fusion propulsion system, the expander can be labeled a magnetic nozzle. One can imagine a spacecraft with a large magnetic nozzle and a Venetian Blind

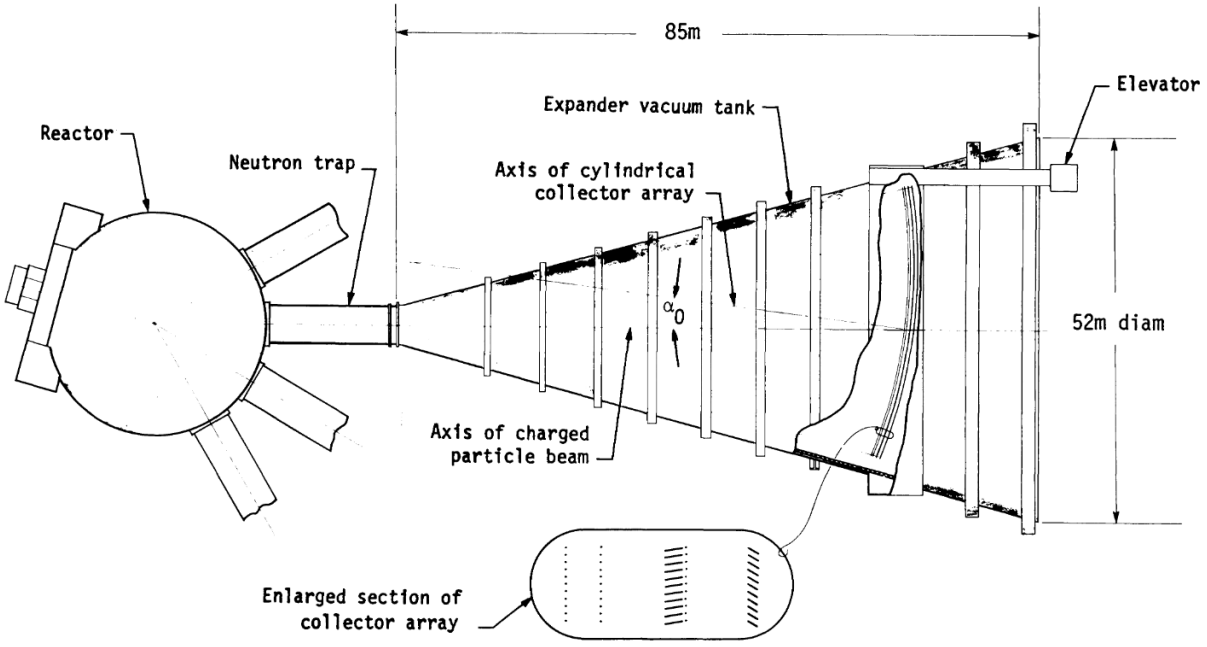


Figure 2.8: A magnetic expander positioned between the fusion reactor and the collection array. The expander transfers energy from $W_\perp$ to $W_\parallel$, accelerating the plasma. An ambipolar electric field transfers energy from electrons to ions. From Barr et al.[2]

collection system situated at the end of the nozzle. The Venetian blinds can occlude some fraction of the plasma exhaust, sufficient to provide the recirculating power in Figure 1.8. The remainder of the exhaust area can be left open to provide thrust.

While this concept is promising for a large fusion power-plant or spacecraft, it is not attractive at the scale of university research. The expander in Figure 2.8 is 85 meters long, which exceeds the research budget and space constraints at UW by multiple orders of magnitude. So, the expander must be omitted. The plasma exhaust of ZaP-HD does not have a high electron temperature. Assuming ion and electron energies of approximately 50 eV, a grid charged to -100 V should reflect the electrons and accelerate the ions. This results in an energy loss, so it is not a practical solution for an operational product, but it is suitable for research purposes.

2.3.2 Space Charge Effects

Conceptually, direct collection by electrostatic fields works like a gridded ion thruster in reverse. The electric field between grids decelerates the ions, trading kinetic energy for electrostatic potential energy. However, both ion thrusters and electrostatic collectors can be space charged limited. Space charge occurs because the plasma between the grids is not quasi-neutral, resulting in a net charge-density. This modifies the electric potential field as described by Poisson's equation, $\nabla^2\phi = -\rho_c/\mathcal{E}_0$.

An ion thruster produces a mono-energetic beam. However, a realistic VB collector must decelerate ions with a spectrum of energies. The space charge limit for such a device must be computed numerically. However, the mono-energetic case has an analytical solution which provides a useful approximation.

Consider a mono-energetic beam of ions with current density j_B and initial kinetic energy W_0. It is directed at a single collector of voltage V_p located at position x_p, shown in Figure 2.9. Ground voltage is established by a screen grid at $x = 0$. A perfectly efficient collector has $V_p = W_0/qZ$ where Z is the charge number of the ion. In steady-state, j_b must be constant for all x. Since $j = qnv$, this means that density n is increasing as v decreases. Curve 'a'

represents the vacuum potential. Since $E = -\nabla\phi$, the slope of 'a' is the vacuum electric field. As charge density increases, the potential can increase in front of the collector. Curve 'b' represents the space-charge limit where $d\phi/dx = 0$ at x_p. If you attempt to exceed the space charge limit, a potential field like curve 'c' develops, which peaks before the collector. The peak potential exceeds V_p, but the ions had initial energy $W_0 = qV_p$, so they reflect before reaching the collector. If the beam is not mono-energetic, ions with higher energy can still pass through. However, lower energy ions are deflected prematurely, reducing efficiency.

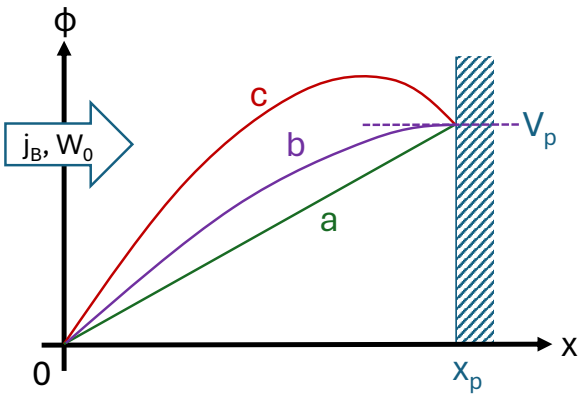


Figure 2.9: Potential diagrams for a 1-D single-plate collector. Case 'a' corresponds to the electric potential in vacuum, case 'b' is the space charge limit, and case 'c' exceeds the space charge limit. Reproduced from Miley[16].

Space charge is mitigated by increasing the electric field (the slope of the potential field). This is achieved by either increasing the voltage or moving the electrodes closer together. The voltage is determined by the energy of the ions to be collected. So, x_p is the only free parameter, and x_{max} is the distance between electrodes where the space-charge limit occurs. This is case 'b' where $E = d\phi/dx = 0$ at $x = x_p$. The solution to this is the well-known

Child-Langmuir law,

$$j_{b,max} = \frac{4}{9}\mathcal{E}_0 \left(\frac{2eZ}{m_i}\right)^{1/2} \frac{V_p^{3/2}}{x_p^2}, \tag{2.12}$$

$$V_p = W_0, \quad x_p = x_{max}, \tag{2.13}$$

$$x_{max} = \sqrt{\frac{4}{9}\mathcal{E}_0 \left(\frac{2eZ}{m_i}\right)^{1/2} \frac{W_0^{3/2}}{j_b}}. \tag{2.14}$$

As an example, consider a fusion reactor which produces a plasma exhaust with a jet power of $P = 250$ MW and a collector area of $A = 1000$ m^2. This happens to match the benchmark simulation case in Section 4.1. A mono-energetic beam of ions at $W_0 = 170$ keV is assumed. Assuming deuterium ions, $v_0 = 4 \times 10^6$ m/s.

$$j_b = \frac{P}{A}\frac{e}{W_0 e} = \frac{P}{AW_0} \qquad (W_0 \text{ in eV}) \tag{2.15}$$

$$j_b = 1.47 \text{ A/m}^2 \tag{2.16}$$

$$n_0 = \frac{j_b}{v_0 e} = 2.3 \times 10^{12} \text{ m}^{-3} \tag{2.17}$$

Equation 2.14 yields $x_{max} = 1.2$ meters. The benchmark case is designed to operate at the space charge limit and is 4 meters long. So, the mono-energetic calculation provides a conservative estimate, accurate to within an order of magnitude.

As another example, consider ZaP-HD. As discussed in Section 4.2.1, the mean ion energy in the exhaust is only 55 eV, with a density of 4×10^{19}. This is extremely dense and extremely slow compared to the benchmark. Assuming hydrogen gas, $v_0 = 10^5$ m/s, and $j_b = 6.6 \times 10^5$ A/m^2. The result is $x_{max} = 5$ μm, which is far too small to manufacture. To achieve a spacing of 1 meter, density needs to be reduced to 10^9 m^{-3}, 10 orders of magnitude. Fundamentally, this is why VB collectors are impractical for ion energies lower than 10 keV. Ion energy must be high to allow reasonable power densities.

Chapter 3

PARTICLE-IN-CELL CODE DEVELOPMENT

Direct Collection of charged particles by means of Venetian Blinds is modeled using a Particle-in-Cell (PIC) code developed by the author from a clean sheet. Work on this Python code began as a series of class exercises in AA 545: Computational Methods for Plasmas during the spring of 2023. After the class concluded, the author sought to apply this code to study other problems. The PIC method is particularly suitable to studying plasmas which are far from thermal equilibrium, such as those produced when particles are sorted by kinetic energy in DEC systems. This research required the development of many unique features, resulting in a new branch of the author's PIC code named *ZaP-DEC*.

This chapter explains how the PIC method works, and why it is a good choice for studying direct collection. ZaP-DEC uses a simplified plasma physics model, and justifications for these simplifications are provided. Finally, the architecture of the code is discussed, detailing how the code's classes and functions interact with each other. This section should be particularly useful for future researchers interested in using the code. The full text of the code in its entirety, including sample cases, is included in Appendix B.

3.1 Introduction to the PIC Method

The Particle-in-Cell method simulates how individual particles respond to electromagnetic fields, and how the particles affect those fields. Importantly, *PIC simulations are not N-body models*. An N-body model needs to simulate the interactions of each particle with every other particle. This requires $\mathcal{O}(N^2)$ calculations per time step, which can quickly become intractable as N becomes large.

To reduce the number of interactions, the PIC method incorporates a spatial computa-

tional grid upon which the fields are calculated. Particle properties $(q, \vec{v})$ are weighted to nearby grid points, then $\vec{E}$ and $\vec{B}$ fields are calculated based on charge and current density at the grid nodes. For M grid nodes, this requires $\mathcal{O}(M^2 + N)$ floating point operations (FLOPS) per time-step, which significantly reduces computational complexity if $N >> M$. More efficient algorithms exist which achieve this in $\mathcal{O}(M \ln(M) + N)$ operations, but ZaP-DEC is not this efficient.

Computational complexity is further reduced using super-particles, meaning each simulated particle actually represents the mass and charge of a large number of individual particles. There is an additional advantage to super-particles: it allows particle collection to be simulated statistically. This is discussed in detail in Section 3.4.1.

3.2 Suitability of PIC to VB Collection

The reader may be asking themselves: Why use a PIC code over the alternatives?

Particles models (such as PIC and N-body) are distinct from fluid models. A fluid model tracks statistical properties of the plasma rather than individual particles. If a plasma is in local thermodynamic equilibrium, then the velocity distribution of its particles can be described by a Maxwellian function. Thus, a large number of particles can be represented statistically using properties such as density, velocity, and temperature. By doing this, fluid models can significantly reduce computational complexity, allowing much larger problems to be simulated. For example, MHD fluid models can be used to simulate the stability of entire fusion reactors. However, if the plasma deviates from local thermodynamic equilibrium, more statistical properties must be tracked, such as skew and kurtosis. As more properties are added, the governing equations become more numerous and complex.

As described in Chapter 2, Venetian Blind collectors exploit their angular dependent transparency to sort particles by kinetic energy and thus improve collection efficiency. If one were to use a fluid code to model Venetian Blind collectors, the distribution functions need to be modified wherever particle collection occurs. The collection is highly directional, and the resulting distributions are far from equilibrium, making a statistical representation

impractical. In contrast, particle methods track individual trajectories, and it is easy to calculate how the plasma is affected by collection.

In summary, PIC is suitable for this problem because the collection process results in a plasma far from equilibrium. In fact, it depends on the plasma remaining far from equilibrium between each successive stage of collectors. This implies that Venetian Blinds only work for plasmas which remain in a non-equilibrium state as they pass through the device. Thus, these plasmas must be nearly collisionless. In the ZaP-DEC code, collisions are not calculated, which is a suitable approximation for these devices.

3.3 ZaP-DEC: Unique Features and Physics Scope

In pursuit of this research, many custom features had to be developed. The most obvious new feature is the inclusion of Venetian blind collectors, also called ribbon grids, which exhibit angular-dependent opacity. The collectors are charged to an electric potential, meaning that ZaP-DEC requires Dirichlet boundary conditions instead of the periodic boundaries implemented in AA 545. Solving for the electric field now requires decomposing the domain into regions bounded by electrodes up- and down-stream, so considerable overhead is added to handle an arbitrary number of regions. Finally, many new visualization functions had to be developed to understand the physics being simulated. This research had to be completed in a reasonable amount of time, so a physics model had to be selected which is simple enough to be implemented quickly while also yielding reasonable results.

ZaP-DEC is designed as a 1-Dimensional Electrostatic PIC code with particle kinetics extended to 2D. Magnetic fields are not simulated, and the electric field is only simulated in one dimension, z, which corresponds to the axis of a Z pinch. Thus, particles may only be accelerated in the z direction. However, the position of the particles in y and z are evolved over time. This feature is required to determine the angle of each particle's trajectory when it is incident on a Venetian Blind collection grid. A particle's vertical velocity is constant, but may be unique. Typically, v_y is determined by the beam incidence angle and plasma temperature. The computational domain is periodic in the y-direction, but is necessarily

finite in the z direction. Collection grids, also called electrodes, are modeled as infinite planes at known z-coordinates. They have a known potential, and the program must solve for the electric field between each grid.

This approach greatly simplifies the physics to be simulated. The first benefit is performance: a 2D solver requires a 2D grid, which squares the number of grid points for a square domain. This nearly squares the number of FLOPS per time-step. The second benefit is avoiding the complexity of handling boundary conditions in the y-direction. The third benefit is that angular dependant collection behavior can be abstracted statistically, as explained in Section 3.4.1.

However, the simplified approach has some obvious drawbacks. Since fields are not simulated in the y-direction, predictions are only valid for a plasma with no vertical variation. In reality, the collection grids have finite height, and there is a large gap between the vertical extent of the grids and the vacuum vessel, which is grounded. Free thermal expansion and space charge effects causes the plasma to spread out vertically, decreasing the density of the plasma and changing the v_y component of each particle, modifying particle trajectories. One can neglect these effects if the grid spacing is relatively small compared to their height, which may not be the case. Thus, this model over-predicts the plasma density between grids. This results in conservative predictions because the space charge limit decreases with density, as discussed in Section 2.3.2. If the beam incidence angle is zero, particle trajectories are not affected along the axis of a real device due to symmetry. Unfortunately, the beam incidence angle must be non-zero, so some of these effects persist at the axis, but this should be small. Therefore, predictions from this model should be tested near the z-axis.

Another simplification of this model is the omission of magnetic fields. However, Venetian blind collection is impractical if the ions are magnetized. In absence of collisions, strongly magnetized ions follow magnetic field lines, making it impossible to sort particles by kinetic energy using this method. If particles are moderately magnetized such that the gyroradius is approximately equal to the scale length of the collectors, then its components v_y and v_x oscillate as it gyrates around the field lines, changing the trajectory and resulting in

premature collection of many particles. As discussed in Section 2.3.1, a practical system utilizes a magnetic expander or nozzle so that perpendicular thermal energy can be converted into a bulk fluid velocity in z. The collection array itself must be located far downstream of the expander where the magnetic field can be neglected. It is therefore appropriate to neglect that magnetic field for this analysis.

3.4 ZaP-DEC Code Architecture

An explicit electrostatic PIC algorithm has the following high-level steps:

1. Weight charge from particles to grid.
2. Calculate the electric field on the grid.
3. Weight electric field from grid to particles.
4. Push particles in response to the field.
5. Repeat.

In this section, it is demonstrated how these steps are implemented in the ZaP-DEC application. The program is divided among several files which contain classes, their associated methods, and stand-alone functions. The inter-dependency of these modules is shown in Figure 3.1 with the high-level steps marked (1) through (5). Note that the class `ZaP_DEC()` contains the main loop that runs the program. A `ZaP_DEC` object contains all the information about a particular analysis: the design of all electrodes, their positions and potentials, the computational grid, all particles, and their associated histories. This allows the user to run many different simulations without mixing up what data belongs to which analysis.

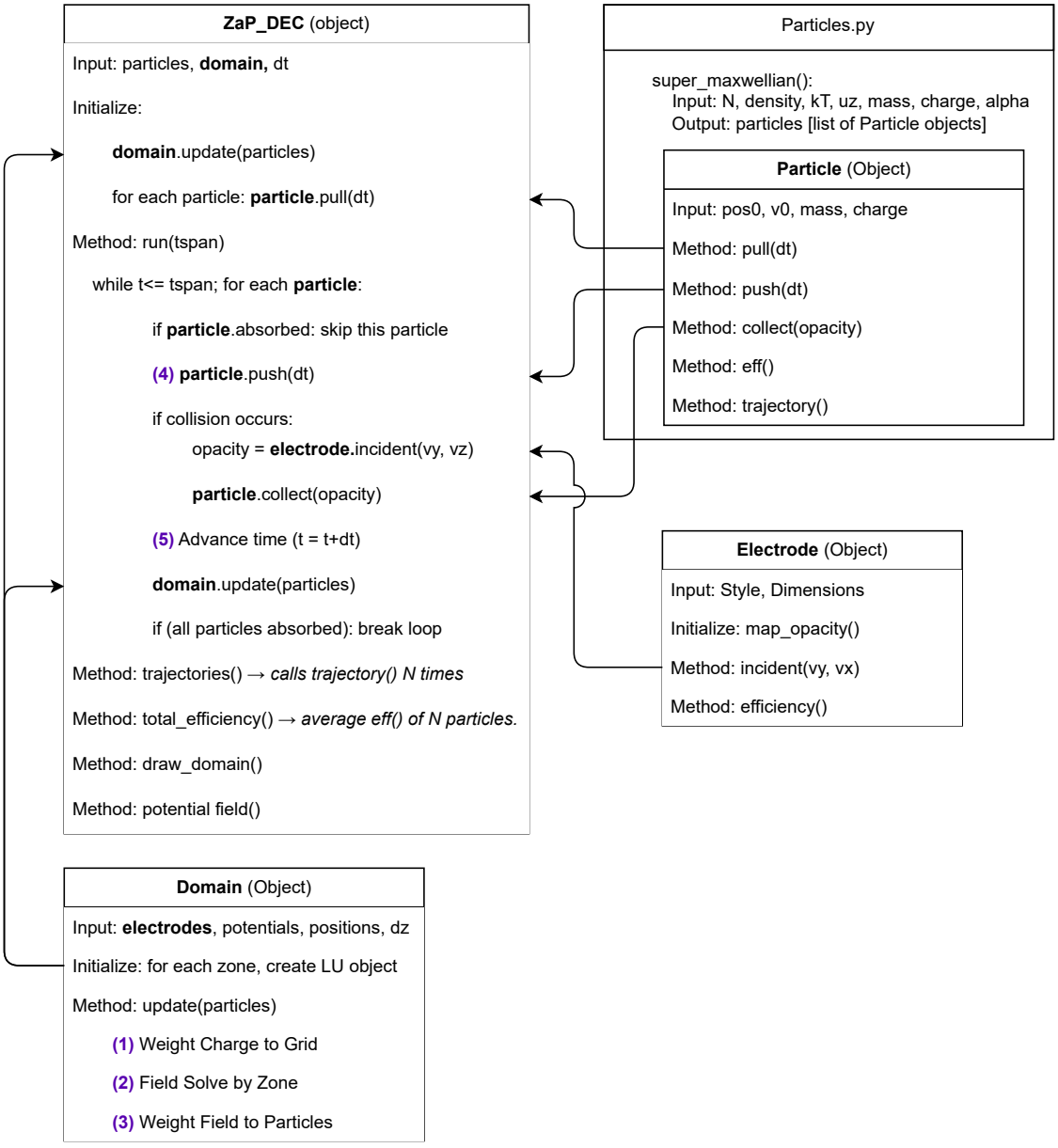


Figure 3.1: A pseudocode overview of ZaP-DEC. Numbers in purple correspond to the steps of an electrostatic PIC algorithm. Bold font indicates 'objects' (instances of a class).

The main loop of the program is executed by the method `ZaP-DEC.run()`. In addition to the high-level steps, the main loop orchestrates the collection of particles on electrodes, allowing a calculation of collection efficiency. The simulation runs until the end of the desired time-span, or when all particles have been absorbed, whichever comes first. If a time-span is not long enough to absorb all particles, `ZaP-DEC.run(tspan)` may be called with a larger time-span. This continues the analysis from where it left off without needing to re-run previous time-steps.

3.4.1 Electrodes

As described in Chapter 2, Venetian blind collectors exploit their angular dependent opacity to sort particles by their kinetic energy and thus improve collection efficiency. It is computationally expensive to simulate collisions of individual particles with individual blinds on a collector. However, it is computationally cheap to determine when a particle passes through a particular z-coordinate. The collection of individual particles can be represented statistically by collecting some fraction of a super-particle when it passes through a grid of known opacity. The `Electrode()` class allows a user to specify the angular dependent opacity of each grid in a simulate DEC system.

In ZaP-DEC, electrodes may have three different styles: wall, screen, and ribbon. The simplest case is the wall: it has 100% opacity, meaning that any super-particle is completely absorbed if it hits a wall. ZaP-DEC requires that a wall is specified at the first and last z-coordinates of the domain, and it may be placed nowhere else. A screen electrode has a fixed opacity of 1% by default, but the user may change it as needed. Screens are used to represent the ground grid and electron reflector grid.

Finally, the most important style is 'ribbon'. A Venetian blind collector is composed of many ribbons, and the opacity of the collector depends on the angle, length, and pitch of the ribbons. When a ribbon electrode is initialized, the program runs a method called `map_opacity()`. This draws three vectors, each corresponding to a ribbon. Next, a flow vector is drawn at zero degrees. The length and pitch of the blinds is projected in the direction

of the flow vector. The ratio of (projected length)/(projected pitch) represents what fraction of area is occluded by ribbons for a given flow direction. The program repeats this analysis in a loop, varying the flow vector from 0 to 360 degrees with 1 degree increments, generating a vector of opacities. This process can even be animated by calling `map_opacity()`. A single frame of such an animation is shown in Figure 3.2.

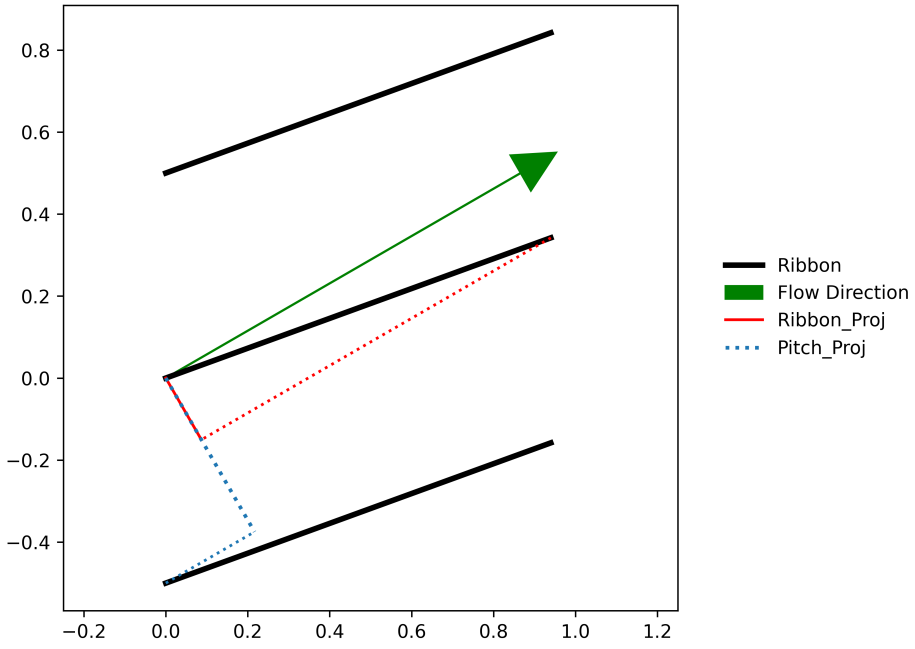


Figure 3.2: Ribbon length and pitch are projected along the flow vector to determine opacity. This is repeated over 1 degree increments to produce a table of opacity versus incident angle.

