Use Authorization

In presenting this dissertation in partial fulfillment of the requirements for an advanced degree at Idaho State University, I agree that the Library shall make it freely available for inspection. I further state that permission to download and/or print my dissertation for scholarly purposes may be granted by the Dean of the Graduate School, Dean of my academic division, or by the University Librarian. It is understood that any copying or publication of this dissertation for financial gain shall not be allowed without my written permission.

Signature _____

Date _____

A PRACTICAL APPROACH TO HYPERVELOCITY PROJECTILE FUSION

by

Lucas Beveridge

Dissertation

submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy in the Department of Nuclear Engineering and Health Physics

Idaho State University

Summer 2019

To the Graduate Faculty:

The members of the committee appointed to examine the thesis of LUCAS BEVERIDGE find it satisfactory and recommend that it be accepted.

_____

George Imel,
Major Advisor

_____

Jay Kunze,
Committee Member

_____

Chad Pope,
Committee Member

_____

William Taitano,
Committee Member

_____

Kenneth Bosworth,
Graduate Faculty Representative

VITA

Lucas was born and raised in Oregon. From a very young age, he was inspired to be engineer and pursued electronics and model rockets. In high school, he led a group of fellow students in the Team America Rocketry Challenge, which went to the national finals in 2003. He then attended Embry Riddle Aeronautical University to pursue Aerospace Engineering and Space Physics and graduated in 2010 with a B.S. in Space Physics. Lucas got his masters in Nuclear Science and Engineering at Idaho State University in 2016 and recently began working at Los Alamos National Laboratory as an R&D Engineer.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

MCF	Magnetic Confinement Fusion
ICF	Inertial Confinement Fusion
MTF	Magnetized Target Fusion
DT	Deuterium/Tritium
DD	Deuterium/Deuterium
FEL	Free-Electron Laser
LiD	Lithium Deuteride
HEDP	High-Energy Density Plasma
MHD	Magnetohydrodynamics
MCNP	Monte Carlo N-Particle
DOE	Department of Energy
SNF	Spent Nuclear Fuel

A PRACTICAL APPROACH TO HYPERVELOCITY PROJECTILE FUSION

Dissertation Abstract

Idaho State University (2019)

The goal of this thesis work is to investigate a type of inertial confinement fusion concept that primarily uses hypervelocity projectiles to heat and compress the fusion fuel. The basic concept was originally developed in the 1960's, but required impact velocities up to 1000 km/s (about 0.3% of the speed of light). Conventional inertial confinement requires extremely high densities to achieve ignition, which causes Rayleigh Taylor and related instabilities. The new concept includes two key ideas, with a number of other possible improvements, that should alleviate these issues. First, the fuel is preheated just before impact so that the necessary impact velocity is reduced to 10's of km/s. Second, the fuel begins at a lower density such that when it is compressed, the fuel density is comparable to the surrounding density, thus preventing the growth of Rayleigh-Taylor instabilities. Methods for improving the performance further were also investigated, including amplifying the energy release by having a shell that releases additional energy, having multiple smaller fueled cavities to help pre-heat the main fuel cavity, as well as theta-pinch and flux compression. Methods for preheating the fuel, economics, spin-off applications, and a possible experiment were also studied. The likely best configuration would have a coil-gun firing a lead projectile at 15-20 km/s, with a lithium-6 metal target with an initial fuel density around 50 kg/m^3. Preheating could be accomplished with a combination of methods including Joule heating, external neutrons beams, or other pulsed plasma heating methods. It appears that this approach is feasible with existing technology, and would require minimal research and expense to reach ignition.

Key Words: Inertial confinement, fusion, impact fusion, magnetized target fusion, ICF, MTF

Chapter I: Introduction

Basics

The goal of this project was to establish a feasible method for using hypervelocity projectiles to heat and compress fusion fuel to at least break-even conditions, if not ignition conditions. The primary heating mechanism is adiabatic compression, but additional heating methods are required because the required confinement time for impact heating alone is so short as to require velocities as high as 1000 km/s, which is not practical. Other approaches to fusion have proven to be more difficult and expensive than originally anticipated, and as of present none of them has reached ignition. This approach promises to be far simpler and easier to develop. In principle, it uses existing technology and will require minimal development and expense to reach ignition.

Most fusion energy concepts fall into two main regimes: low-density and long confinement time (magnetic confinement fusion, MCF), and high-density, short confinement time (inertial confinement fusion, ICF). There is also a middle regime that includes magnetized target fusion (MTF), where the densities aren't as high, but the confinement time is longer than for ICF. This concept requires densities higher than typical MTF concepts, but much lower than ICF. The density and temperature conditions are driven by the Lawson Criterion [1], which is a rough means to determine the necessary conditions to reach positive energy gain by balancing energy production and losses:

$$nT\tau_E \geq \frac{12k_B}{E_{ch}}\frac{T^2}{\langle\sigma v\rangle} \quad (1)$$

where n is the fuel atom density, τ_E is the confinement time, k_B is Boltzmann's constant, T is plasma temperature (in keV), and $\langle\sigma v\rangle$ is the reaction cross section averaged over a Maxwellian velocity distribution at a given temperature, and E_{ch} is the energy of the charged particles

produced by the fusion reactions. This expression is a bit simplistic, as it does not include many important loss mechanisms such as heat conduction, bremsstrahlung, radiation, or instabilities. It only includes energy density. Nonetheless, it is useful for developing intuition. These other losses are discussed below along with brief descriptions of other fusion concepts. The minimum of the right-hand side of equation 1 for deuterium and tritium fusion is about $3x10^{21}$ [keV s/m^3] which occurs at a temperature of 14 [keV]. The confinement time can be approximated as the radius or linear dimension of the cavity divided by the thermal velocity. So, if a cavity with a radius of 1mm, and if the fuel has the same density as uranium (which is a probable target material, and would minimize instabilities), the left-hand side is about $6.72x10^{22}$ [keV s/m^3], which is indeed much greater than the minimum.

Fusion Approaches

Magnetic confinement is one of the approaches that have been heavily funded in the form of Tokamaks and Stellarators. The details of the individual reactor designs are not relevant here other than to mention that all MCF concepts have issues with magnetohydrodynamic (MHD) instabilities. The instabilities combined with the necessary confinement time to reach break-even (several seconds) make them very complex and expensive. Hence the reason they have taken decades to develop.

Inertial confinement fusion has had similar difficulties, although the conditions are very different. One form of inertial confinement has already been proven to work, but requires a nuclear weapon to ignite the plasma, so is not practical for power generation. For power generation, other means of compression are required, such as high-powered lasers, or particle beams. The details of the various permutations are beyond the scope of this introduction, but some general statements can be made.

Confinement times of only nanoseconds are required, but the fuel densities at ignition can be as high as 500-1000 g/cm^3. This causes Rayleigh-Taylor instabilities to form. Rayleigh-Taylor instability magnifies small imperfections (such as a bump on the interface surface between the fluids) in the interface between the fuel and imploding capsule, and grows exponentially with the ratio of the difference in material densities [2]. This is also a major loss mechanism in other inertial confinement schemes [3]. The approximate growth rate of an initial amplitude of h_0 on the interface surface under acceleration is:

$$h \approx h_0 \, Exp\left[t\sqrt{\frac{\rho_h - \rho_l}{\rho_h + \rho_l} \, g \, \alpha}\right] \qquad (2)$$

h is the height of the disturbance at time t, ρ_h and ρ_l are the densities of the two fluids, g is the acceleration (which is quite large during compression), and α is the spatial wavenumber. This expression is only valid for the initial growth. As the instability grows, it becomes more complicated and Kelvin-Helmholtz instability (vortices) can form along the filaments, although it can also form under other circumstances (such as behind a shock moving tangential to a wall). Ideally, the densities of the two fluids will be similar, so that the time-scale of the growth rate is shorter than the confinement time. This will minimize the mixing. However, because ICF plasmas reach such high densities, that is not feasible and this the fuel capsule must be extremely smooth to reduce h_0. An illustration of the progression of Rayleigh-Taylor is shown below:

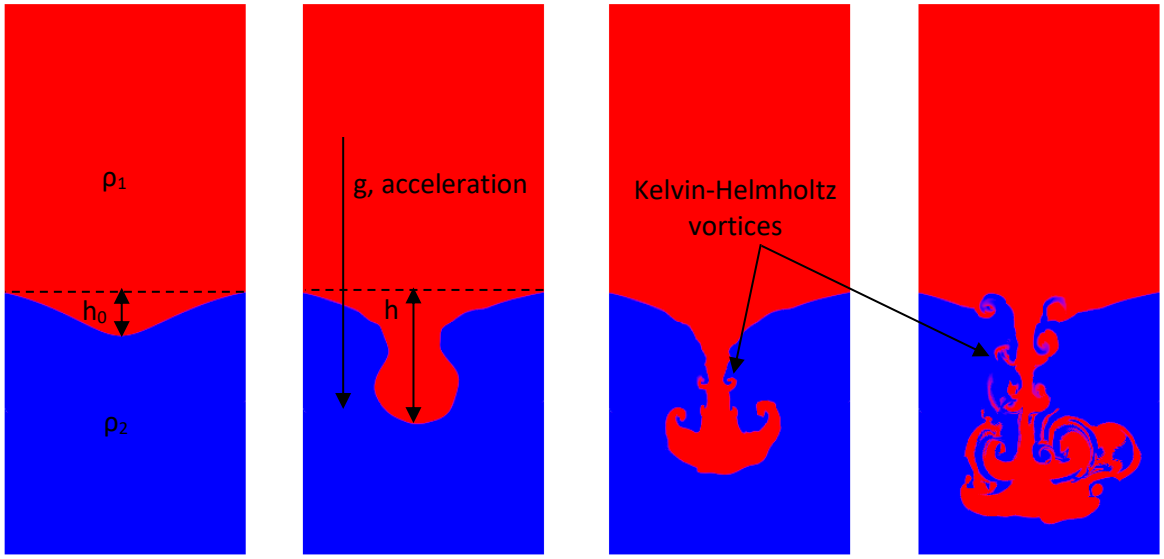


Figure 1. Illustration of Rayleigh-Taylor Instability

Magnetized target fusion MTF is an interesting concept and has similar issues, although the density is much lower than ICF, with confinement times on the order of a microsecond. These approaches appear promising, but have not had as much support. However, there is some recent work on this by private companies such as General Fusion [4].

Hypervelocity Guns

If a gun were used to reach the same conditions as typical ICF (plasma densities of 1000 g/cm^3, and temperatures ~10 keV) without any additional heating, the projectile would have to be fired at speeds up to 1000 km/s [5]. However, this is not practical, and the implosion would not be symmetrical enough to prevent hydrodynamic instabilities at such high densities.

There are a few options for firing projectiles at velocities up 10-20 km/s. The goal of this project was to make use of guns that already exist, or could easily be built based on existing technology. Two-stage light gas guns routinely achieve 7-8 km/s for research use [6]. Rail-guns and coil-guns can also reach similar velocities with large projectiles, and should be able to reach even higher velocities. Gas-guns are limited to the speed of sound in the accelerating gas, which

is unlikely to get much higher than about 8 km/s. Additionally, gas guns cannot be fired rapidly since they rely on disposable components that have to be replaced between shots. Similarly, rail-guns suffer tremendous ablation during a shot due to the projectile needing to be in physical contact with the rails, and would not be able to fire many rounds before requiring major servicing. This effectively rules out both as options for a practical power generation system. Coil-guns however show great promise because they do not require physical contact with the projectile; they have essentially no theoretical limit on the maximum speed, and can have a high repetition rate which will be necessary for a power plant [7].

It is unclear if coil-guns have been investigated for shooting at such high speeds, but there is nothing apparent that would prevent it. These devices have been studied primarily as weapons, and as such, do not need to reach such high velocities. The primary focus of this project was investigating the reactor physics of the plasma, but rough calculations indicate it should be possible since the velocities are only somewhat greater than what has been demonstrated.

Justification

A few simple calculations can be done to determine if the overall concept is feasible. The densities and fuel volumes differ significantly from other ICF or MCF fusion approaches. The fuel density will be assumed to be 20 g/cm^3 which is roughly equal to depleted uranium or lead. Since the target will likely be made from one of those two, this will prevent Rayleigh-Taylor instabilities from growing as the plasma reaches ignition temperatures since the densities are similar. The temperature will be assumed to be 25 keV, which is below the temperature for peak cross section for deuterium-tritium (DT) fusion, but is also easier to reach, and is a more conservative value. For inertial confinement fusion, the confinement time can be estimated as the

ratio of the radius (or characteristic dimension) R and the thermal velocity [8]:

$$\tau = \frac{R}{\sqrt{\frac{k_B T}{m_i}}} \tag{3}$$

where k_B is Boltzmann's constant, T is temperature, and m_i is the average ion mass. In this case, the final dimension (radius) was assumed to be 1mm, after being compressed down from several centimeters, which results in a confinement time of ~1 ns. This is likely conservative since a cavity of several centimeters collapsing at 10 km/s will take several microseconds to reach the minimum volume. The total time where significant fusion energy release occurs is thus likely much longer than 1 ns.

Combining the confinement time with the Lawson Criterion (equation (1)), the condition for ignition can be derived:

$$\rho\, R \geq 1 g/cm^2 \tag{4}$$

Using the values from above the product of density and radius is 2 g/cm^2, which easily satisfies this relationship. Furthermore, the energy gain can also be computed using the confinement time:

$$Q = \frac{\langle \sigma v \rangle\, E_{tot}\, n\, \tau}{k_B T} \tag{5}$$

where E_{tot} is the total energy released by fusion (including neutrons and (n,α) reactions), again, using the values from above, Q ~ 1800. This is probably optimistic, since it does not include losses such as heat conduction and bremsstrahlung radiation, but is nonetheless encouraging.

Bremsstrahlung can be minimized relative to fusion energy at an optimum burn temperature of about 13 keV, and the maximum ratio of fusion energy to bremsstrahlung is about 280 [8]. Blackbody radiation losses are negligible at these densities since the plasma is optically thin [8]. This can be verified by assuming the fuel is at solid-state density (n = 5x10^{22} cm^{-3}), and computing the cross section from the optimal burn temperature $\sigma = 2\, T^{-3.5}$ which is about

$2x10^{26}$ cm^2 for DT and leads to a mean free path of about 1000 cm, which is far larger than any target.

The next major element is preheating the fuel. There are a multitude of possible means to accomplish this, but the necessary energy required to heat the fuel can be estimated to determine if this is feasible. Using 500 eV as the preheat temperature, the energy required is:

$$E_{pre} = \frac{3}{2} N k_B T \quad (6)$$

Using the final fuel density and 1mm radius to compute the total number of atoms, we get N = $2x10^{22}$ atoms, and E_{pre} = 400kJ. This requirement is quite modest compared to the energy released and may be possible to achieve with lasers or plasma injectors in the near future. Other possibilities were investigated during this project, such as using external neutron beams to induce (n,α) reactions in the fuel cavity.

Other System Enhancements

Other methods of compressing and heating plasmas such as amplifying the energy released with a shell that releases additional energy upon neutron capture, theta-pinch (or other magnetic confinement), and stepping up the energy release with multiple cavities in tandem could also be included to enhance it. Adding a theta-pinch would not fully confine the plasma, but could help reduce electron conduction loss, and also result in flux-compression which would cause positive feedback in pressure and temperature as the cavity is compressed. This could also be used in tandem with amplification (this is a new term used here to refer to something similar to boosting, but works differently in this context where the cavity could be surrounded by depleted uranium or Lithium-6), causing multiple positive feedbacks helping to compress the fuel. Multiple sequential steps could be used as well to increase the energy yield, with the projectile collapsing and heating progressively larger fueled regions. None of these, apart from

magnetic fields, can realistically be used with conventional ICF, giving this approach a significant advantage. Additionally, external neutron beams could help preheat the fuel and breed the tritium required.

All of the above-mentioned elements for achieving ignition will be discussed in more detail later, but clearly, numerous configurations could be tested. The major advantage being that many of these permutations could be tested with a single gun facility, making experimental iterations much less expensive and simple to test.

Chapter II: Literature Review

Introduction to the Literature on Hypervelocity Fusion

A number of fusion concepts using hypervelocity guns have been proposed [5], [9], [10], [11], [12] but most require velocities of 500-1000 km/s. There are two main exceptions to this. One requires a hypothetical super-explosive caused by the impact (at around 50 km/s) to briefly drive atoms in the target material into higher energy levels, which would then produce powerful x-rays [8]. The other is being developed by a British startup company that uses hydrodynamic instabilities caused by the impact to confine the plasma [13]. They have not released much information about this, and it's unclear how realistic this is. The most relevant concept (by Winterberg) is discussed below.

There have been a large body of work in the field, which led to a conference on the subject at Los Alamos in the 1970's [14]. However, most of this work concerned the acceleration method, some quite exotic, but all assuming extremely high impact velocities.

In a textbook on inertial confinement fusion also authored by Winterberg, he suggests using magnetic fields to reduce heat conduction losses [8], which would reduce impact velocity, but also requires preheating the fuel to high temperature, which he appears to dismiss. At the time he wrote the book, that may have seemed reasonable, since the required temperature was 100's of eV. Later we will show that this is not such an onerous limitation.

Original Winterberg Concept

The idea of using hypervelocity projectiles was first suggested by Winterberg [5] in the 1960's. The idea is to heat ~10^{22} atoms to a temperature of about 4 keV for ignition, which would require a velocity of 10^8 cm/s (1000 km/s). This speed is also necessary to minimize the confinement time and thus losses. They suggested using traveling wave accelerators with

superconducting projectiles to reach these enormous speeds. There are a several shortcomings to this. First, the assumed heat conduction loss is over-estimated because they use a diffusion approximation, which is not valid at such large temperature gradients. In fact, the maximum heat flux due to conduction is about $1/10^{th}$ of the electron free-stream limit (the maximum current of electrons in the plasma) [15]. Second, the compression cannot be perfectly spherical, although it can be approximated (see Fig. 1), the resulting hydrodynamic losses would be much greater than they anticipated. The losses will be discussed in greater detail later, but this could lead to non-Raleigh Taylor instabilities such as Kelvin-Helmholtz instability. The other losses will be discussed in the next chapter.

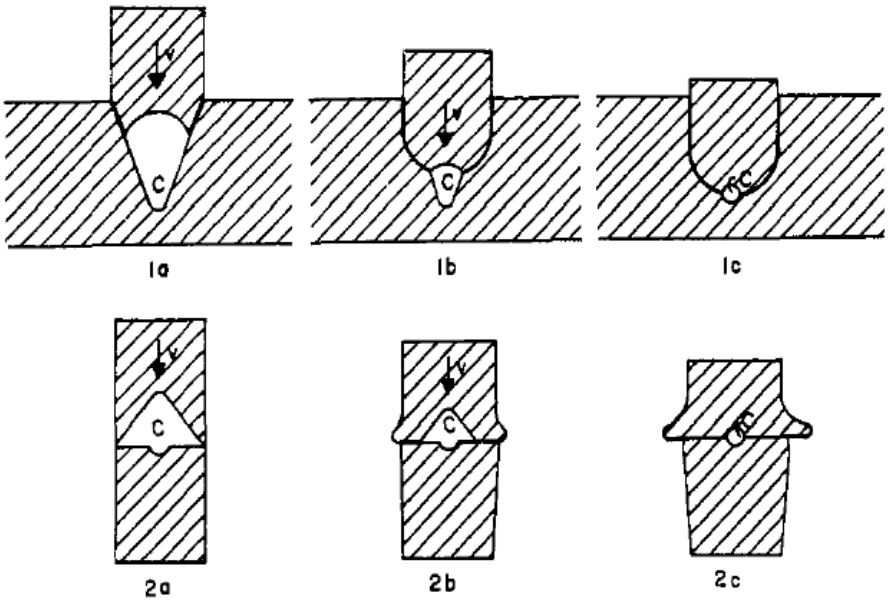


Figure 2. Two examples of conical projectiles collapsing to approximate spherical compression [5]

Gas Guns

There have been quite a variety of acceleration methods proposed for hypervelocity fusion ranging from gas guns (which will be discussed in more detail shortly), to exotic methods such as laser ablation rocket guns [16]. Due to the constraint of lowering the impact speed, the work here focused on more conventional methods such as coil guns and gas guns, which have been demonstrated at the required velocities.

The first concept that will be discussed is perhaps the most obvious, since it is similar to firearms. Gas guns are limited to the speed of sound in the working fluid. Hydrogen is typically used due to its low molecular weight, and high speed of sound, which at room temperature is about 8 km/s. The guns usually function by first accelerating a piston by conventional means, sometimes a gun-powder charge. The piston compresses the light gas in a cylinder until a critical pressure is reached, causing a burst disc to rupture releasing the light gas into a narrower cylinder where the projectile is already loaded. An illustration is shown in the figure below:

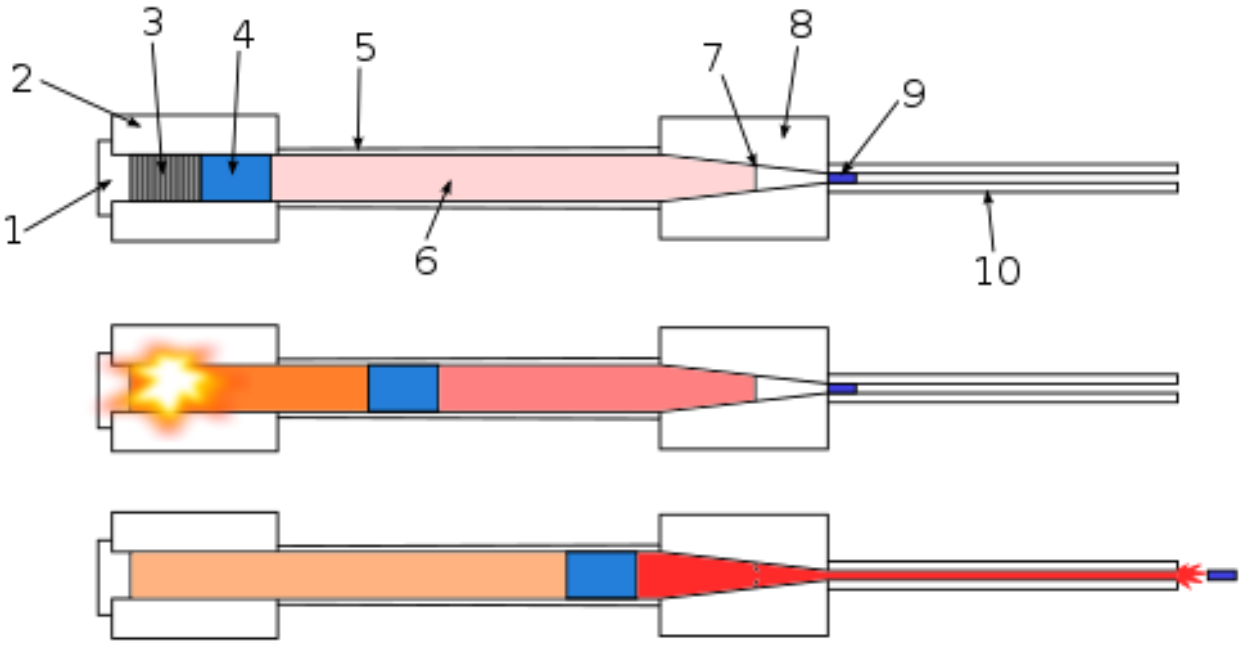


Figure 3. Illustration of a light-gas gun (Johan Fredriksson [CC BY-SA 3.0 (https://creativecommons.org/licenses/by-sa/3.0)]).