Whenever a particle crosses an electrode, the method `incident(vy, vy)` is called. If the electrode in question is a grid or a screen, the fixed opacity is returned. However, if it is a ribbon grid, the incident angle is calculated from the velocity components. The opacity of the ribbon grid is interpolated between values in the opacity vector, and opacity is returned.

Note that `Electrode()` objects do not store any information about their position in the domain or their electric potential. Both of those are specified in `Domain()`. However, electrode objects have two useful attributes called `collected` and `wasted`. These attributes

start at zero, but are updated by `ZaP_DEC()` whenever particles are incident on grids. This lets the user track collection efficiency of individual grids using `Electrode.efficiency()`, making it easy to see which grids need design changes. Collection efficiency is also tracked for each individual super-particle, which is discussed in Section 3.4.3.

3.4.2 Domain

After electrodes are defined, the computational domain can be created as an instance of the `Domain()` class. The user must provide 3 python lists: a list of electrode objects, a list of their electric potentials, and a list of their positions along the z-axis. Their positions must be ordered from left to right, and all three lists must correspond to each other. Finally, the user must also specify the grid spacing (dz). When the domain is initialized, the program automatically decomposes the domain into regions bounded by grids, shown in Figure 3.3.

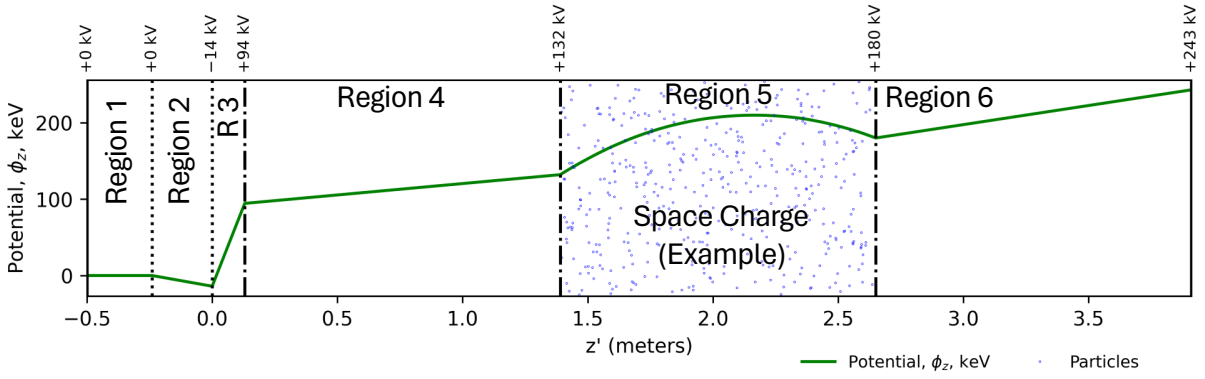


Figure 3.3: An example of a domain after the electric field has been solved. Region 5 has charged particles that modify the potential. This example exceeds the space charge limit.

Several rules must be followed when defining the domain. First, every electrode must reside exactly on a grid point, so their coordinate must obey $z = N(dz)$, where N is an integer. Second, grids must be at least two grid points apart, but it is recommended to have many more cells between grids. Third, the domain boundaries must be 'wall' electrodes

because particles are not allowed to pass through them. The last rule is that there must be a screen electrode at z=0. This is the electron reflector grid, and all plasma upstream of this grid is quasi-neutral. Downstream, electrons are not simulated.

Some practices are recommended but not required. It is recommended that users calculate what potential successfully deflects nearly all of the electrons in the upstream plasma. Electron screening is hard-coded into ZaP-DEC, but the voltage of the reflector grid is not enforced. This is done because electron screening is necessary for DEC to function, but electrons themselves are not simulated, and no assumptions are made regarding the electron velocity distribution. In the case where a magnetic expander is used, electrons have low energy compared to ions. However, in cases where no expander is used, the assumption $T_e \approx T_i$ can be made in certain situations. Failure to calculate this correctly can result in invalid collection efficiency predictions. Note that particle collection on the reflector grid is simulated, so incident ions drive a negative current, representing an energy loss proportional to the reflector grid voltage.

It is recommended to include one screen grid upstream from the reflector grid, called the ground grid. This grid, and the left wall, should have a potential of zero, resulting in a vacuum electric field of zero in region 1. This is the region in which particles should be initialized because they will have an electric potential energy of zero. When calculating collection efficiency, the initial kinetic energy is accounted for, but not the initial electrostatic potential energy, so it is wise to avoid this error. However, this is not required because the user may want to simulate a case where electrons have already been removed by magnetic fields, such as a cusp device as depicted in Figure 2.6. In that case, the reflector grid may have a voltage and opacity of zero, and the upstream wall may have a voltage of zero.

Once per time-step, ZaP-DEC calls the Domain.update() method. This is an extremely important section of code that performs steps 1, 2, and 3 of the high-level PIC algorithm. The next three subsections explain these steps in detail.

PIC Step 1: Weight Charge to Grid

In the PIC method, particles are often treated as a 'cloud' of charge whose shape determines how charge is distributed to nearby grid points. In ZaP-DEC, each particle's cloud is as wide as a grid cell (dz), with a charge density of (q_i/dz) as shown in Figure 3.4. This charge can be weighted proportionally to neighboring grid cells to simulate the macroscopic electric fields observed in plasmas over length scales greater than the Debye length. This is called Linear Grid-Point Weighting. With ions indicated by subscript i and grid points indicated by subscript j,

$$a = z_{j+1} - z_i \qquad b = z_i - z_j \tag{3.1}$$

$$\rho_j = \rho_j + \left(\frac{q_i}{dz}\right)\frac{a}{dz} \tag{3.2}$$

$$\rho_{j+1} = \rho_{j+1} + \left(\frac{q_i}{dz}\right)\frac{b}{dz}, \tag{3.3}$$

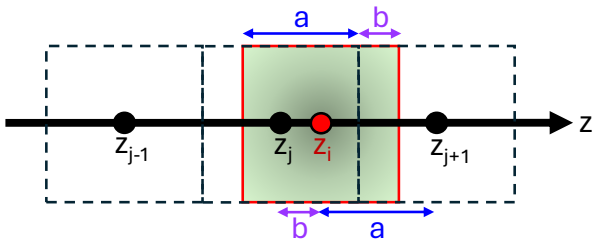


Figure 3.4: Charge distributed over a particle's shape which is apportioned to adjacent grid cells. Above, distances a and b show what fraction of the charge is be weighted to grid cells Z_j and Z_{j+1}, respectively. Below, a and b are relocated to demonstrate the inverse relationship between distance to the particle and the apportioned charge.

Whenever `domain.update()` is called, the charge density assigned to all grid-points is reset to zero. Next, the program loops through each particle and weights its charge to adjacent grid-points. After the loop is completed, all of the charge is accounted for.

For regions upstream of the electron reflector grid ($z < 0$), the plasma is assumed to be quasi-neutral. Electrons are not simulated explicitly. Instead, the average charge density of each region is subtracted from each of its constituent points. Thus, the total charge of the region is zero, but local space charge can exist. It is assumed that zero electrons exist downstream of the reflector grid.

If one chooses dz to be smaller than the Debye length, plasma dynamics such as wave propagation can be accurately simulated. The Debye length is

$$\lambda_D = \sqrt{\frac{\epsilon_0 k T_e}{n_e q_e^2}}. \tag{3.4}$$

For dense plasmas, the Debye length can be extremely small. If the domain is much larger than the Debye length, it becomes computationally difficult to achieve $dz = \lambda_D$. In Chapter 4, it is shown that this is indeed the case, so wave propagation is not investigated in the simulated DEC system.

PIC Step 2: Field Solve on the Grid

The electric potential at each grid point is solved using Poisson's equation,

$$\nabla^2 \phi = -\frac{\rho}{\epsilon_0}. \tag{3.5}$$

In our case, the domain and ρ are discrete, so the Laplacian operator is approximated using the second-order central finite-difference equation,

$$\frac{\phi_{j-1} - 2\phi_j + \phi_{j+1}}{(dz)^2} = -\frac{\rho_j}{\epsilon_0}. \tag{3.6}$$

Each region of '$m + 1$' points is bounded by electrodes with known potentials, ϕ_0 and ϕ_m. Solving Eq. (3.6) at the boundaries,

$$\text{Left side of domain:} \quad \frac{\rho_1}{\epsilon_0} - \frac{\phi_0}{(dz)^2} = \frac{-2\phi_1 + \phi_2}{(dz)^2}, \tag{3.7}$$

$$\text{Right side of domain:} \quad \frac{\rho_{m-1}}{\epsilon_0} - \frac{\phi_m}{(dz)^2} = \frac{\phi_{m-2} - 2\phi_{m-1}}{(dz)^2}. \tag{3.8}$$

This can be written as a linear algebra problem of the form $Ax = b$, where A is a matrix representing the Laplacian operator with periodic boundary conditions,

$$\boldsymbol{A} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_{m-2} \\ \phi_{m-1} \end{bmatrix} = \begin{bmatrix} \frac{\rho_1}{\epsilon_0} - \frac{\phi_0}{(dz)^2} \\ \rho_2 \\ \vdots \\ \rho_{m-2} \\ \frac{\rho_{m-1}}{\epsilon_0} - \frac{\phi_m}{(dz)^2} \end{bmatrix} \tag{3.9}$$

$$\text{where} \quad \boldsymbol{A} = (dz)^{-2} \begin{bmatrix} -2 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 \end{bmatrix} \tag{3.10}$$

Solving $x = A^{-1}b$ inside of the loop is computationally expensive, requiring $\mathcal{O}(m^3)$ operations for each time-step, where m is the number of interior grid points. Instead, the matrix A is pre-constructed for each region during initialization of the domain, then LU decomposition is performed on each matrix. This involves factoring A into a lower triangular matrix (L) and an upper triangular matrix (U) such that A = LU. This requires $\mathcal{O}(m^3)$ operations, but is only performed once per analysis. Inside of the loop, `domain.update()` solves the two triangular systems $Ly = b$ and $Ux = y$ using using forward and backward substitution, respectively. This has a complexity of $\mathcal{O}(m^2)$ operations for each time-step.

After finding potential, the electric field is calculated as $E = -\nabla\phi$. This is performed over the entire domain in one step, it is not done by region. The built-in numpy function `gradient()` solves this using the central difference formula for interior points and the forward and backward difference formulas at domain boundaries.

PIC Step 3: Weight Field to Particles

Now that the electric field is known at each grid point, one must interpolate the field values between those points to determine how much electric field is 'felt' by each particle. This is the inverse of Eq. (3.2) and (3.3), except that electric field is weighted instead of charge.

$$a = z_{j+1} - z_i \qquad b = z_i - z_j \tag{3.11}$$

$$E_i = E_j \left(\frac{b}{dz}\right) + E_{j+1} \left(\frac{a}{dz}\right) \tag{3.12}$$

3.4.3 Particles

Each super-particle is an instance of the class `Particle()` with the following attributes:

1. The initial kinetic energy and initial mass of the super-particle.
2. How much of the super-particle's energy has been collected.
3. The particle's current charge and mass, which decreases during collection events.
4. Absorbed: A boolean which is true if the particle no longer exists.
5. Position (y, z), stored at each time-step.
6. Velocity, where v_y is constant and v_z is stored for each time-step.
7. Velocity at one half step $(\Delta t/2)$ before the current time-step.
8. The electric field strength felt by the particle at the current time-step.
9. A list of y-boundary crossings to facilitate plotting with periodic boundary conditions.

The electric field strength has already been discussed: this value is weighted to particles by `Domain.update()`. The remainder of the attributes are discussed in this section, except for the last one. Y-boundary crossings are discussed in Section 3.4.4.

Of particular importance is the attribute 'absorbed' which indicates that a particle is in its final state and not to be updated. When this attribute is true, `ZaP_DEC()` omits it from the loop, saving computational power. A simulation ends when `ZaP_DEC()` detects that all particles have been absorbed.

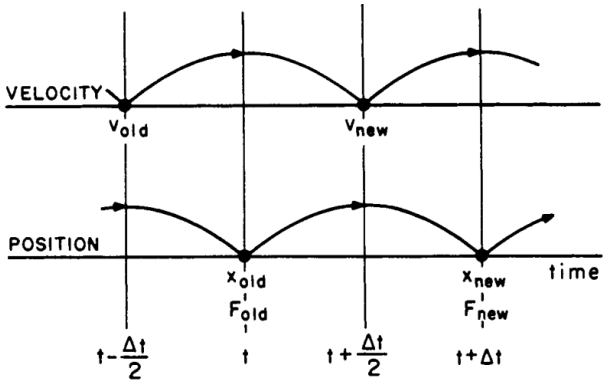


Figure 3.5: Leap Frog Method: Force is time centered while advancing velocity. Velocity is time centered while advancing position.[5]

PIC Step 4: Push particles in response to field

At each time-step, particles accelerate in response to the electric field. This acceleration can be integrated over time to find velocity. The velocity can be integrated to find position. When performing numerical integration, it is important to choose an integration method which is computationally efficient while minimizing the accumulation of errors over many time-steps. ZaP-DEC uses the Leap-Frog Method, which has second-order accuracy, $\mathcal{O}(\Delta t^2)$. This means that the numerical solution v_h has error $E = |v - v_h| \propto \Delta t^2$, where Δt is the time-step. This algorithm is used once per time-step when the main loop calls `Particle.push()`.

The Leap-Frog Method applies a force which is centered in time between the old velocity and the new velocity. In practice, this means that velocities must be calculated one half-step before and after the force is calculated, as shown in Figure 3.5.

$$v_{z,new} = v_{z,old} + E_z \frac{q}{m} \Delta t \tag{3.13}$$

$$z_t = z_{t-1} + v_{z,new} \Delta t \tag{3.14}$$

After this step, $v_{z,new}$ and $v_{z,old}$ are averaged to find v centered at time t. This final result is stored for every time-step so that visualizations of trajectories in position and

velocity space are consistent with respect to time. However, the centered velocity is never used by `Particle.push()` during subsequent timesteps. Instead, $v_{z,new}$ at step t-1 becomes $v_{z,old}$ at step t. This is why $v_{z,old}$ is stored as an attribute for all particles.

At the start of an analysis, `Particle.push()` cannot be used because $v_{z,old}$ is not known. So, the method `Particle.pull()` is used once for each particle. This uses Euler's method to integrate backwards by a single half-step in time:

$$v_{z,old} = v_z - \frac{\Delta t}{2} E_z \frac{q}{m}. \tag{3.15}$$

Euler's method has only first-order accuracy, $\mathcal{O}(\Delta t^1)$, but the errors do not accumulate because it is only used once per analysis.

Since there is never any acceleration in the y direction, v_y is constant. Therefore, there is no need to use leap-frog to solve for the vertical position y. Instead, $y_t = y_{t-1} + v_y \Delta t$, with an error equal to machine precision at each time-step.

Particle Collection

The Venetian blind electrodes are charged to a positive potential which is intended to retard the velocity of incident particles. This converts some of the particles kinetic energy into electrostatic potential energy. When the particle touches a grid, it is assumed that the particle becomes neutralized. This pulls an electron out of the electrode. A large number of such events drives a current at the electrode's potential, resulting in collected power, $P = IV$. To achieve 100% collection efficiency, a particle's velocity must equal zero in the limit where distance to the electrode approaches zero. Any kinetic energy that remains is deposited into the electrode as heat, so it is wasted.

However, ZaP-DEC uses super-particles, representing many particles. When a super-particle is incident on an electrode whose opacity is less than 100%, a fraction of the particle is absorbed. So, super-particles may need to collide with several electrodes before they are completely absorbed, demonstrated by Figure 3.6. During each collision, some electrostatic energy is collected, and some kinetic energy is lost.

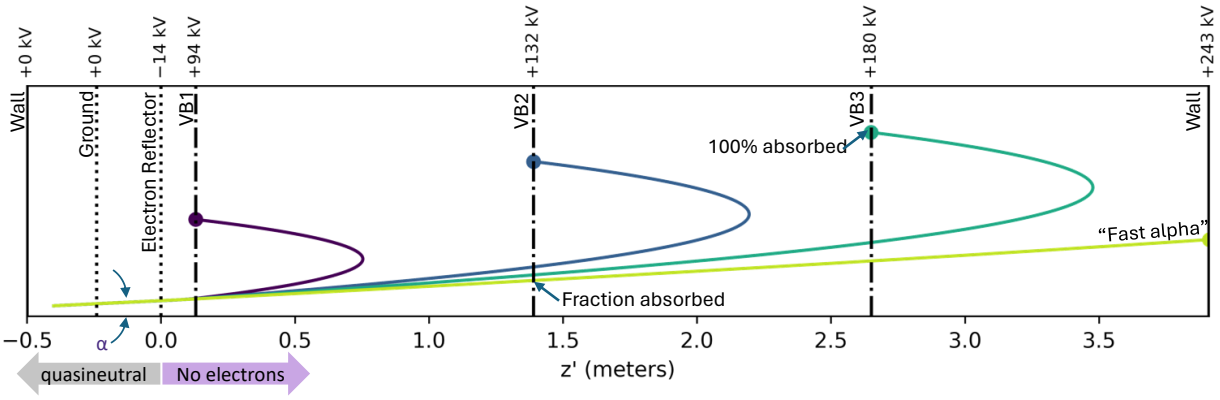


Figure 3.6: Super-particles experiencing several collection events before full absorption.

After a super-particle's position has been advanced by `Particle.push()`, the program checks to see if an electrode exists between z_t and z_{t-1}, which implies that the particle crossed through it. If such a crossing has occurred, the particle's velocity components are passed to `Electrode.opacity()`, described in Section 3.4.1, returning the opacity.

The opacity and electrode potential are passed to `Particle.collect()`. The opacity value represents what fraction of the super-particle's mass and charge is absorbed when it passed through the electrode. This allows us to determine how much energy is collected, and how much of the particle's kinetic energy is lost. The new mass and charge of the particle are also updated. With opacity represented by K,

$$E_{collected} = q\phi K \tag{3.16}$$

$$E_{lost} = \frac{1}{2}m(v_y^2 + v_z^2) \tag{3.17}$$

$$m = m(1 - K) \tag{3.18}$$

$$q = q(1 - K) \tag{3.19}$$

At this point, if less than 1% of the particle's original mass remains, it is flagged as absorbed. It is now in its final state, and is skipped in all future calculations. This is why the initial mass of the particle is stored as an attribute.

Collection Efficiency

There are three different ways of measuring efficiency in ZaP-DEC: efficiency per particle, efficiency per electrode, and total efficiency for the whole analysis.

Efficiency per particle is calculated using `Particle.eff()`. This takes the sum of all energy collected and divides it by the initial kinetic energy. The definition of initial kinetic energy can be calculated with 2 or even 3 dimensions. When particles are initialized, the argument `KEx=True` may be specified. This assumes that a particle has equal kinetic energy in the x and y dimensions, even though the x dimension is not simulated. While this is not accurate for a single particle, it can yield a fair approximation for a large number of particles.

$$\text{if KEx=True:} \quad W_0 = .5m(v_{0z}^2 + 2v_{0y}^2) \tag{3.20}$$

$$\text{else:} \quad W_0 = .5m(v_{0z}^2 + v_{0y}^2) \tag{3.21}$$

Using this feature always results in decreased efficiency because $v_\perp$ components cannot be recovered by E_z electrostatic fields

Efficiency per electrode is calculated using `Electrode.efficiency`. This calculation is similar, but it does use E_{lost} because it needs to track where exactly the energy is lost. This is helpful for a user to determine where the design of the system can be improved. The grid with the lowest efficiency may need a change to the angle of the blinds, their spacing, the axial position of the collector, or its potential. Because this is the only use-case for the feature, no capability is added to estimate losses in a 3D system.

Total efficiency sums the energy collected for every particle and divides it by the sum of the initial kinetic energies of all particles. This is performed by `ZaP_DEC.total_efficiency`, and it represents the final result of the analysis.

Creating super-particles in large numbers

It can be useful to initialize particles one at a time and to simulate their trajectory through a series of VB grids. However, to simulate more realistic scenarios, and to evaluate space

charge limitations, it is necessary to create large numbers of super-particles. This can be achieved using `super_maxwellian()`, a function in `particles.py`.

This function creates a list of N super-particles with random uniform distribution in space and with a maxwellian distribution of velocities. In 1-D, a maxwellian distribution is

$$f(v) = \left(\frac{1}{2\pi v_t^2}\right)^{\frac{1}{2}} \exp\left(\frac{-(v-u)^2}{2v_t^2}\right), \quad v_t = \sqrt{\frac{kT}{m_s}}, \tag{3.22}$$

where $f(v)$ is the probability of a particle having velocity v, v_t is the thermal velocity at temperature T, m_s is super-particle mass, and u is the average fluid velocity. This is identical to a normal distribution with standard deviation equal to thermal velocity ($\sigma = v_t$).

Since this analysis is two dimensional, it requires a maxwellian velocity distribution for each direction. In the case of ZaP-HD, or other SFS Z pinches, the plasma exhaust is expected to have a significant axial fluid velocity u_z. As discussed in Section 3.3, this analysis is assumed to apply at the Z-Pinch axis and to have no vertical variation. Due to radial symmetry, $u_r = 0$ at the axis, which translates to $u_y = 0$ in Cartesian coordinates. So, $f(v_z)$ is evaluated with $u = u_z$, while $f(v_y)$ uses $u = 0$. The temperatures are assumed to be equal in both directions. However, the benchmark case described in Section 4.1 utilizes a magnetic expander, so temperature $T_\perp = T_y$ is assumed to be zero. For such cases, a user may include the argument `beam=true`, but it is false by default.

As described in Section 2.1.2, the collection grids are not normal to the z-axis. Instead, they are rotated by a small angle 'α'. So, after calculating v_y, v_z for every super-particle, a coordinate transformation is performed:

$$v_z' = v_z \cos(\alpha) - v_y \sin(\alpha) \tag{3.23}$$

$$v_y' = v_z \sin(\alpha) + v_y \cos(\alpha) \tag{3.24}$$

Formally, the rest of the ZaP-DEC program operates in the coordinate system (y', z'), and plots label the horizontal axis as z', but the distinction is dropped everywhere else.

Note that Eq. (3.22) uses the thermal velocity of super-particles with mass m_s. At this point, it is necessary to define how super-particles relate to their constituent particles

explicitly. In the present of an electric field, an ion's acceleration is

$$\frac{d\vec{v}_i}{dt} = \vec{E}\left(\frac{q_i}{m_i}\right). \tag{3.25}$$

If the charge to mass ratio is unchanged, the acceleration is unchanged. Thus, the following properties are defined,

$$m_s = m_i\frac{N_{plasma}}{N_{model}}, \quad q_s = q_i\frac{N_{plasma}}{N_{model}}, \quad n_s = n_i\frac{N_{plasma}}{N_{model}}, \tag{3.26}$$

where m_s and q_s represent mass and charge of a single super-particle, while n_s is the density of the super-particles in a fluid that represents ions with density n_i.

For a given simulation, a user can exert control over the computation time by selecting the number of super-particles, N_{model}, to represent a plasma with density n_i. However, finding the number of constituent particles, N_{plasma}, requires a volume. The PIC algorithm is 1-dimensional, so a range of z-values is sufficient for this purpose:

$$N_{plasma} = n_i(z_{0,max} - z_{0,min}), \tag{3.27}$$

which has units of particles per square meter. In a 1-D analysis, particles are considered to have infinite extent in the non-resolved dimensions, so these particles are infinite planes located at coordinate z_i. As described in Section 3.3, it is recommended to set this range equal to the limits of region 1, where the vacuum electric field is zero.

Users can also choose to initialize particles over a range of y values. This has no effect on super-particle properties or plasma dynamics. The feature is included to permit realistic looking visualizations, but it makes the visualizations less useful because particle trajectories are more difficult to interpret. It is recommended to set the y range to $[y_0, y_0]$ such that all particles have the same initial y-coordinate.

When using super-particles, an important consideration is that the Debye length of the plasma changes:

$$\lambda_D = \frac{v_t h}{\omega_p} = \sqrt{\frac{\epsilon_0 kT_i}{n_i q_i^2}} = \sqrt{\frac{\epsilon_0 kT_s}{n_i q_s^2}}\sqrt{\frac{N_{model}}{N_{plasma}}}. \tag{3.28}$$

To reduce computation time, $N_{model} \ll N_{plasma}$, meaning the Debye length gets much smaller. As discussed in Section 3.4.2, resolving a Debye length is important to study plasma dynamics such as waves, but Eq. (3.28) makes this even more difficult.

In summary, `super_maxwellian()` requires a user to specify the number of super-particles N_{model}, the spatial range over which they should be distributed, the mass m_i and charge q_i of the constituent (non-super) particles, the plasma density n_i, the fluid velocity u_z, the temperature T_e in eV, and a beam angle α. The function outputs a list of super-particles with the desired properties. To simulate a multi-species plasma, `super_maxwellian()` can be called multiple times to make lists which can then be concatenated together. The new list, along with a `Domain` object and time-step 'dt', are passed to `ZaP_DEC()` to create an analysis object which can be run.

3.4.4 Visualization

This code required new efforts to visualize the results, beyond the tools developed for the original PIC code developed in AA 545. The original PIC code used periodic boundary conditions in both directions. This is handled in a simple manner: after `particle.push()` updated the position of a particle, $x = x\%L_x$, where L_x is the length of the domain and % is the modulus operator. Modulus computes the remainder after division, which implements periodic boundary conditions. However, plotting trajectories of particles with lines caused artefacts: vertical or horizontal lines appeared wherever a particle crossed a boundary. In AA545, this is handled by not using lines. Instead, points are placed at every time-step for each particle.

While that method worked, the nature of VB collection requires a large number of particle trajectories which must be distinguished from each other as they cross the domain. If they cannot be distinguished, it is difficult to tell which trajectories are problematic for energy recovery. So, tools are developed to automatically plot particle trajectories one line-segment at a time, with Figure 3.7 being an example.

To do this, every particle has an attribute called '`cross`'. During `Particle.push()`,

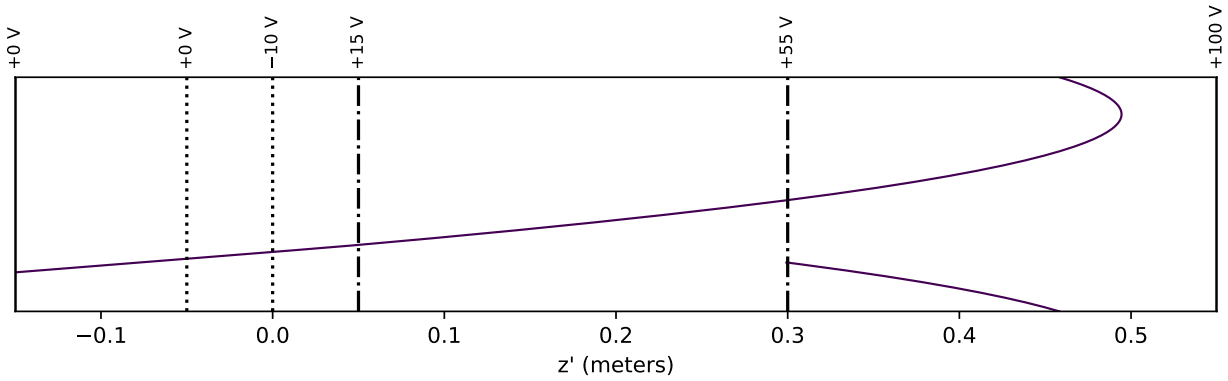


Figure 3.7: Plotting line segments over a periodic domain. Plotting particle trajectories with lines, as opposed to dots, is easier to interpret.

whenever a particle crosses a y-boundary, a linear interpolation is performed to determine the z-coordinate of the crossing:

$$f = (y_b - y1)/(y2 - y1) \tag{3.29}$$

$$z_c = f(z_t - z_{t-1}) + z_{t-1} \tag{3.30}$$

$$\text{crossing} = [y_b, z_c], \tag{3.31}$$

where y_b is the boundary coordinate and z_c is the intercept with y_b. The value of crossing shown here is appended to the attribute `cross`, resulting in a list of coordinates. These represent intermediate points not associated with a time-point. The actual position of the particle, which is integrated over time, is updated to wrap around the domain. So, the accuracy of the simulation is not affected, but these intermediate points can be inserted into the list of points being plotted to represent end points of line segments. If there is no crossing, *NaN* (meaning 'not a number') is appended to the `cross` attribute.

There is a method called `Particle.trajectory()` which takes a `matplotlib` axis object as an argument. It can also take a color, label, and linewidth as optional arguments. This allows a user (or another function) to define a plot, then call the trajectory of any particle of interest to populate the plot. It works by first creating a list of indexes that contain

crossings. In a loop, it plots the position data from a start index and start point until it reaches an index where a crossing occurred, called the 'crossing entrance'. That point is overwritten with the interpolated crossing point. Next, it defines a new start point and start index. The new start point matches the coordinates of the crossing point, except it is moved to the other boundary (the 'crossing exit'). The new start index is chosen to be just before the simulation data. As a result, two intermediate points are inserted for plotting purposes without modifying the physics model.

Other plotting tools are designed in the same modular fashion: accepting axis objects as arguments so that more complex plots can be built by calling individual functions. For example, `ZaP_DEC.draw_domain()` populates a plot with lines representing each electrode and labels their potentials. Different electrode types use different line styles. Using the option 'kV=True' divides all potentials by 1000 and label them as kV.

Using `ZaP_DEC.potential_fields()` superimposes a plot of plasma or vacuum potentials over the domain at a given snapshot in time, including the positions of particles that are modifying the field. This is already demonstrated in Figure 3.3. It is particularly useful for evaluating space charge effects.

As shown in Figure 3.2, the design of VB electrodes themselves can be plotted, and opacity vs incident angle may be animated using the `Electrode.map_opacity()` method.

Chapter 4

SIMULATION RESULTS AND ANALYSIS

ZaP-DEC is utilized to study a variety of scenarios. First, a benchmark case is studied to verify that the model can make predictions to a reasonable degree of accuracy. Next, plasma parameters relevant to the ZaP-HD experiment are studied. Due to the high density and low energy of the ZaP-HD exhaust, the space charge limit is exceeded. Further simulations are performed to determine how experiments may be conducted to yield useful results. Finally, the model is used to make predictions for a notional SFS Z-Pinch Fusion Reactor operating at $Q \gg 1$.

4.1 Benchmark Case

As a benchmark, ZaP-DEC is used to reproduce results from *'A preliminary engineering design of a "Venetian blind" direct energy converter for fusion reactors'* by Barr, et al.[2]

As discussed in Section 2.3.1, this case assumes that plasma is extracted from a Magnetic Mirror fusion reactor. These devices operate at low plasma density and trap particles by exploiting conservation of the magnetic moment. However, some particles have a velocity vector which resides in the loss cone of the mirror, and escape. These particles do not have a maxwellian energy distribution. In fact, most particles that escape are almost reflected in the mirror, so they have high velocity in the perpendicular directions (x, y), and low velocity in the axial direction (z).

Next, they enter the magnetic expander, which accelerates ions and electrons in the axial direction while decreasing their perpendicular velocities. The electrons try to attain extremely high speeds, but are impeded by an ambipolar electric field. The electrons lose energy to the ions, making them easy to deflect. From all of the above considerations,

Barr et al. performed a numerical calculation of the plasma exhaust velocity distribution. However, a curve of best fit is not provided in the paper. So, for the purposes of this study, a maxwellian distribution is produced which approximates the plasma properties.

The proposal assumed a plasma exhaust power of 1000 MW to be handled by 4 DEC modules, each intercepting 250 MW over a 1000 m^2 collection area. Barr et al. calculated a mean ion energy of 167 keV in each module. The current carried by each species is

$$I_{D^+} = 915 \quad \text{A}, \tag{4.1}$$

$$I_{T^+} = 570 \quad \text{A}, \tag{4.2}$$

$$I_{He^{2+}} = 26 \quad \text{A}. \tag{4.3}$$

The authors note that "although the He ions are born with 3.52 MeV, their mean escaping energy is only about twice that of the D^+ or T^+ and is a factor 100 down in intensity." In this study, a mean energy of 334 keV is used for He^{2+}, while 167 keV is used elsewhere.

This information is sufficient to create a Maxwellian approximation. Since current and collection area are known, the average current density can be determined. Next, the average particle energy is used to determine average velocity (u) per species. Finally, number density is computed from $j = qnu$.

Species	**J** (A/m^2)	**Mass** (u)	**q** (C/e)	**u** (m/s)	**n** (m^{-3})
I_{D+}	0.915	2	1	4.00×10^6	1.43×10^{12}
I_{T+}	0.57	3	1	3.27×10^6	1.09×10^{12}
I_{He2+}	0.026	4	2	4.00×10^6	2.03×10^{10}

Table 4.1: Calculated plasma properties for each species.

Next, the temperature is approximated. Due to the expander used in this example, $T_\perp = 0$ is assumed. As for the axial direction, Barr et al. state that incident particles had a minimum energy cutoff of 90 keV, which is 77 keV below the mean energy of 167 keV. As previously stated, a Maxwellian distribution is a normal distribution whose standard

deviation (σ) is the characteristic thermal velocity (v_{th}). For any normal distribution, 68% of samples fall within $\sigma \pm 1$, 95% fall within $\sigma \pm 2$, and 99.7% fall within $\sigma \pm 3$. For this study, $\sigma = 2$ is selected. The temperature is $T \propto v_{th}^2$, so 2 sigma is spanned by 4T. So, the 77 keV span is divided by four, yielding a temperature of 19.25 keV. At this temperature, 95% of particles have energies between 90 and 244 keV. So, only 2.5% of particles are be below the cutoff.

The proposed collector design in Barr, shown in Figure 4.1, featured grid potentials with similar numbers. The first grid is charged to +94 kV, and the last is charged to +243 kV. Any particle whose energy is less than 94 keV is deflected, and lost. This figure is used to specify `Electrode()` and `Domain` objects for the benchmark. The position of each grid had to be shifted to ensure the electron reflector resides at z=0, as described in Section 3.4.2. A grid point spacing of $dz = 0.01$ is selected, requiring 442 grid points.

Populations of super-particles are generated by `super_maxwellian()` for each species using the properties stated in Table 4.1. In total, 1000 super-particles are used: 450 D^+, 450 T^+, and 100 He^{2+}.

Next, a `ZaP-DEC()` object is initialized. This requires the following arguments: a python list of particles, a domain object, and an appropriate time-step. The time-step is chosen such that the fastest particles in the system cross a grid cell in two time-steps. A particle whose energy is 2σ above the mean has 244 keV of energy and a velocity $v_z \approx 6 \times 10^6$ m/s, yielding a time-step of,

$$dt = \frac{1}{2}\frac{dz}{(6 \times 10^6)} \tag{4.4}$$

$$= 4.16 \times 10^{-10} \text{ sec} = 0.416 \text{ ns} \tag{4.5}$$

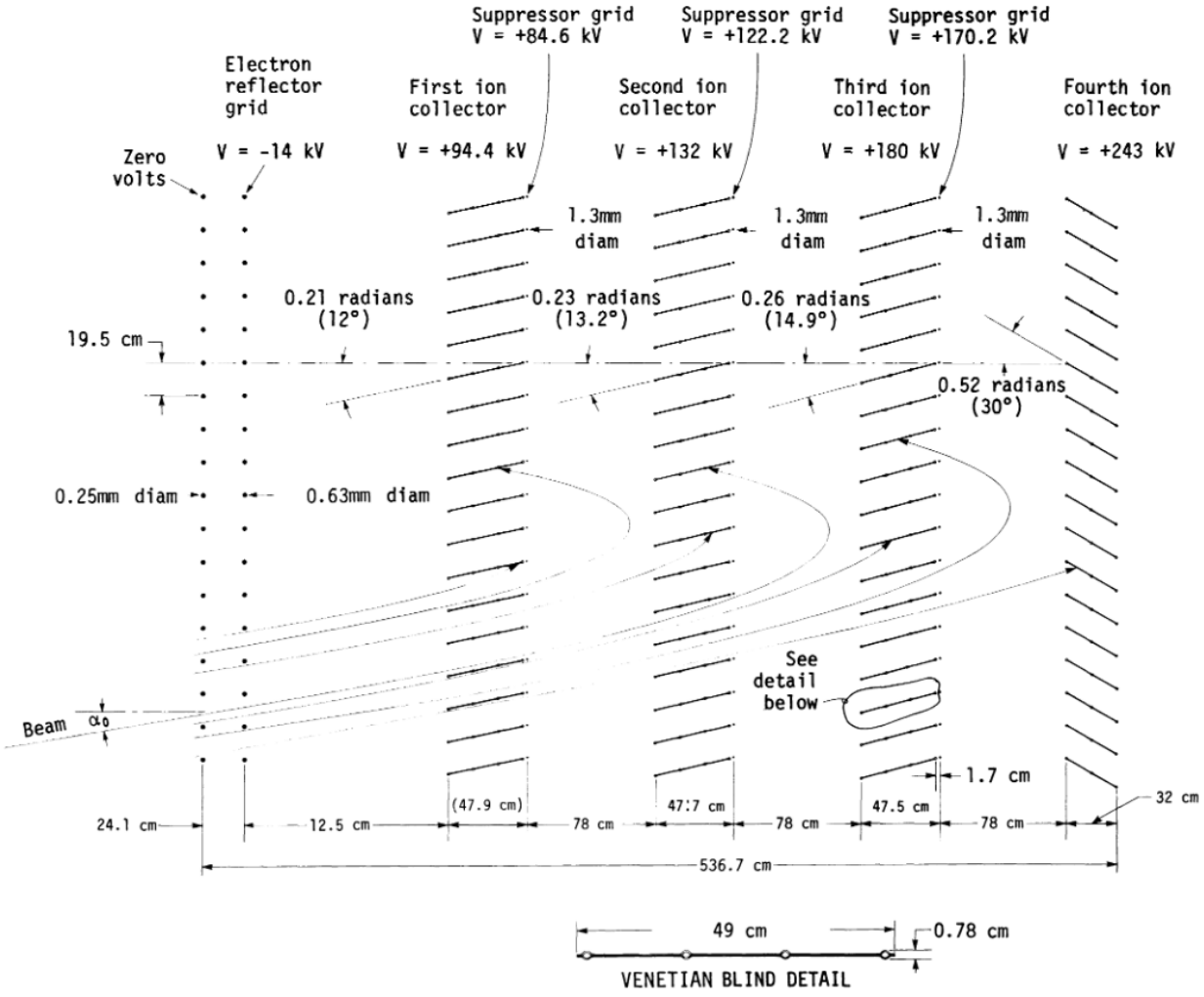


Figure 4.1: Benchmark Case Collector Design from Barr et al[2]. A beam of plasma enters the DEC system at angle α, passing the ground grid before reaching the electron reflector. After electrons are removed, ions follow parabolic trajectories. VB grids are designed to have low opacity to approaching ions, but high opacity to retrograde ions. The fourth ion collector retains a VB design, but is oriented to capture all incident ions. This allows neutral particles to be pumped out of the system.

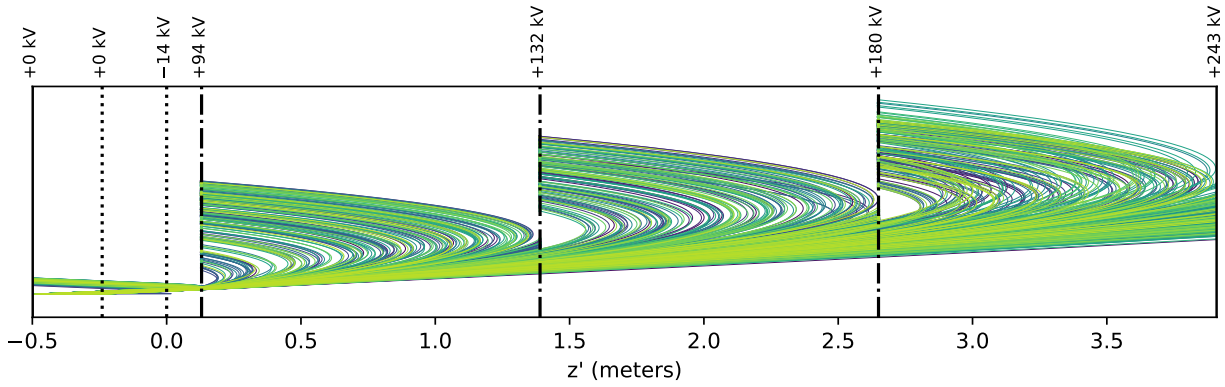


Figure 4.2: Benchmark Case reproducing the design in Figure 4.1. Screen grids are dashed lines, VB 'ribbon' grids are dashed-dotted lines. Particles are initialized between the left wall and the first screen grid at t=0. They travel to the electron reflector (-14 kV) which removes electrons and accelerates ions. Next, particles with energies < 132 kV are reflected and collected by the 94 kV grid. The process repeats with subsequent grids. After each ribbon electrode, a parabolic-shaped gap in ion trajectories is observed. Any particle whose trajectory would fill those gaps must be incident on the adjacent collection grid at a shallow angle, resulting in collection. Some particles are observed to impact the downstream wall because their energy exceeds 243 keV. The predicted collection efficiency is 70.8%.

The results of the simulation are shown in Figure 4.2, with a predicted efficiency of 70.8%. Barr et al. predicted an efficiency of 65% for this design, but that includes losses present in the overall system. If one only includes losses due to the injection angle, interception by non-ideal electrodes, angular spread, parallel motion, and space charge, Barr's stated efficiency is 71.4%. Considering that an approximation is used for the plasma distribution function, a 0.6% error is considered a satisfactory result.

In Section 2.3, it is stated that it is difficult, if not impossible, to employ an expander on the ZaP-HD experiment. In that case, the plasma has a finite temperature in the radial direction. As a numerical experiment, the benchmark case is run again with $T_z = T_\perp =$

Electrode	Type	Voltage, kV	Efficiency
3	ribbon	94	49.5%
4	ribbon	132	64.3%
5	ribbon	180	76.5%
6	wall	243	76.7%

Table 4.2: Collection efficiency at each electrode for the benchmark case with a finite temperature in both directions, $T_z = T_\perp = 19.5$ keV.

19.25 keV. No other changes are made to the design of the grids or plasma parameters. The result, shown in Figure 4.3, is a collection efficiency of 50.0%.

A report of collection efficiency at each electrode (Table 4.2) provides additional insight. Efficiency at the first ribbon grid is close to the overall total efficiency, while later grids gave high efficiency. The obvious conclusion is that the first grid acts as a filter: particles who pass through the first grid must have a trajectory close to the design intent. Thus, their trajectories are be suitable for the downstream grids. One should expect efficiency to decrease as $T_\perp$ increases.

4.2 Application to ZaP-HD

Next, the ZaP-DEC code is applied to predict collection efficiency from the exhaust plume of ZaP-HD. As noted in Section 2.3.1, ZaP-HD lacks an expander, meaning electrons have high energy, and the plume has significant thermal energy in the radial direction. Thus, one should expect results which are far from the ideal case. The intent of the simulation is to determine what can be tested, and how the experimental limitations may be accommodated.

4.2.1 Exhaust Plume Properties

As before, the first step is determining the plasma properties to be simulated. The plasma exhaust of ZaP-HD was recently measured by Jared Smythe for his 2023 master's thesis[30].

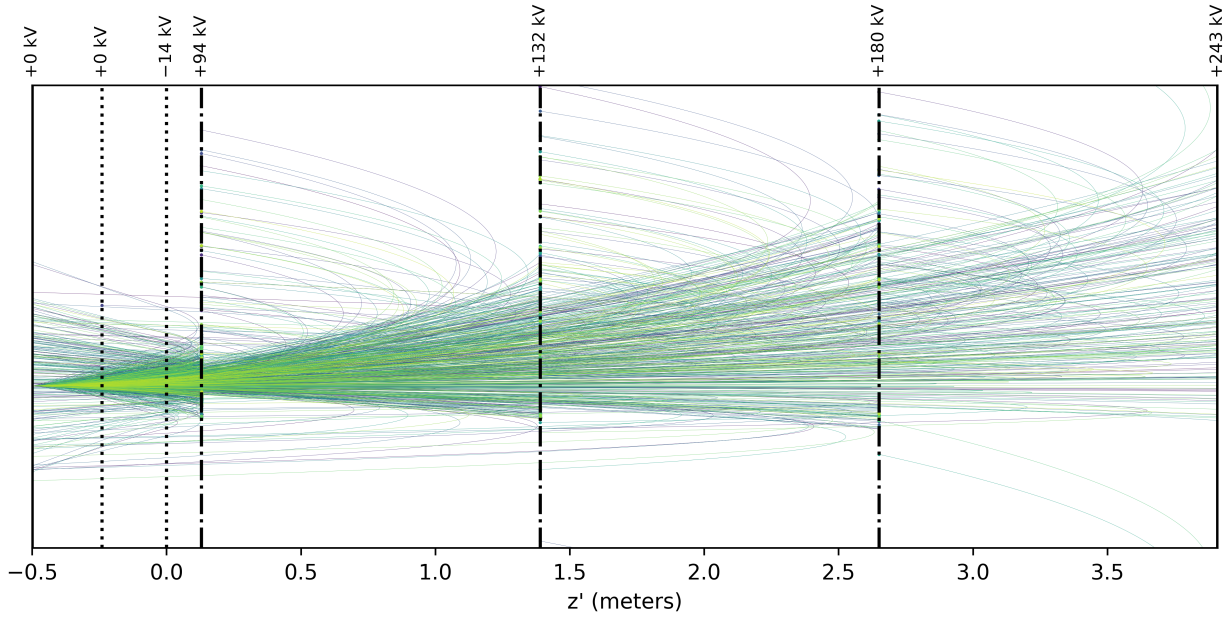


Figure 4.3: Benchmark Case when $T_z = T_\perp$. The variation in vertical velocities reduces efficiency of the first ribbon grid. However, the first ribbon grid acts as a filter which allows ions with optimal trajectories reach downstream grids. As a consequence, the efficiency of downstream grids is high.

Using a Langmuir probe, Smythe measured electron temperature and density while varying the axial and radial location of the instrument. He also varied the end-wall geometry. ZaP-HD's anode (the outer electrode) has an end wall which allows current to travel radially inward to the axis of the device. Current can then travel through the Z pinch to the cathode, where it exits the device.

The original design of ZaP-HD has a closed end-wall which blocks 94% of the cross sectional area of the device. Due to recent interest in studying the SFS Z pinch configuration as a propulsion device, this geometry is problematic as it blocks most of the exhaust flow. So, a new 'Spoked end-wall' was designed which allows radial current flow through the spokes while leaving large openings for plasma to escape. The spokes block 45% of the cross section

	u (km/s)	**Te** (eV)	**ne** (m^{-3})	$\mathbf{W}_z/\mathbf{W}_{tot}$
Closed End-Wall	27 ± 20	60	6×10^{19}	0.27
Spoked End-Wall	47 ± 19	10	1×10^{21}	0.62
No End-Wall	103 ± 11	20	4×10^{19}	0.77

Table 4.3: Plasma Exhaust Properties vs Configuration[30]. Axial energy added by author.

area. It is also possible to form a Z pinch with no end-wall at all, but the downstream end of the pinch must flare radially outward to close the circuit. Pinches with this feature are not compressed over their entire length, so this is detrimental to performance.