Legend: 1. Breech block, 2. Chamber, 3. Propellant charge (gun powder), 4. Piston, 5. Pump tube, 6. Light gas (helium or hydrogen), 7. Rupture disk, 8. High pressure coupling, 9. Projectile, 10. Gun barrel

As mentioned previously, this has some practical issues that would need to be fixed before it could be used in a power-plant. Because these guns are typically used for research, they aren't designed to fire at a high repetition rate. If a single shot releases 30 GJ of energy, the gun would have to fire once every ten seconds to produce 1 GWe on average. Alternatively, the firing rate for a gun could be decreased by adding additional guns. However, currently large gas guns sometimes take up to a day to reset for another shot. If it proves feasible to use a projectile at 7-8 km/s, then this may still be the most appealing option, even if it requires significant design improvements.

Electromagnetic Guns

There are two primary electromagnetic acceleration methods. The first is the rail gun that uses the Lorentz force produced by current flowing through the projectile and a magnetic field to do the acceleration. This is being investigated for military use by various governments, but may not be practical here. Because the current must flow through the projectile, the rails must be in physical contact with it until it leaves the barrel. At the extreme velocities under consideration, the friction is enormous, and the resulting ablation will cause rapid wear of the rails. This may be an insurmountable practical problem for a power plant that must operate for years with minimal maintenance. For this reason, rail guns weren't considered in this work. A simplified illustration

of a rail gun is shown below to illustrate how they work.

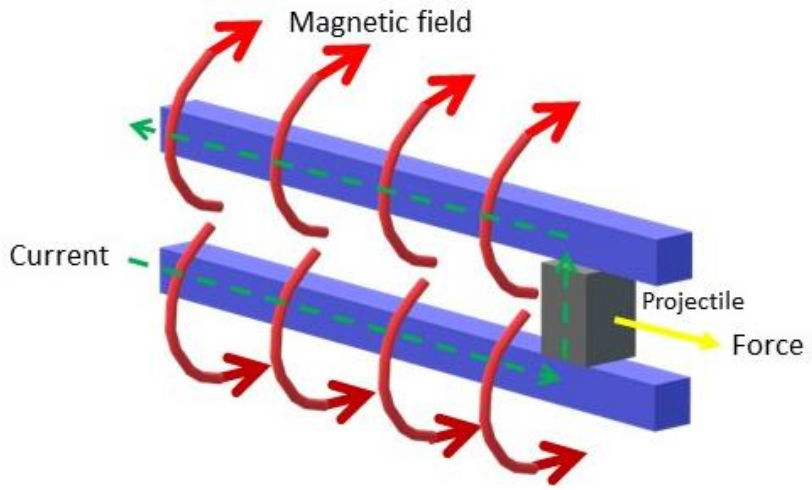


Figure 4. Illustration of a rail gun

Coil guns are the other main option. These are more promising since they can theoretically reach much higher velocities than a gas gun, but don't require physical contact with the projectile. Timing the firing of the coils can be a complicated design problem, but with modern electronics is not difficult. A simplified diagram of a coil-gun is shown below. This shows a rotary switch to illustrate the function of the coils, which have to fire sequentially.

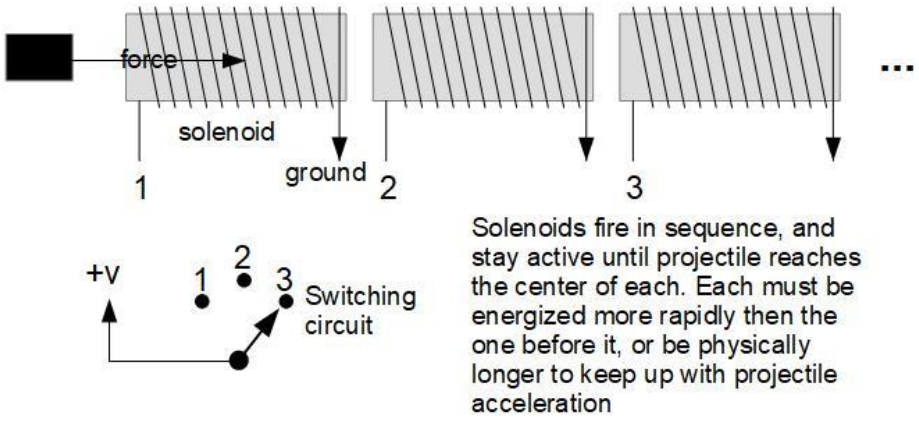


Figure 5. Illustration of a coil gun

Plasma Preheating

There are numerous options for preheating the fuel to the needed minimum initial temperature of several hundred eV. The first option is Joule heating [17], where current is passed through the plasma, and is heated by electrical resistance. Another option is preheating with free-electron lasers (FEL), which already operate at extremely high-power during pulses [18], and could potentially produce the hundreds of kilojoules needed in the short time scale of a shot. There's also the possibility of using plasma injectors, which are fairly well developed for this type of application, but for that reason weren't examined in detail here. An additional class of options is combining other concepts that could help release additional energy beyond just the fuel in the cavity, such as amplification, stepped cavities, and having solid breeding fuel in the cavity. For instance, a powerful external neutron source could be used to preheat a ^{6}Li pellet with deuterium gas or LiD pellet inside the cavity by driving (n,α) reactions and breeding tritium. Energy is released through this reaction in the form of kinetic energy in both an alpha and a triton: $n + Li^{6} \xrightarrow{yields} He^{4}(2.1MeV) + T(2.7MeV)$. This possibility was examined in some detail.

Magnetic Confinement/Flux Compression

Flux compression is a well-known technique for generating extremely high magnetic fields [19], [20], [21]. The concept relies on the field lines being frozen in a conductor, which is then rapidly compressed by explosives or some other method. As long as the time scale is short enough that resistive losses in the conductor (in this case plasma) is negligible. As the device is compressed, current is induced in the conductor, which then produces its own magnetic field that adds to the existing field increasing the contained flux.

The presence of a magnetic field can help confine the electrons within the plasma even without the help of flux compression. Confining the electrons reduces conduction loss since they tend to move towards a wall and carry away heat with them.

This has only had some minor study done in the context of inertial confinement fusion. Winterberg mentions that magnetic fields could be used to reduce electron conduction loss, since the electrons are more strongly affected (due to their low mass) than the ions at high plasma densities [8]. The University of Rochester OMEGA laser facility has shown greater fusion energy yields using magnetic fields and flux compression as well [22]. They got significant results using an 8T magnet, but modern high-temperature superconductors can reach much higher field strengths. This was studied here as another means of adding pressure to the plasma, producing a positive feedback mechanism.

Chapter III: Methodology

Zero-Dimensional Model

Extensive calculations have been done on the new concept, and numerous iterations were tested. The basic approach was to integrate the reaction rate using an adiabatic temperature increase with the temperature dependent reaction cross section to compute the average energy released over the time span of the impact. The speed of the fuel cavity collapse is the same as the impact speed. The stopping point is the moment when the dynamic pressure of the projectile ($\frac{1}{2}\rho v^2$) is equal to the pressure of the plasma. The deceleration of the projectile was not modeled for simplicity, but this should be reasonable since any significant slowing doesn't occur until right before maximum pressure.

Two primary models were eventually settled on, a cylindrical and a spherical case. The main difference mathematically are the expressions for the volume and surface area vs time. Both had a linear dimension (either height or radius) that decreased linearly at a rate equal to the projectile velocity. The calculations are summarized below.

First, adiabatic relationships were used to find the temperature and pressure:

$$T = T_0 \frac{V_0^{\gamma-1}}{V(t)^{\gamma-1}}$$
$$P = P_0 \frac{V_0^{\gamma}}{V(t)^{\gamma}} \quad (7)$$

where T is the plasma temperature, P is the pressure, γ is the ratio of specific heats (c_p/c_v) for a monatomic gas/plasma (5/3), t is time, V_0 is the initial cavity volume, and V is the cavity volume. The length of time to fully compress the fuel was found by setting the dynamic pressure of the projectile to the pressure of the plasma and solving for time. Since the impact pressure isn't being modelled with the plasma, this will approximately account for the back pressure against the piston which eventually stops the compression. There are two different expressions,

one for the cylindrical case, and one for the spherical case:

$$t_{max,cyl} = \frac{1}{v}\left[h_0 - \frac{2^{1/\gamma}\left(\frac{v^2\rho_{proj}}{P_0 V_0^{\gamma}}\right)^{-1/\gamma}}{A(t)}\right]$$
$$t_{max,sph} = \frac{1}{v}\left[R_0 - \left(\frac{\frac{2^{2\gamma-1}\pi}{3}v^2\rho_{proj}}{P_0 V_0^{\gamma}}\right)^{\frac{1}{3\gamma}}\right] \quad (8)$$

Here, ρ_{proj} is the mass density of the projectile, h_0 and R_0 are the initial height and radius of the cavity respectively, and v is the impact velocity. These were used to compute the average gross energy released, as well as total losses. The heating and losses were computed separately and then summed to obtain the net energy released. The fusion energy released over the whole time of compression:

$$E_{gross} = \int_0^{t_{max}} \left(\frac{1}{4}n(t)^2\langle\sigma v\rangle Q_{tot}V(t)\right)dt \quad (9)$$

Once again, V is volume of the cavity, n(t) is the fuel number density as a function of time, Q_{tot} is the total energy released per fusion reaction (23 MeV, including (n,α) reactions in lithium used to breed tritium), and $\langle\sigma v\rangle$ is the reaction rate averaged over a Maxwellian distribution for deuterium and tritium as a function of temperature (in keV) from [23]:

$$\langle\sigma v\rangle = 3.68\times10^{-12}T^{-2/3}\exp(-19.94T^{-1/3})[cm^3/s] \quad (10)$$

This expression for the cross section is only valid up to temperatures of about 25 keV, but the plasma temperature was unlikely to exceed this in the model.

Numerous loss mechanisms were initially included, but proved to be negligible. These include energy lost in deforming the projectile/target, heat conduction through the solid wall of the target (which is only negligible due to the short time-scales), energy lost accelerating the

projectile, and radiation (which was small since the plasma proved to be optically thin, this fact was later also found in Winterberg's textbook). A summary of all the losses considered is shown in the table below for the cylindrical case. The spherical case was similar, and shows that most of these losses were far less than 1% of the energy released.

Table 1. Quasi-1-D model losses

Loss mechanism	Loss [J]	% of energy produced
Deformation	13,611	0.00025
Heat conduction through wall	4,362	0.00008
Radiation	2.55E-41	0
Kinetic energy of projectile	1.50E+06	0.03
1/10 free-streaming conduction	5.32E+09	96.18
Bremsstrahlung	1.21E+08	2.19

The most significant losses were due to electron bremsstrahlung and heat conduction [15] through the plasma to the wall. The expression for bremsstrahlung is from [24]:

$$E_{Brem} = 1.58 \times 10^{-38} z^2 n(t)^2 \left(\frac{T}{e}\right) [Watt/m^3] \tag{11}$$

z is the atomic number for the ions (2.5 for deuterium/tritium), and e is electron charge in Coulomb. As mentioned previously, heat conduction is below what diffusion theory would suggest to about 1/10 of the electron free-stream limit [15] (with units of power):

$$E_{cond} = \frac{1}{10}\frac{3^{3/2}}{2} n(t) k_B T(t) \left(\frac{k_B T(t)}{m_e}\right) A(t) \tag{12}$$

m_e is the mass of an electron, and A(t) is the surface area of the cavity as a function of time. This is the maximum possible loss, so it's a deliberate over-estimate for the sake of being conservative. The two losses are integrated over time to obtain the total energy loss.

$$E_{loss,tot} = \int_0^{t_{max}} E_{Brem} dt + \int_0^{t_{max}} E_{cond} dt + E_{other} \tag{13}$$

This is simply subtracted from the gross energy produced to obtain the net energy released. The

energy output gain Q is computed as the ratio of the energy produced from energy lost and adjusted with the plant thermodynamic efficiency. The Wolfram Mathematica [25] notebooks with the complete calculations for typical results of both the spherical and cylindrical cases are included in Appendices A and B respectively.

One-Dimensional Model

At the suggestion of William Taitano [26], the main model was based on the fluid Euler equations. This is a relatively straightforward system of equations to solve in principle. However, it was not necessary to write a code from scratch. An unnamed open-source finite-volume code written with CLAWPACK's "wave-propagation method" was found that solved the system of a piston in a cylinder with compressible fluids. Although the code was meant to study the interface between two fluids in the cylinder, it was easily modified for one fluid. The original source code can be found at [28], and the files that were modified for this project can be found in appendix C.

There are three main conservation equations in the compressible form of Euler's equations for the fluid mass, momentum, energy, and an equation of state to complete the system:

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \frac{\partial (\rho u)}{\partial x} &= 0 \;\; Mass\ Continuity \\ \frac{\partial (\rho u)}{\partial t} + \frac{\partial (P + \rho u^2)}{\partial x} &= 0 \;\; Momentum \\ \frac{\partial (\rho \epsilon)}{\partial t} + \frac{\partial [(P + \rho \epsilon) u]}{\partial x} &= 0 \;\; Energy \\ P = P(\rho, \epsilon) \;\; &Equation\ of\ State \end{aligned} \tag{14}$$

Again, ρ is density, u is velocity, P is pressure, ε is the specific energy of the fluid (J/m^3), and the single space dimension is x. The code solves these with CLAWPACK, which is a set of libraries and code designed to solve hyperbolic conservation law systems using a finite volume method [29], and this specific code has modified routines for a moving mesh [30]. Originally, the code

was written to study the interface between two fluids, and verified by solving a shock-tube problem. For this study, the fluid properties on either side of the boundary were set equal to one another, and dimensions changed to match the cylindrical case in the quasi-1-D model, as well as increasing the initial piston velocity to 10 km/s. The time limits for each simulation were set according to $t_{max,cyl}$ from equation (8), again this approximates the effect of back pressure.

The code outputs fluid temperature, pressure, and density. These can be used to compute fusion reaction rates, bremsstrahlung, and electron conduction losses. These are the same expressions used in the quasi-1-D code. The Matlab scripts used for post processing and generating plots are in Appendices D, and E. The post processing scripts were used to compute total energy gains by integrating the instantaneous power per cell over all cells and total simulation time:

$$\mathrm{E}_{tot} = \sum_{t=0}^{t_{end}} \sum_{x=1}^{1/\Delta x} p_{cell}\, \Delta t \qquad (15)$$

E_{tot} is the net energy produced, t_{end} is the time when the collapse ends, $1/\Delta x$ is the number of grid elements, Δt is the time step length, and p_{cell} is the instantaneous net power in each cell.

Several cases were studied with various fractions of the worst-case conduction loss, which is equal to ~$1/10^{th}$ of the electron free-streaming limit. The best case with no conduction loss was also examined, this represents an ideal case where magnetic fields practically eliminate it. Another calculation was run with a longer time limit, allowing the plasma to reach a pressure equivalent to a 15 km/s impact, this was meant to represent the additional pressure exerted by feedback from an exploding shell around the cavity or magnetic flux compression.The code uses a high-resolution finite-volume wave-propagation method developed in [31]. The moving grid has a uniform spacing, and a typical grid for this project is shown below with 1000 mesh

elements. The y-axis is time, and the x-axis is the space dimension along the length of the cylinder. The grid elements are so close together that they were made different colors so that they could be distinguished. The right side of the mesh can be seen sloping to the left. This is the piston boundary moving in time.

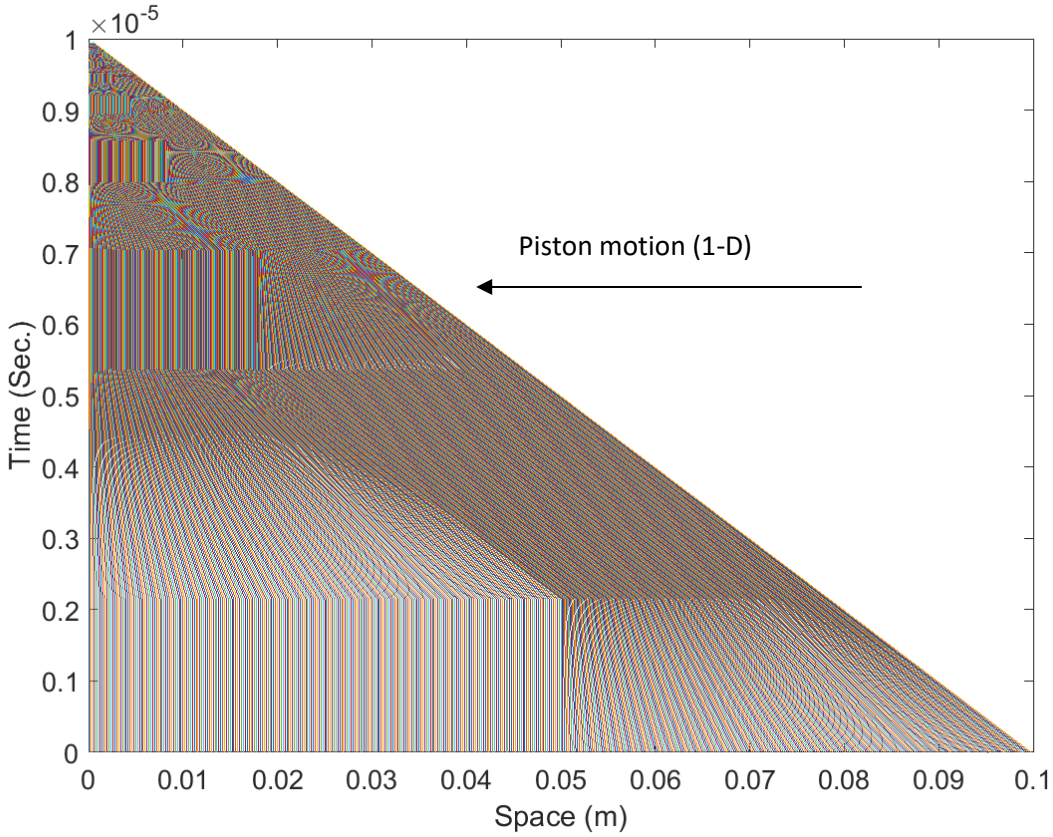


Figure 6. A typical mesh for a collapsing cylinder of fuel with 1000 mesh elements

The code has some limitations that should be mentioned. First, it would not converge when the source terms were added to the energy equation. However, it was tested with source terms set to zero, and with the source terms subroutine file included in the main program, still would not converge. This appears to be a bug in the original code. If time allowed, this would have been fixed, but it was decided to be non-essential.

Additionally, setting the initial conditions with a high energy density fuel (500 eV and 250 g/cm^3) also prevented the code from converging. This appears to simply be a shortcoming of the CLAWPACK solver, and was avoided by increasing the initial temperature to about 1.5 keV and decreasing the initial density. The essential results were still very useful, and could be compared to other calculations.

One-Dimensional Model Code Verification

The code is has been verified against a standard shock-tube problem [30]. However, due to the extreme conditions in our system, we have also compared predictions of pressure and temperature versus time to a simple adiabatic model. Each data point corresponded to the moment when a shock-wave reached an opposite wall and the conditions in the cavity were uniform. This avoids having to find some average value across strong shocks. The adiabatic model was the same as Eq. (7) , taking the ratio of T/T_0 and p/p_0. The initial conditions for both the adiabatic calculation and the simulation were the same. The figures below show the comparison plots with error bars on the simulation data points corresponding to +/- 10%. This shows that all the simulation data points were well within 10% of the adiabatic relationships, and therefore still accurate even at the high energy densities of the plasma.

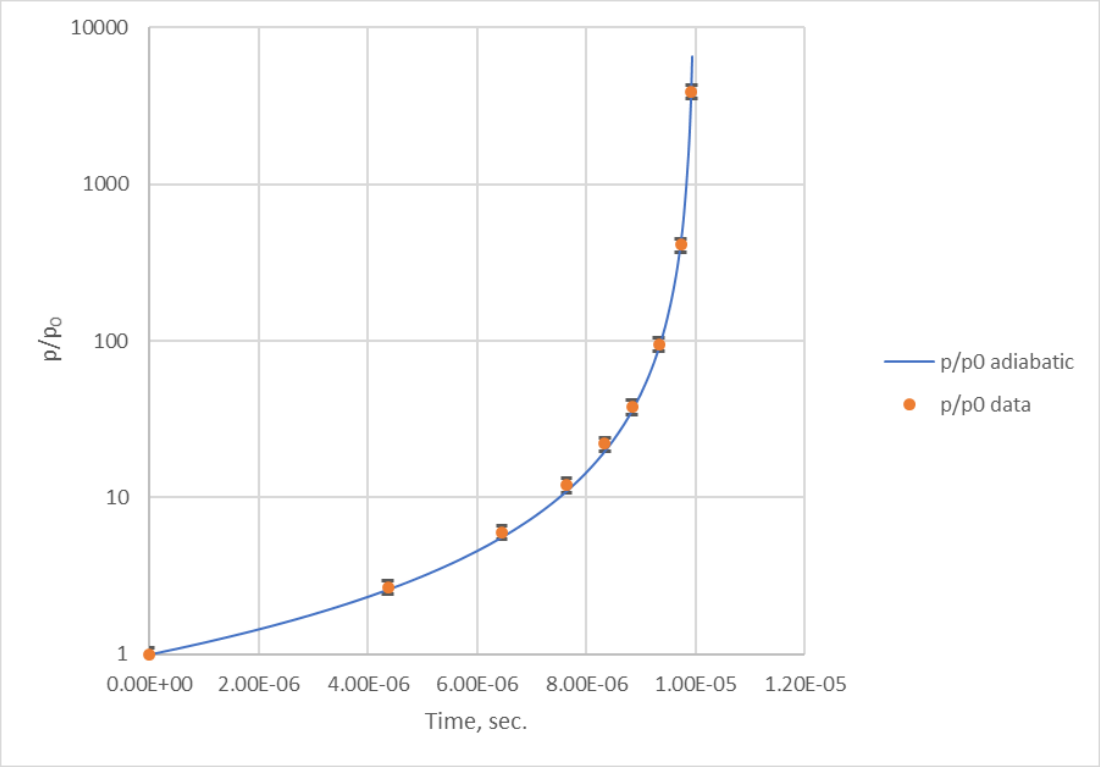


Figure 7. Comparison of pressure predictions

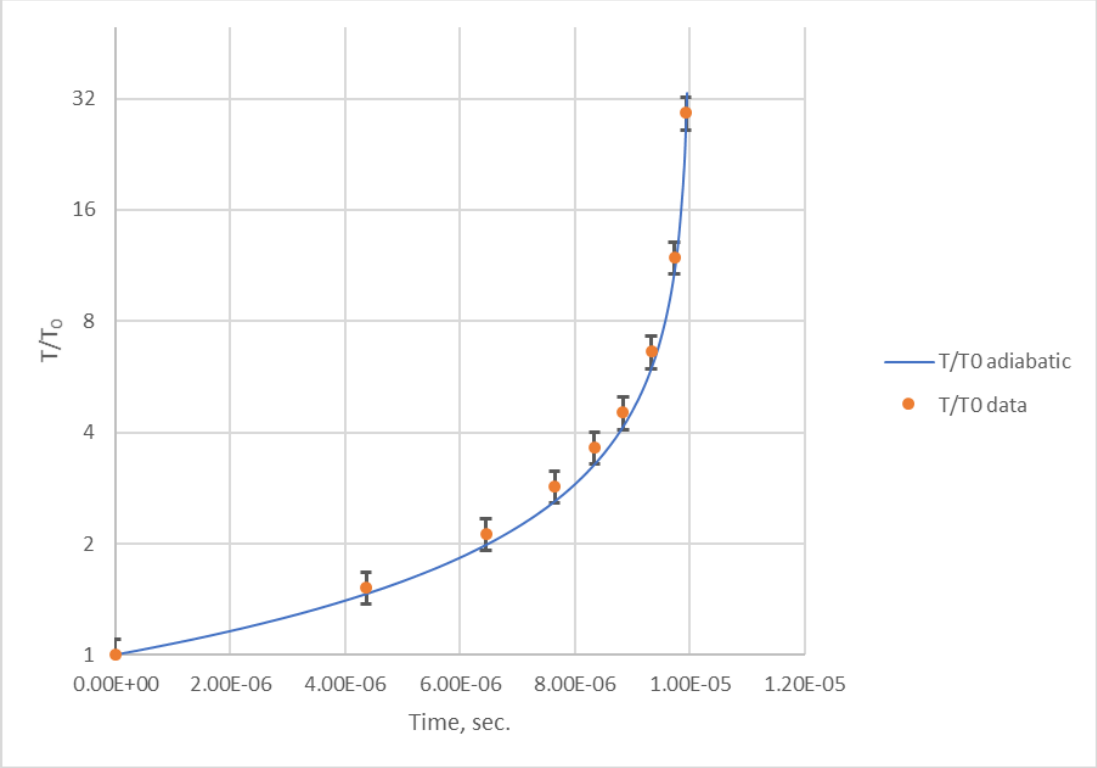


Figure 8. Comparison of temperature predictions

Clearly, there is quite good agreement between the CLAWPACK predictions and analytical theory. The Mathematica notebook used to do these calculations is in Appendix F.