Figure 4.4: End wall configurations on ZaP-HD[30]. The spoked end wall is designed to maximize the plasma exhaust area while allowing pinch compression along its full length. Omitting the end wall results in a plasma exhaust with higher axial velocity but poor pinch compression.

Smythe's measurements are summarized in Table 4.3. These lack the nuance of radial or axial variation, and should be taken as a qualitative approximations suitable for a preliminary study such as this one. The 'No end-wall' configuration is selected for further study because it has the highest axial energy and the lowest density. Assuming the exhaust is pure hydrogen gas, a mean velocity of 103 km/sec equates to a mean particle energy of 55 eV.

At this point, it is important to compare these values to the benchmark case in Table

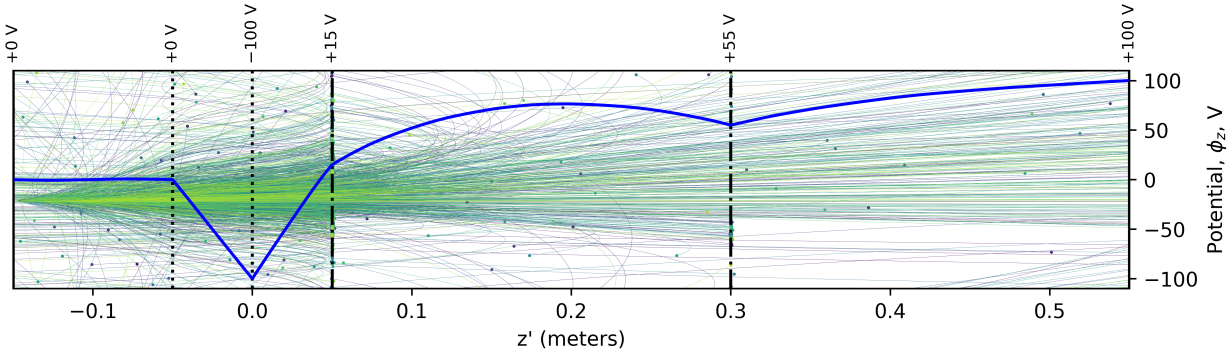


Figure 4.5: VB collection in ZaP-HD. The analysis is paused before collection of all particles to observe space charge effects between the grids. As expected, the space charge limit is exceeded, but higher energy particles still reach the rear wall.

4.1. The benchmark has mean velocities on the order of 10^6 m/s and densities of 10^{12} m^{-3}, while ZaP-HD's exhaust is an order of magnitude slower and seven orders of magnitude more dense. Based upon our understanding of space charge in Section 2.3.2, there are clearly issues with the space charge limit.

4.2.2 Simulation Results

Due to the expected space charge limitations, there is no attempt to design perfectly optimal grid spacing. A design with 3 evenly spaced grids is selected. The total system length is 0.6 meters, which fits in the new Exhaust Chamber discussed in Chapter 5. The central grid is charged to a potential of 55 V because the mean particle energy is $\overline{W} = 55$ eV. In the benchmark case, the DEC system captured particles over a span of energies equal to $\overline{W} \pm 4T$. In this case, since the temperature is 20 eV, this can't be replicated. Instead, the system spans approximately $\overline{W} \pm 2T$: the first grid is charged to 15 V and the downstream grid is charged to 100 V. Since there is no expander, electrons have significant energy, $T_e \approx T_i$. So, the electron reflector grid is charged to -100V. This accelerates the ions significantly, and it reduces space charge limitations in front of the first ribbon grid.

A single test particle is created with 90 eV of energy, and the program is instructed to print the angle of incidence with each grid. This information is used to adjust the angle of each grid before the next run.

Next, 1000 super-particles are generated with the properties stated in Table 4.3 for the No End-Wall case. The time-step is set such that a 95 eV particle crosses one grid space in two time-steps. This is run for 1000 time-steps, and halted to generate Figure 4.5. At this time-point, many particles still have yet to be collected (represented as dots). The potential field is superimposed over the domain to observe space charge effects. As previously predicted, some particles do make it through the elevated potentials. As seen the benchmark case, particles which can reach downstream regions are more likely to have optimal trajectories because the blinds act as filters.

The simulation is then permitted to finish. The final particle trajectories are not much different than Figure 4.5, so it is not repeated. However, the final collection efficiencies are of interest. The total collection efficiency is 24.2%, which is surprisingly high given the conditions. The efficiency per grid increases downstream as expected.

1. Electrode 3, (ribbon), efficiency = 17.8
2. Electrode 4, (ribbon), efficiency = 22.9
3. Electrode 5, (wall), efficiency = 28.3

While this result is attractive, some skepticism is warranted about the validity of the result. Particles are initialized at the beginning of the analysis, and faster particles will race ahead of slower particles. The faster particles, with the highest energy, can therefore outrun some of the space-charge effects. This is less important in a design which is not space-charged limited, but may be important in this case. This is discussed in Chapter 6 as a topic for future work.

4.3 Application to D-T Fusion SFS Z pinch with Q≫1 (D-T)

Recent work by Takagaki[31] has studied the possibility of alpha heating in a SFS Z-pinch fusion reactor. As described in Section 1.2, alpha particles from the reaction can heat the plasma, improving its performance. It was determined that a pinch current of $I_p = 1.35$ MA results in a gain of $Q = 15$, while a pinch current of $I_p = 2.55$ MA results in $Q = 152$.

Interestingly, the alphas may be accelerated to high axial velocity. Alphas are born with high energy, so many of them have high radial velocity. This radial motion interacts with the azimuthal magnetic field by the Lorentz force, $\vec{F} = \vec{j}_r \times \vec{B}_\theta$, causing an axial acceleration of alphas along the Z pinch. Higher axial velocity increases the pinch current, causing B_θ to increase, improving confinement of the alphas. This improved confinement reduces their radial velocity. These predictions show that the alpha particle population exits a SFS Z-pinch reactor with high mean axial velocity. This effect was studied in the context of a propulsion system, so thrust and specific impulse was predicted, shown in Table 4.4

The alpha particles have high energy compared to deuterium and tritium. The thrust prediction makes clear that the majority of the momentum is carried by alphas despite a density 1000 times lower than deuterium or tritium. So, this case was simulated with alpha particles only. The grid voltages can be set high enough that deuterium and tritium will be reflected before they reach the first grid. In comparison to the ZaP-HD case, the high velocity and low density of the alpha particles reduce space charge effects, but the high temperature relative to mean energy are expected to decrease collection efficiency. It is assumed that the electron temperature is close to the temperature of deuterium and tritium, so most are deflected by a 40 keV grid.

Previously in this work, it was noted that the first grid acts as a filter for particle trajectories. So, the voltage of the first grid is critical. By trial and error, the voltages shown in Figure 4.6 were determined to yield efficiencies which are not optimum, but near the maximum. Interestingly, the efficiency is increased when the voltage of the first grid exceeds $\overline{W}_z/Z$, where Z is the charge number. This is sufficient to deflect particles with mean energy.

		α	D	T
m_i	kg	6.69E-27	3.35E-27	5.02E-27
I_{sp}	s	400,000	8,000	8,000
v_e	m/s	3,924,000	78,480	78,480
$\dot{m}$	kg/s	9.2E-04	7.0E-03	1.0E-02
F_{thrust}	N	3,600	550	785
T	keV	700	10	10
$\overline{W}_z$	keV	321	0.064	0.096
n	m^{-3}	3.5×10^{16}	2.7×10^{19}	2.5×10^{19}

Table 4.4: Fusion Z pinch Properties from Figure 5.12 of Yu Takagaki's dissertation, corresponding to pinch current $I_p = 1.35$ MA and $Q = 15$. The mean axial kinetic energy $\overline{W}_z$ is derived from exhaust velocity and ion mass. The linear number density is calculated as $\dot{m}/(m_i v_e)$, then multiplied by a collection area of 1 m^2 to derive number density n. Temperatures are suggested by Takagaki for a Z-Pinch plasma after alpha heating.

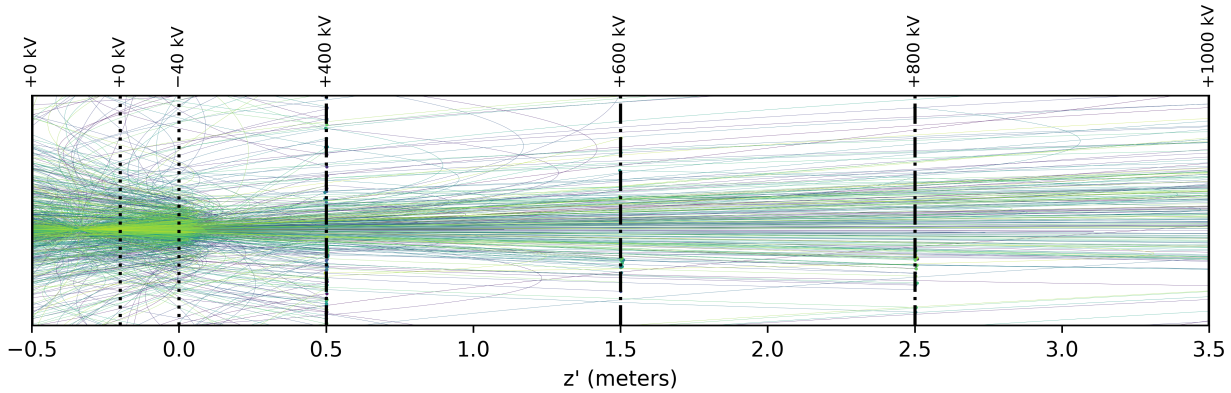


Figure 4.6: Simulation results for alpha particles with properties from Table 4.4. Note that alpha particles have Z=2, so a 400 keV particle can be collected by a 200 kV grid with 100% efficiency. Particles with the mean energy are deflected, which was found to improve collection efficiency. For this case, a collection efficiency of 26.9% is predicted.

Chapter 5

DESIGN OF ZAP-HD EXHAUST CHAMBER

A SFS Z pinch produces a high speed plasma exhaust which can be utilized for thrust, which has been studied on both FuZE[26] and ZaP-HD[29]. At present, the exhaust enters a region called the extension chamber, which is an empty space with diagnostic access for optical instruments and Langmuir probes. This has been sufficient for initial research of the plume, but it has limitations. The extension chamber has a diameter of 12 inches, greatly limiting the experiments which can be installed there. Experiments which can be performed may have erroneous results because the plasma exhaust may impinge on the chamber walls before it can be measured by most instruments. So, it is desirable to build a new extension chamber with a larger diameter. The new design is named the Exhaust Chamber, shown in Figure 5.1.

5.1 Design Goals and Proposed Experiments

The primary design goal for the Exhaust Chamber is to enable new experiments, including direct energy conversion. Direct collection by Venetian blinds is discussed in Section 5.5, but other DEC methods may be attempted in the future. The design of the chamber interior includes stainless steel rails with regularly spaced holes to support a variety of experiments in the future.

There is also interest in studying magnetic nozzles. In particular, how does the plasma behave at the interface between the azimuthal magnetic field of the Z pinch and the magnetic nozzle, which has radial and axial components? ZaP-HD may also be an interesting platform for studying demagnetization of ions in a magnetic nozzle. An optimal location for mounting a magnetic nozzle is the upstream wall of the Exhaust Chamber, so this includes a pattern

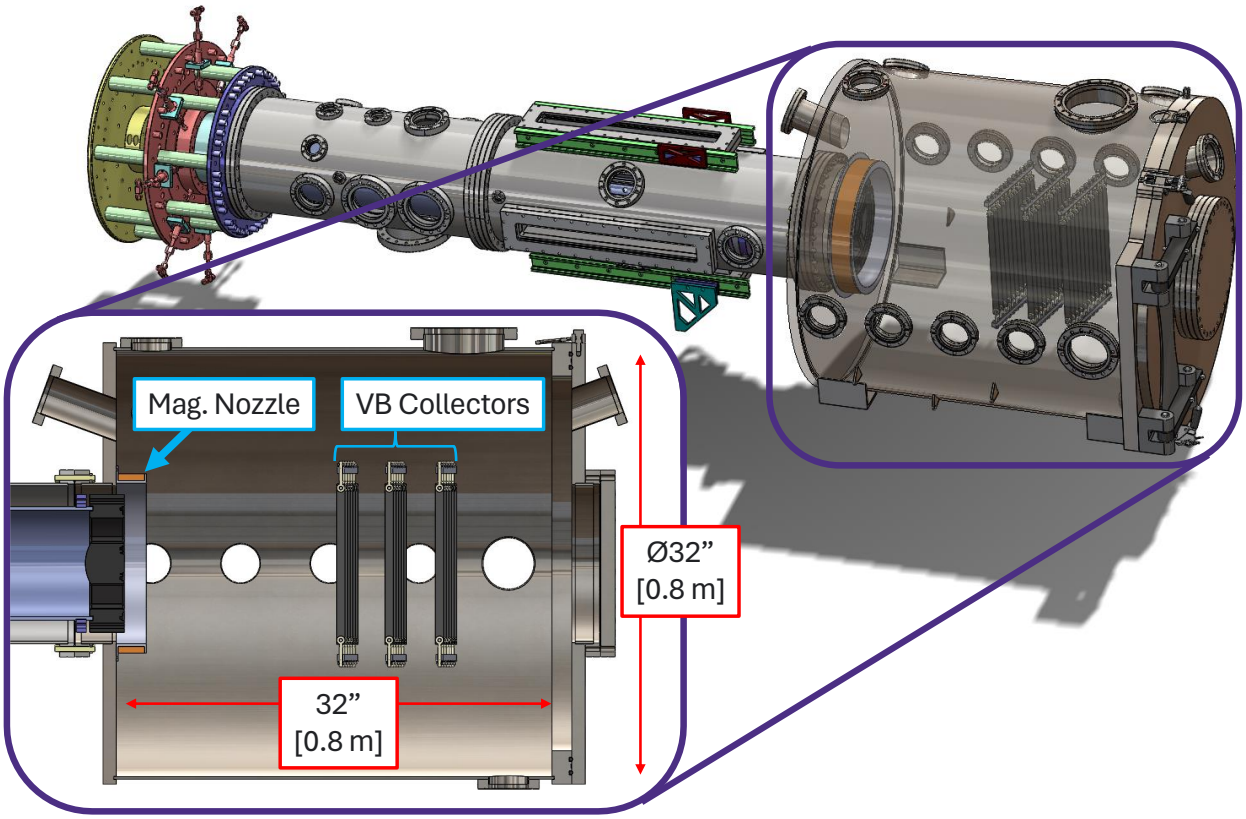


Figure 5.1: CAD Model of Exhaust Chamber attached to the ZaP-HD device, with a detail view of the chambers interior dimensions. Preliminary designs for magnetic nozzles and VB collectors are visible.

of regularly spaced threaded holes.

Secondary design goals include operational improvements. The first is improved access to the interior of the chamber to install experiments including thrust measurement devices, Langmuir probes, retarding potential analyzers, and DEC experiments. At present, access to the chamber interior requires removal of 30 bolts to detach a 14 inch diameter ConFlat flange. The new chamber, with a diameter of 32 inches, exceeds the maximum standard ConFlat size of 16 inches. So, a door and hinge has been designed which swings open for access. A 14 inch diameter ConFlat flange is included in the center of the door to replicate the existing

interface on the current extension chamber. This is done so that existing instruments using that interface are still supported. Due to its complexity, the door design is presented along with its structural analysis in Section 5.4.3.

Another design goal is making ZaP-HD easier to maintain. Occasionally, components inside the machine need to be repaired, or a new design requires major disassembly of the machine. At present, it is difficult to gain access to B-dot probes or the complete nosecone assembly. So, a rail system has been designed to support the Exhaust Chamber. After the chamber is brought to atmospheric pressure, fasteners connecting the 12-inch diameter vacuum chamber sections may be removed, and the aft section can translate with the Exhaust Chamber on rails.

Finally, a tertiary design goal is a desire to use our lab space more efficiently. Many of our diagnostics require the use of lasers which pass through or over the machine. The beam paths are routed by optical elements mounted to optics tables, which take a lot of space. To rectify this, the chamber weldment has anchor points for optical breadboards. The breadboards can pivot about the upper anchor points, but they can be secured and leveled using turnbuckles (Figure 5.2). When not in use, a quick-release pin may be removed and they fold out of the way. One should note that a disadvantage of this design is its relatively poor isolation from vibration caused by vacuum pumps. The existing optics tables are mounted to the floor and include vibration isolation supports to ensure stability of the optical components mounted to them. The optical breadboards lack these features, so they may not be useful for applications which require optical stability.

Another example of efficient use of lab space is the new location of the turbomolecular pump (hereafter referred to as a turbopump). The relatively large volume of the new chamber made it desirable to move the turbo-pump from the upstream chamber to the Exhaust Chamber to minimize vacuum conductance losses. By situating the turbopump on top of the chamber, the volume underneath the machine can be utilized for optical beam paths, such as for Schlieren imaging. It also keeps harnessing and plumbing above the machine and out of the way of workers in the lab. A final benefit is that it should reduce the amount of

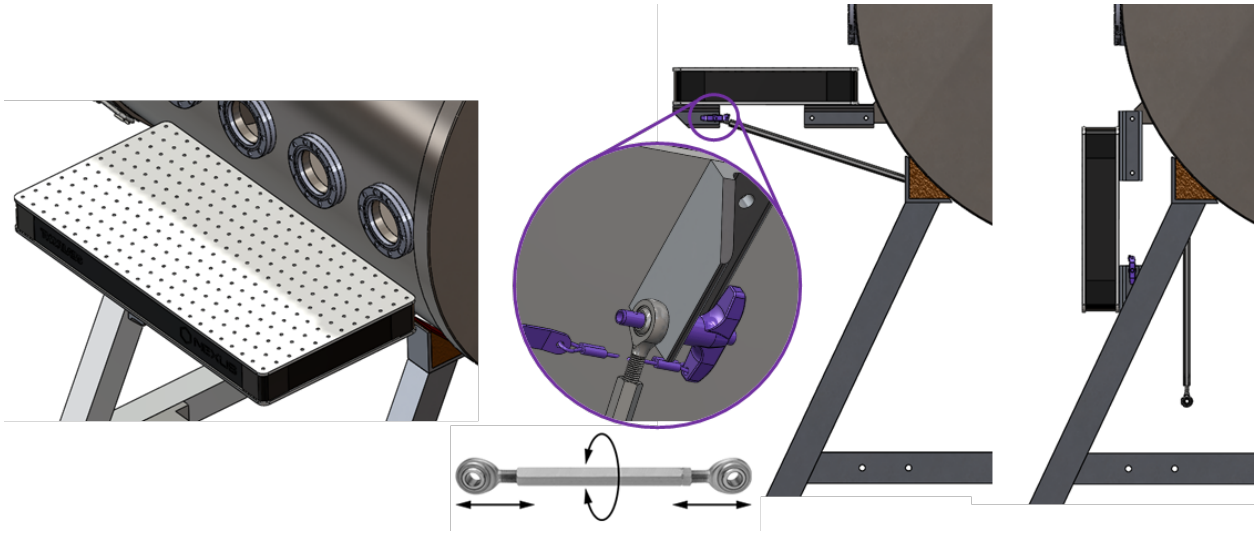


Figure 5.2: Deployable optical breadboards to conserve lab space

debris ingested by the turbopump.

All previous functionality from the old Extension Chamber had to be maintained such as ConFlat ports for Langmuir probes, electrical pass-throughs, and Ion-Doppler Spectroscopy (IDS) telescopes. The telescopes in particular presented a design challenge. Their first constraint is that they must be mounted at an angle relative to z-axis of the ZaP-HD device. The second constraint is that line of sight from ConFlat to the axis should be identical between both telescope locations so that a common telescope design can be used. Due to packaging constraints, this forced one of the telescopes to be mounted on the door.

5.2 Facility Constraints

The largest constraint on the design is the facility. ZaP-HD is situated in a room whose doors are 34.625 inches wide, and removing a wall is not an option. The chamber weldment is designed to be 34 inches wide, clearing the doorway by 5/16 inch per side. The UW AA department has only one mobile lifting device which can fit through the doorway and navigate around the machine to install a chamber: a Genie SLA-10 Superlift^{TM}. The SLA-

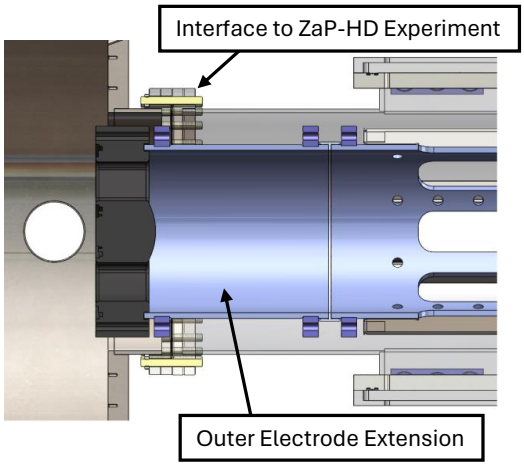


Figure 5.3: ZaP-HD interface and the Outer Electrode Extension. The interface consists of a 14" diameter ConFlat flange that requires 30 all-thread fasteners and nuts. The electrode extension increases the length of Z-Pinches produced by the ZaP-HD device.

10 has a rated load capacity of 1000 lbs, but this capacity decreases linearly after the load center exceeds 24 inches. Therefore, this placed a simultaneous constraint on the length and mass of the chamber weldment with the door and all ConFlats removed.

5.3 Mechanical Interfaces

The Exhaust Chamber interfaces to ZaP-HD's 14 inch diameter downstream ConFlat flange, which includes 30 threaded holes. Therefore, the mating interface on the Exhaust chamber must have 30 clearance holes. They are fastened together using silver-plated stainless steel all-thread, as shown in Figure 5.3. To accommodate this interface, the Exhaust chamber is required to have a short 12 inch diameter tube welded on. The end-wall on ZaP-HD's outer electrode terminates upstream of this interface, which defeats one of the primary design goals for the chamber. So, ZaP-HD requires an extension to its outer electrode. This means that ZaP-HD must produce longer pinches than before, and its performance will change.

When the door of the chamber is open, its weight exerts significant torque. Unfortunately,

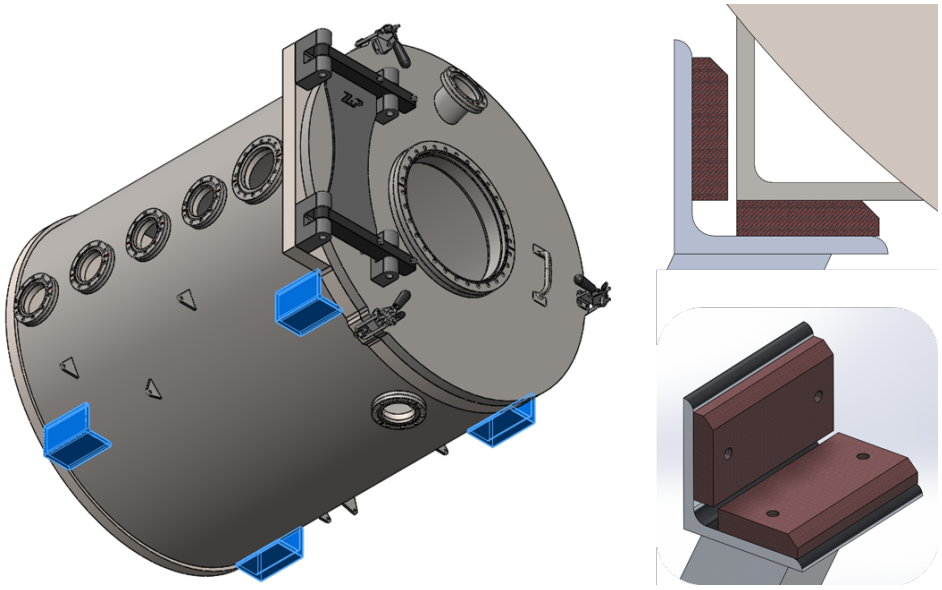


Figure 5.4: Exhaust Chamber's interface to frame. Left: Welded angle-irons shown in blue. Top right: Angle-irons on phenolic blocks. Bottom right: Blocks installed to frame.

the rest of the ZaP-HD experiment is unconstrained against axial rotation; it sits on a phenolic V-Block interface. So, the exhaust chamber required a structural interface which can react this torque. This is accomplished by welding four angle-irons to the bottom of the chamber. These interface with phenolic blocks which are fastened to the frame.