Although the original code authors did a shock-tube verification, this behavior was also tested in this work since the values being computed are far more extreme than their calculations (their calculations were meant more as toy models for demonstrating the capabilities of CLAWPACK). Strong shocks can be seen in plots traveling back and forth between the fixed wall and the piston with pressure, density, and temperature increasing behind the shock. Below is a set of plots for two different time steps to illustrate this:

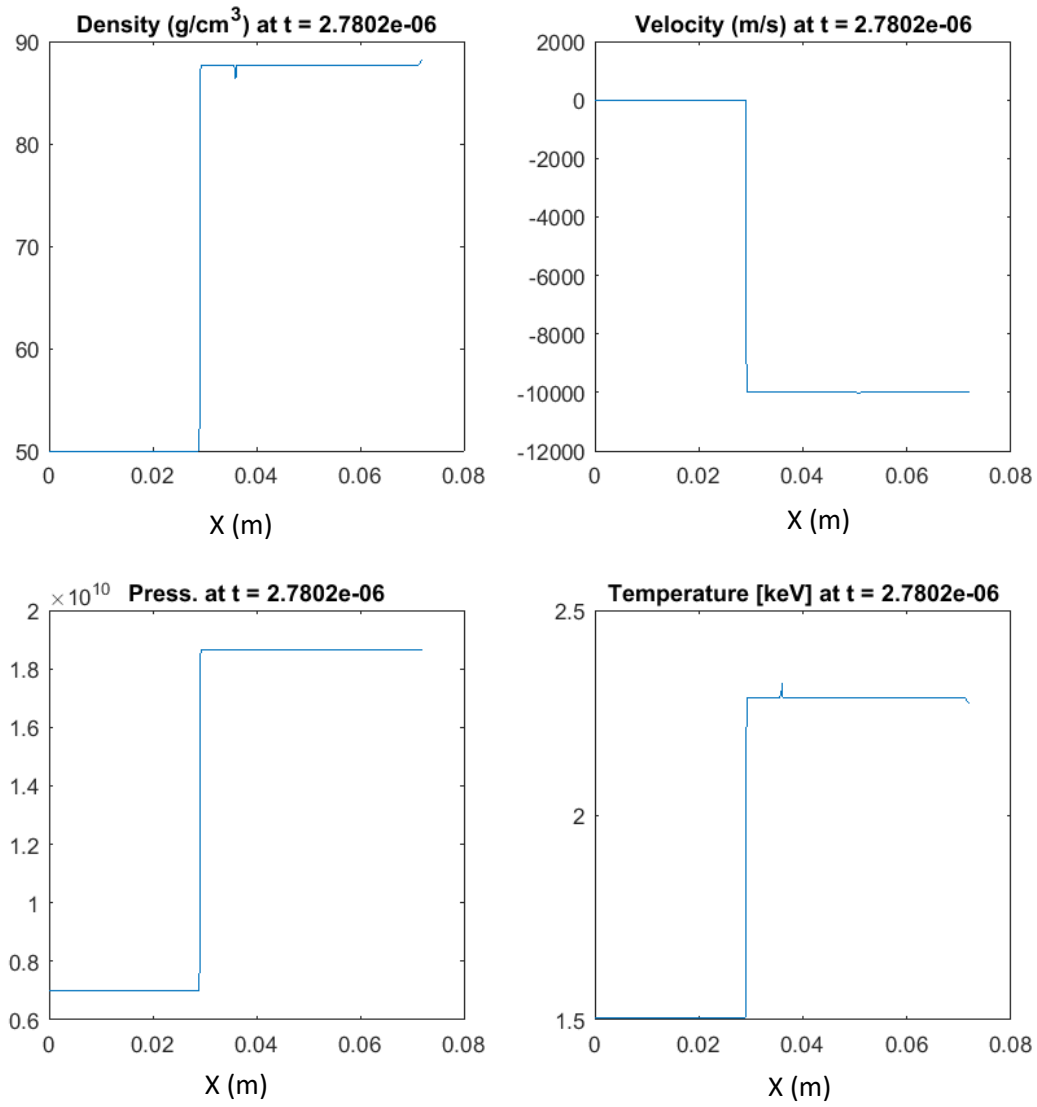


Figure 9. Plots of values along the length of cavity cylinder at 2.8μs. The x-axis is the distance between the wall and piston in meters, and pressure is in Pascals.

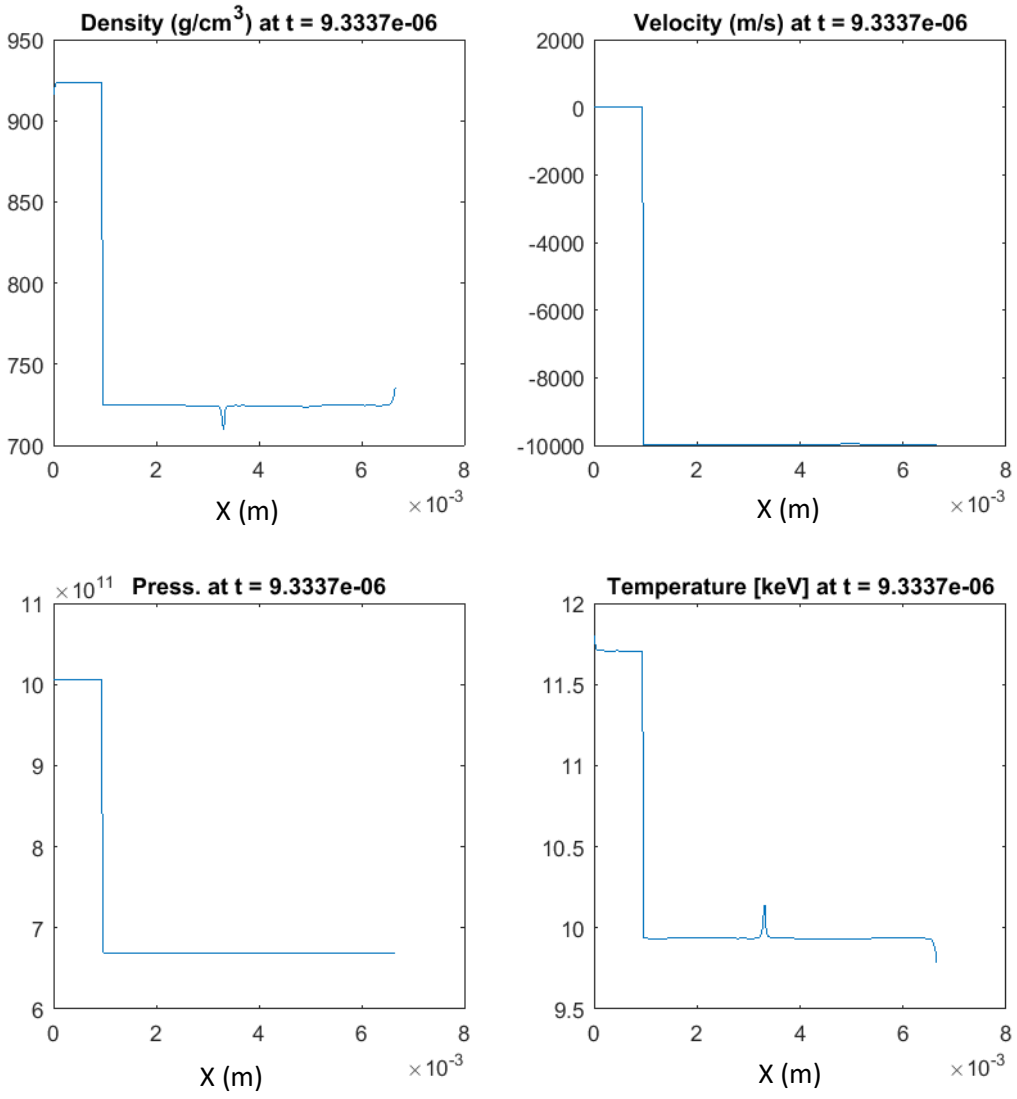


Figure 10. Plots of values along the length of cavity cylinder at 9.3μs (roughly where the unamplified case ends). The x-axis is the distance between the wall and piston in meters, and pressure is in Pascals.

In the preceding plots, the x-axis scale in them is different due to the compression. It should be noted that there are obvious spikes in the middle of the plots; these are numerical artifacts caused by the boundary of the two fluids. The effect of the artifact is negligible for this work.

Preheating and Stepped Ignition

Since preheating the fuel is a critical element in this concept, a number of options were examined. There are far more possible methods of heating the plasma than could be examined in detail here, but many of those are already well developed for fusion systems, such as powerful lasers, and plasma injectors.

The hotter the plasma is initially; the less energy is required from the projectile to reach ignition, even though this also increases the initial pressure that the impact must overcome for compression. We can use the initial temperature from the quasi-1-D models (100eV) as a lower limit to estimate how much energy is required for this heating. For the quantity of fuel in these simulations, this is on the order of 1MJ via eq. (6). This is quite substantial, and may require several methods be employed simultaneously.

The first method considered is Joule heating. This is the simplest method, and has been used in fusion experiments for decades. It has some limitations however. The maximum temperature it can reach is asymptotic and is a function of density and the distance between electrodes. The relevant expressions (which assume Spitzer resistivity) for calculating this are shown below and come from [17]:

$$T_m = \frac{2.65\text{x}10^{27}\ [K]}{N} \tag{16}$$

N is the line-density of the fuel:

$$\text{N} = \int_0^a 2\pi n r\ dr \tag{17}$$

Here, a is the radius of the plasma and n is the number density. The time required to reach 90% of T_m is:

$$\tau = 3.25\text{x}10^{-10}\ a^2\ {T_m}^{3/2} \tag{18}$$

It can be easily seen that Joule heating by itself can't reach high enough temperatures at these densities, but it could be used for heating a lower density fuel in an adjacent cavity compressed by the projectile, which could serve as a neutron source for heating a denser plasma, which is also then compressed by the projectile. A simple MCNP6 model of this scenario was carried out with a deuterium/tritium neutron spectrum uniformly distributed in a smaller fuel cavity above the main fuel cavity [37]. The main fuel cavity also had a pellet of lithium deuteride (LiD) that would undergo (n,α) reactions that both breed tritium and heat the plasma while releasing about 4.8 MeV. The energy deposition was tracked with an F6 tally, and the neutron flux plotted for the whole geometry. The F6 tally is a volume integrated flux multiplied by cross sections and energy for reaction to estimate the energy deposited in the volume. Three versions of this were examined. One that has a graphite target to thermalize neutrons to increase capture probability in the LiD. This would allow the walls around the cavity to also release energy and potentially exert pressure and produce tritium. The LiD shell will be discussed in more detail in the next section. The other has a target made out of LiD. The third has heated fuel within the main cavity surrounding the LiD pellet. This technique in general will be referred to as "stepped ignition". The MCNP input files are in Appendices G, H, I.

Another method of preheating the fuel is to have an external neutron beam bombard the target. This is similar to the stepped ignition case in that a pellet of Li-6 or LiD is placed inside the main fuel cavity and the resulting reactions heat and breed tritium. This was modeled as a large beam of 0.025eV neutrons impinging on the target. Again, the energy released was tracked with an F6 tally, and the neutron flux was plotted over the whole geometry as a diagnostic for the models. The flux plots were not used for any calculations. The input file is in Appendix J.

One method of producing a neutron beam was examined (although there are others that are already well studied, such as spallation). A beam of 10 MeV ^{4}He ions impacting on a beryllium target would induce (α,n) reactions. The needed beam current for an accelerator can be found from the energy released by the (n,α) reactions in the LiD. The (α,n) cross section for ^{9}Be at 10 MeV is only about 0.7 barns, and so the resulting reaction rate requires an enormous beam current of about 750 kA. Although this is quite high, it is over a short period of time and may still be feasible. Nonetheless, another source of neutrons may make this approach more appealing.

Amplification and Feedback

Another enhancement that could greatly augment the preheating and kinetic impact is to use the neutrons produced by the first fusion reactions during compression to release additional energy in the material around the cavity, imparting additional pressure. There are a number of material options for the target, including metals and alloys that undergo (n,α) reactions, depleted uranium, or even spent nuclear fuel (SNF). Uranium and SNF are not desirable due to the production of fission products and the associated complexity of handing the contamination in the target chamber, but has the benefit of consuming what is currently waste.

The concept was modelled in MCNP6 with ^{10}B, ^{6}Li metal, LiD, LiD liner and ^{238}U, and ^{238}U metal for a range of cavity heights (representing the collapse during impact). The energy deposited in the target material was tracked with F6 tallies. This shows how the effect changes as the cavity collapses. The neutron flux at the inside surface of the cavity was also tracked with an F2 tally. Unlike the F6, an F2 tally only tracks the instantaneous flux through a surface. This shows how the flux at the cavity surface changes as it is compressed. The input files are in Appendices K through O.

The net effect is a positive feedback that increases the confinement pressure and density. A great deal of compression needs to be provided by the projectile, but the increased pressure reduces the required impact velocity. This will be discussed more in the results. Feedback loop calculations were not done because there's too many unknowns without more sophisticated 3D calculations with coupled physics to model the plasma/target interface. To approximate the effect of amplification, a 1D model was run to a greater time limit (and thus higher pressure and density) to determine the effect on gain. The impact velocity was still 10 km/s, but the final pressure and density were equivalent to a 15 km/s impact. It should be noted that this additional pressure will be exerted mostly in the radial direction, which the code cannot account for, but would produce a radially converging shock which would better compress the fuel since it's closer to spherical compression. Winterberg talks about other methods of doing this by shaping the projectile and target. This was shown in Figure 2.

Chapter IV: Results

Quasi-One-Dimensional Model

This basic model was used to test various ideas to make improvements on the hypervelocity fusion concept. Numerous combinations of preheating (increased initial temperature), initial fuel density, fuel mass, projectile velocities, cavity volumes, and cavity dimensions were tried. Through trial and error, two key improvements were identified. First, reducing the initial fuel density from the solid density of deuterium/tritium ice allows the final density at fusion temperatures to be made nearly identical to the surrounding material, which practically eliminates Rayleigh-Taylor instability. Second, preheating the fuel before or during impact to a temperature of 100's of eV (millions of Kelvin) allows for a far lower impact velocity (10-20 km/s) to reach break-even or ignition.

The model was verified by reproducing Winterberg and others results. Lowering the initial temperature to ambient conditions, and setting the initial fuel density equal to the solid density of DT ice resulted in a required impact velocity of about 1000 km/s to exceed break-even. The results also broadly agree with simple triple-product calculations (30GJ for triple product, ~16-18GJ for the quasi-1-D code).

There were two main cases studied. First, a spherical cavity with isentropic compression, which is somewhat idealized. Second, a cylindrical case that was compressed only in the axial direction. In reality, it would be some combination of these two, so it was thought they could provide rough upper and lower bounds on performance.

The results for the two cases are summarized in the table below:

Table 2. Quasi-1-D model results

	Impact speed (km/s)	R_0/h_0 (mm)	ρ_0 (kg/m^3)	ρ_f (kg/m^3)	Q	Net Energy per Shot [GJ]
Cylinder	10	5/100	103.7	18,532	1.01	18.7
Sphere	15	5/NA	62.27	17,995	1.40	16.3

R_0/h_0 is the ratio of initial radius to initial height, ρ_f is the final density of the fuel at the moment that plasma pressure is equal to impact pressure. The energy gain for both of these is modest, but these models were adjusted to determine the minimum conditions for break-even. Higher velocities and larger fuel volumes can dramatically increase this. Also note that the final fuel densities are similar to the density of depleted uranium (or lead which roughly doubles in density upon impact). The initial temperature for both of these was set to 100eV (1.16×10^6 K). The net energy released through fusion is also comparable to the energy calculated through the triple product. Below are plots of cavity volume, pressure, temperature, and number density for the cylindrical case.