The purpose of the phenolic (on the ZaP-HD device and the Exhaust Chamber) are to prevent an electrical path to ground through the frame. Instead, the experiment is grounded to a 'star ground' which prevents ground loops. Due to electromagnetic induction, current can be produced in ground loops which is a source of noise for scientific instruments. The phenolic blocks are also the 'shim' interface for the system. During the first installation of the chamber, misalignment between the the ConFlat interfaces will be measured, then phenolic blocks will be machined to a precise thickness. If too much is machined, shim stock can be added between the blocks and the frame.

The final external interface is between the frame and the floor (Figure 5.5). As previously

Figure 5.5: The Exhaust Chamber frame is guided by a rail. The chamber can be removed from the ZaP-HD device without use of a crane, improving safety while reducing the time required to maintain or repair the device.

discussed, the frame is designed to interface with a rail. The rail is a simple angle iron welded to a flat plate which is bolted to the floor. V-groove wheels ride on the angle-iron, providing lateral shear support. Using a single rail is easier to align during installation. As a consequence, the frame does not sit level unless the rail is installed. This frame design is preliminary, and will change after publication of this work. The next design iteration will support the weight of both the exhaust chamber and the assembly region chamber to improve machine maintenance.

5.4 Structural Analysis

Structural analysis for the Exhaust Chamber includes separate FEA models for the weldment and door hinge, while traditional analysis is performed via spreadsheet for pins and fasteners. This section covers each analysis which is performed.

5.4.1 Safety Factors and Stress Allowables

It is critical that the design is safe for students, staff, faculty, and the public. Safety factors from NASA-STD-5005D Section 5.1.2 were selected requiring a Factor of Safety (FoS) of 2 against detrimental yielding, and a FoS of 3 against collapsing, buckling, exceeding the ultimate load, or failing to support the design load.

The chamber is manufactured from annealed 304 Stainless Steel. A-basis stress allowables are retrieved from the Metallic Materials Properties Development and Standardization (MMPDS) Handbook, 10th edition. This is an aerospace industry standard source for material properties recognized by the FAA, DoD, and NASA. These allowables, modified by the safety factors, are shown in Table 5.1. The driving requirement is the yield case, requiring tensile stresses below 13 ksi.

	Property (ksi)	**FoS**	**Allowable**
Fty [ksi]	26	2	13.0
Ftu [ksi]	73	3	24.3
Fsu [ksi]	50	3	16.7

Table 5.1: Stress allowables after application of safety factors. Fty, Ftu, and Fsu refer to the tensile yield, tensile ultimate, and shear ultimate stresses of the material, respectively. When the design load is applied, predicted stresses must be less than the stress allowable.

5.4.2 Chamber Stress and Stability Analysis

A finite element model (FEM) is used to predict stress and stability of the Exhaust Chamber (see figure 5.6). This required creation of a separate CAD model which contains simplified geometry for many components. Fillet radii and chamfers are removed because they increase mesh density and none of them are intended to reduce stress concentrations. The door hinge interface is simplified to reduce mesh density in that region. O-ring grooves, and other

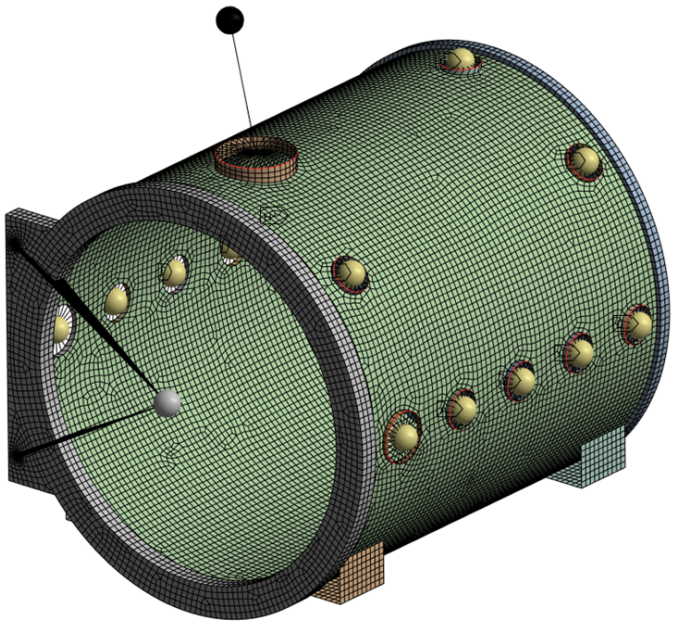


Figure 5.6: Finite element mesh of Exhaust Chamber. Sheet-metal skins are modeled using parabolic shell elements, while the door flange and upstream wall are modeled with hex-dominant parabolic solid elements. Components are abstracted using point masses attached with rigid elements.

channels intended to aid in manufacturability, are deleted.

The majority of the Exhaust Chamber is a thin-walled vessel, meaning thickness $t \ll L$, where L can be the circumference or the length of the vessel. So, the chamber is modeled predominantly with parabolic shell elements using ANSYS Mechanical. Most of the chamber skin is manufactured from 3/16 inch thick sheet metal, but thickness differs in select locations. For example, the angle-irons are 1/4 inch thick, and each ConFlat location requires a welded tube whose thickness varies depending on ConFlat diameter. Wherever sheet bodies intersect, 'share topology' is used to merge nodes, efficiently welding the bodies together. The mesh density for most shell elements is 0.5 inches.

The chamber's upstream wall and downstream door flange are both 1 inch thick. At a

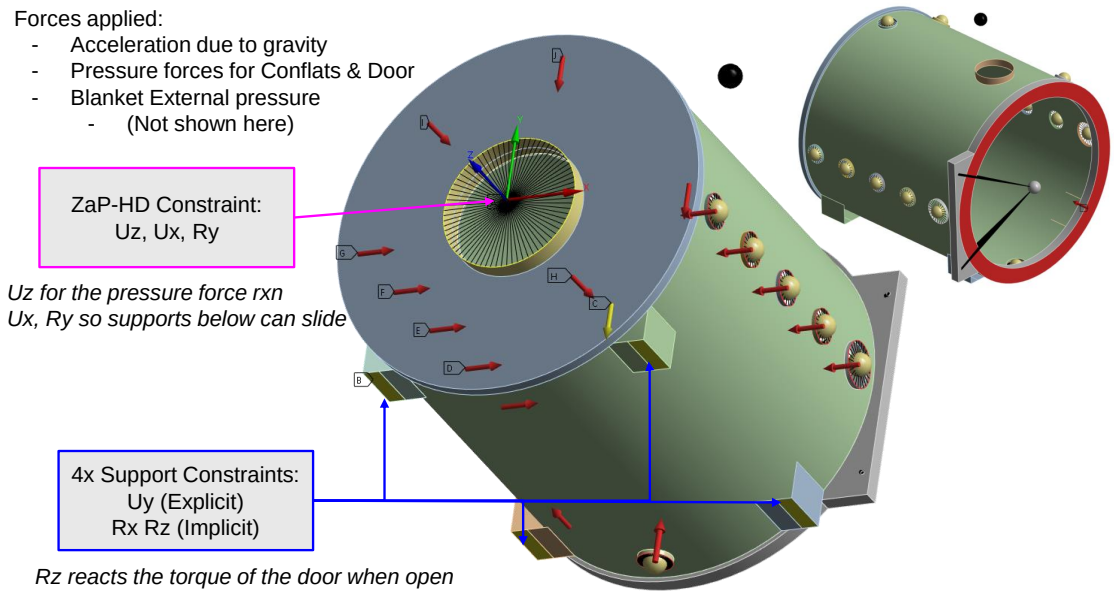


Figure 5.7: Loads and constraints applied to the Exhaust Chamber FEM. The ZaP-HD constraint corresponds to the 14-inch diameter ConFlat interface, and it reacts two translations and one rotation. Four supports on the bottom of the chamber are explicitly fixed in the vertical direction, resulting in two implicit rotational constraints. One atmosphere is applied to exterior surfaces, and a pressure $\times$ area force is applied to the flange of each aperture.

mesh density of 0.5 inches, use of solid elements is justified for these components. So, a solid, hex-dominant mesh with parabolic elements is selected. Interfaces between solid and shell elements are bonded together, but nodes are not merged.

There are many other components with significant mass, but whose geometry is unimportant for the stress model. For example, ConFlat flanges. In the CAD model, a solid ConFlat 'blank' is added to every ConFlat flange, and the coordinates of the center of gravity (CG) for each flange+blank assembly is extracted. These coordinates are used to populate the FEM with point masses. Rigid multi-point constraints (MPCs) are used to attach each point mass to its associated welded tube in the assembly. These rigid elements provide significant radial

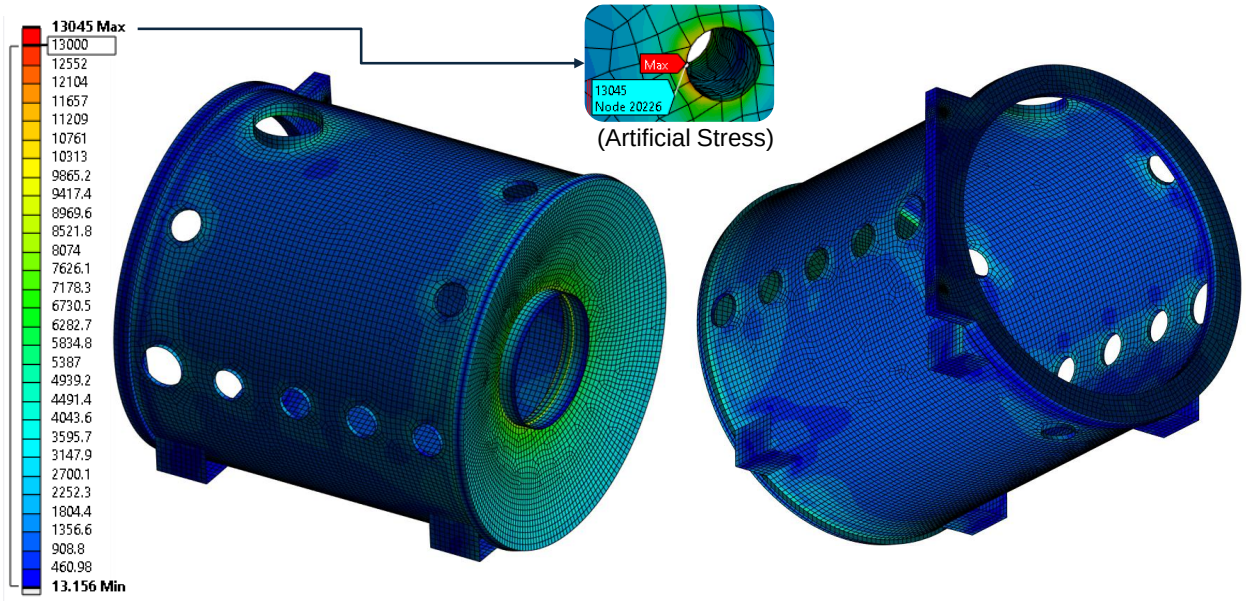


Figure 5.8: Chamber stress predictions are lower than stress allowables with the exception of artificial stresses at the hinge joint. Artificial stresses are caused by simplifications of the finite element model. Analysis of shear pins and bolted joints is performed separately to resolve this issue.

stiffness to the welded tubes, but this is appropriate due to the comparatively high radial stiffness of ConFlat flanges.

A similar process is followed to represent the turbopump, gate-valve, and the door. The turbopump and gate valve are installed together as an assembly, and their combined CG is represented as a single mass element connected with MPCs, shown as the black sphere in Figure 5.6. The door is also represented as a point mass, shown as the grey sphere. This mass includes the door itself, the hinges, clamps, the handle, the ConFlat telescope mount, and a central 14 inch diameter conflate flange and blank. Rigid elements connect it to the weldment's door flange at two points. As described in Section 5.4.3, the door hinge is actually fastened by a combination of bolts and pins, but that level of detail is modeled elsewhere. The simplified geometry is sufficient to approximate the load-path so that skin stresses and

stability may be evaluated.

After creation of the mesh, loads and constraints are defined, shown in Figure 5.7. The mechanical interfaces, described in 5.3, are carefully abstracted. The interface to ZaP-HD is a 14 inch diameter ConFlat. Rigid MPCs connect the welded tube to a remote point to which constraints are applied. The point is fixed in the axial (z) and transverse (x) directions, but not vertically (y). During assembly, the exhaust chamber is supported vertically by the frame before it is bolted to ZaP-HD, so it is assumed that its weight is not carried by ZaP-HD. In reality, some small vertical load is applied to the ZaP-HD device's 14" ConFlat flange, but for the purposes of this model it is conservative to maximize the load flowing through though the thin skin of Exhaust Chamber. This constraint also fixes rotation about the y-axis (yaw), which is necessary to fully constrain the model.

An explicit vertical constraint is added to each of the angle irons on the bottom of the chamber. The constraint area is smaller than the real design so that a conservative prediction can be made for adjacent weld stresses. At the assembly level, this results in an implicit constraint on rotation about the x (pitch) and z (roll) axes, so all 6 degrees of freedom (DoF) are accounted for. Horizontal constraints are not added because the chamber is free to expand and contract in those directions. In reality there is friction, but modeling friction requires a nonlinear model.

The pressure force acting on the door and each ConFlat blank are calculated via a spreadsheet and applied to the edge of each aperture. For the door, it is applied to the contact area between the door and flange. For all remaining external surfaces, atmospheric pressure is applied. A vertical acceleration of 1 gravity is applied to the whole model.

Results of the stress analysis are presented in Figure 5.8. Recalling the stress allowable of 13 ksi from Section 5.1, it is shown that the structure is much stronger than required. The only exceedance exists at one of the door hinge attach points. From the analysis in Section 5.4.3, this is proven to be an artificial stress. For conservatism, this analysis is repeated with different constraints: the angle iron interfaces to the frame are fixed in all directions. This did not result in a meaningful change to the stresses.

However, stress is typically not the driver in the design of a vacuum vessel. The larger concern is typically buckling, which is analyzed next. A linear buckling analysis is performed with the same loads and constraints applied as above. The first buckling mode, shown in Figure 5.9, is predicted to occur at 16.5 external atmospheres, which far exceeds our safety requirement of 3 atmospheres.

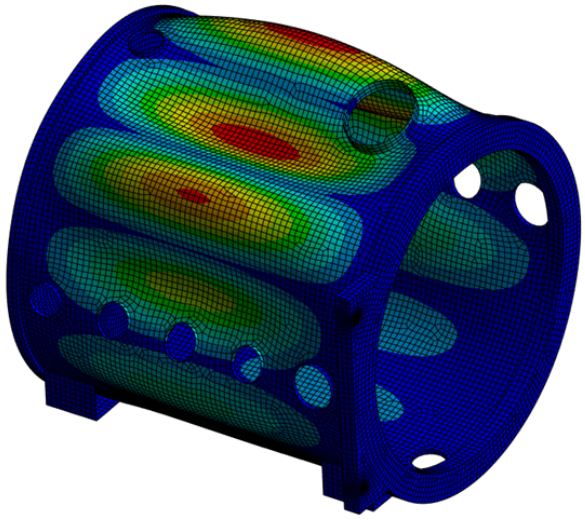


Figure 5.9: Linear buckling analysis produces eigenvectors representing the shape of buckling modes, and associated eigenvalues representing the load factor at which instability occurs. All loads and constraints from Figure 5.7 are applied for the analysis. The first eigenvalue is 16.5, meaning this buckling shape is predicted at 16.5 atmospheres and gravities.

5.4.3 Door Hinge Fastener Analysis

The door hinge design is the most mechanically complex feature of the Exhaust Chamber design. As shown in Figure 5.10, the door features a double hinge. The outer hinge allows rotations exceeding 180°, while the inner hinge has a small but adjustable range of motion. The inner hinge allows the door to rotate independently from the outer hinge so that the

sealing o-ring can be evenly compressed over the full range of o-ring compression states. Without this feature, the o-ring sees more compression on the side adjacent to the hinge.

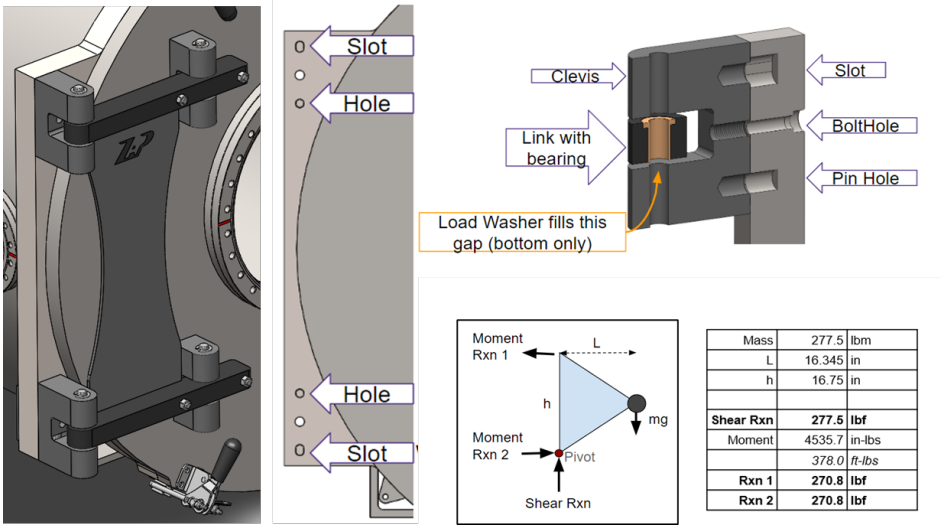


Figure 5.10: Hinge design and pin Sizing. The hinge joints carry tension loads with fasteners and shear loads with pins. Pin-hole joints carry shear in two directions, while pin-slot joints carry shear in only one. Free body diagrams are used to derive fastener loads in agreement with previous FEA predictions. Calculations are performed in a spreadsheet to determine minimum fastener sizing.

Since the door is heavy and may be moved often, it is important that the joint is well defined and that it won't slip over time. It is also safety critical, as a failed hinge joint can result in injury. Figure 5.10 shows a free body diagram which is used to calculate the reaction forces for each clevis. When the door is closed, or open by 180°, moment reactions appear as shear forces, driving pin selection. When the door is open by 90°, the reactions are tension forces, driving bolt selection. The hand-derived reaction forces are compared to those extracted from the FEA results in Figure 5.8 and found to be in close agreement. The inner and outer hinges have identical interfaces except that bolts install from the outside for the inner hinge. Loads are higher on the outer hinge, so it is the only one which is analyzed.

Each clevis is supported by one tension fastener and two shear pins. One shear pin interfaces with a hole, and the other to a slot. The slot is critical because it prevents the design from becoming over constrained and unmanufacturable. The pin-hole interface constrains two DoF (both rotations), the pin-slot constrains one DoF, and the tension fastener works with mating faces to constrain three DoFs. Horizontal shear is divided between two pins, while vertical shear is carried by only the pin-hole joint. So, the root sum square of the vertical shear and half the horizontal shear determines the pin load.

Shear and bearing allowables are calculated for every standard pin size between 1/4 and 1/2 inches. After applying the safety factors from Section 5.1, the minimum pin size is found to be 3/8 inch. These pins provide enough margin for a 250 lb person to do a pull-up on the door while maintaining FoS=3 against failure. However, it takes little space to buy more margin, so larger pins may be selected for the final design.

Fastener margins are calculated using the commercial software MechaniCalc which utilizes the aerospace industry standards detailed in Shigley's[6]. Lubricated 5/16-24 bolts are analyzed with an install torque of 67.5 in-lbs. Results of this analysis can be found in Appendix A. The driving case is thread shear in the parent material, as annealed stainless has low shear strength. Most of the shear force is caused by preload, but a preload of at least 10% of bolt yield is recommended to prevent loss of torque. The result is FoS=3 with the door weight alone. It is desirable to match the failure load of the shear pins, so a thread insert such as a Heli-Coil may be used to increase thread shear strength.

5.4.4 Door Hinge Stress Analysis

A FEM is created for the door hinge, whose setup is shown in Figure 5.11. The pin-hole and pin-slots are constrained as previously described. The clevis interfaces transfer the moment reactions through horizontal pin shear, but weight of the door causes as a vertical shear load in the arms. This is transferred to clevises through a planar Oilite^{TM} bearing. These are porous bronze bearings which have absorbed oil to lubricate the joint.

A sheet-metal shear web is welded to each of the hinge arms. Due to warping from the

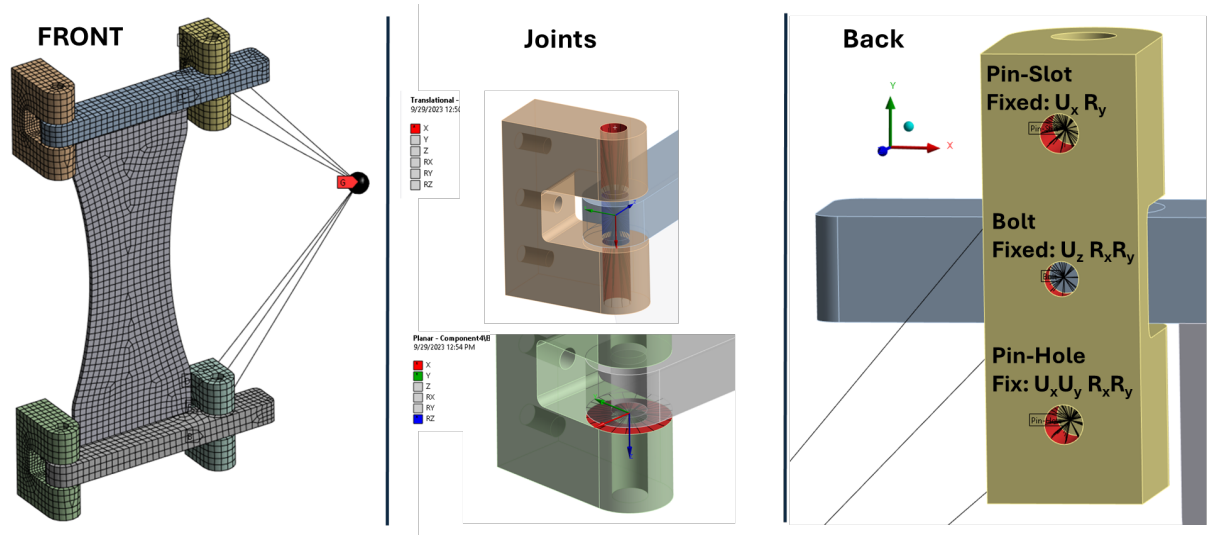


Figure 5.11: Mesh for Hinge FEM. Hex-dominant solid shell elements are used for clevises and arms, while the shear web is modeled with shell elements. Pivot joints had locked rotation to constrain the model.

welding process, the pivot holes will be machined after welding. The shear web is modeled with shell elements and bonded to the arms, modeled with solid elements. The clevises are also modeled with solid elements. The result is low stresses in the hinge (4.3 ksi), deflections of only 0.002" under load, and weld loads of 144 lb/in. This results in a weld stress of 1.6 ksi. Detailed results can be found in Appendix A.

5.5 Implementation of DEC Experiment on Zap-HD Device

In this section, a preliminary design for a VB collection experiment is proposed. As described in Section 4.2.2, this experiment is space charge limited. However, some ions pass through the first grid before it becomes limited, and measurements can be taken to validate the model.

A mechanical design goal is to minimize cost and manufacturing time as much as possible. It is prudent to test something simple on the first attempt. If results are promising, a more

complex design can be pursued.

The design consists of many ribbons within a grid module, and many grid modules within a DEC system. Each grid module is charged to a different potential. It is critical that the position and angle of each ribbon be well defined within a grid module. Similarly, it is important that each grid module can be in a precise position with a precise angle. The ribbons themselves will be made of stainless steel and should be thin (perhaps .005" to .010"), which is apt to warp unless they are under tension. Thin sheets also have low torsional stiffness, so angular constraints must be imposed at both ends of the ribbon.

For the above reasons, the proposed design relies upon a small piece of aerospace hardware referred to as a Positive Index Locking Device (PILD), specified by National Aerospace Standard NAS1193E4C. These are a set of washers with serrated faces that lock together when compressed. One of them locks to a shaft while the other locks to the face of a housing. Traditionally, these are used in aerospace to prevent unwanted rotation of turnbuckles during torque. A turnbuckle changes its length when it is rotated, which is important for setting the trim of control surfaces or adjusting range of motion of mechanisms. But they must be secured by the torque of a nut, which can change the turnbuckle length if it is not locked. In this case, PILDs allow us to tension each ribbon by applying torque to a nut while maintaining angular control of the ribbon.

Each ribbon is assembled into a grid module, shown in Figure 5.13. These modules must be hung from above and supported from below by additional structure which is not shown. The interface to this structure may use additional positive index locking devices on both ends of the module. This allows us to set the beam angle (α) described in Section 2.1.2 and shown in Figures 2.2.

The grid module's horizontal members have castellations which are interfaces to the PILD. This ensures that ribbons are evenly spaced and that their angle is locked relative to the beam. These castellations may be machined at arbitrary angles such that the ribbons are controlled to any angle desired. Tolerances of $\pm 1^\circ$ should be achievable. A full DEC experiment, installed in the ZaP-HD Exhaust Chamber, is visualized in Figure 5.14.

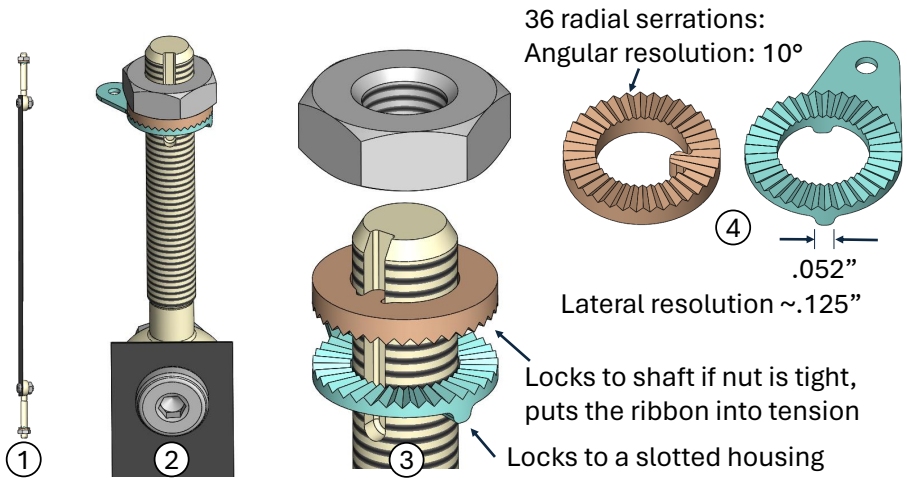


Figure 5.12: (1) A full ribbon, (2) Closeup, (3) Exploded view, (4) PILD detail.

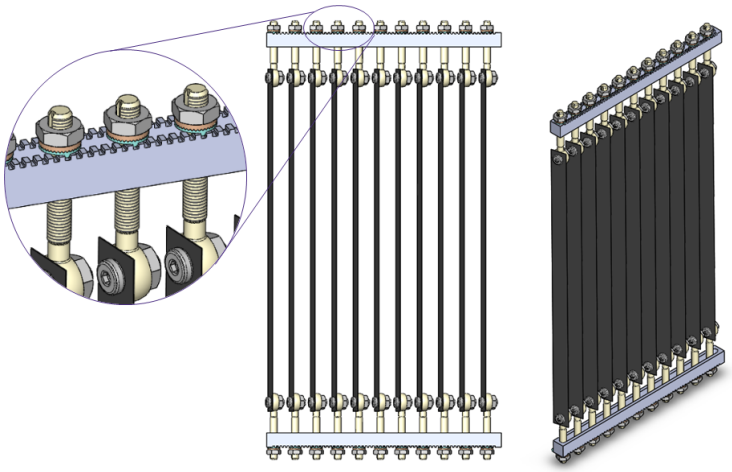


Figure 5.13: A full grid module for a proposed VB Collection Experiment. The detail view shows castellations which control the position and orientation of ribbons in the module.

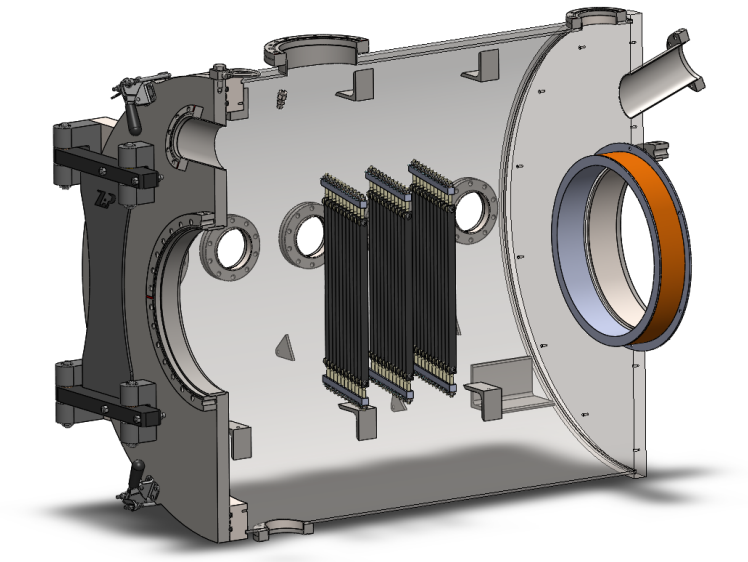


Figure 5.14: CAD model of VB collection experiment in ZaP-HD Exhaust Chamber. An optional magnetic nozzle is shown upstream. An electron reflector screen (not shown) is also required.

Chapter 6

SUMMARY, CONCLUSIONS, AND FUTURE WORK

This work details the development and application of a custom Particle-in-Cell code named ZaP-DEC which was designed to study direct energy conversion by means of Venetian blinds. For each simulation, a user specifies the design of the electrodes, their position, and electric potentials. Next, plasma properties are defined, and the plasma is represented by the program using super-particles. This allows particle collection to be abstracted statistically due to the angular dependant opacity of Venetian blind electrodes. Super-particle trajectories are simulated as they pass through planes representing electrodes, and the energy collected is tabulated. This is compared against the sum of initial kinetic energies for all particles to determine collection efficiency.

ZaP-DEC is verified against a benchmark case in the published literature, and the simulated collection efficiency of 70.8% is within 1% of the benchmark. ZaP-DEC is also applied to simulate energy collection from the exhaust plume of the ZaP-HD device at the University of Washington with a predicted efficiency of 24.2%. A final case simulates collection from the exhaust of a SFS Z-Pinch fusion reactor with a predicted efficiency of 26.9%.

While ZaP-DEC has been numerically verified, it has yet to be experimentally validated. A preliminary design for a Venetian blind collection experiment is presented that controls the orientation and position of each electrode. This experiment is too large to fit in the extension chamber of the ZaP-HD device, so a new chamber design, called the Exhaust Chamber, is proposed. The design goals, requirements, and features of the Exhaust Chamber are detailed, and mechanical analysis is performed to demonstrate that the design meets safety requirements.

6.1 Conclusions

ZaP-DEC is verified against a benchmark case featuring a plasma which is collimated by a magnetic expander, accelerating the plasma axially while cooling the temperature in the perpendicular direction. This enables high collection efficiency, but requires a large physical space. The benchmark case was modified to have a finite temperature in the perpendicular direction to demonstrate the resulting loss of efficiency from 70.8% to 50.0%. However, the modified case uses a temperature which is still a small fraction of the mean axial energy. The ZaP-HD exhaust case and the D-T fusion reactor case exhibit temperatures which are much higher relative to the mean energy, and predicted efficiencies are less than a heat engine. Venetian blind DEC systems clearly require a small transverse temperature relative their mean energy to achieve high efficiencies.

Despite this, the ZaP-HD and D-T Reactor cases show that useful efficiencies may be achievable without an expander. Such systems may be desirable in applications where a heat engine takes too much space, or is unreliable. Incorporating a magnetic expander will require a large volume and necessitate a vacuum chamber, which is heavy and expensive for many terrestrial applications. However, in the vacuum of space, a magnetic expander can be relatively lightweight, and the improved efficiency will reduce spacecraft radiator mass significantly. The collimated exhaust will also improve the thrust efficiency of a fusion space thruster.

6.2 Future Work

ZaP-DEC provides useful predictions, but it can be improved. At present, a user must apply rules of thumb or trial and error to design an efficient collection system. ZaP-DEC would be more useful if it included tools that systematically optimize the design of collection grids for a set of plasma parameters. Analytic solutions to this optimization problem exist in the literature for well-collimated plasmas[2], but cases with finite transverse temperature may require numerical optimization algorithms. One possible approach is an optimization routine

which accepts an initial guess from the user and automatically runs test cases with a small number of particles and a coarse computational grid. It could iterate until a maximum is determined within some reasonable tolerance, then the grid can be refined and the particle count increased. This can be repeated until the algorithm converges on an optimum within a target tolerance.

Another shortcoming with ZaP-DEC's current implementation is that steady-state operation is not simulated. For each simulation, all particles are initialized in the region between the upstream wall and the first grid, where the electric potential energy is zero. These particles have some initial kinetic energy as described by a drifting Maxwellian distribution function, and faster particles race ahead of the slower particles. For the ZaP-HD exhaust case in this study, the simulated time from initialization to collection of all particles requires about 20 microseconds, which is similar to the timescale for each shot of ZaP-HD. For an accurate simulation, particles should be created continuously until collection efficiency reaches a steady-state value, or until the elapsed time exceeds the duration of a shot.

ZaP-DEC should be validated by experiment, and the apparatus and facilities necessary to do so are proposed in this work. However, many details for these experiments have yet to be completed. The Exhaust Chamber and supporting frame require drawings with well-specified tolerances. The Venetian blind experiment requires a support structure with suitable insulators. No design currently exists for an electron reflector grid, which is required for a DEC system to operate.

While a preliminary mechanical design is presented, the associated electrical engineering has not been attempted. An energy conversion experiment will require dedicated electronics to charge each grid to the required voltages and to accurately measure collection efficiency. This is a difficult challenge on a SFS Z-Pinch device due to the very short time-scales over which they operate. Accurate data acquisition systems will be required with temporal resolutions on micro-second timescales. Ideally, these systems should be synchronized with existing diagnostics used on the ZaP-HD device, allowing researchers to correlate the collected current with other measurements.

BIBLIOGRAPHY

[1] I. F. W. Aston. The Mass-Spectra of Chemical Elements. *Philosophical Magazine and Journal of Science*, 36:611–625, 1920.

[2] W. L. Barr, R. J. Burleigh, W. L. Dexter, R. W. Moir, and R. R. Smith. A preliminary engineering design of a "Venetian blind" direct energy converter for fusion reactors. *IEEE Transactions on Plasma Science*, 2(2):71–92, June 1974. Conference Name: IEEE Transactions on Plasma Science. doi:10.1109/TPS.1974.6593737.

[3] W. L. Barr, B. C. Howard, and R. W. Moir. Computer Simulation of the Periodic Electrostatic Focusing Converter. *IEEE Transactions on Plasma Science*, 5(4):248–258, December 1977. Conference Name: IEEE Transactions on Plasma Science. URL: https://ieeexplore.ieee.org/document/4317060, doi:10.1109/TPS.1977.4317060.

[4] Hans A. Bethe. Energy Production in Stars. *American Scientist*, 30(4):243–264, 1942. Publisher: Sigma Xi, The Scientific Research Society. URL: https://www.jstor.org/stable/27825955.

[5] Charles K. Birdsall. *Plasma physics via computer simulation*. McGraw-Hill, New York, 1st edition edition, January 1985.

[6] Richard Budynas and Keith Nisbett. *Shigley's Mechanical Engineering Design*. McGraw Hill, New York, NY, 10th edition edition, January 2014.

[7] Francis Chen. *Introduction to Plasma Physics and Controlled Fusion*. Springer, New York, NY, 3rd ed. 2016 edition edition, December 2015.

[8] Rodger Dyson, et al. Nuclear Electric Propulsion Brayton Power Conversion Working Fluid Considerations. Cleveland, OH. NTRS Author Affiliations: Glenn Research Center, Los Alamos National Laboratory, Analytical Mechanics Associates (United States), National Aeronautics and Space Administration NTRS Report/Patent Number: E-20018 NTRS Document ID: 20220002293 NTRS Research Center: Glenn Research Center (GRC). URL: https://ntrs.nasa.gov/citations/20220002293.

[9] Wenzhe Fa and Ya-Qiu Jin. Quantitative estimation of helium-3 spatial distribution in the lunar regolith layer. *Icarus*, 190(1):15–23, September 2007. URL: https://www.sciencedirect.com/science/article/pii/S0019103507001285, doi:10.1016/j.icarus.2007.03.014.

[10] Eleanor G. Forbes, Uri Shumlak, Harry S. McLean, Brian A. Nelson, Elliot L. Claveau, Raymond P. Golingo, Drew P. Higginson, James M. Mitrani, Anton D. Stepanov, Kurt K. Tummel, Tobin R. Weber, and Yue Zhang. Progress Toward a Compact Fusion Reactor Using the Sheared-Flow-Stabilized Z-Pinch. *Fusion Science and Technology*, 75(7), July 2019. Institution: Lawrence Livermore National Lab. (LLNL), Livermore, CA (United States) Number: LLNL-JRNL-811187 Publisher: American Nuclear Society. URL: https://www.osti.gov/pages/biblio/1632373, doi:10.1080/15361055.2019.1622971.

[11] Jeffrey P. Freidberg. *Plasma Physics and Fusion Energy*. Cambridge University Press, Cambridge, 1st edition edition, August 2008.

[12] S. J. Gitomer. A Direct Convertor Based upon Space Charge Effects. *IEEE Transactions on Plasma Science*, 5(1):18–22, March 1977. Conference Name: IEEE Transactions on Plasma Science. doi:10.1109/TPS.1977.4316998.

[13] Dan M. Goebel, Ira Katz, and Ioannis G. Mikellides. *Fundamentals of Electric Propulsion, Second Edition*. Wiley, Hoboken, New Jersey, 2nd edition edition, November 2023.

[14] Elliot L Claveau. *Axial Evolution of a Sheared-Flow-Stabilized Z Pinch*. Thesis, 2020. Accepted: 2020-10-26T20:38:43Z. URL: https://digital.lib.washington.edu:443/researchworks/handle/1773/46361.

[15] R.M. Mayo, R.L. Mills, and M. Nansteel. On the potential for direct or MHD conversion of power from a novel plasma source to electricity for microdistributed power applications. *IEEE Transactions on Plasma Science*, 30(4):1568–1578, August 2002. Conference Name: IEEE Transactions on Plasma Science. doi:10.1109/TPS.2002.804170.

[16] George H. Miley. *Fusion Energy Conversion*. American Nuclear Society, January 1976.

[17] R. W. Moir and W. L. Barr. "Venetian-blind" direct energy converter for fusion reactors. *Nuclear Fusion*, 13(1):35, January 1973. URL: https://dx.doi.org/10.1088/0029-5515/13/1/005, doi:10.1088/0029-5515/13/1/005.

[18] R. W. Moir, W. L. Barr, and G. H. Miley. Surface requirements for electrostatic direct energy converters. *Journal of Nuclear Materials*, 53:86–96, September 1974. URL: https://www.sciencedirect.com/science/article/pii/0022311574902256, doi:10.1016/0022-3115(74)90225-6.

[19] M. L. Oliphant, P. Harteck, and Rutherford. Transmutation Effects observed with Heavy Hydrogen. *Nature*, 133(3359):413–413, March 1934. Publisher: Nature Publishing Group. URL: https://www.nature.com/articles/133413a0, doi:10.1038/133413a0.

[20] M. L. E. Oliphant and Lord Rutherford. Experiments on the Transmutation of Elements by Protons. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 141(843):259–281, 1933. Publisher: The Royal Society. URL: https://www.jstor.org/stable/96218.

[21] T. A. Oliphant, F. L. Ribe, and T. A. Coultas. Direct conversion of thermonuclear plasma energy by high magnetic compression and expansion. *Nuclear Fusion*, 13(4):529, August 1973. URL: https://dx.doi.org/10.1088/0029-5515/13/4/006, doi:10.1088/0029-5515/13/4/006.

[22] Bryan Palaszewski. Atmospheric Mining in the Outer Solar System. *41st AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*, 2005. URL: https://www.academia.edu/1776497/Atmospheric_Mining_in_the_Outer_Solar_System.

[23] Cecilia Payne-Gaposchkin. *Stellar atmospheres; a contribution to the observational study of high temperature in the reversing layers of stars*. PhD thesis, The Observatory, Cambridge, Mass, 1925.

[24] L. J. Perkins, G. H. Miley, and B. G. Logan. Novel fusion energy conversion methods. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 271(1):188–196, August 1988. URL: https://www.sciencedirect.com/science/article/pii/016890028891145X, doi:10.1016/0168-9002(88)91145-X.

[25] Marek Rubel. Fusion Neutrons: Tritium Breeding and Impact on Wall Materials and Components of Diagnostic Systems. *Journal of Fusion Energy*, 38(3):315–329, August 2019. doi:10.1007/s10894-018-0182-1.

[26] U. Shumlak. Z-pinch fusion. *Journal of Applied Physics*, 127(20):200901, May 2020. Publisher: American Institute of Physics. URL: https://aip.scitation.org/doi/10.1063/5.0004228, doi:10.1063/5.0004228.

[27] U. Shumlak and C. W. Hartman. Sheared Flow Stabilization of the m=1 Kink Mode in Z Pinches. *Physical Review Letters*, 75(18):3285–3288, October 1995. Publisher: American Physical Society. URL: https://link.aps.org/doi/10.1103/PhysRevLett.75.3285, doi:10.1103/PhysRevLett.75.3285.

[28] U Shumlak, H S McLean, B A Nelson, A Schmidt, E L Claveau, E G Forbes, R P Golingo, D P Higginson, A D Stepanov, K T Tummel, T R Weber, and Y Zhang. A Compact Fusion Device based on the Sheared Flow Stabilized Z-Pinch, August 2017.

[29] U. Shumlak, B. A. Nelson, E. L. Claveau, E. G. Forbes, R. P. Golingo, M. C. Hughes, R. J. Oberto, M. P. Ross, and T. R. Weber. Increasing plasma parameters using sheared flow stabilization of a Z-pinch. *Physics of Plasmas*, 24(5):055702, February 2017. doi:10.1063/1.4977468.

[30] Jared Keaton Smythe. *Electrode Geometry Effects on Plume Characteristics and Thruster Performance of ZaP-HD*. Thesis, University of Washington, 2023. Accepted: 2023-08-14T17:01:38Z. URL: https://digital.lib.washington.edu:443/researchworks/handle/1773/50200.

[31] Yu Takagaki. *Development of a 5N-moment Multi-Fluid Plasma Model for D-T Fusion in an Axisymmetric Z Pinch*. Thesis, 2023. Accepted: 2024-02-12T23:38:23Z. URL: https://digital.lib.washington.edu:443/researchworks/handle/1773/51074.

[32] H. Takeno, Y. Yasaka, M. Ishikawa, Y. Nakashima, M. Kume, N. Sotani, Y. Munakata, S. Harada, D. Akashi, T. Kawaguchi, S. Miyazaki, and K. Ichimura. Recent Results in Research on Direct Energy Conversion for a Fusion System. *Fusion Science and Technology*, 63(1T):131–134, May 2013. Publisher: Taylor & Francis _eprint: https://doi.org/10.13182/FST13-A16888. doi:10.13182/FST13-A16888.

[33] Hiromasa Takeno, Kazuya Ichimura, Satoshi Nakamoto, Yousuke Nakashima, Hiroto Matsuura, Junichi Miyazawa, Takuya Goto, Yuichi Furuyama, and Akira Taniike. Recent Advancement of Research on Plasma Direct Energy Conversion. *Plasma and Fusion Research*, 14(0):2405013–2405013, January 2019. URL: https://www.jstage.jst.go.jp/article/pfr/14/0/14_2405013/_article, doi:10.1585/pfr.14.2405013.

[34] Hiromasa Takeno, Kazuhiro Shibata, Satoshi Nakamoto, Takeru Furukawa, and Yousuke Nakashima. Examination of Efficiency Dependence on Deceleration Voltage in a Traveling Wave Direct Energy Converter Simulator. *Plasma and Fusion Research*, 18:2405053–2405053, 2023. doi:10.1585/pfr.18.2405053.

[35] Samuel E. Wurzel and Scott C. Hsu. Progress toward fusion energy breakeven and gain as measured against the Lawson criterion. *Physics of Plasmas*, 29(6):062103, June 2022. arXiv:2105.10954 [physics]. URL: http://arxiv.org/abs/2105.10954, doi:10.1063/5.0083990.

Appendix A

CHAMBER DOOR HINGE ANALYSIS

Tensile Force: 271 lbf

Shear Force: 0 lbf

Bolt Material: 17-4PH, H925 (Bar, Profile, <= 3.0 in)

Thread Size: (•) Inch ( ) Metric — 5/16-24

Preload (% Yield)	Preload Force	Torque Coefficient, K_T	Install Torque
12%	1,080 lbf	0.20	67.50 in*lbf

	$FS_{separation}$	$FS_{bolt\ yield}$	$FS_{thread\ shear\ bolt}$	$FS_{thd\ shear\ internal}$
Nominal:	4.67	200.13	13.99	3.25
Max Bolt Load:	5.83	193.30	---	---
Min Bolt Load:	3.03	209.68	---	---

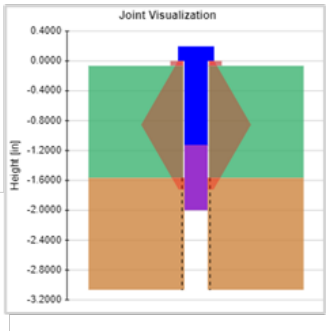


Figure A.1: Fastener Analysis Results. This is the minimum allowable fastener size, the final design might be larger. Alternatively, a thread insert can improve margins.

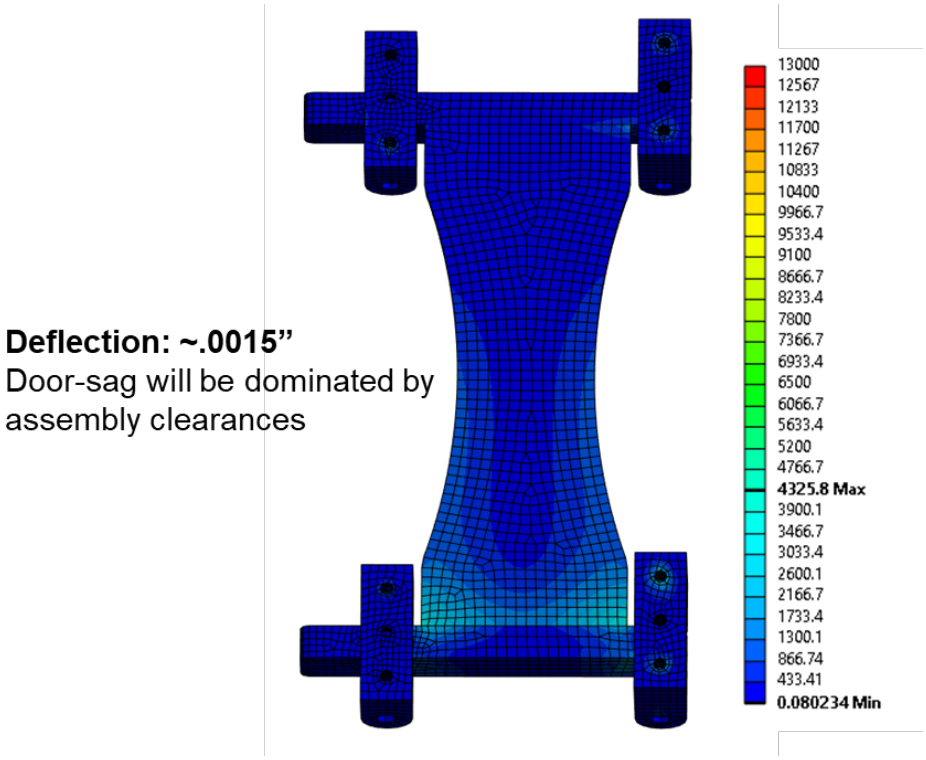


Figure A.2: Hinge stresses and deflections.