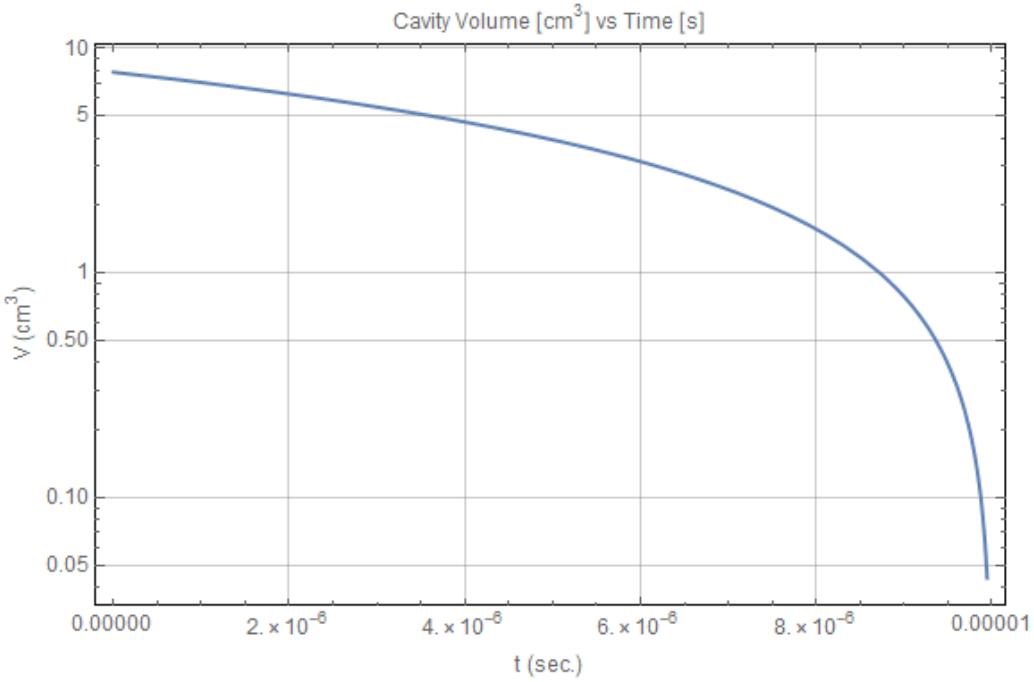


Figure 11. Plot of cavity volume in cylindrical case

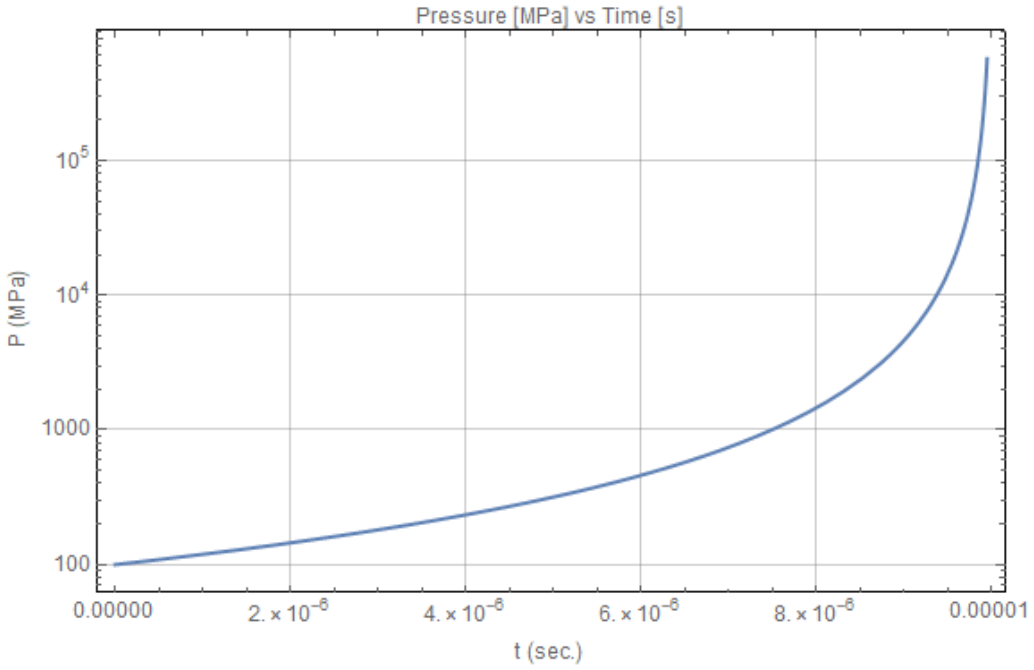


Figure 12. Plot of pressure in cylindrical case

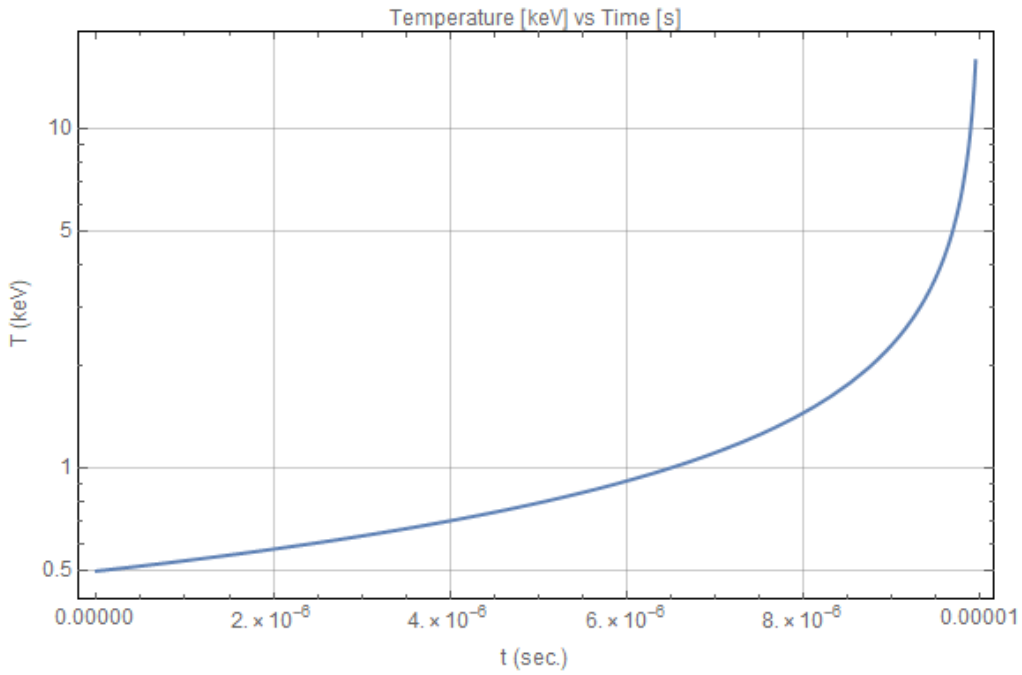


Figure 13. Plot of temperature in cylindrical case

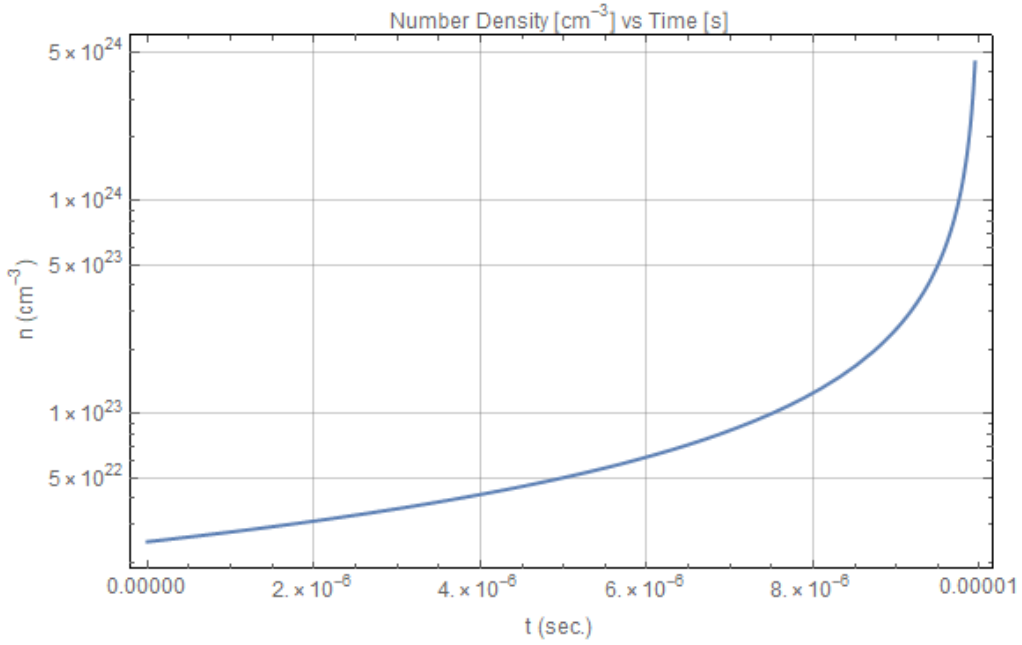


Figure 14. Plot of number density in cylindrical case

Below are plots of volume, pressure, temperature, and number density for the spherical uniform compression case.

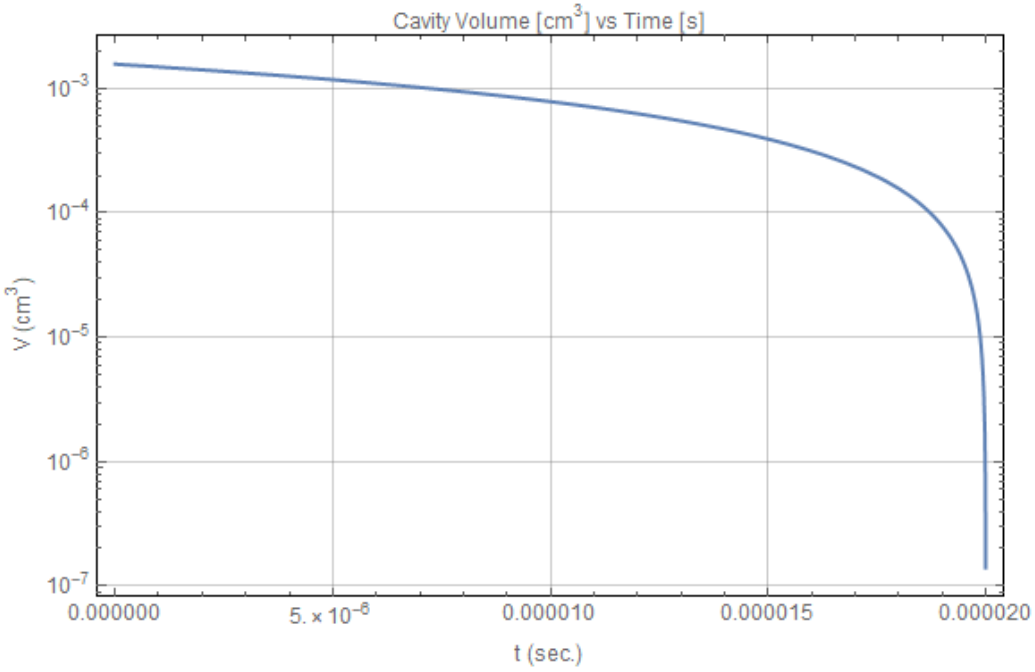


Figure 15. Plot of cavity volume in spherical case

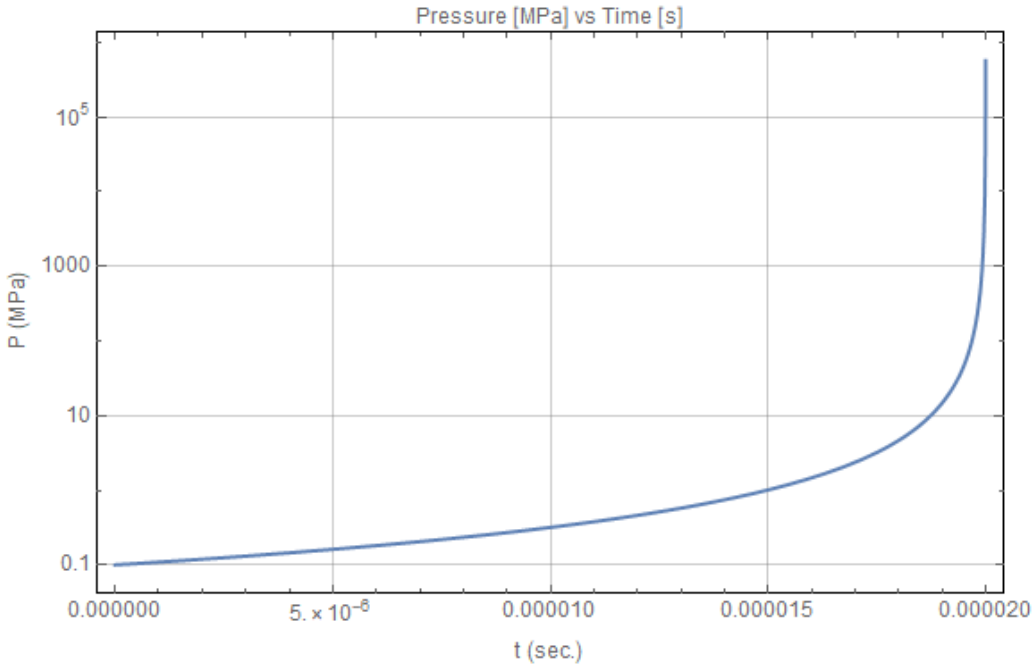


Figure 16. Plot of pressure in spherical case

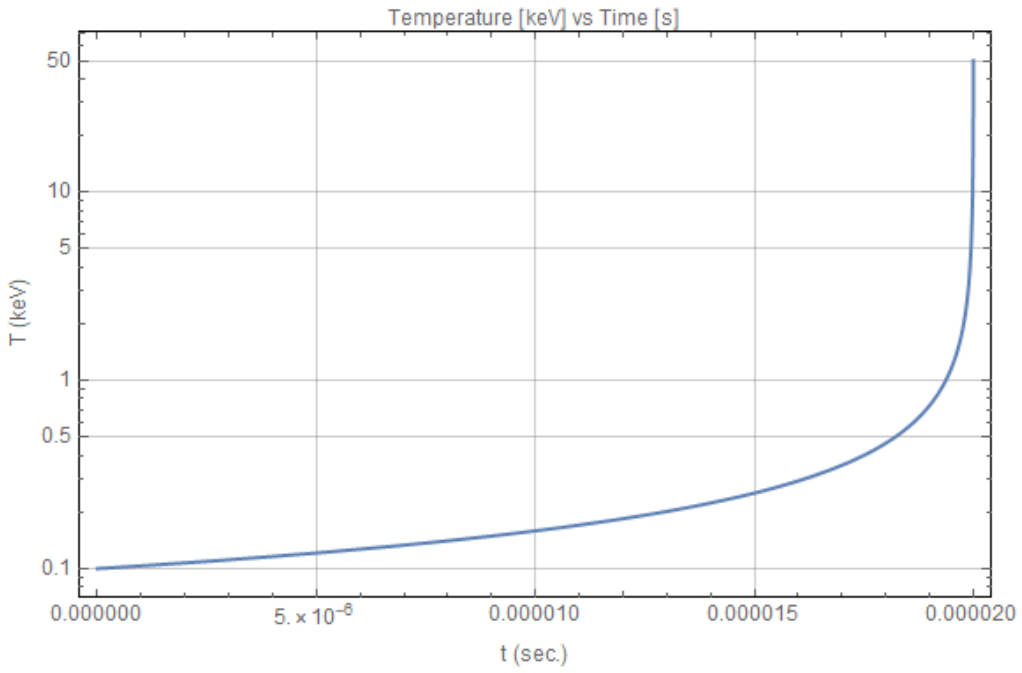


Figure 17. Plot of temperature in spherical case

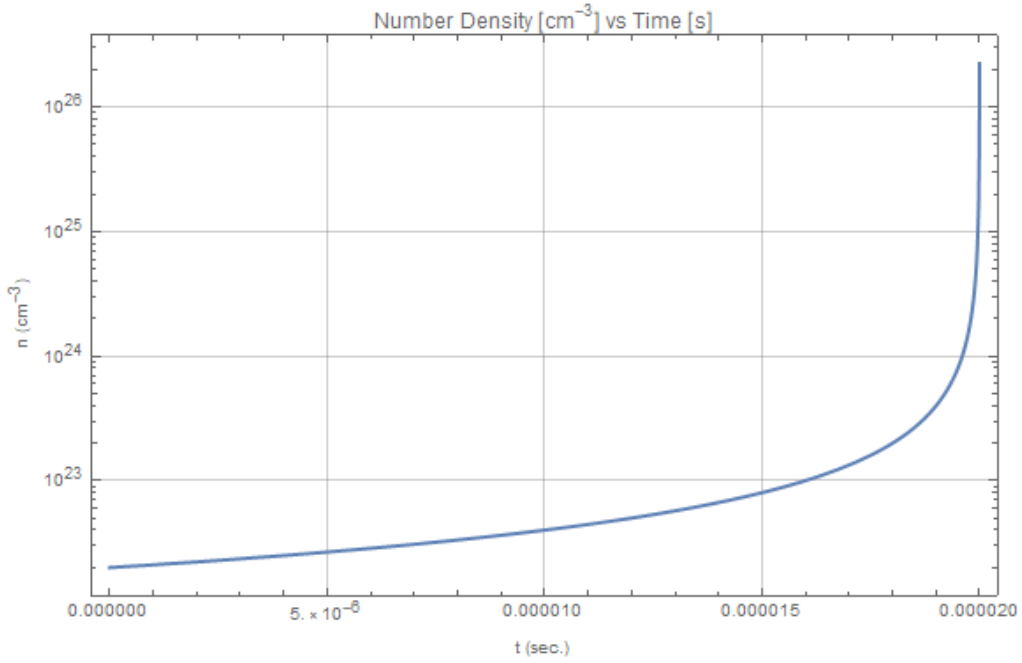


Figure 18. Plot of number density in spherical case

The spherical heating happens far more rapidly than the cylindrical case since the volume of a sphere decreases with collapse speed cubed (v^3), whereas a cylinder volume only decreases by speed (v) if it is collapsed axially. This decreases the effective confinement time, and reduces

losses. Adding “amplification” to a cylindrical system would cause radial compression, and thus more closely approximate uniform spherical compression.

It's worth mentioning that the high-energy neutrons (14.3 MeV) could also allow depleted uranium and other actinides to be burned outside the target chamber. A hybrid system like this could still produce useful energy, even if net gain can’t be achieved with fusion alone.

One-Dimensional Model

The models simulated with the hydro-code all had approximately the same initial conditions as the quasi-1-D code. The cavity was 10cm x 1cm, with an initial density of 50 kg/m3, and an impact of 10 km/s. However, the initial temperature was much higher in order to make the code converge. All the simulations started out at 1.5 keV (1.74×10^7 K) instead of 100eV (1.16×10^6 K), but the principle is the same.

Although the source-terms had to be neglected from the energy equation in the CLAWPACK code, two primary simulations could be used to test a variety of cases. Since the piston boundary does not slow down, running the model beyond the point where impact pressure equals plasma pressure will emulate the effect of “amplification”. In effect, it acts as if the impact velocity is greater than it really is. So, two cases were run. The first had the time limit set to 9.215μs (where impact pressure reaches equilibrium with the plasma), and another set to 9.500μs which produced a pressure equivalent to a 15 km/s impact (roughly double the pressure). The time limit for the second model is longer to increase pressure, but was somewhat arbitrary since it couldn’t be estimated how much additional pressure would be added, but the general effect could be illustrated. Plots of cell-averaged pressure for both cases are shown below.

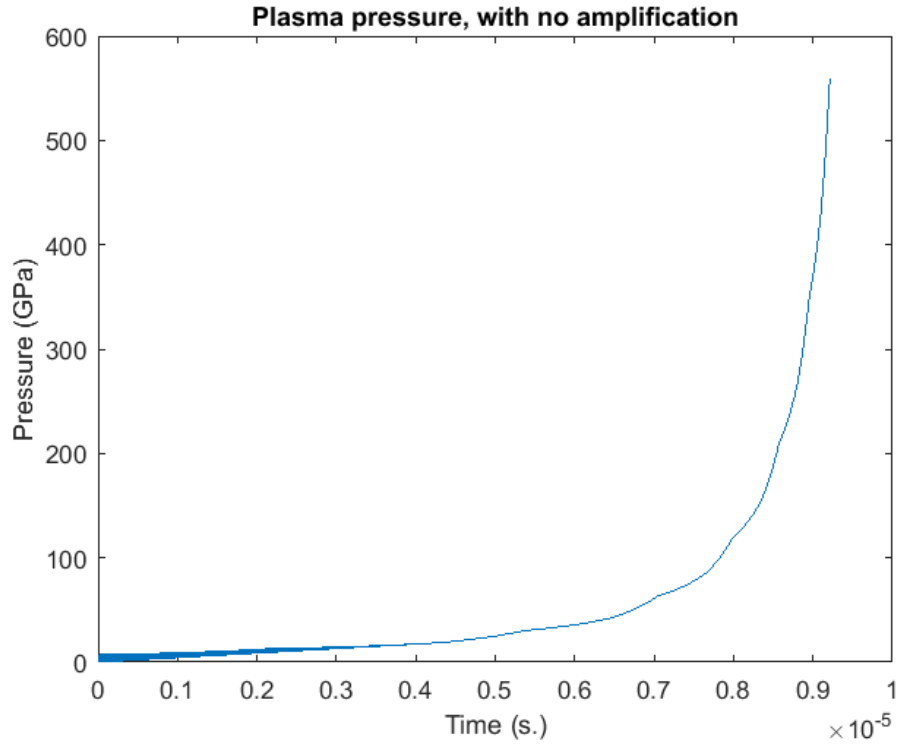


Figure 19. Pressure vs time for unamplified case

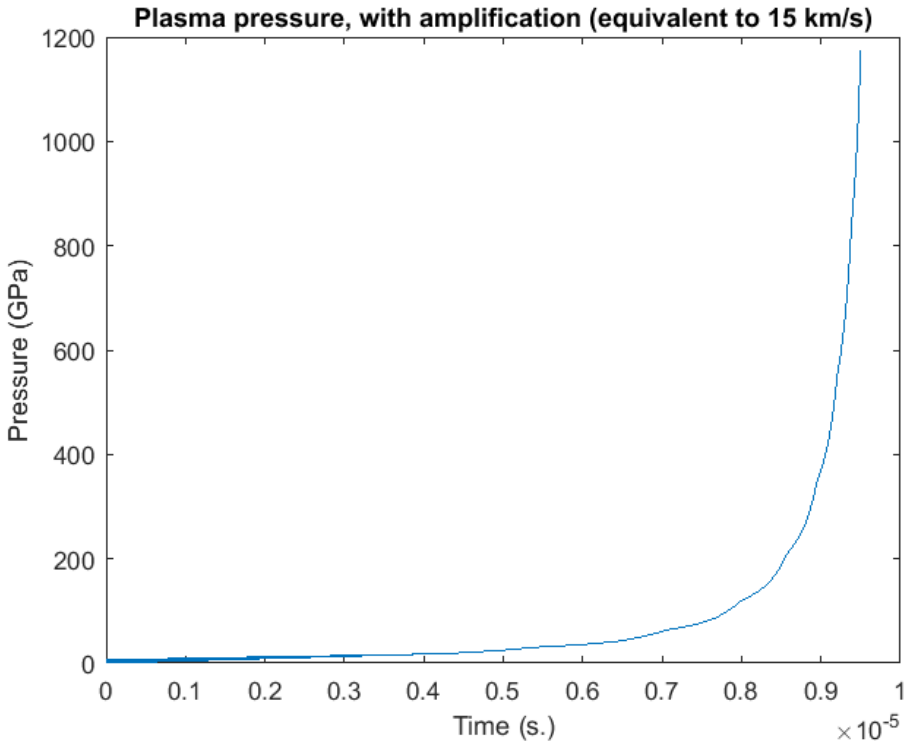


Figure 20. Pressure vs time for amplified case

The code outputs pressure, temperature, and density which were used to compute the energy gains and losses. The temperature and densities reached in these models was not as great as in the quasi-1-D models, and yet net energy gain was achieved. This is likely due to the higher initial temperature and resolution of the output data. Plots of temperature and density for both simulations are shown below.

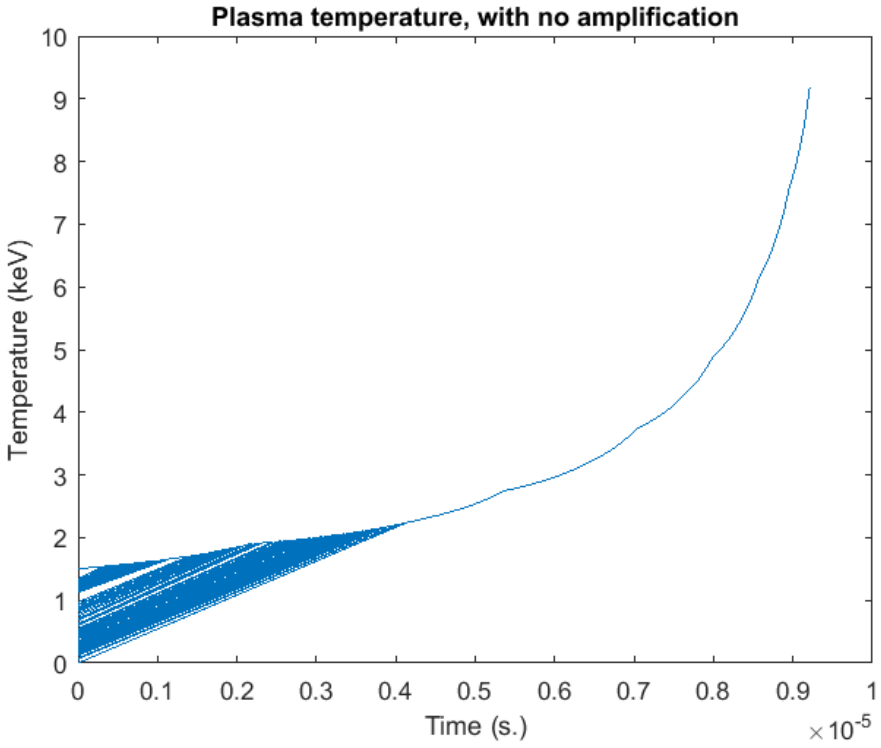


Figure 21. Temperature vs time for unamplified case

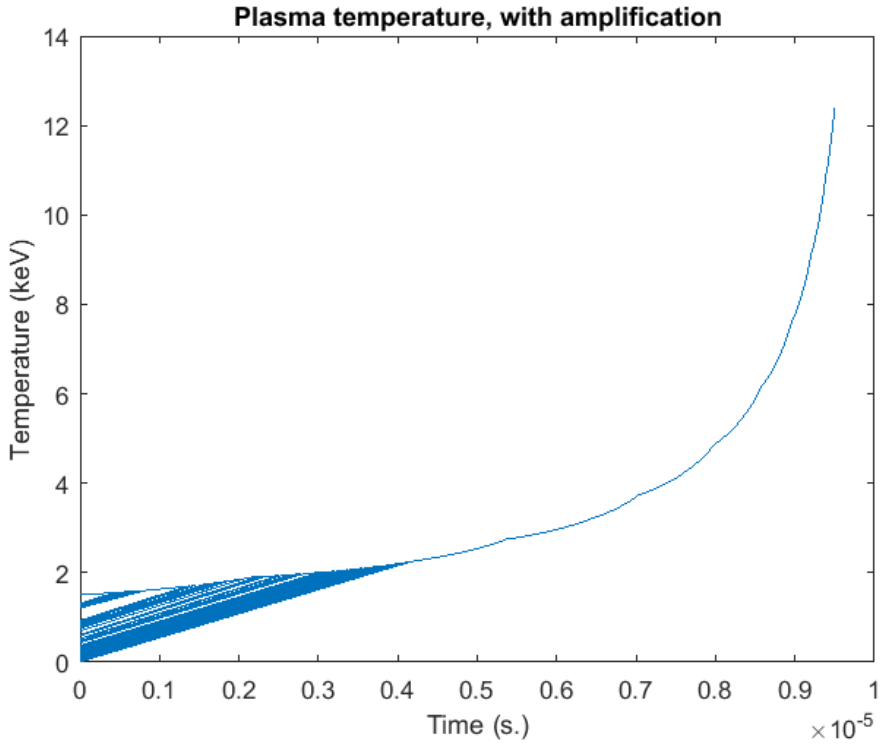


Figure 22. Temperature vs time for amplified case

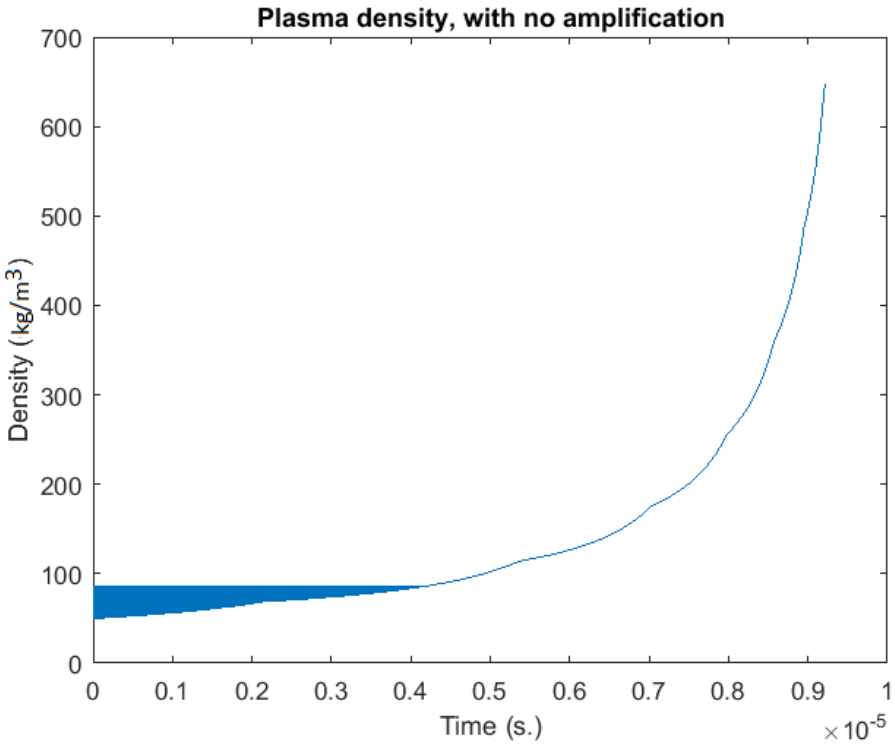


Figure 23. Density vs time for unamplified case

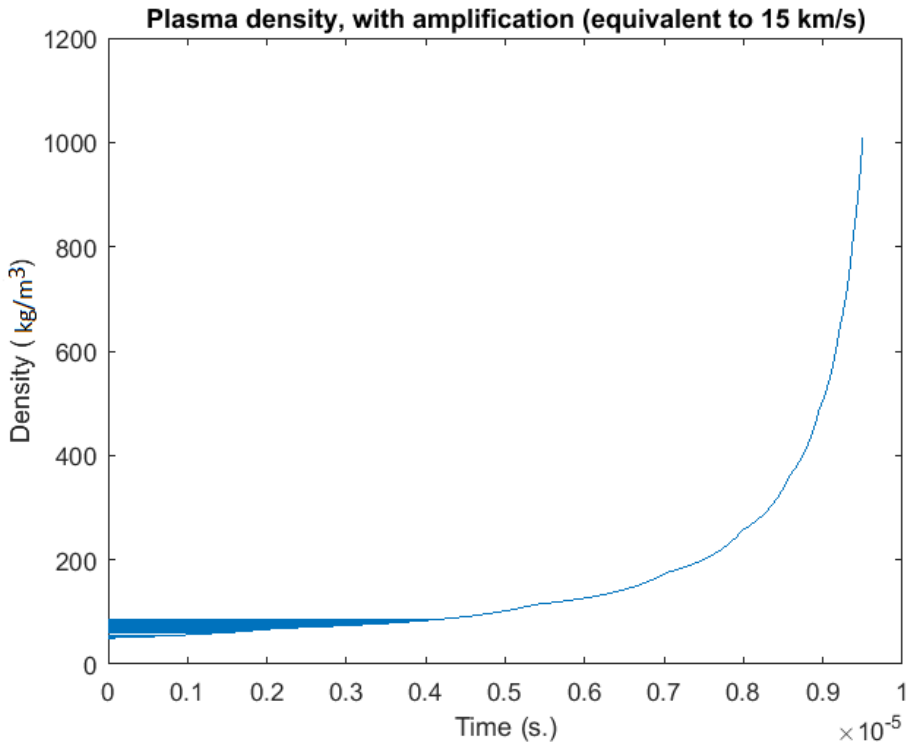


Figure 24. Density vs time for amplified case

In addition to the regular and amplified models, the effect of reducing conduction was approximated by running four separate cases for each simulation with varying amounts of conduction loss. As previously mentioned, a magnetic field could help reduce this loss. They went from a best case with no loss to a worst case with a loss equal to the free streaming limit. The plots of cell-averaged power output versus time for all four, for both amplified and unamplified cases are shown below.

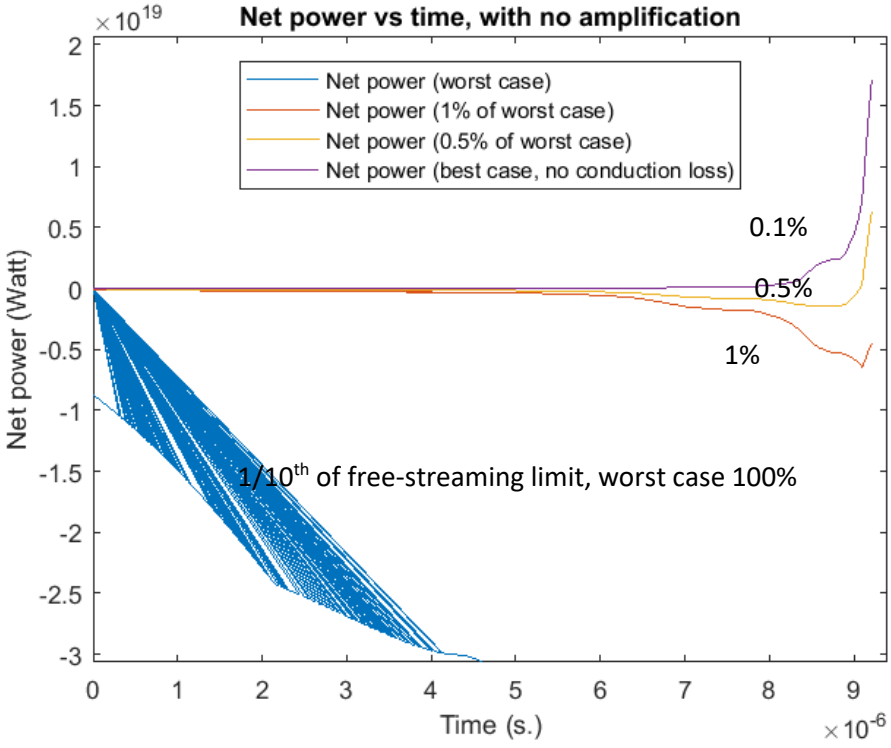


Figure 25. Power vs time for unamplified case

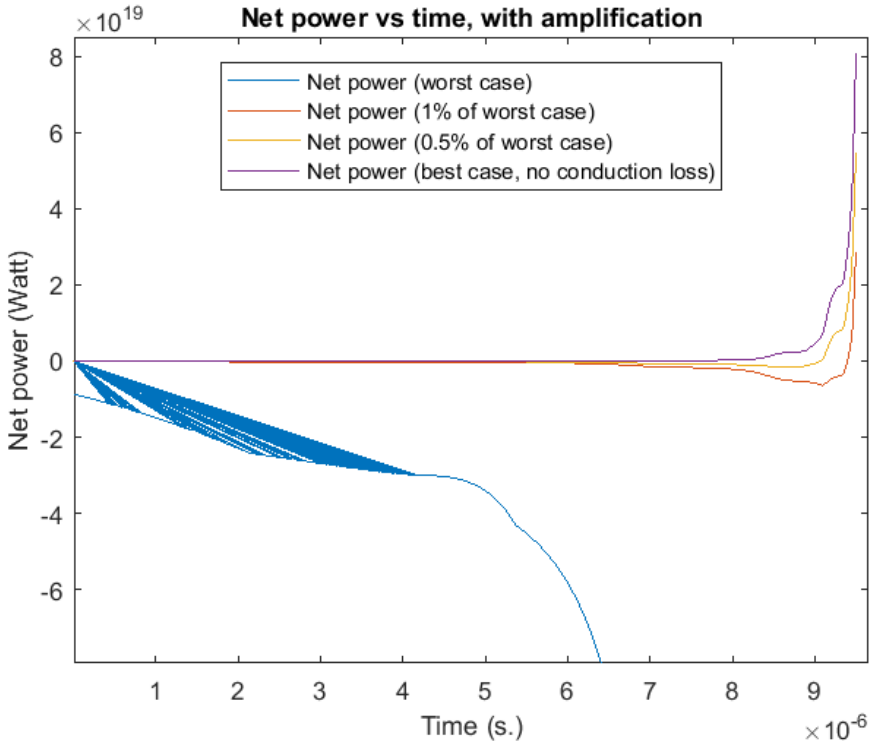


Figure 26. Power vs time for amplified case

First, in the above plots, there were some numerical artifacts apparent in the worst-case data up to about 4μs. These are also present in the other plots, but post processing magnified them. However, this had no effect on the results since all of the energy generation and loss occurs after about 6μs.

The difference between the two is quite clear. The losses are much greater for the unamplified case regardless of how bad the conduction loss is, and the peak power is greater. Table 3 shows the integrated total energy production for all cases. An additional case with a higher average atomic weight (7.5 g/mol as opposed 2.5 g/mol for DT alone) was also computed. This was meant to approximate the effect of the target/projectile material contaminating the fuel. However, just as with instabilities and amplification, a more precise estimate cannot be done without more detailed 3D models.

Table 3. Total energy released for all cases, all energies in Joules

	Worst case (free streaming limit)	**1% of worst case**	**0.5% of worst case**	**Best case (no conduction loss)**
Unamplified	-1.32E+15	-9.35E+12	-2.73E+12	3.89E+12
Amplified	-2.16E+15	-8.78E+12	2.06E+12	1.29E+13
Contaminated + amplified	-7.21E+14	-5.80E+12	-2.18E+12	1.43E+12

Not surprisingly, the amplified case is far better than any other case. The positive net energies are much higher (by about 100-1000) than the triple product and quasi-1-D calculations. This is likely a consequence of the code not properly accounting for losses and low resolution. Most of the energy gains and losses occur in the last few hundred nanoseconds, but the time steps are just

under 10ns, so this accounts for only a handful of the time-steps. Most of these cases also show negative total energies. Because the code could not properly account for the losses while solving the Euler equations, it couldn't properly produce the non-trivial effects. These negative energies are obviously not physical, but do show how significant the impact of the losses can be. If time allowed, the solvers in the hydro code would have been changed so that they would converge with source terms added. However, it was decided that this would not have been worth the extra effort, since a 1-D code can only roughly approximate the real physics, and would show the same trends. The next logical step beyond this is to obtain a proper HEDP code that can handle thermonuclear burn, as well as MHD effects to study adding magnetic fields. This will allow much deeper study of many of the elements touched on in this project. Eventually, 3D models combined with experiments could be used to study it even further.

Impact Models

It was unclear initially at what speed the compression would be transmitted through the target since the dynamics can be fairly complicated. There is substantial published research on this [38], [33] since the same sort of hypervelocity impacts affect weapon systems, spacecraft materials, planetary surfaces, etc. However, to get more useful and specific data, computer models were required.

There was no computer code available that could model both the impact and the high-density plasma. However, there are codes that can quite accurately model the impact dynamics alone. Most of these codes are maintained by national labs, and are not generally available. One major exception to this is ANSYS Autodyn [34], which is a smooth particle hydrodynamic (SPH) Lagrangian code used for modelling solid mechanics with explosives and high-energy collisions.

SPH Lagrangian solvers of this type were developed in the in the early 1990's [35] [36]. SPH codes don't have meshes in the usual sense. As the name implies, it models materials as a collection of particles that interact. This makes it well suited for systems with complex boundary dynamics and open fluid flow, allowing accurate modeling of fragmentation, crack growth, fracture, etc. All of which are relevant to this problem. Autodyn also includes an extensive library of materials with non-linear properties included.

Because there's no code available that can couple the high-velocity impact dynamics with high energy-density plasma (HEDP) physics, it had to be tested separately. The primary goals of this were to determine the maximum pressure exerted as a function of velocity, determine how quickly the target collapses, and to find how the density of the target material is affected since it will have an effect on instabilities. For both the quasi-1-D and 1-D models, it was assumed for the purpose of finding the maximum simulation time that the pressure was equal to the dynamic pressure of the projectile:

$$p = \frac{1}{2}\,\rho_{proj}\,{v_i}^2 \tag{19}$$

ρ_{proj} is the density of the projectile (likely lead, tungsten, or uranium), and v_i is the impact speed.

A simplified model with a 3cm long hollow cavity with lead for both the target and projectile was run over a range of impact speeds from 5 km/s to 30 km/s. Using lead for targets and/or projectiles allows the metal to be recycled as coolant. It also has a high density that exerts more pressure on impact. The cylinders had equal diameters, as earlier iterations of these models seemed to indicate this was optimal for fully compressing the target with minimal loss of kinetic energy. The maximum pressure from each run was extracted and plotted against the dynamic pressure prediction. This is shown in Figure 27.

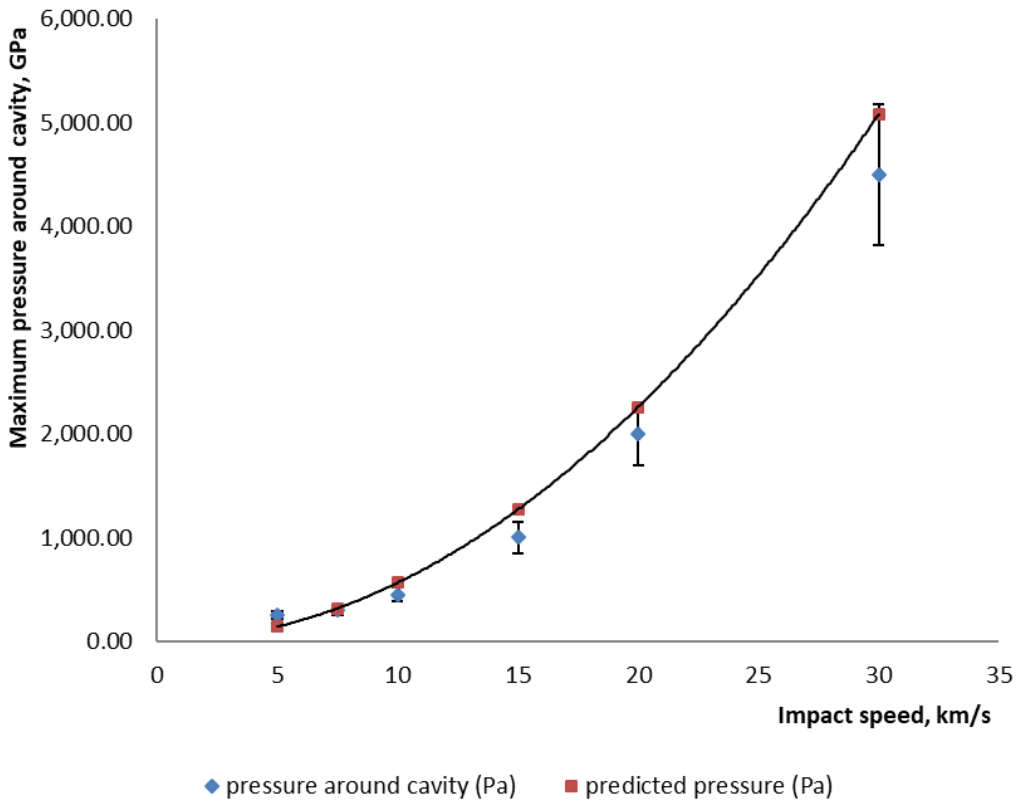


Figure 27. Impact pressure simulations vs dynamic pressure

The error bars represent ±15% of the data point since this is roughly the maximum error. This shows that even at very high impact speeds, the dynamic pressure model is always within about 15% of the simulation. Therefore, the assumption used in the plasma models appears to be quite reasonable.

The density of the lead roughly doubled as well in all these models. This seems to be the limit before it begins heavily fragmenting. Also, the difference between actual cavity collapse time versus h_0/v_i (the collapse time if it collapsed at the speed of impact) was computed.

Table 4 below shows this difference for the full range of impact speeds.

Table 4. Comparison of collapse times

speed (m/s)	collapse time (sec.)	h_0/v_i (sec.)	multiple of h_0/v_i, f(h/v)
5000	2.90E-06	6.0E-06	0.48
7500	1.30E-06	4.0E-06	0.33
10000	1.30E-06	3.0E-06	0.43
15000	8.70E-07	2.0E-06	0.44
20000	6.00E-07	1.50E-06	0.40
30000	4.00E-07	1.0E-06	0.40

Clearly, the cavity collapsed much faster than expected, and by the same roughly the same relative amount (the standard deviation in the ratio is only 0.048). In reality, the plasma interacting with the shock will slow this down, perhaps to about the same speed as the shock through the solid material (which is roughly the same as the impact speed on this scale). However, this will only occur after the collapse has progressed quite far since the plasma starts at a far lower density, and doesn't reach a comparable density until collapse is nearly complete. This indicates that the interaction between the plasma, the wall of the target, and impact shocks may be quite complicated. This is not too surprising given the known complexity of target/fuel interactions in other ICF systems.

Instabilities

A detailed study of instabilities was not possible without a higher dimensional model. However, as discussed earlier, the main instabilities are Rayleigh-Taylor related, such as Richtmyer-Meshkov which is the same as Rayleigh-Taylor in the limit of a sudden impulse (such as a strong shock from an impact, as opposed to a steady acceleration such as gravity) [39], [40], [41]. Given that the densities of the target and fuel are known for a range of cases, the growth rate can be estimated for a worst case to show that starting with a lower average fuel density can

practically eliminate this. Equation (2) describes the amplitude of a perturbation at the fuel/wall interface. The acceleration, g, at 10 km/s impacting a 1cm cavity is about $2x10^9$ m/s^2. A worst case can be taken from the density data for unamplified 1-D model, which had a maximum fuel density of 600 kg/m^3, and a target density of about 20,000 kg/m^3. The time t, is 0.00001 sec. For a spatial wavenumber equal to $1/h_0$, the ratio of h/h_0 is only 3.94. Even for a wavenumber 10x larger, the ratio is still only ~76. Compare this to a typical ICF target with a density of 1,000,000 kg/m^3. Clearly, this is not such an issue in this configuration, as intended. If fuel densities can be made to be closer to the density of the target material, this can be made negligible.

The other instability that may cause an issue is Kelvin-Helmholtz instability. This could be caused along the filaments produced in Rayleigh-Taylor, or by shock waves propagating normal to a surface producing a tangential flow and shear forces. It's unclear if this will be a serious issue without higher-dimensional modelling.

Finally, if magnetic fields are used to help compress and reduce conduction loss, a whole zoo of potential MHD instabilities may become an issue. However, since the fields would have a relatively weak effect due to the high density of the plasma, and the theta-pinch configuration tends to resist these, these are not expected to be a serious problem. The added gain from reducing conduction losses would likely compensate for any new losses these produce, as appears to the case in experiments [22].

Preheating and Stepped Ignition

In the previous chapter, several preheating methods were mentioned, but only two were examined in depth. The first was to use a lower-density fuel in a secondary cavity above the main fuel. This would not reach ignition itself, but could produce enough neutrons to induce (n,α) reactions to pre-heat the main fuel. In theory, this could be done in two or three stages, but for simplicity, only two stages were tested. Below are MCNP mesh plots in figures 28, 29, and 30 for neutron flux in two cases with a secondary cavity to provide a qualitative picture of the systems. The mesh plots were not used for any calculations, but did show how the neutrons generally behaved. The first has a target made of graphite intended to moderate the neutrons and improve capture, and the other has a LiD target, which would also release additional energy. In these plots, the flux is normalized per source neutron. The large cavity and small cavities are from top to bottom respectively, with the LiD pellet placed at the top of the large cavity. In the third case, the pellet is in the center, and the outer shell is also LiD.

Figure 30 is for a case with only one cavity, but with the gaseous fuel preheated so that the initial compression produces neutrons that get captured in a LiD pellet in the center. This third case is similar to amplification, but instead of generating extra external pressure, it heats and pressurizes the fuel from within. This is an interesting case, and the dynamics would be much different than the others, with a feedback loop that operates quite differently than amplification alone. It also produces the tritium fuel needed to burn with the deuterium.