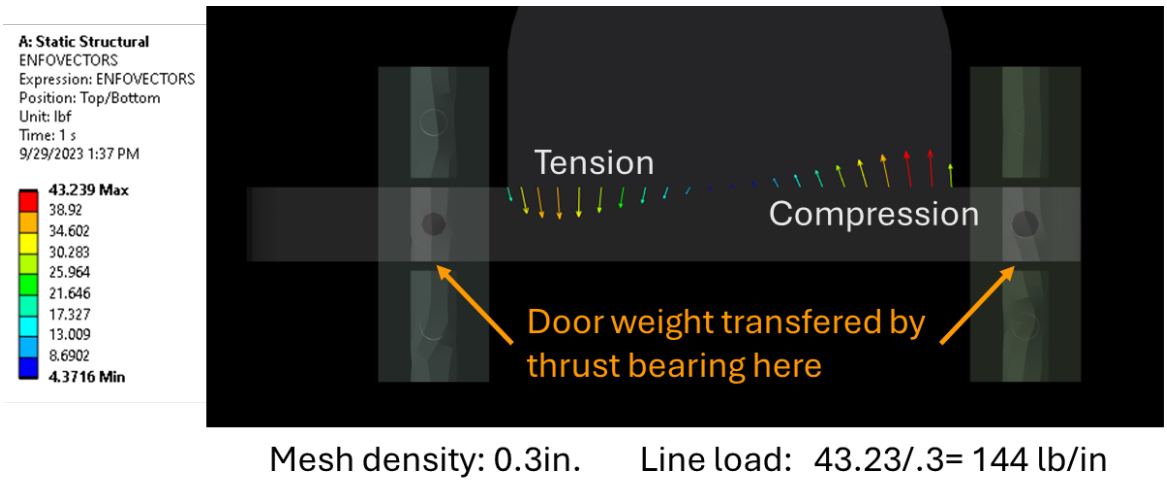
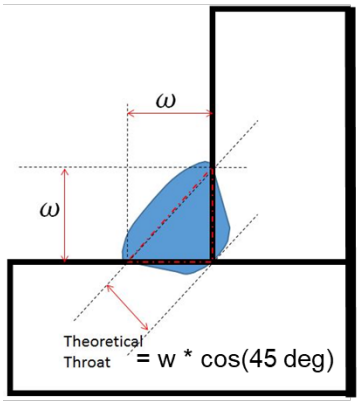


Figure A.3: Predicted weld line-load from FEM



(Minimum weld size per AWS D1.1/D1.1M:2020 Structural Welding Code (Steel), Table 7.7.

w	0.125	in
throat	0.0884	in
L	0.3	in
A	0.027	in2
F	43.3	lbf
stress	1632.9	psi

Weld stresses are very low.

Figure A.4: Weld stress prediction.

Appendix B

ZAP-DEC PYTHON CODE

ZAP_DEC.py

```
"""****************************
OVERVIEW OF THE ZAP_DEC PROGRAM
- ZAP_DEC is used to study DEC from a Flow Z-Pinch with VB collectors.
- This is a 1D PIC code, but particle trjectories are tracked in 2D.
- Tracking 2D trajectories allows visulization of the angular dependency of collection.
- The Z direction is perpendicular to the retarding electric field. Y is perp to Z.
- Finite domain in Z, periodic in Y. Ey=0 everywhere.

******************************
USING THE CODE:
- A run script should be made to actually run cases. You don't run it from this file.

******************************
The ZAP_DEC() class:
- This actually runs the analysis, stores inputs and outputs, etc.
- It ties together the Particle(), Domain(), and Electrode() classes
    - A list of Particle() objects is called particles
    - A list of Electrode() objects is called electrodes

List of particles can be created manually, or with one of the functions in Particles.py

ZAP_DEC.run() actually runs the PIC algorithm by:
    1) Weighting charge to the domain cells & solving fields using Domain.update()
    2) Push each particle, and check each particle for collection.
    3) Increment time forward.

Particle absorption, energy collection:
    Every electrode has some opacity.
        'Screen' has fixed opacity.
        'Collectors' have angular dependent opacity.
        'Wall' (at zmin and zmax, respectively) have 100% opacity.

    Every 'particle' is actually superparticle representing many smaller particles.
```

```
    When a particle impact an electrode, charge and mass are multiplied by the opacity.
    The energy wasted is equal to the kinetic energy of the particle during impact.
    The energy collected is equal to (q_particle) * (electrode_voltage)
        Note1: Particles hitting Ground at zmin are absorbed with 0% collection.
        Note2: Particles hitting electron reflector collect negative energy!

    Each particle object keeps track of:
        - Current kinetic energy, charge, and mass
        - Energy absorbed
        - Energy wasted
    When only 1% of the particle mass remains, the particle is completely lost.

Some visualization functions are here to easily plot the results of a particular case.
"""

import numpy as np
import matplotlib.pyplot as plt

class ZAP_DEC:
    def __init__(self, particles, domain, dt=1):
        domain.update(particles) # Solve for cell potentials given particle positions.

        self.particles = np.array(particles) # An array of particle objects
        self.domain = domain
        self.dt = dt
        self.tpoints = [0]
        self.count_absorbed = 0

    def run(self, tspan, report=False):
        particles = self.particles
        npart = len(particles) # How many particles are there
        count_absorbed = self.count_absorbed # When this == npart, stop run.

        domain = self.domain
        elec_pos = domain.positions
        potentials = domain.potentials
        zmin, zmax = domain.zmin, domain.zmax

        dt = self.dt
        t = self.tpoints[-1]

        if t >= tspan:
            raise ValueError('Chosen "tspan" has already been simulated. Increase tspan.')
```

```
        return

    while t <= tspan:
        if t+dt > tspan:
            break

        for idx, particle in enumerate(particles):
            if particle.absorbed == True:
                continue # try the next particle

            if t == 0:
                particle.pull(dt) # Find vx_old

            particle.push(dt)
            z0, z1 = particle.z[-2], particle.z[-1]
            left, right = min(z0,z1), max(z0,z1)

            # Check for collection
            if z1 <= zmin:
                incident_electrode=[0]
            elif z1 >= zmax:
                incident_electrode=[-1]
            else:
                incident_electrode = [elec_idx for elec_idx, \
                                      pos in enumerate(elec_pos) if left < pos < right]
            for elec_idx in incident_electrode:
                # If no grids are crossed, nothing happens.
                # If one or more grids are crossed, absorb at each.
                opacity = domain.electrodes[elec_idx].incident(particle.vy,\
                                                               particle.vz[-1])
                absorbed, collected, wasted = particle.collect(opacity,\
                                                               potentials[elec_idx])
                if absorbed:
                    count_absorbed += 1
                domain.electrodes[elec_idx].collected += collected
                domain.electrodes[elec_idx].wasted += wasted
                if report:
                    print(f'Particle hit electrode {elec_idx} at angle '
                          f'{particle.angle():.1f} deg.')

        t += dt # Advance time
        self.tpoints.append(t)
```

```
            # Advance fields to n+1
            self.domain.update(particles)

            # Check collection
            if count_absorbed == npart:
                print(f'All particles absorbed at t={t}')
                break

        self.count_absorbed = count_absorbed

    # methods below plot results for a given ZAP_DEC object
    def trajectories(self, ax, kV=False, linewidth=0.1):
        particles = self.particles
        n = len(particles)
        ymax = particles[0].ymax
        zmax = self.domain.zmax
        zmin = self.domain.zmin

        # Unique color for each particle:
        colors = plt.cm.viridis(np.linspace(0, .9, n)) #tab10
        idx = 0

        for particle in particles:
            particle.trajectory(ax, color=colors[idx], label=idx, linewidth=linewidth)
            idx += 1

        self.draw_domain(ax, ymax, kV=kV)
        ax.set_xlim(zmin,zmax); ax.set_xlabel('z\' (meters)')

        ax.set_ylim(0,ymax); #ax.set_ylabel('y')
        ax.set_yticks([])

    def fields(self, ymax='particle', figsize=(10,3), rightspine=1.1):

        if ymax == 'particle':
            ymax = self.particles[0].ymax
            # User can override in case there are no particles (vacuum field)

        # Plot the positions of each particle, the potential, and the E field
        pz = np.array([particle.z[-1] for particle in self.particles])
        py = np.array([particle.y[-1] for particle in self.particles])

        cz = self.domain.z
```

```
    potential = self.domain.phi
    E = self.domain.E

    # Particle positions
    fig, ax1 = plt.subplots(dpi=600, layout='constrained', figsize=figsize)
    ax1.plot(pz, py, label='Particles', color='orange', marker='o', linewidth=0)
    ax1.set_xlabel('z')#; ax1.set_xticks([1,2,3,4]); ax1.grid('True')
    ax1.set_ylabel('y'); ax1.set_ylim(0,ymax)

    # Potential
    ax2 = ax1.twinx()
    ax2.plot(cz, potential, label='Potential, $\phi_z$', color='green')
    ax2.spines['right'].set_position(('axes', 1.0)) # adjust right spine position
    ax2.set_ylabel('Potential, $\phi_z$')#; ax2.set_ylim(2000,6000)

    # E-Field
    ax3 = ax1.twinx()
    ax3.plot(cz, E, label='Electric Field, $E_z$', color='blue')
    ax3.plot(cz, np.zeros_like(E), label='E=0', color='blue', linestyle='--')
    ax3.spines['right'].set_position(('axes', rightspine)) # adjust right spine
    ax3.set_ylabel('Electric Field, $E_z$')#; ax3.set_ylim(2,20)

    # Legend
    lines1, labels1 = ax1.get_legend_handles_labels()
    lines2, labels2 = ax2.get_legend_handles_labels()
    lines3, labels3 = ax3.get_legend_handles_labels()
    lines = lines1 + lines2 + lines3; labels = labels1 + labels2 + labels3
    ax1.legend(lines, labels, frameon=False, fontsize='small',
               ncol = 4, bbox_to_anchor=(0.5, -0.1), loc='upper center')

    return ax1, ax2, ax3, ymax

def potential_field(self, ax1, ymax, keV=False):
    # Plot the positions of each particle and the potential
    pz = np.array([particle.z[-1] for particle in self.particles])
    py = np.array([particle.y[-1] for particle in self.particles])

    cz = self.domain.z
    potential = self.domain.phi

    # Potential
    if keV == True:
        ax1.plot(cz, potential/1000, label='Potential, $\phi_z$, keV', color='green')
```

```
            ax1.set_ylabel('Potential, $\phi_z$, kV')#; ax2.set_ylim(2000,6000)
        else:
            ax1.plot(cz, potential, label='Potential, $\phi_z$, eV', color='green')
            ax1.set_ylabel('Potential, $\phi_z$, V')#; ax2.set_ylim(2000,6000)
        # ax1.spines['right'].set_position(('axes', 1.0)) # adjust right spine position


        # Particle positions
        ax2 = ax1.twinx()
        ax2.plot(pz, py, label='Particles', color='blue', \
                 marker='o', markersize=.2, linewidth=0)
        ax2.set_xlabel('z\'')#; ax1.set_xticks([1,2,3,4]); ax1.grid('True')
        ax2.set_ylim(0,ymax)

        zmax = self.domain.zmax
        zmin = self.domain.zmin
        ax1.set_xlim(zmin,zmax); ax1.set_xlabel('z\' (meters)')
        ax2.set_ylim(0,ymax); #ax.set_ylabel('y')
        ax2.set_yticks([])

        # Legend
        lines1, labels1 = ax1.get_legend_handles_labels()
        lines2, labels2 = ax2.get_legend_handles_labels()
        lines = lines1 + lines2; labels = labels1 + labels2
        ax1.legend(lines, labels, frameon=False, fontsize='small',
                   ncol = 4, bbox_to_anchor=(0.8, -0.2), loc='upper center')

        return ax1, ax2

    def draw_domain(self, ax, ymax, kV=False):
        """
        Draws the domain and the grids, labeling grids with potentials
        """
        positions  = self.domain.positions
        potentials = self.domain.potentials
        if kV:
            scale=1/1000
        else:
            scale=1

        for idx in range(len(positions)):
            style = self.domain.electrodes[idx].style
            if    style=='wall':   linestyle='-'
```

```
        elif style=='screen': linestyle=':'
        elif style=='ribbon': linestyle='-.'
        if kV == True:
            label = f'  ${potentials[idx]*scale:+.0f}$ kV'
        else:
            label = f'  ${potentials[idx]*scale:+.0f}$ V'
        ax.plot([positions[idx], positions[idx]], [0, ymax], \
                linestyle=linestyle, color='black')
        ax.text(positions[idx], ymax, label, fontsize=8,\
                rotation=90, ha='center')

def total_efficiency(self):
    collected = 0
    total = 0
    for particle in self.particles:
        collected += particle.energy_collected
        total += particle.KE0
    return collected/total
```

electrodes.py

```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image # Used for animation

#%%

class Electrode():
    """
    Defines an electrode, which is then abstracted as a plane.
    -------------------------
    TYPES:
        'screen' electrodes have fixed opacity
        'ribbon' electrodes have angular dependant transparency
    PARAMETERS
        fixed_opacity : Only works for 'screen' electrodes
        ribbon_angle : float, DEGREES
            Angle of the ribbons
        ribbon_L : float
            Length of the ribbons in a ribbon grid.
        pitch : float
            Spacing between ribbons, measured from leading edge of each ribbon.
    -------------------------
    METHODS

        incident(vy, vz, print_out=False) :
            Particles hit ribbons at incident angle determined by velocities.
            Returns fraction of particles absorbed.
            vy, vz: Vertical and horizontal velocities respectively.
            print_out : Optional argument to print the opacity

        map_opacity(plot=False, animate=False) :
            Calls opacity() in a loop with 1 degree increments.
            Arg plot=True shows the map, animate=True shows all the plots from opacity().
            WARNING: Animate makes a lot of files (359x .png, 1x .gif)

        opacity(theta, plot=False) :
            Calculates opacity at a given angle. Arg plot=True visualizes this.
            theta : float, RADIANS (For all methods)
                Angle of particle velocity vector. 0 degrees is horizontal to the right.

        plot_vec() :
```

```
        Helper function used for plotting in opacity().
        Not intended to be called externally.
"""

def __init__(self, style, # Style options are 'screen' and 'ribbon'
             fixed_opacity=.01, ribbon_angle=5, ribbon_L=1, pitch=1/2.5):
            # defaults are close to those in barr et.al. (1974)
            # "1% of total beam current impionges on the grid" pg 80 (10 in pdf)

    self.style=style

    if style=='ribbon':
        self.ribbon_angle = float(ribbon_angle)
        self.ribbon_L = float(ribbon_L)
        self.pitch = float(pitch)
        self.angles, self.opacities = self.map_opacity()

    elif style=='screen':
        self.fixed_opacity = fixed_opacity

    elif style=='wall':
        self.fixed_opacity = 1

    else:
        raise ValueError('Style must be ribbon, screen, or wall')

    # For tracking collection efficiency
    self.collected = 0
    self.wasted = 0

def incident(self, vy, vz, print_out=False):
    # This method will be called every time a particle hits an electrode

    theta = np.arctan2(vy, vz)

    if self.style=='ribbon':
        opacity = np.interp(theta, self.angles, self.opacities)
        if print_out == True:
            print(f'Interpolated Opacity: {opacity*100:.1f}%')
        return opacity

    if self.style=='screen' or 'wall':
        return self.fixed_opacity
```

```
def map_opacity(self, plot=False, animate=False):
    Angles = np.arange(-180, 180, 1)*np.pi/180 # Evaluated once per degree
    Opacities = []
    file_names = []

    for theta in Angles:
        opacity = self.opacity(theta, plot=animate)
        # plot=animate because we only need to plot at this step if we plan to animate
        Opacities.append(opacity)

        if animate==True:
            file_name = f"frame_{theta:03d}.png"
            plt.savefig(file_name, dpi=300)
            plt.close()  # Close the figure to free up memory
            file_names.append(file_name)

    if plot==True:
        fig, ax = plt.subplots(dpi=600)
        ax.plot(Angles*180/np.pi, Opacities)
        ax.set_title('Opacity vs Incident Angle of Particles\n'
            f'{self.ribbon_L} long, '
            f'{self.ribbon_angle}$\degree$ angle, {self.pitch} pitch.')
        ax.set_xlabel('Angle, degrees')
        ax.set_ylabel('Opacity')

    if animate==True:
        images = [Image.open(file_name) for file_name in file_names]
        images[0].save("ribbon_animation.gif", save_all=True, \
                       append_images=images[1:], duration=100, loop=0)

    return Angles, np.array(Opacities)

def opacity(self, theta, plot=False):
    ribbon_angle = self.ribbon_angle
    ribbon_angle *= np.pi/180 # Radians

    ribbon_L = self.ribbon_L
    pitch = self.pitch

    # Unit vec perpendicular to theta
    view_vec = np.array([np.cos(theta +np.pi/2), np.sin(theta +np.pi/2)])
```

```
# Draw ribbon and its projection
ribbon_vec = np.array([ribbon_L * np.cos(ribbon_angle), \
                       ribbon_L*np.sin(ribbon_angle)])
ribbon_proj = np.dot(view_vec, ribbon_vec)*view_vec

# Show the particle flow direction
particle_vec = np.array([ribbon_L * np.cos(theta), ribbon_L * np.sin(theta)])

# Project the pitch between ribbons (max size of each opening)
if ribbon_proj[1] < 0:
    pitch *= -1
pitch_vec = np.array([0,pitch])
pitch_proj = np.dot(view_vec, pitch_vec)*view_vec

# Compute magnitudes of projections
mag_ribbon_proj = np.linalg.norm(ribbon_proj)
mag_pitch_proj = np.linalg.norm(pitch_proj)

# Compute opacity, which is 1-transparency
if mag_pitch_proj == 0:
    opacity = 1
else:
    opacity = mag_ribbon_proj/mag_pitch_proj
    if opacity > 1:
        opacity = 1

if plot==True:
    fig, ax = plt.subplots(dpi=600)

    self.plt_vec(ax, ribbon_vec, label='Ribbon', color='k', pitch=pitch, lw=3)
    ax.arrow(0,0,particle_vec[0],particle_vec[1], color='green', \
             head_width=0.1, head_length=0.1, label='Flow Direction' )

    self.plt_vec(ax, ribbon_proj, label='Ribbon_Proj', color='r')
    self.plt_vec(ax, ribbon_proj, origin=ribbon_vec, color='r', ls=':')
    self.plt_vec(ax, pitch_proj, label='Pitch_Proj', ls=':', color='C0', lw=2)
    self.plt_vec(ax, pitch_proj, origin=(0,pitch), ls=':', color='C0')

    ax.set_aspect('equal')
    ax.set_xlim(-1.25*ribbon_L,1.25*ribbon_L)
    ax.legend(frameon=False, bbox_to_anchor=(1.05, 0.5), loc='center left')

return opacity
```

```
def plt_vec(self, ax, vec, origin=(0,0), pitch=0, **kwargs):
    """
    Convenience function for plotting a vector.
    Providing pitch will make 2x vectors, vertically spaced by pitch.
    **kwargs is any argument for ax.plot(); i.e. color, label
    """
    ax.plot( (origin[0],vec[0]), (origin[1],vec[1]), **kwargs)

    if pitch != 0:
        if "label" in kwargs:
            kwargs["label"] = None
        ax.plot( (origin[0],vec[0]), (origin[1]+pitch,vec[1]+pitch), **kwargs)
        ax.plot( (origin[0],vec[0]), (origin[1]-pitch,vec[1]-pitch), **kwargs)

def efficiency(self):
    total = self.wasted+self.collected
    if total == 0:
        efficiency = 0
    else:
        efficiency = self.collected / total
    return efficiency
```