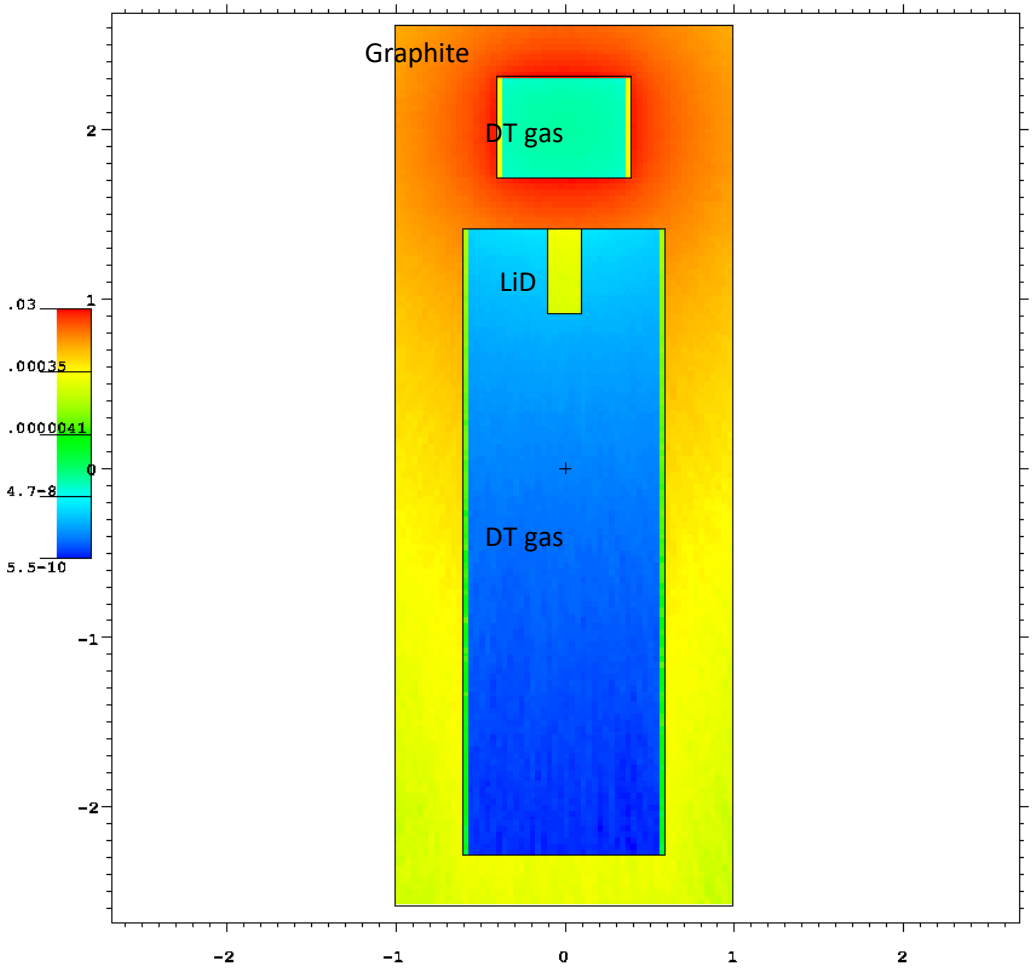


Figure 28. MCNP mesh plot of neutron flux for a graphite shell

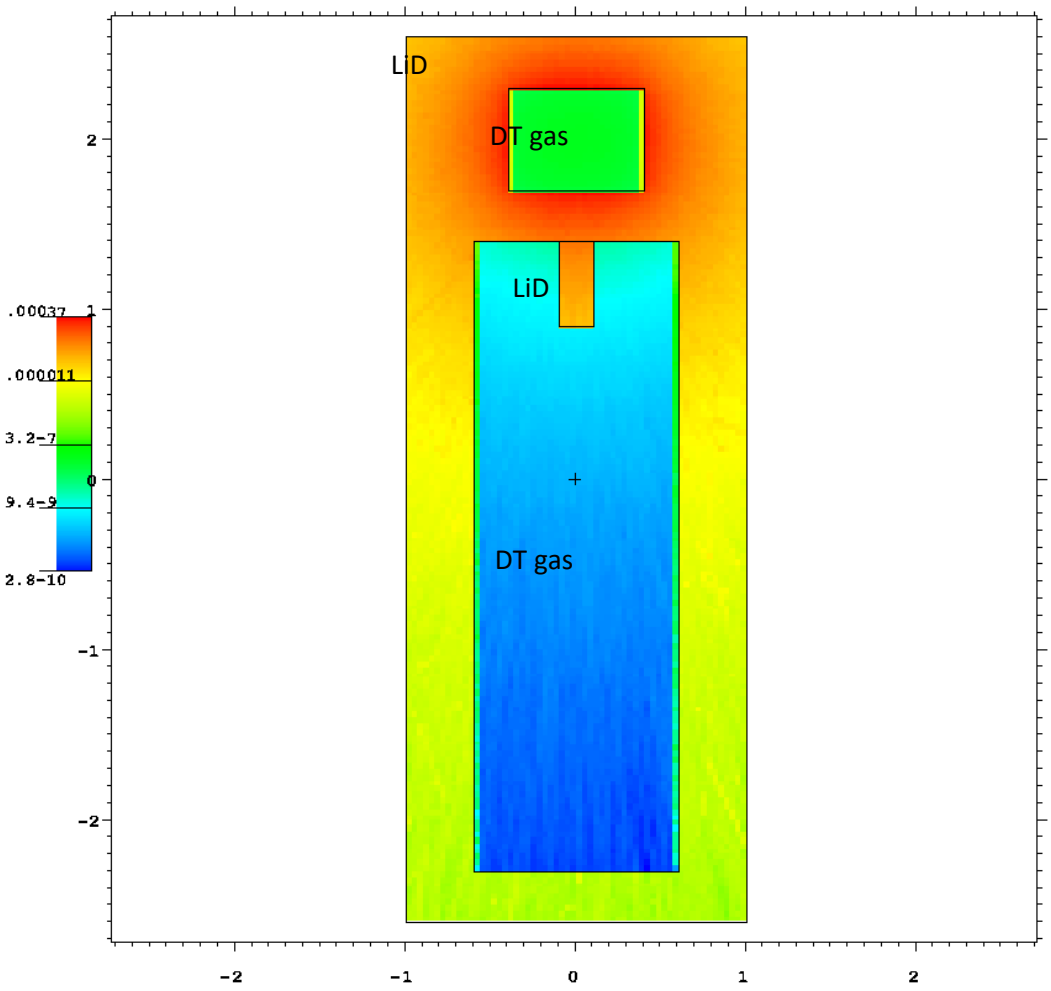


Figure 29. MCNP mesh plot of neutron flux for a LiD shell

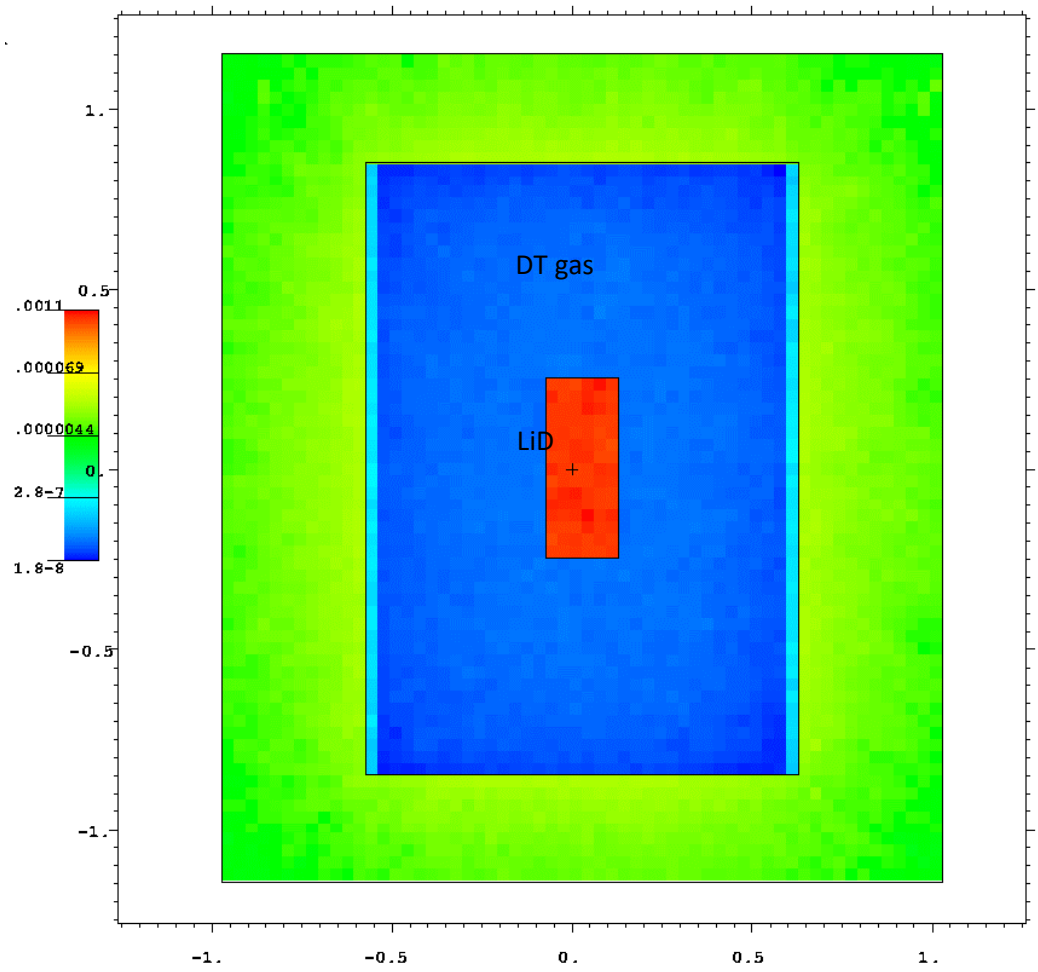


Figure 30. MCNP mesh plot of neutron flux for a LiD pellet with preheated gas fuel

The F6 tally results for the first two cases are not encouraging. They both require more than 1GJ of fusion energy from DT reactions (to produce enough neutrons) in the first stage to reach the necessary 500kJ of preheat in the main stage. The third case is much better, but still requires more than 100MJ to reach the necssary temperature. This is still well below the estimated net energy per shot of 10's of GJ, but could be quite difficult to reach without ignition itself. The table below summarizes these results.

Table 5. Summary of stepped preheating results

	Graphite shell	σ	LiD shell	σ	LiD pellet	σ
mass, g	3.14E-03		3.14E-03		1.26E-02	
F6 tally, MeV/g/src neutron	8.05E-02	0.0039	7.49E-02	0.0042	2.12E-01	0.0062
Energy, MeV/src neutron	6.48E-03	1.2E-5	5.61E-03	1.3E-5	4.50E-02	7.8E-5
energy required from first stage (J)	1.10E9	2.03E6	1.27E9	2.94E6	1.59E8	2.75E5

Finally, the other preheating technique examined was to bombard the target with a neutron beam generated from an external source. The geometry was very similar to the stepped cases. A pellet of LiD was placed in the center, but this had an aluminum shell to allow more neutrons to pass into the cavity. The beam was all thermal 0.025eV neutrons with a diameter slightly larger than the cavity height. Figure 31 is a mesh plot of normalized neutron flux that clearly shows the beam emitted from above and impacting the pellet leaving a shadow behind it.

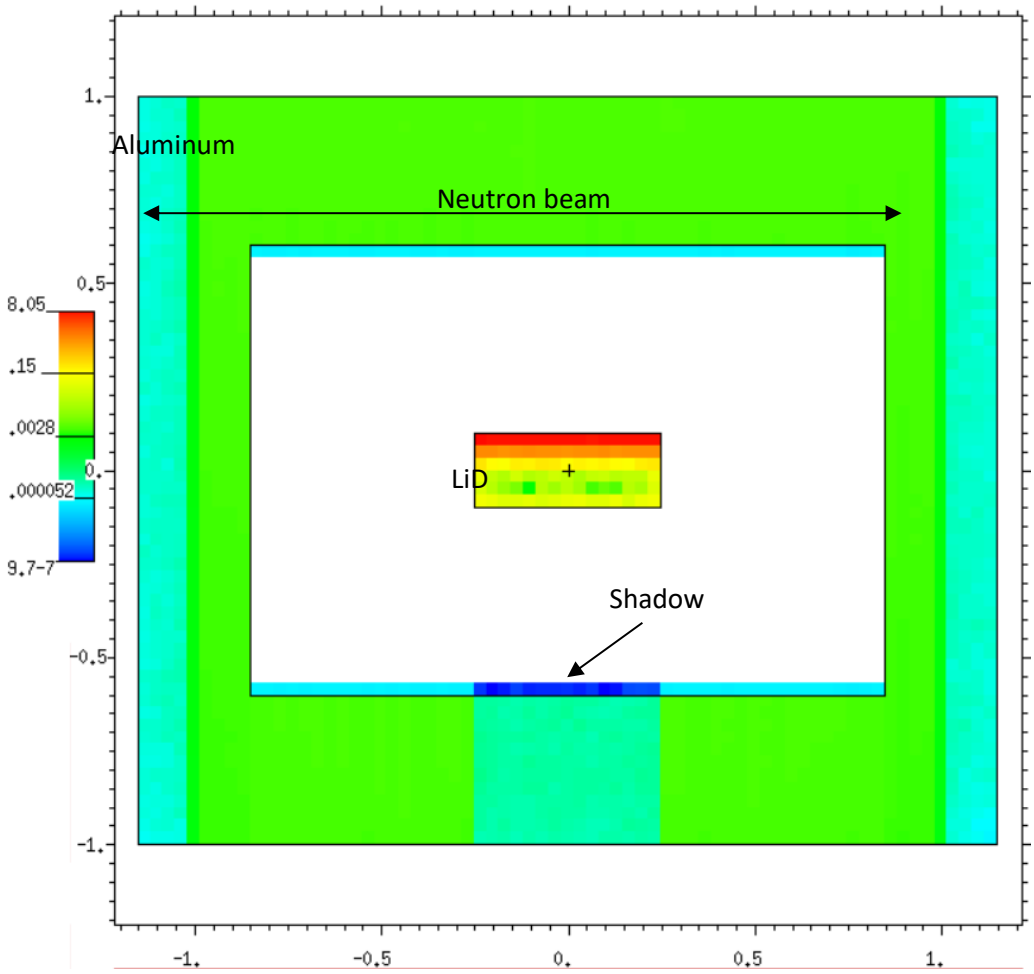
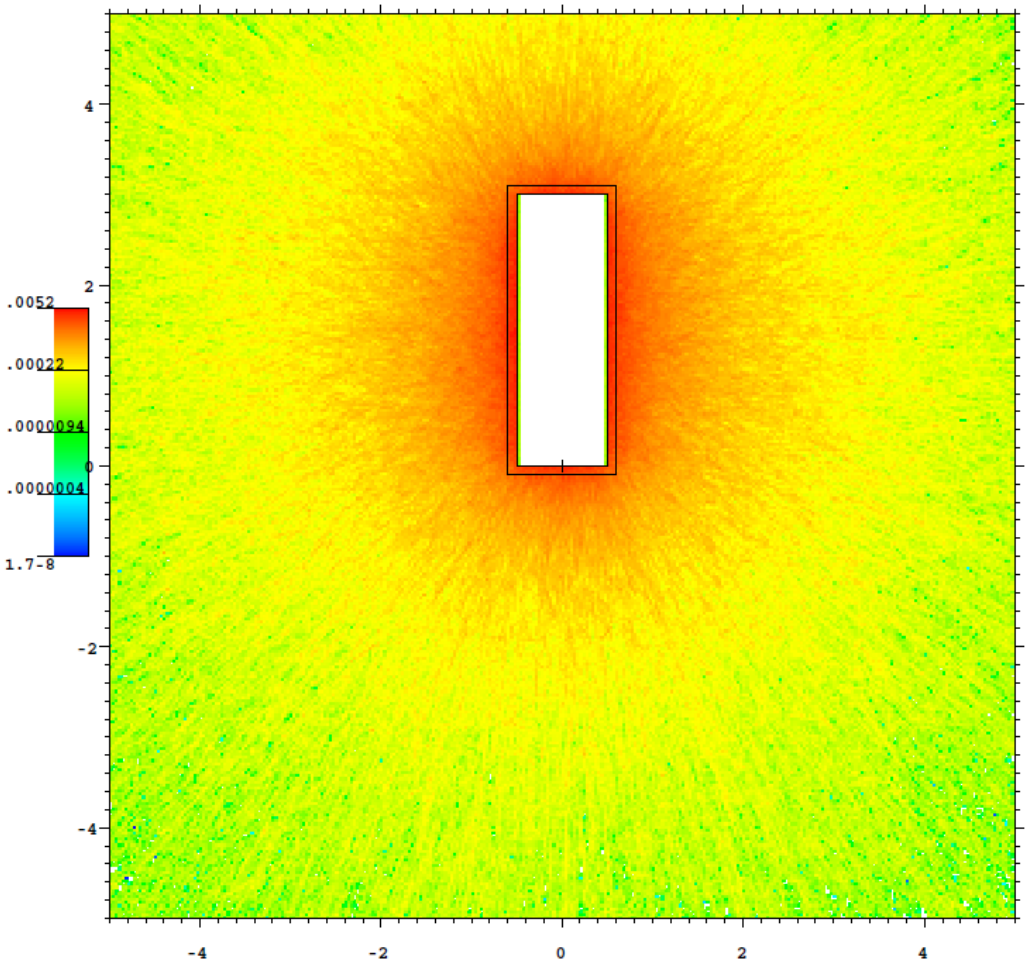


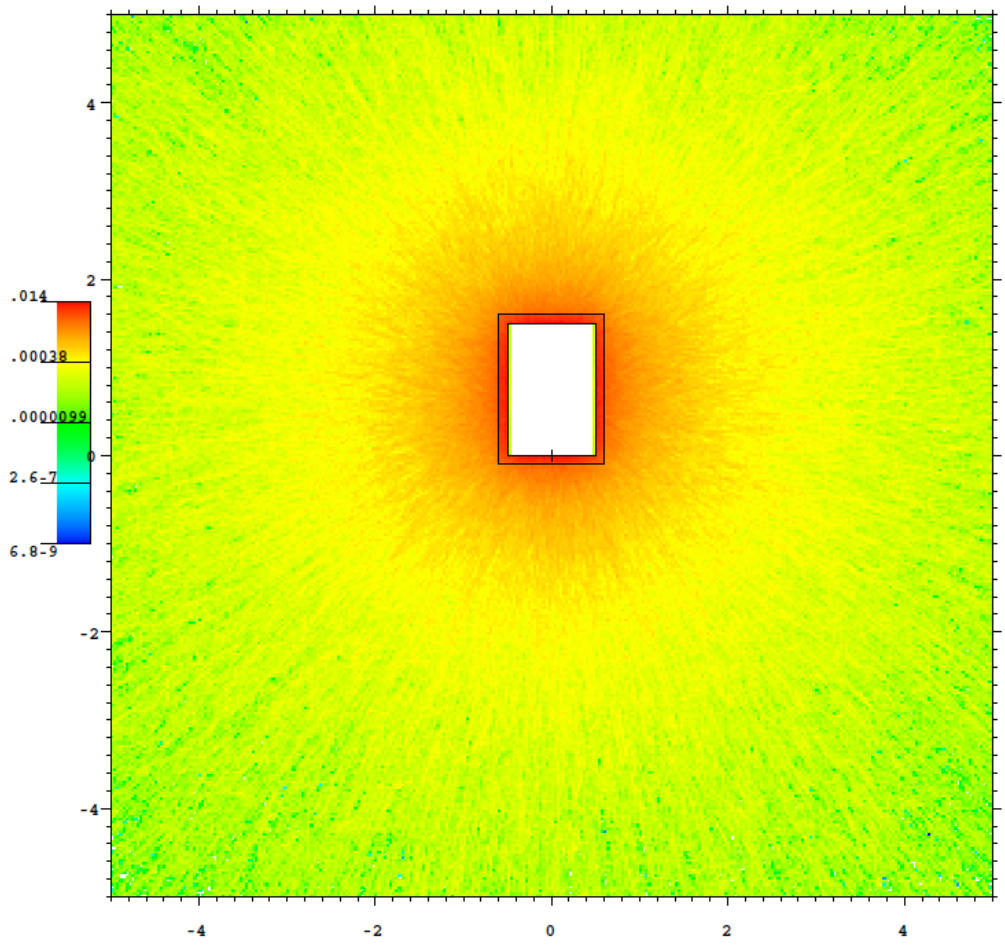
Figure 31. MCNP mesh plot of neutron flux for a LiD pellet with external beam

The maximum normalized flux is clearly much larger in the LiD pellet than it is in the stepped cases. The F6 tally for the pellet produces 150keV per source neutron. This could be increased with a more focused beam. Producing a neutron beam like this is beyond the scope of this project, but the tally result can be used to roughly estimate the beam current required of an accelerator that fires alpha particles at a ^{9}Be target at 10MeV which is the peak of the (α,n) cross section at about 0.7barn. To reach the 500kJ energy, roughly 2.4x10^{24} alphas/s are required, which is a current of 768 kA. This is very large but there are potentially better means of generating such a neutron beam.

Amplification and Feedback

A few representative mesh plots of a typical case are shown below. F6 tallies were used to determine the relative amount of energy deposited in the shell of each target. F2 flux tallies were also taken for each case.




.014
.00038
.0000099
2.6-7
6.8-9
4
2
0
-2
-4
-4
-2
0
2
4

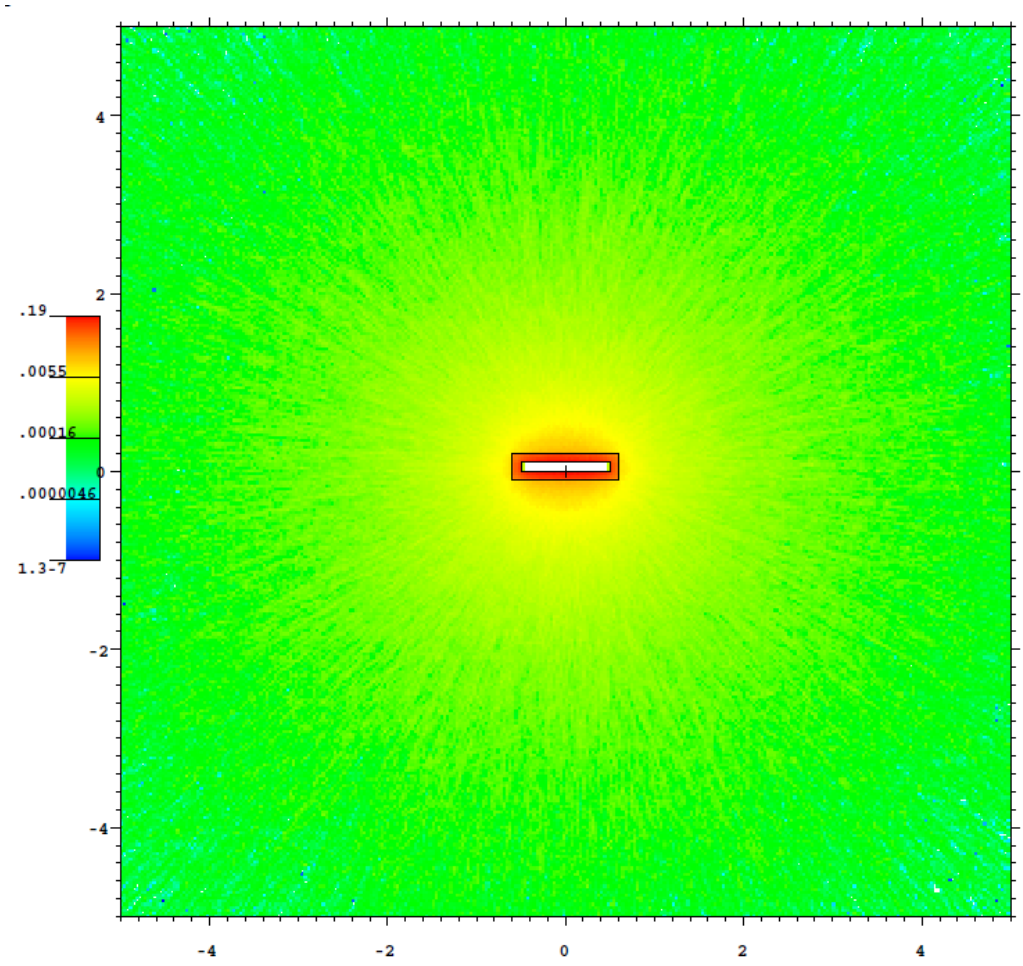


Figure 32. MCNP mesh plot of neutron flux for a sequence of cavity heights representing collapse

Five cases were examined: ^{10}B, ^{6}Li metal, LiD shell surrounded by ^{238}U, LiD, and ^{238}U metal. In all models the density was adjusted for each step in the sequence to correct for compression. Below are plots of the F6 tallies and the F2 tallies (at the inner surface of the cavity). Error bars were not included since the uncertainties in the data points would produce error bars too small to discern in the plot (all were in the range from 0.0002 to 0.0006).

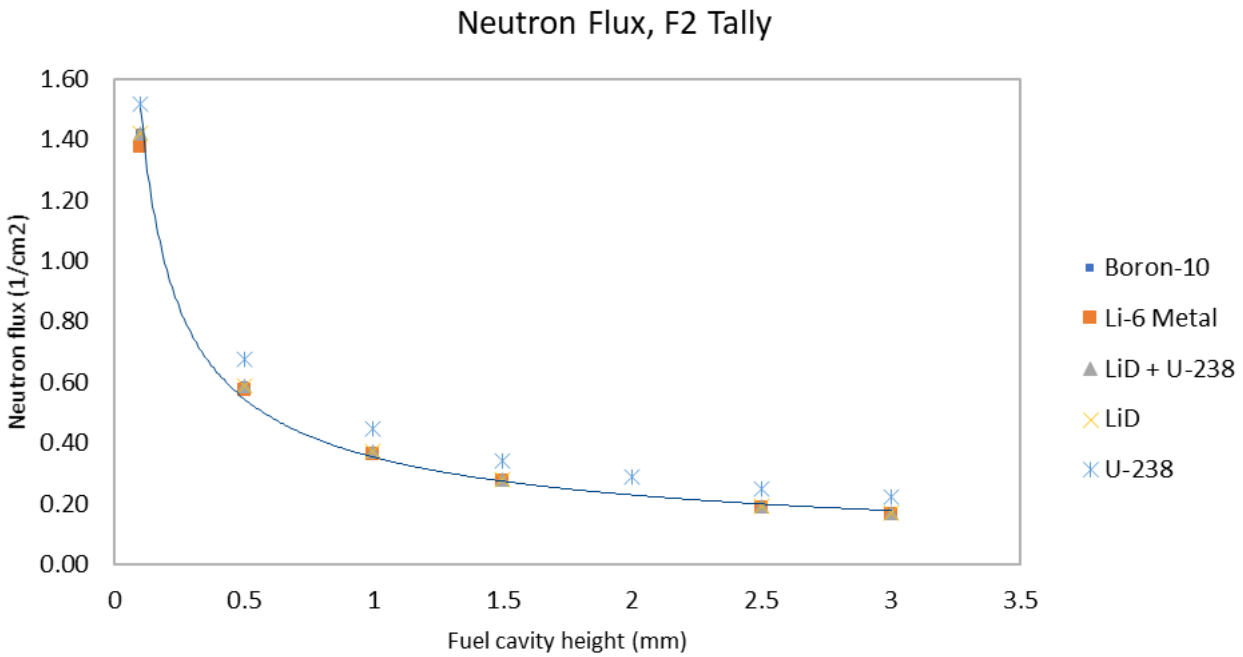


Figure 33. MCNP F2 tally results

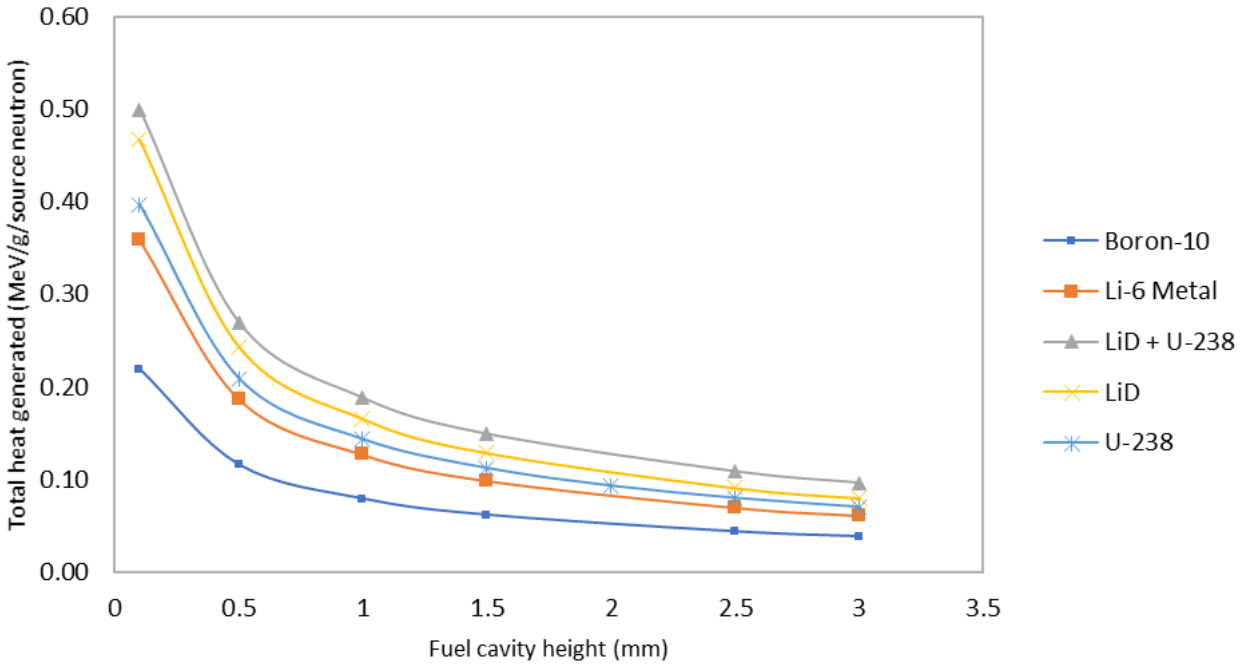


Figure 34. MCNP F6 tally results

Both of these plots clearly show a strong positive feedback that increases exponentially as the cavity collapses. The F6 tally shows high energy return as well (roughly 3.5% of the

neutron energy is returned through exothermic reactions). All of these cases performed quite well, but LiD and LiD + ^{238}U performed the best. LiD may be the most desirable since it also breeds tritium that can be used in future shots after being extracted from with the target fragments and unburned fuel.

All of the preceding MCNP models used room temperature cross section data, so the amplification cases for ^{10}B, LiD, and ^{238}U were also run with cross sections for 2500 K. The tally results were identical to the room temperature cases within the uncertainty. Doppler broadening makes little difference since there are no resonances in the cross sections at the 14 MeV energies of the neutrons. The input files for these cases are identical to the room temperature cases, but with 2500 K ENDF/B-VII.0 cross sections.

Magnetic Fields (θ-pinch and flux compression)

Another element that could be added to the system is a theta pinch. A theta pinch is caused by an axial current, which induces a magnetic field in the plasma in the theta direction (hence the name). Theta-pinches also tend to resist instabilities, and so shouldn't contribute any new losses. Because theta-pinch includes a magnetic field, as the plasma is compressed, so too would the magnetic flux (since the plasma is a conductor), increasing the field strength. The field would likely not be strong enough to fully confine the plasma due its high density, but it would have the effect of preventing some of the electrons from reaching the wall, reducing the conduction loss, and reducing the impact speed required. The 1-D code was not able to account for this of course, and it would have complicated the project, but some basic calculations could be done to demonstrate the effect. The magnetic pressure can be found for a given initial magnetic field, and a ratio of initial to final cross section area:

$$p = \frac{(B_0 f)^2}{2 \mu_0} \tag{20}$$

B_0 is the initial magnetic field strength, f is the ratio of the cross-sectional areas $f = A_0/A_1$, and μ_0 is the permeability of free space. This pressure is in addition to the energy from amplification and impact, thus providing another positive feedback mechanism acting to multiply those effects as well. This a fairly naïve calculation since the actual geometry of the plasma is not uniform during compression, which will cause the field to be more complex and likely cause MHD instabilities that would reduce the increase in gain. However, it does give an indication of how strong the effect could be. Two estimates of pressure were calculated this way, one for radial compression, and the other for the axial compression. The initial radial compression could be provided by the amplification. Starting with 20T magnetic fields, the pressure for the axial and radial directions respectively are 1,590 GPa and 314 GPa. This is comparable to the pressure from the impact alone. This is very good, but as mentioned previously, probably too optimistic. Experiments with laser-driven systems have shown very promising results [22], although not as great as would be expected from this simple estimate.

Economics

Since there are many variables and potential configurations, estimating operating costs for a power plant using this concept is difficult at this point. However, it may be useful to estimate the necessary revenue per shot to make the plant profitable, as this will be a critical constraint in any future design. The average current price of electricity across all sectors in the USA as of March 2019 is about $0.1038/kWh [42]. The quasi-1-D model and the triple product calculation show a net energy release on the order of 30GJ per shot. If we assume the plant utilizes steam turbines similar to those used in current nuclear plants (~30% efficient), this will result in 10GJe to the grid per shot. This would produce about $288 gross revenue per shot, or

about $2.5M per day if the repetition rate is 6 shots/minute. This must cover all expenses averaged over each shot. Staffing and maintenance must be minimized as much as possible. The figure below shows a schematic concept for a power plant.

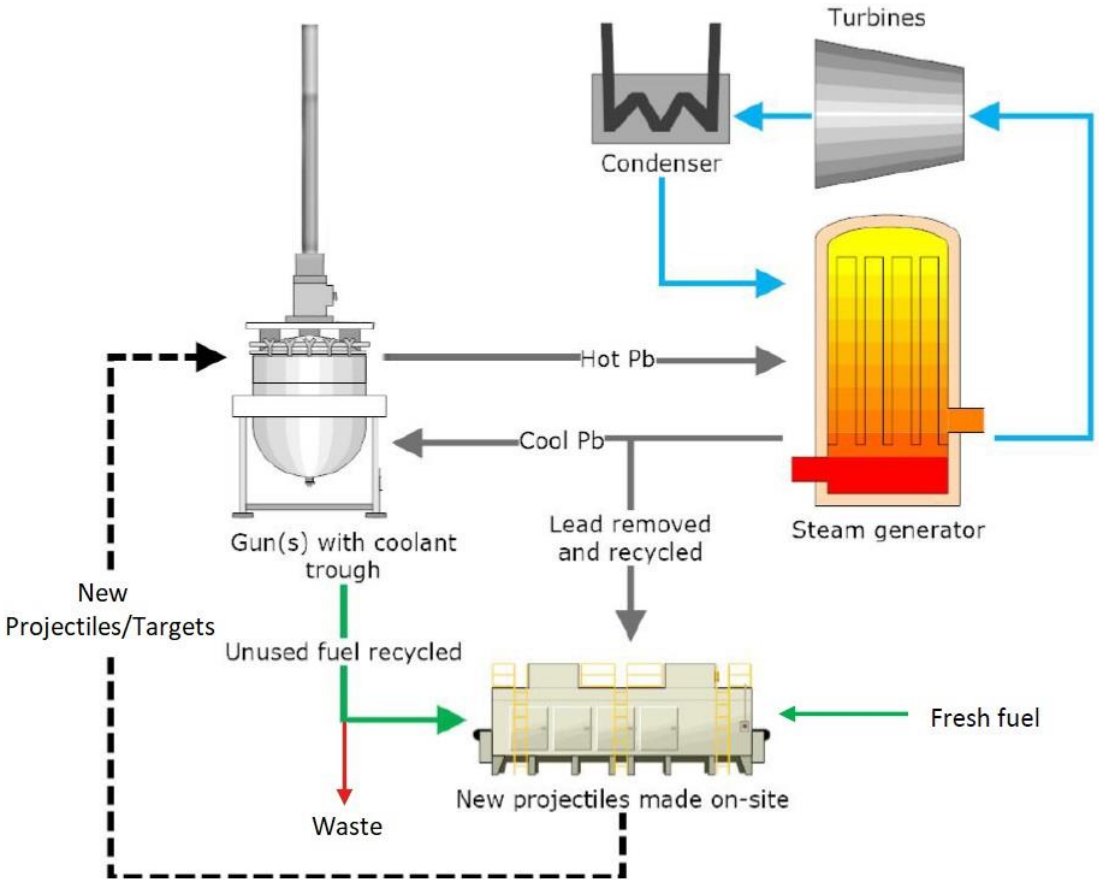


Figure 35. Simplified power plant

It's assumed there is a jacket of molten lead around the target chamber for absorbing the high-energy neutrons. This allows the lead used in targets and projectiles to be recycled throughout the system without a separate loop for collecting target/projectile fragments. Likewise, the unspent fuel and ash gasses will be extracted from the target chamber by pumps since a vacuum must be maintained anyway. If LiD is used as a fuel, it will have to be manufactured on-site. If DT ice is used instead, then facilities will be needed for combining and freezing them into

precise ice pellets. In either case, these precise fuel pellets must be manufactured (with very-high quality) at a rate of at least 6 per minute. Furthermore, a combined cycle system with uranium would have to deal with possible radioactive contamination and debris. Clearly, this system will be quite complicated, making it a challenge to keep the cost per shot reasonable. However, this is certainly no worse than other ICF approaches.

It should be pointed out that the complexity of the system would not change much if the energy per shot was increased. Therefore, the cost per shot would likely not change much either, making larger systems far more economical, regardless of average output power. An interesting consequence of this, is that the lower the repetition rate for a given average power, the more economical it is.

Possible Experiment(s)

A good first step in testing the whole concept would be to show that preheating and magnetic flux compression could produce high enough temperatures to reach fusion at the impact velocities of an existing gun.

This could be done with Joule heating produced with a high-voltage power supply in a small (2.5mm diameter) target filled with deuterium gas (since obtaining and handling tritium is more difficult) at low pressure (~1 atm). Although this would not suffice for a power plant, under these conditions, the initial temperature would be about 2 keV (2.32×10^7 K). This higher temperature is essential if deuterium alone is used since the DD cross section has no peak. The target must be this small to be fully compressed by BB-sized projectiles used in light-gas gun facilities. It also reduces the risk of activating the experimental equipment with neutrons. The magnetic field could be provided by a solenoid or rare-earth magnet around the target (assuming the facility operators allow a strong magnet to be placed in the target chamber). The permanent

magnets would likely be destroyed on each shot, but small and strong rare-earth magnets are quite inexpensive, and extras could be purchased for the experimental runs. A solenoid could be made large enough to avoid damage from the impact and debris.

Verifying that fusion has occurred would have to be done by detecting neutrons. A good ^{3}He neutron detector and very fast single-channel analyzer or oscilloscope would have to be used to verify that neutrons were only produced during the impact. The time resolution needs to be less than 1μs (since this is roughly the time scale of the impact). Activation foils placed at a precise distance from the target with and without a moderator could be used to estimate the total energy released. The analysis would be quite simple since the target is effectively a point source, and the cross sections are all well known.

Multiple trials would be done, testing one variable at a time. Starting with a control test shot, without preheating or magnetic fields. Next, adding Joule-heating to verify that some fusion occurs, and finally adding magnetic fields as well. If the neutron detectors have short enough time resolution, some transient effects may become visible.

The preparation would be fairly inexpensive since the guns already exist and are available to researchers already. A glovebox or fume-hood would be needed for melting and casting several uniform lead targets, as well as a weak neutron source for calibrating the detectors, and a detector system for precisely measuring the activity in the foils after each test.

Establishing the feasibility with a simple experiment such as this would facilitate in obtaining funding for more expensive and sophisticated experiments. This simple experiment would not test the viability of reducing Rayleigh-Taylor instabilities, or the other methods of preheating. The more advanced experiments would have to be used for that purpose.

Chapter V: Conclusions

Discussion of Research Findings

The quasi-1-dimensional model was quite informative. It was written in a way that any number of losses could be added, and indeed many potential losses were included that ended up being negligible. The only losses that were significant (greater than 1%) were bremsstrahlung and electron conduction. However, since the different physics were only loosely coupled, calculating a conduction loss using the temperature gradient was not possible, so the worst-case free-streaming loss was used as a conservative estimate. The models were iterated to test numerous ideas, including the reduction of density to effectively eliminate instability losses, and preheating the fuel. The results agree broadly with a simpler calculation for both net energies released and confinement time (although confinement time is difficult to define here since the nuclear physics are decoupled from the compressible fluid behavior). It indicated that the overall concept had merit, but needed more study. It also confirmed that radiation loss was insignificant, which was determined by other calculations later. The results were encouraging enough to continue work

The 1-D model was intended to more realistically calculate the compressible fluid behavior while including the gains and losses from thermonuclear burn, free-streaming conduction, and bremsstrahlung. The model was modified from an open source code written to demonstrate the capabilities of the CLAWPACK libraries for solving conservation law systems of equations. Therefore, it was already heavily verified by the original authors, but was still verified for this purpose by comparing the temperature and pressure at the extreme conditions of impact to adiabatic estimates, and they agreed almost perfectly. Unfortunately, the code would not converge with any source terms added due to an apparent bug. So, the gain and loss terms

had to be partially decoupled by calculating them from the code's output data. Because of this, the results are probably not very accurate, since the losses would lower the temperature of the plasma until late in the compression. Nonetheless, the results were informative for examining the effect of reducing the conduction loss and adding additional confinement pressure.

The conduction loss reduction would have to be quite significant to make any real difference in energy gain (upwards of 99%). However, the conduction loss calculated was a worst case, so this may not be such an impediment. Making a larger target, and increasing the fuel density (and thus fuel mass) could mitigate this as well via the squared cubed law.

Adding additional confinement pressure (such as would be the case with "amplification") made a tremendous difference. The pressure was equivalent to a 15 km/s impact, and increased the best-case net energy by a factor of more than 3x over the unamplified case.

Amplification was also combined with contamination where the average atomic weight was tripled to approximate the addition of some lead to the fuel. The best-case net energy for contamination with amplification was an order of magnitude lower than with pure fuel. The contamination was not bad (only tripling the average atomic weight), but still had an enormous effect on the gain. This is a significant factor that will need to be studied further. The mitigation of instabilities by tuning the initial fuel density should help prevent this mixing from occurring.

Preheating the fuel is a necessary part of reducing the necessary impact velocity, and may require upwards of several megajoules of energy delivered over a span of a few microseconds. This should be achievable without any new technology, but is more challenging than originally anticipated. Joule heating alone is not capable of reaching the requisite temperatures by itself, but could be used as the first stage of heating when combined with other techniques. One possibility is starting with a gaseous D+T fuel (which can be heated to 100's of eV) which then heats a

^{6}Li + DT ice or LiD pellet which captures the neutrons that are produced as the cavity is collapsed. LiD is a desirable option here since it also breeds the tritium fuel required. Based on the MCNP models that were run, only the configuration with hot fuel surrounding the pellet appeared to be a feasible option. The multi-step configurations reduced the neutron flux substantially.

The other main option for preheating that was examined was an external neutron beam that similarly heating a pellet within the cavity, but this too appears problematic due the large beam current required. However, if the beam could be made more collimated, or it is combined with another heating method, this could be a viable option. There are numerous other potential heating methods were not investigated, such as simply using a particle beam directly as a plasma injector, or using lasers to compress and heat a separate fuel capsule to generate neutrons. This is an area that needs further investigation.

Amplification as it is used here is somewhat similar to boosting in weapons, but still relies on the compression of the projectile to drive it until ignition occurs. The actual effect could not be estimated very well without coupled physics, but MCNP models and approximations with the 1-D model are very encouraging. This could well be an essential element in a practical design.

Magnetic fields could also be added to enhance confinement and reduce conduction loss. Past experiments show promise, and calculations show potential for tremendous gain (although these are a bit optimistic). The magnetic flux compression and amplification both add strong positive feedbacks that would multiply each other.

Overall, it appears that the best configuration would have an electromagnetic accelerator such a coilgun, since these can be scaled up to reach whatever velocity is necessary. A gas gun

does not have this flexibility. At the moment, and pulsed external neutron source appears to be the most feasible option for preheating, although other options may prove fruitful (there are numerous other heating methods developed for MCF). More research is needed here. A lithium metal target with a lead projectile would reduce the necessary fuel density, and reduce mixing with the fuel during compression as well as facilitating amplification. External magnets would provide further positive feedback in gain.

Future Research Possibilities

Many of these things could be tested at minimal expense using an existing light-gas gun facility. The additional instrumentation and equipment are not expensive compared to a traditional fusion experiment that would require whole new facilities, and many years of development.

Follow-up work will focus on developing and modelling practical preheating methods, modelling the instabilities with the interaction between the fuel and target walls, and thermonuclear burn with a more sophisticated code that can more completely represent the physics. Many of the codes available are maintained by the DOE at various national labs, and access is strictly controlled. However, the FLASH code developed by the University of Chicago is available and open source. It is an HEDP code used for inertial confinement fusion and astrophysical applications such as modelling supernovae. Modelling thermonuclear burn is one of its primary capabilities, as well as the other physics required.

A spin-off project is planned that will examine the use of this system for space-propulsion. Most of the energy is released as high-energy neutrons, which can be thermalized to extract the energy and recharge the system. The thermal neutrons could then be used to induce (n,α) reactions in either ^{6}Li or ^{10}B propellant. Since the neutrons have such a short mean-free

path compared to the resulting high-energy α-particles (which would serve as exhaust), a block of Li or B would produce thrust with a specific impulse in excess of 1,000,000 s.

References

[1] J. D. Lawson, "Some Criteria for a Power Producing Thermonuclear Reactor," vol. 70, no. 1, 1957.

[2] P. G. Drazin, "The Stability of a Shear Layer in an Unbounded Heterogeneous Inviscid Fluid," *J. Fluid Mech.*, p. 214, 1958.

[3] "Rayleigh-Taylor Instabilities," in *An Introduction to Inertial Confinement Fusion*, Boca Raton, CRC Press, 2006, pp. 126-141.

[4] M. Laberge, S. Howard, D. Richardson, A. Froese, V. Suponitsky, M. Reynolds and D. Plant, "Acoustically Driven Magnetized Target Fusion," in *2013 IEEE 25th Symposium on Fusion Engineering (SOFE)*, San Francisco, 2013.

[5] F. Winterberg, "Implosion of a Dense Plasma by Hypervelocity Impact," vol. 10, no. 55, 1968.