domain.py

```
'''
The domain is the computational grid on which the PIC analysis runs.
Particles have their own domain in z and y, and the z domains should match.
Since collectors can also be called "grids", I use the words "cells, nodes, and domain".

To run a Particle-in-cell analysis, we need cells. We weight particle charge to the cells,
then solve for the electric field between the nodes. A cell is the space between nodes.

Different regions of the domain are calculated differently:
    Z < 0: Quasineutral Zone
        This is where particles are normally spawned. Only ions are simulated directly.
        A neutralizing background of electrons is sim'd by subtracting avg charge density.

    Z > 0: Space Charge Zones
        Electrons are screened from these zones, so space charge effects are possible.

    Zones are seperated by electrodes.
    For example (See Fig 10 & 13 of [1]):
        'Ground' at zmin (sets reference voltage for the system)
        'Electron-reflector' at z=0 (Screen electrode with negative voltage)
        'Collector(s)': Electrodes at z>0 (positive voltage)
        'Wall': at zmax (positive voltage)

    This class only calculates fields in the presence of particles.
    Particle collisions are calculated by ZAP_DEC after particle.push().

    [1]Barr, W. L., R. J. Burleigh, W. L. Dexter, R. W. Moir, and R. R. Smith.
        A Preliminary Engineering Design of a Venetian Blind Direct Energy Converter
       for Fusion Reactors. IEEE Transactions on Plasma Science 2, no. 2 (June 1974)
       https://doi.org/10.1109/TPS.1974.6593737.
'''
#%% Header
import numpy as np
import matplotlib.pyplot as plt
from scipy.sparse import spdiags
from scipy.sparse.linalg import spsolve, splu

class Domain:
    def __init__(self, electrodes, potentials=[0, -10, 100], \
                 positions=[-1, 0, 1], dz=0.1, SI=False):
        # 'electrodes' is a list of Electrode class objects. Stores opacity information.
```

```
# 'positions'  is a vector of z-coordinates for each corresponding electrode. 0 is
    required.
# 'potentials' is the 'voltage' of each corresponding electrode.
# dz is the spacing of cell nodes on the computational domain.

# Check inputs for errors:
if not all([len(electrodes) == len(potentials), \
            len(electrodes) == len(positions)]):
    raise ValueError('Each electrode must have a position and potential.')

if not all([electrodes[0].style == 'wall', electrodes[-1].style == 'wall']):
    raise ValueError('The first and last electrodes must be walls.')

if 0 not in positions:
    raise ValueError('One electrode must be located at z=0. '
                     'This should be the electron reflector.')

if any(pos+dz in positions for pos in positions):
    raise ValueError('Minimum seperation between electrodes is 2*dz')

# Initialize cell nodes globally
zmin, zmax = positions[0], positions[-1]  # These are domain endpoints
z = np.arange(zmin, zmax+dz, dz)
z = np.round(z, decimals=8) # Stupid floating point errors

if dz != np.round(z[1] - z[0], decimals=8):
   raise Exception('Linspace did not match user-specified dz')

if not all(pos in z for pos in positions):
    raise ValueError("Every electrode position must be on the computational grid "
                     "(divisible by dz).")

# Define each zone and make laplacian operator for each:
nzones = (len(positions)-1)
zones = np.zeros((nzones,2), dtype=int)
    # zones is a 2D array where each row reads
        # [zone number, left z index, right z index]
    # Each z-index corresponds to the electrode position at the zone boundary
LU = []
    # List of LU objects. These are LU decompositions of laplacian for each zone.

for i in range(nzones):
    zones[i,0] = np.argwhere(z==positions[i])[0,0]
```

```
            zones[i,1] = np.argwhere(z==positions[i+1])[0,0]
            m = zones[i,1]-zones[i,0]+1 # Number of points in the zone

            d1 = np.ones(m) # A diagonal of 1s
            A = spdiags([d1, -2*d1, d1],
                        [-1,     0,  1], (m-2,m-2), format='csc')
            # Size is m-2 because grid potentials are known.

            A /= dz**2
            LU.append(splu(A)) # I can reuse this many times.

        # Store attributes
        self.electrodes = electrodes
        self.potentials = potentials
        self.positions = positions
        self.z, self.dz = z, dz
        self.zmin, self.zmax = zmin, zmax

        self.nzones = nzones
        self.zones = zones
        self.LU = LU
        self.SI = SI


    #%% Update function
    def update(self, particles):
        """
        This method takes the following steps:
            1) Weights charge from particles to the grid
            2) Electric field solve on grid (piecewise by zone)
            3) Weights Electric field to the particles

        Input is a list of particles of Particle class.
        """
        # ===================================================================================
        # 1) WEIGHT TO GRID
        # ===================================================================================

        zCells = self.z   # Positions of grid points
        rho = np.zeros_like(zCells) # Charge density at each gridpoint
        dz = self.dz

        for particle in particles:
```

```
        zPart = particle.z[-1] # Current position of the particle
        q = particle.q         # Current charge of the particle
        # Find the indexes above and below
        idx_above = np.searchsorted(zCells, zPart)
        idx_below = idx_above-1

        # If particle is beyond highest grid point, it must have been collected
        if idx_above > len(zCells)-1:
            # No charge to weight here
            # print('Warning: Particle which should have been collected is off grid')
            return

        # If particle is exactly on a gridpoint, idx_above will match
        # If exactly on the left endpoint, idx_below will be -1
        if zCells[idx_above] == zPart:
            rho[idx_above] += q/dz
            break

        # Weight charge
        a = zPart - zCells[idx_below] # This is why I needed 'if' just above
        b = dz-a

        rho[idx_below] += q*b/(dz**2)
        rho[idx_above] += q*a/(dz**2)

    # Add neutralizing background charge for z<0
    idx_z0 = np.argwhere(zCells==0)[0,0] # Corresponds to z=0
    if idx_z0 > 2: # Doesn't make sense otherwise...
        rho[1:idx_z0] -= np.mean(rho[1:idx_z0])
        # I exclude charge weighted to the endpoints (electrodes)

    self.rho = rho # Store this for debugging purposes.

    # ============================================================================
    # FIELD SOLVE
    # ============================================================================
    # Electric potential (phi) solve:    ^2(phi) = -rho
    phi = np.array([self.potentials[0]])
    if self.SI:
        E0 = 8.85419e-12 # F/M
    else:
        E0 = 1
```

```
for i in range(self.nzones):

    pleft, pright = self.potentials[i], self.potentials[i+1]
    idx_left, idx_right = self.zones[i,0], self.zones[i,1]

    b = -rho[idx_left+1:idx_right]/E0 # Truncate the end points (potential known)
    b[0]  -= pleft/dz**2   # RHS of poissons eq for boundary points
    b[-1] -= pright/dz**2
    phi = np.append(phi, self.LU[i].solve(b)) # potentials at interior points
    phi = np.append(phi, pright) # potential at right side of domain

self.phi = phi # Save the potentials

"""
Electric field solve: E=   phi
np.gradient uses central difference on interior points, fwd/back diff at ends
"""
E = -np.gradient(phi,dz) # Takes the gradient with spacing dz
self.E = E

# ================================================================================
# WEIGHT ELECTRIC FIELD TO PARTICLES
# ================================================================================

for particle in particles:
    zPart = particle.z[-1] # Current position of the particle
    # Find the indexes above and below
    idx_above = np.searchsorted(zCells, zPart)
    idx_below = idx_above-1

    # If particle is beyond highest grid point, it is between x[-1] and x[0]
    if idx_above > len(zCells)-1:
        raise ValueError('Particle is outside of the grid')

    # If particle is exactly on a gridpoint, idx_above will match
    # If exactly on the left endpoint, idx_below will be -1
    if zCells[idx_above] == zPart:
        particle.E = E[idx_above]
        break

    # Weight field
    a = zPart - zCells[idx_below]
    b = self.dz-a
```

```
particle.E  = E[idx_below]*b/dz
particle.E += E[idx_above]*a/dz
```

particles.py

```
import numpy as np
# import matplotlib.pyplot as plt

empty = np.array([np.nan, np.nan]) # Used to fill vectors in various places

class Particle: # Track a particle's motion in response to forces
    """
    Initial position (pos0) and velocity (v0) are tuples or vectors: (y,z), (vy, vz).
    The y dimension is periodic, from 0 to ymax [m].
    There are no forces in y, so vy is constant always. These can be superparticles.

    Mass and charge are normalized to proton mass and e, respectively.
    If SI=True, mass and charge get modified internally (specify # nucleons and # protons)

    KEx accounts for energy loss in the x dimension by doubling energy loss in y.
        Keep off for beams. Consider using for maxwellians.
    """
    def __init__(self, pos0=(0,0), v0=(1,1), ymax=1, \
                 mass=1, charge=1, SI=False, KEx=False):
        if len(pos0)!=2 or len(v0)!=2:
            raise ValueError('pos0 (y,z) and v0 (vy,vz) must have length 2.')

        self.z = [pos0[1]]
        self.y = [pos0[0]]

        self.vy = v0[0]   # Always constant
        self.vz = [v0[1]] # List of vz

        self.ymax = ymax

        if SI:
            mass   *= 1.6726e-27 # proton mass
            charge *= 1.6022e-19 # proton charge

        self.m, self.m0 = mass, mass # Current mass and initial mass
        self.q = charge
        self.E = 0 # V/m, Electric field felt by particle in z direction

        if KEx:
            self.KE0 = .5*mass*( v0[1]**2 + 2*v0[0]**2 )
            # Initial KE of particle. vx estimated.
```

```
        else:
            self.KE0 = .5*mass*( v0[1]**2 + v0[0]**2 )
            # Initial KE of particle. vx is not accounted for.

        self.cross = [empty]            # History of when and where I cross y boundaries.
                                        # Contains coords of crossing over last timestep.
                                        # nan, nan means no crossing
        """
        Track whether this particlar particle has been collected or lost (absorbed)
        Electric energy collected
        """
        self.absorbed=False
        self.energy_collected=0

    def pull(self, dt):
        # Integrate backwards by half of a timestep to find vx_old
        # This only needs to happen when vx_old is unknown
        # NOTE: Use same dt as push(), the 1/2 is added in the function!
        vz = self.vz[-1]
        accel = self.E*self.q/self.m
        self.vz_old = vz - dt/2*accel

    def push(self, dt):
        # if self.absorbed == True:
        #     return # This code was redundant with ZAP_DEC

        y1, z1 = self.y[-1], self.z[-1]

        # Leapfrog velocity, then position by 1 timestep. (They are 1/2 step apart)
        accel = self.E*self.q/self.m
        vz_new = self.vz_old + dt*accel
        y2 = y1 + self.vy*dt
        z2 = z1 + vz_new*dt

        # Periodic BCs (y only)
        ymax = self.ymax # Note: ymin is always 0

        if y2 >= ymax:
            f = (ymax-y1)/(y2-y1)
            zc = f*(z2-z1)+z1 # This is where the trajectory crossed the boundary
            cross = np.array([ymax,zc]) # 1 means particle entered ymax boundary
        elif y2 < 0:
            f = -y1/(y2-y1)
```

```
            zc = f*(z2-z1)+z1
            cross = np.array([0,zc]) # 0 means particle entered ymin boundary
        else:
            cross = empty # no crossing

        y2 %= ymax  # remainder wraps over domain

        # Update centered velocity (For plots, not used in algorithm)
        self.vz.append(( vz_new + self.vz_old )/2)

        # Prepare for the next timestep
        self.y.append(y2)
        self.z.append(z2)
        self.vz_old = vz_new
        self.cross.append(cross.copy())

    def collect(self, opacity, potential):
        collected = self.q * potential * opacity

        wasted = .5*self.m*opacity * (self.vy**2+self.vz[-1]**2)

        self.m *= (1-opacity)
        self.q *= (1-opacity)
        self.energy_collected += collected

        if self.m/self.m0 < .01:
            self.absorbed = True # If only 1% of the mass is left, lets call it a day.
        return self.absorbed, collected, wasted

    def remaining(self):
        remain = self.m/self.m0*100
        print(f'{remain:.1f}% of particle mass remaining')

    def eff(self):
        eff = self.energy_collected/self.KE0
        print(f'{eff*100:.1f}% of particle energy collected')

    def trajectory(self, ax, color='C0', label='', linewidth=0.1):
        """
        Plots the trajectory of a single particle.
        Plotting is broken into segments delineated by periodic boundary crossings.
        All segments and the final marker have the same color, chosen at random.
        """
```

```
posA = np.array([self.y, self.z]) # Positions
crossA = np.array(self.cross) # y-crossings
idxs = np.argwhere(~np.isnan(crossA[:,0])).flatten() # Indices of crossings
    # ~ is the bitwise NOT operator. Converts false to true.
    # This find indices of non-NAN values in first column of crossA.

if len(idxs)==0: # If no crossings, this function is very simple
    y, z = posA[0,:], posA[1,:]
    ax.plot(z, y, color=color, label=label, linewidth=linewidth)
    ax.plot(z[-1], y[-1], color=color, marker='o', markersize=0.5)
    # Final location (collected or lost here)
    return

"""
First plot a line from start point until the index where it has crossed.
Overwrite the last point with the crossing location.
New start point has same crossing location but on the next boundary.
"""
start_idx = 0 # The start index. This changes every loop.
start_pt = posA[:,0] # The start point. This changes every loop.
ymax = self.ymax

for idx in idxs:
    y, z = posA[0,start_idx:idx+1], posA[1,start_idx:idx+1]
    # Plot until it crosses
    y[0], z[0] = start_pt # Overwrite first point to be start point.
    y[-1], z[-1] = crossA[idx,:] # Overight final point as crossing entrance
    ax.plot(z, y, color=color, linewidth=linewidth) # Plot the segment

    # Find crossing exit for next segment
    start_idx = idx.copy()
    start_pt = crossA[idx,:]
    if start_pt[0]==0:
        start_pt[0] = ymax
    else:
        start_pt[0] = 0

# Plot the final segment after the last crossing
y, z = posA[0,start_idx:], posA[1,start_idx:]
y[0], z[0] = start_pt # Overwrite first point to be start point.
ax.plot(z, y, color=color, label=label, linewidth=linewidth)
ax.plot(z[-1], y[-1], color=color, marker='o', markersize=0.5)
# Final location (collected or lost here)
```

```
    def angle(self, idx=-1):
        if idx > len(self.vz)-1:
            idx = -1
        angle = np.arctan2(self.vy, self.vz[idx])*180/np.pi
        return angle

def super_maxwellian(N, density, yrange, zrange, kT, uz, \
                     mass=1, q=1, theta=0, ymax=[], SI=False, beam=False):
    '''
    Makes a maxwellian distribution of N superparticles!
    kT is the temperature. If SI is on, this is in eV.
    uz is the drift velocity (m/s in SI)
    theta specifies the incidence angle of the beam on the grids
    mass and q are for actual particles, NOT superparticles.
        Should always be normalized to proton mass & charge.
    beam=True makes temperature only apply in z. Beam is cold in x and y.
    '''
    if ymax==[]:
        ymax = yrange[1]

    # Superparticals
    Nplasma = density*(max(zrange)-min(zrange))
    scale = Nplasma/N

    # Find v_thermal of the actual particles
    if SI:
        mp = 1.6726e-27 # proton mass
        e  = 1.6022e-19 # proton charge
    else:
        mp = 1
        e = 1
    v_th = np.sqrt(kT*e/(mass*mp))

    # Positions and Velocities
    y =  np.random.uniform(yrange[0], yrange[1], N)
    z =  np.random.uniform(zrange[0], zrange[1], N)
    vz = np.random.normal(uz, v_th, N)
    if beam:
        vy =  np.zeros(N) # kTy = 0
        KEx=False # kTx = 0
    else:
        vy = np.random.normal(0, v_th, N)
```

```
    KEx=True # kTx = kTy

# Rotate velocity vectors by given angle
theta *= np.pi/180
vz_prime = vz*np.cos(theta) - vy*np.sin(theta)
vy_prime = vz*np.sin(theta) + vy*np.cos(theta)

pos0 = np.array([y, z]).T
v0 = np.array([vy_prime,vz_prime]).T

particles = [Particle(pos0[i], v0[i], ymax, mass*scale, q*scale, SI=SI, KEx=KEx)\
             for i in range(N)]
# Note: particles is always defined with normalized mass and charge
return particles
```

RUN benchmark beam.py

```
'''
Benchmark beam case

'''

#%% Header
# My files:
from ZAP_DEC import ZAP_DEC
from particles import Particle, super_maxwellian
from domain import Domain
from electrodes import Electrode

# Packages:
import numpy as np
import matplotlib.pyplot as plt

#%% Constants
# mp = 1.6726e-27 # proton mass
# e  = 1.6022e-19 # proton charge

#%% Define eletrodes and domain
left_wall = Electrode('wall')
ground = Electrode('screen')
reflect = Electrode('screen')
rib1 = Electrode('ribbon', ribbon_angle=12)
rib2 = Electrode('ribbon', ribbon_angle=13.2)
rib3 = Electrode('ribbon', ribbon_angle=14.9)
right_wall = Electrode('wall')

electrodes = [left_wall, ground, reflect, rib1, rib2, rib3, right_wall]
positions = np.array([-.5, -0.24,   0, 0.13, 1.39, 2.65, 3.91])
potentials= np.array([  0,     0, -14, 94.4,  132,  180,  243])*1000 # Volts

dz=0.01
domain = Domain(electrodes, potentials, positions, dz=dz, SI=True)

#%% Particles
zrange = [-.5, -.24]
yrange = [.25, .25]
ymax = 2.5
a0 = 7 # degrees, the beam angle
```

```
kT = 19.25*1000 # eV was 25.6

Dpart = super_maxwellian(450, 1.43E+12, yrange, zrange, kT, \
        uz=4e6, mass=2, q=1, theta=a0, ymax=ymax, SI=True, beam=True)

Tpart = super_maxwellian(450, 1.09e12, yrange, zrange, kT, \
        uz=3.27e6, mass=3, q=1, theta=a0, ymax=ymax, SI=True, beam=True)

Hepart = super_maxwellian(100, 2.03e10, yrange, zrange, kT, \
        uz=4e6, mass=4, q=2, theta=a0, ymax=ymax, SI=True, beam=True)

particles = Dpart+Tpart+Hepart


#%% Analysis
dt = dz/6e6/2
# dt = .000001 # Need to figure out for SI units
test = ZAP_DEC(particles, domain, dt=dt)
test.run(dt*10000)

#%% Plot
fig, ax = plt.subplots(dpi=600, figsize=(10,2))
test.trajectories(ax, kV=True);
# plt.savefig("benchmark beam.pdf", bbox_inches="tight")

for idx, el in enumerate(electrodes):
    print(f'Electrode {idx}, ({el.style}), efficiency = {el.efficiency()*100:.1f}%')

print(f'Total efficiency: {test.total_efficiency()*100:0.1f}%')
```

RUN benchmark maxwellian.py

```
"""
BENCHMARK WITH MAXWELLIAN CASE
"""

#%% Header
# My files:
from ZAP_DEC import ZAP_DEC
from particles import Particle, super_maxwellian
from domain import Domain
from electrodes import Electrode

# Packages:
import numpy as np
import matplotlib.pyplot as plt

#%% Constants
# mp = 1.6726e-27 # proton mass
# e  = 1.6022e-19 # proton charge

#%% Define eletrodes and domain
left_wall = Electrode('wall')
ground = Electrode('screen')
reflect = Electrode('screen')
rib1 = Electrode('ribbon', ribbon_angle=12)
rib2 = Electrode('ribbon', ribbon_angle=13.2)
rib3 = Electrode('ribbon', ribbon_angle=14.9)
right_wall = Electrode('wall')

electrodes = [left_wall, ground, reflect, rib1, rib2, rib3, right_wall]
positions = np.array([-.5, -0.24,   0, 0.13, 1.39, 2.65, 3.91])
potentials= np.array([  0,     0, -14, 94.4,  132,  180,  243])*1000 # Volts

dz=0.005
domain = Domain(electrodes, potentials, positions, dz=dz, SI=True)

#%% Particles
zrange = [-.5, -.24]
yrange = [1.75, 1.75]
ymax = 5
a0 = 7 # degrees, the beam angle
kT = 19.25*1000 # eV was 25.6
```

```
Dpart = super_maxwellian(450, 1.43E+12, yrange, zrange, kT, \
        uz=4e6, mass=2, q=1, theta=a0, ymax=ymax, SI=True, beam=False)

Tpart = super_maxwellian(450, 1.09e12, yrange, zrange, kT, \
        uz=3.27e6, mass=3, q=1, theta=a0, ymax=ymax, SI=True, beam=False)

Hepart = super_maxwellian(100, 2.03e10, yrange, zrange, kT, \
        uz=4e6, mass=4, q=2, theta=a0, ymax=ymax, SI=True, beam=False)

particles = Dpart+Tpart+Hepart


#%% Analysis
dt = dz/6e6/2
# dt = .000001 # Need to figure out for SI units
test = ZAP_DEC(particles, domain, dt=dt)
test.run(dt*100000)

#%% Plot
fig, ax = plt.subplots(dpi=600, figsize=(10,4))
test.trajectories(ax, kV=True);
# plt.savefig("benchmark maxwellian.pdf", bbox_inches="tight")

for idx, el in enumerate(electrodes):
    print(f'Electrode {idx}, ({el.style}), efficiency = {el.efficiency()*100:.1f}%')

print(f'Total efficiency: {test.total_efficiency()*100:0.1f}%')
```

RUN zap_hd grids v1.py

```
#%% Header
# My files:
from ZAP_DEC import ZAP_DEC
from particles import Particle, super_maxwellian
from domain import Domain
from electrodes import Electrode

# Packages:
import numpy as np
import matplotlib.pyplot as plt

#%% Constants
mp = 1.6726e-27 # proton mass
e  = 1.6022e-19 # proton charge

#%% Define eletrodes and domain
left_wall = Electrode('wall')
ground = Electrode('screen')
reflect = Electrode('screen')
rib1 = Electrode('ribbon', ribbon_angle=10.9)
rib2 = Electrode('ribbon', ribbon_angle=15.8)
right_wall = Electrode('wall')

electrodes = [left_wall, ground, reflect, rib1, rib2, right_wall]
positions = np.array([-.15, -0.05,   0, 0.05, 0.3, 0.55])
potentials= np.array([  0,      0, -100,   15,  80, 160])

dz=0.001
domain = Domain(electrodes, potentials, positions, dz=dz, SI=True)

#%% Single test particle

a0 = 10 # degrees, the beam angle

vz = np.sqrt(90*e*2/mp) # Find the trajectory that hits the back wall
vy = vz*np.tan(a0*np.pi/180)

test_particle = [Particle(pos0=(.05,-.15), v0=(vy,vz), ymax=.3, \
                          mass=1, charge=1, SI=True, KEx=False)]

dt = dz/vz # Takes 2 time steps to cross one grid-space
```

```
test = ZAP_DEC(test_particle, domain, dt=dt)
test.run(dt*10000, report=True)


#%% Plot
fig, ax = plt.subplots(dpi=600, figsize=(10,2))
test.trajectories(ax, kV=False, linewidth=1);

for idx, el in enumerate(electrodes):
    print(f'Electrode {idx}, ({el.style}), efficiency = {el.efficiency()*100:.1f}%')

print(f'Total efficiency: {test.total_efficiency()*100:0.1f}%')
plt.savefig('periodic.pdf', bbox_inches="tight")

#%%
# rib2.map_opacity(plot=True)

#%% Particles for maxwellian (no end wall)
zrange = [-.15, -.05]
yrange = [.2, .2]
ymax = .5
kT = 20 # eV

particles = super_maxwellian(1000, 4e12, yrange, zrange, kT, \
        uz=103000, mass=1, q=1, theta=a0, ymax=ymax, SI=True, beam=False)


#%% Analysis, stop for space charge

test = ZAP_DEC(particles, domain, dt=dt)
test.run(dt*1000)

#%% Plots
fig, ax = plt.subplots(dpi=600, figsize=(10,2))
test.trajectories(ax, kV=False);

potential = domain.phi
cz = domain.z

ax2 = ax.twinx()
ax2.plot(cz, potential, label='Potential, $\phi_z$, eV', color='blue')
ax2.set_ylabel('Potential, $\phi_z$, V')#; ax2.set_ylim(2000,6000)
plt.savefig('zap space charge.pdf', bbox_inches="tight")
```

```
#%% Finish

test = ZAP_DEC(particles, domain, dt=dt)
test.run(dt*1000000)

#%% Plots
fig, ax = plt.subplots(dpi=600, figsize=(10,2))
test.trajectories(ax, kV=False);

for idx, el in enumerate(electrodes):
    print(f'Electrode {idx}, ({el.style}), efficiency = {el.efficiency()*100:.1f}%')

print(f'Total efficiency: {test.total_efficiency()*100:0.1f}%')
```

RUN DT Fusion take 6.py

```
'''
DT Fusion Case
'''
#%% Header
# My files:
from ZAP_DEC import ZAP_DEC
from particles import Particle, super_maxwellian
from domain import Domain
from electrodes import Electrode

# Packages:
import numpy as np
import matplotlib.pyplot as plt

#%% Constants
mp = 1.6726e-27 # proton mass
e  = 1.6022e-19 # proton charge

#%% Define eletrodes and domain
left_wall = Electrode('wall')
ground = Electrode('screen')
reflect = Electrode('screen')
rib1 = Electrode('ribbon', ribbon_angle=9)
rib2 = Electrode('ribbon', ribbon_angle=11)
rib3 = Electrode('ribbon', ribbon_angle=15)
right_wall = Electrode('wall')

electrodes = [left_wall, ground, reflect, rib1, rib2, rib3, right_wall]
positions = np.array([-.5, -0.2,   0, 0.5, 1.5, 2.5, 3.5])
potentials= np.array([  0,     0, -40, 400,  600,  800,  1000])*1000 # Volts

dz=0.01
domain = Domain(electrodes, potentials, positions, dz=dz, SI=True)

#%% Test particle
a0 = 7 # degrees, the beam angle
W0 = .320e6*6 * e # Twice the max grid voltage bc Z=2 for alphas

vz = np.sqrt(2*W0/(4*mp)) # Find the trajectory that hits the back wall
vy = vz*np.tan(a0*np.pi/180)
```

```
test_particle = [Particle(pos0=(.05,-.15), v0=(vy,vz), ymax=1, \
                                mass=4, charge=2, SI=True, KEx=False)]

dt = dz/(2*vz) # Takes 2 time steps to cross one grid-space
test = ZAP_DEC(test_particle, domain, dt=dt)
test.run(dt*10000, report=True)

#%% Plot test particle
fig, ax = plt.subplots(dpi=600, figsize=(10,2))
test.trajectories(ax, kV=True, linewidth=1);

for idx, el in enumerate(electrodes):
    print(f'Electrode {idx}, ({el.style}), efficiency = {el.efficiency()*100:.1f}%')

print(f'Total efficiency: {test.total_efficiency()*100:0.1f}%')
# plt.savefig('periodic.pdf', bbox_inches="tight")

#%% Particles
zrange = [-.4, -.25]
yrange = [1.25, 1.25]
ymax = 3
kT = .7e6 # eV

Hepart = super_maxwellian(1000, 3.5e16, yrange, zrange, kT, \
        uz=3.9e6, mass=4, q=2, theta=a0, ymax=ymax, SI=True, beam=False)

particles = Hepart

#%% Analysis
vmax = np.max([particle.vz for particle in particles])
dt = dz/(2*vmax)
test = ZAP_DEC(particles, domain, dt=dt)
test.run(dt*10000)

#%% Plot
fig, ax = plt.subplots(dpi=600, figsize=(10,2))
test.trajectories(ax, kV=True);

for idx, el in enumerate(electrodes):
    print(f'Electrode {idx}, ({el.style}), efficiency = {el.efficiency()*100:.1f}%')

print(f'Total efficiency: {test.total_efficiency()*100:0.1f}%')
```