[6] R. Hibbert, M. J. Cole, M. C. Price and M. J. Burchell, "The Hypervelocity Impact Facility at the University of Kent: Recent Upgrades and Specialized Capabilities," in *14th Hypervelocity Impact Symposium 2017, HVIS2017, 24-28 April 2017*, Canterbury, Kent, UK, 2017.

[7] P. M. H. Kolm, "Basic principles of coaxial launch technology," *IEEE Transactions on Magnetics*, vol. 20, no. 2, pp. 227-230, March 1984.

[8] F. Winterberg, The Release of Thermonuclear Energy by Inertial Confinement: Ways Towards Ignition, Singapore: World Scientific Publishing Co. Pte Ltd., 2010.

[9] K. Ikuta, "A Shaped Projectile for Hypervelocity Impact as a Driver of Nuclear Fusion," *Japanese Journal of Applied Physics,* vol. 25, no. 7, pp. L587-L588, 1986.

[10] Y. Lei, J. Liu and Z. Wang, "Hypervelocity Macroscopic Particle Impact Fusion with DT Methane," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment,* vol. 606, no. 1-2, pp. 157-160, July 2009.

[11] F. Winterberg, "Acceleration of a dense jet to thermonuclear velocities by hypervelocity impact," *IAEA Nulcear Fusion Letters,* vol. 7, no. 289, pp. 289-290, 1967.

[12] S. Olariu, "Hypervelocity-impact ignition of nuclear fusion reactions in laser-compressed deuterium-tritium pellets," *Nuclear Instruments and Methods in Physics Research A,* vol. 366, pp. 376-384, 1995.

[13] N. Cranenburgh, May 2018. [Online]. Available: https://www.createdigital.org.au/fusion-startup-pistol-shrimp-energys-holy-grail/.

[14] Los Alamos Scientific Laboratory, "Impact Fusion Workshop," US Department of Energy, Los Alamos, New Mexico, 1979.

[15] D. Shvarts, J. Delettrez, R. L. McCrory and a. C. P. Verdon, "Self-Consistent Reduction of the Spitzer-Harm Electron Thermal Heat Flux in Steep Temperature Gradients in Laser-Produced Plasmas," *Physical Review Letters,* vol. 47, no. 4, pp. 247-250, July 1981.

[16] J. S. DeGroot and T. E. McCann, "Laser Driven Macroparticles," in *Proceedings of the Impact Fusion Workshop*, Los Alamos, 1979.

[17] M. G. Haines, "The Joule Heating of a Stable Pinched Plasma," *Proc. Phys. Soc.*, vol. 76, p. 250, 1960.

[18] C. Pellegrini, "The history of X-ray free-electron lasers," *Eur. Phys. J. H,* vol. 37, no. 5, pp. 659-708, 2012.

[19] lA. P. TERLETSKII, "Production of Very Strong Magnetic Fields by Rapid Compression of Conducting Shells," *JETP,* vol. 5, no. 2, pp. 301-302, 1957.

[20] S. Younger, I. Lindemuth, R. Reinovsky, C. M. Fowler, J. Goforth and a. C. Ekdahl, "Scientific Collaborations Between Los Alamos and Arzamas-16 Using Explosive-Driven Flux Compression Generators," *Los Alamos Science*, pp. 48-71, 1996.

[21] J. C. SOLEM and M. G. SHEPPARD, "Experimental Quantum Chemistry at Ultrahigh Magnetic Fields: Some Opportunities," *International Journal of Quantum Chemistry,* vol. 64, no. 5, p. 619–628, 1997.

[22] P. Chang, G. Fiksel, M. Hohenberger, J. P. Knauer, R. Betti, F. J. Marshall, D. D. Meyerhofer, F. H. Se´guin and a. R. D. Petrasso, "Fusion Yield Enhancement in Magnetized Laser-Driven Implosions," *Phys. Rev. Lett.,* vol. 107, no. 035006, 2011.

[23] Naval Research Labratory, NRL Plasma Formulary, Washington, DC: United Staes Navy, 2013.

[24] K. Miyamoto, Plasma Physics for Nuclear Fusion, Revised Edition, MIT Press, 1989.

[25] Wolfram Research, *Mathematica*, 2018.

[26] W. Taitano, Interviewee, *Staff Scientist, Los Alamos National Lab.* [Interview]. 2018.

[27] COMSOL group, *COMSOL Multphysics*, V 5.3a ed.

[28] R. J. LeVeque and R. Fazio, 2012. [Online]. Available: https://staff.washington.edu/rjl/clawpack/movingmesh/ifpiston/. [Accessed February 2019].

[29] R. J. LeVeque, *CLAWPACK Version 2.1*, 1995.

[30] R. Fazio and R. J. LeVeque, "Moving-Mesh Methods for One-Dimensional Hyperbolic Problems Using CLAWPACK," *Computers and Mathematics with Applications,* vol. 45, pp. 273-298, 2003.

[31] R. LeVeque, "Wave propagation algorithms for multi-dimensional hyperbolic systems," *J. Comput. Phys.*, vol. 131, pp. 327-353, 1997.

[32] D. Christman, J. Gehring, C. Maiden and A. Wenzel, "STUDY OF THE PHENOMENA OF HYPERVELOCITY IMPACT," GM Defense Research Laboratories, Santa Barbara, California, 1963.

[33] L. C. Chhabildas, "Hypervelocity Impact Phenomena," Albuquerque, New Mexico, 1995.

[34] ANSYS, *Autodyn.* Software

[35] L. Libersky and A. Petschek, "Smooth Particle Hydrodynamics with Strength of Materials, Advances in the Free Lagrange Method. Lecture Notes in Physics.," vol. 395, pp. 248-257, 1990.

[36] L. Libersky, A. Petschek, A. Carney, T. Hipp, J. Allahdadi and F. High, "Strain Lagrangian hydrodynamics: a three-dimensional SPH code for dynamic material response," *J. Comput. Phys.*, vol. 109, no. 1, pp. 67-75, 1993.

[37] T. Goorley and e. al., "Initial MCNP6 Release Overview," *Nuclear Technology,* vol. 180, pp. 298-315, 2012.

[38] J. G. C. M. A. W. D.R. Christman, "STUDY OF THE PHENOMENA OF HYPERVELOCITY IMPACT," GM Defense Research Laboratories, Santa Barbara, California, 1963.

[39] R. D. Richtmyer, "Taylor Instability in a Shock Acceleration of Compressible Fluids," *Communications on Pure and Applied Mathematics,* vol. 13, no. 2, p. 297–319, 1960.

[40] E. E. Meshkov, "Instability of the Interface of Two Gases Accelerated by a Shock Wave," *Soviet Fluid Dynamics,* vol. 4, no. 5, p. 101–104, 1969.

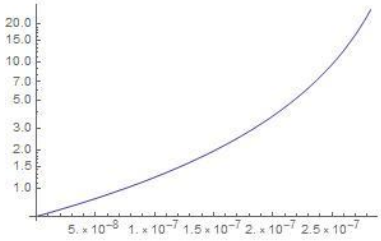
[41] V. Suponitsky, "On the collapse of a Gas Cavity by an Imploding Molten Lead Shell and Richtmyer-Meshkov Instability," General Fusion Inc, 2013.

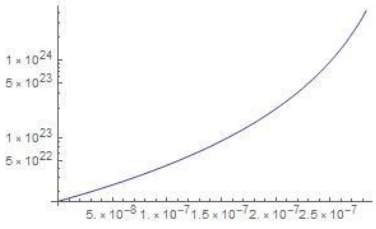
[42] Energy Information Administration, "US Energy Information Administration Electric Power Monthly with Data for March 2019," May 2019.

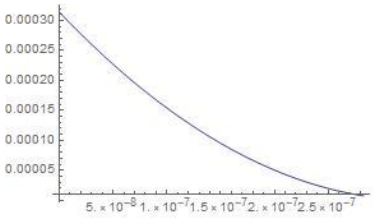
Appendix A, Quasi-1-D Model Spherical Case

```
(******** CONSTANTS **********)

(** NOTE: max pressure ≈ 10^5 MPa from autodyn simulation **)
η = 0.4; (** overall plant efficiency from heat to kinetic energy **)
η_gun = 0.75;
ϵ_0 = 8.85418782 × 10^-12;
e = 1.60217662 × 10^-19; (** electron charge, C **)
L = 1 / 100; (** projectile wall thickness, m **)
kPb = 34.7; (** lead thermal conductivity, W/m*K **)
R_0 = 5 / 1000; (** initial cavity radius, m **)
(*A=(R_0)^2 π; (** cavity area cross section, m^2 **)*)
v = 15000; (** projectile velocity, m/s **)
(*h_0=10/100; (** initial cavity height, m**)*)
Kperkev = 11604525.00617; (** conversion from Kelvin to keV **)
keVperJ = 6.242 × 10^15;
R = R_0 - v t;
SurfArea = (4 π R^2);
V = (4/3) π R^3; (** cavity volume, m^3 **)
V_0 = (4/3) π R_0^3;
T_0 = 0.6 * Kperkev; (** initial temp of fuel gas, K **);
(*T_0=20000; (** alternate initial temperature for testing Winterberg concept **)*)
c_p = 3910.5 T_0^0.1932; (** fuel heat capacity J/kg/K, approximate **)
γ = 5 / 3;
z = 1; (** number of protons in fuel atoms **)
k = 1.38 × 10^-23; (** Boltzmann constant J/K**)
σ = 5.67 × 10^-8; (** Stefan Boltzmann constant, watt/m^2/K^4 **)
m_e = 9.1 × 10^-31; (** electron mass, kg **)
Q = 18200 + 4800; (** energy of each α particle produced per reaction (D+T and ^6Li(n,α)), keV **)
QJ = Q / keVperJ; (** thermal energy produced per reaction, J **)
Ech = 3520 / keVperJ; (** energy of charged particles produced by DT fusion (to determine if ignition has been achieved) **)
mp = 0.05; (** projectile mass, kg **)
ρpb = 11.34; (** lead density, g/cc **)
ρpbSI = ρpb * 1000; (** lead density, kg/m^3 (SI units) **)
utm = 3.087 × 10^9; (** modulus of toughness of Lead, J/m^3 **)
volp = mp / (ρpb * 1000); (** projectile solid volume,
m^3 **)
```

```
(******** ADIABATIC RELATIONSHIPS AND PLASMA PARAMETERS **********)

P0 = 101000; (** initial pressure, Pa **)
(*ρ0=P0 (2.5/1000)/(8.31 T0) *50000;*)
(*n0=ρ0 6.022 10^23/(2.5/1000) (** initial number density, cm^-3 **)*)
n0 = 0.75 * 2 × 10^22 * 100^3;
ρ0 = 0.0025 n0/(6.022 × 10^23)
num = n0 V0;
const = T0 V0^(γ-1); (** adiabtic temp, volume constant **)
const2 = P0 V0^γ; (** adiabatic pressure, volume constant **)
n = num/((4/3) π R^3); (** number density as a function of time **)
σv = (3.68 × 10^-12 T^-2/3 Exp[-19.94 T^-1/3])/100^3; (** fusion cross section for D+T m^3/s **)
P = const2/((4/3) π R^3)^γ; (** plasma pressure, Pa **)
T = const/((4/3) π R^3)^(γ-1) / Kperkev; (** plasma temperature as a function of time, keV **)
(*T=(T0 (P/P0)^((γ-1)/γ)) / Kperkev (** plasma temperature as a function of time, keV **)*)
(*time=h0/v *.985;(** collapse time, s **)*)
time = R0/v - ((2^(-1+2γ) (π/3)^γ v^2 ρpb)/const2)^(1/(3γ))/v (** collapse time for spherical geometry, s **)
LogPlot[T, {t, 0, time}, PlotRange → All]
LogPlot[n/100^3, {t, 0, time}, PlotRange → All]
Plot[SurfArea, {t, 0, time}, PlotRange → All]

62.271670541348406`
2.829148352150036`*^-7
```



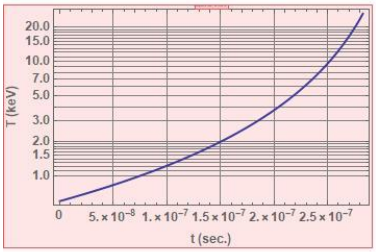


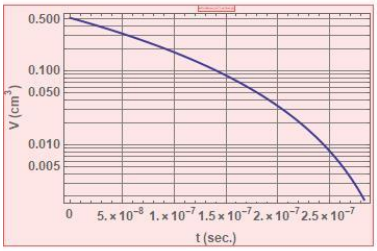


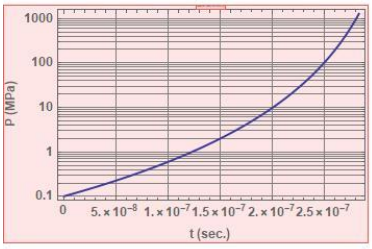
```
(******** ENERGY GAIN **********)

power = 1/4 n^2 σv QJ V; (** function for fusion power release, Watts **)

ThermalEnergy = NIntegrate[power, {t, 0, time}]; (** total fusion energy released from D+T reactions, and (n,α) in ⁶Li **)

energy = η * ThermalEnergy (** net electrical energy output, J **)

work = const2 ((4/3 π (R₀ - v time)^3)^(1-γ) - V₀^(1-γ)) / (1 - γ) ;(** work done on fuel gas during adiabatic compression, J **)

5.652167304969409` *^10

(******** ENERGY LOSSES **********)

Eb = NIntegrate[1.5 × 10^-38 (z)^2 (n)^2 (T / keVperJ / e)^1/2 * V, {t, 0, time}] (** electron Bremsstrahlung loss,

J (from Plasma Physics for Nuclear Fusion, Miyamoto, pg 7) **)

(** Rosseland opacity calculation for radiation loss **)

g_ff = 1; (** gaunt factor, conservative for these temperatures (probably closer to 2 or 3 in reality) **)

κ_ff = 3.68 × 10^18 g_ff 0.0025 n / (6.022 × 10^23) 1 / T^3.5 ;(** free-free opacity, m^2/kg **)

κ_es = 0.02 * 2; (** electron scattering opacity, m^2/kg **)

κ̄ = Mean[{κ_ff, κ_es}]; (** Rosseland mean opacity, m^2/kg **)

λ_R = 1 / (0.0025 n / 6.022×10^23 κ̄) ;

rad = NIntegrate[16/3 λ_R SurfArea σ T^4, {t, 0, time}] (** Radiation loss **)

(*****************************************************)

heat = 4/3 π (R₀ - v time)^3 * 3 num / (4/3 π (R₀ - v time)^3) k const / (4/3 π (R₀ - v time)^3)^(γ-1) (** Energy lost heating fuel up to final temperature and density **)

deform = utm * volp (** energy lost deforming the projectile (integral of stress-strain curve times solid volume) **)

KE = (1 - η) 1/2 mp v^2 (** fraction of kinetic energy lost, 1-η is lost out the cooling stacks **)

(*conduct=NIntegrate[2 kPb A (const / (A(h₀-(t)*v)))^290 / L +2 √(A/π) π (h₀-(t)*v) kPb (const / (A(h₀-(t)*v)))^290 / L ,{t,0,time}] ; (** thermal conduction losses through Pb wall **)*)

conduct = NIntegrate[0.1 3^3/2 / 2 n k const / (4/3 π R^3)^(γ-1) (k const / (4/3 π R^3)^(γ-1) / m_e)^1/2 SurfArea, {t, 0, time}] (** thermal conduction losses through plasma **)

joule = ρ₀ V₀ c_p (T₀ - 10) (** energy required to pre-heat fuel through joule heating/discharge **)

losses = KE + deform + Eb + heat + conduct + joule + rad (** net energy losses **)
1.3849991162223837` *^9
1.3394377029659015` *^-36
9.895700050289223` *^7
13611.111111111113`
3.375` *^6
3.867919518846805` *^10
1.8634779302936353` *^7
4.018517469560737` *^10
```

```
(******** NET ENERGY **********)

netenergy = energy - losses (** net energy produced, J **)
gain = energy / losses (** energy gain **)
workFraction = Abs[work / (1/2 mp v^2)] * 100; (** percent of projectile kinetic energy used to adibatically compress fuel **)

finalρ = ρ0 V0 / ((4/3) π (R0 - v time)^3) (** final mass density of fuel, kg/m^3 **)

(*hf = (h0 - (time) *v) *1000 (** final height of cavity, mm **)*)
finalR = (R0 - v time) * 1000 (** final cavity radius, mm **)
Pfinal = const2 / ((4/3) π (R0 - v time)^3)^γ / 1000000 (** final pressure in MPa **)


1.6336498354086723`*^10
1.4065304799053782`
17995.21494523736`
0.756277471774946`
1275.7500000000043`

(******** DO WE HAVE IGNITION? **********)

IgEnergy = energy / QJ Ech (** total α energy released from D+T reactions **)
IgLoss = Eb / (1 - η) + heat + conduct (** energy loss relevant to ignition **)
gain = IgEnergy / IgLoss (** if this is >1, then ignition has been achieved **)
8.650273440648836`*^9
4.108648404934158`*^10
0.2105381767459233`
```

```
(******** PLOTS **********)

LogPlot[T, {t, 0, time}, PlotLabel → "Temperature [keV] vs Time [s]", PlotRange → All, LabelStyle → {Bold, Medium}, GridLines → Automatic, PlotStyle → {Thick},
 Frame → True, FrameLabel → {"t (sec.)", "T (keV)"}]
LogPlot[V * 100^3, {t, 0, time}, PlotLabel → "Cavity Volume [cm^3] vs Time [s]", PlotRange → All, FrameLabel → {"t (sec.)", "V (cm^3)"}, LabelStyle → {Bold, Medium},
 GridLines → Automatic, PlotStyle → {Thick}, Frame → True]
LogPlot[P / 1000000, {t, 0, time}, PlotLabel → "Pressure [MPa] vs Time [s]", PlotRange → All, FrameLabel → {"t (sec.)", "P (MPa)"}, LabelStyle → {Bold, Medium},
 GridLines → Automatic, PlotStyle → {Thick}, Frame → True]
LogPlot[n / 100^3, {t, 0, time}, PlotLabel → "Number Density [cm^-3] vs Time [s]", PlotRange → All, FrameLabel → {"t (sec.)", "n (cm^-3)"}, LabelStyle → {Bold, Medium},
 GridLines → Automatic, PlotStyle → {Thick}, Frame → True]
```



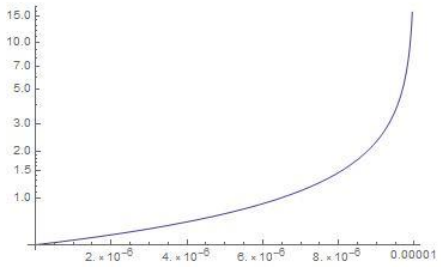


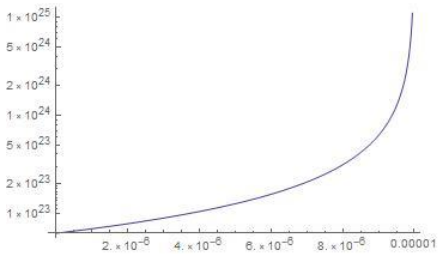


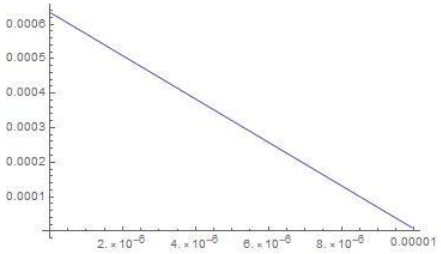
Appendix B, Quasi-1-D Model Cylindrical Case

```
(******** CONSTANTS **********)

(** NOTE: max pressure ≈ 10^5 MPa from autodyn simulation **)
η = 0.4; (** overall plant efficiency from heat to kinetic energy **)
η_gun = 0.75;
ϵ_0 = 8.85418782 × 10^-12;
e = 1.60217662 × 10^-19; (** electron charge, C **)
L = 1 / 100; (** projectile wall thickness, m **)
kPb = 34.7; (** lead thermal conductivity, W/m*K **)
R_0 = 1 / 1000; (** cavity radius, m **)
A = (R_0)^2 π; (** cavity area cross section, m^2 **)
v = 10000; (** projectile velocity, m/s **)
h_0 = 10 / 100; (** initial cavity height, m**)
Kperkev = 11604525.00617; (** conversion from Kelvin to keV **)
keVperJ = 6.242 × 10^15;
V = A (h_0 - t * v); (** cavity volume, m^3 **)
V_0 = A h_0;
T_0 = 0.5 * Kperkev; (** initial temp of fuel gas, K **);
(*T_0=20000; (** alternate initial temperature for testing Winterberg concept **)*)
c_p = 3910.5 T_0^0.1932; (** fuel heat capacity J/kg/K, approximate **)
γ = 5 / 3;
z = 1; (** number of protons in fuel atoms **)
k = 1.38 × 10^-23; (** Boltzmann constant J/K**)
σ = 5.67 × 10^-8; (** Stefan Boltzmann constant, watt/m^2/K^4 **)
m_e = 9.1 × 10^-31; (** electron mass, kg **)
Q = 18200 + 4800; (** energy of each α particle produced per reaction (D+T and ^6Li(n,α)), keV **)
QJ = Q / keVperJ; (** thermal energy produced per reaction, J **)
Ech = 3520 / keVperJ; (** energy of charged particles produced by DT fusion (to determine if ignition has been achieved) **)
mp = 0.05; (** projectile mass, kg **)
ρpb = 11.34; (** lead density, g/cc **)
ρpbSI = ρpb * 1000; (** lead density, kg/m^3 (SI units) **)
utm = 3.087 × 10^9; (** modulus of toughness of Lead, J/m^3 **)
volp = mp / (ρpb * 1000); (** projectile solid volume,
m^3 **)
```

```
(******** ADIABATIC RELATIONSHIPS AND PLASMA PARAMETERS **********)

P0 = 100 * 10^6; (** initial pressure, Pa **)
ρ0 = P0 (2.5 / 1000) / (8.31 T0) * 50000;
n0 = ρ0 6.022 * 10^23 / (2.5 / 1000) (** initial number density, cm^-3 **)
(*n0=2 10^22*100^3;
ρ0=0.0025 n0 / (6.022 10^23) *)
num = n0 V0;
const = T0 V0^(γ-1); (** adiabtic temp, volume constant **)
const2 = P0 V0^γ; (** adiabatic pressure, volume constant **)
n = num / (A (h0 - t * v)); (** number density as a function of time **)
σv = 3.68 * 10^-12 T^-2/3 Exp[-19.94 T^-1/3] / 100^3; (** fusion cross section for D+T m^3/s **)
P = const2 / (A (h0 - t * v))^γ; (** plasma pressure, Pa **)
T = const / (A (h0 - t * v))^(γ-1) / Kperkev; (** plasma temperature as a function of time, keV **)
SurfArea = (2 A + 2 π R0 (h0 - v t));
(*T=(T0 (ρ/ρ0)^((γ-1)/γ)) / Kperkev (** plasma temperature as a function of time, keV **)*)
(*time=h0/v *.985;(** collapse time, s **)*)
time = h0/v - 2^(1/γ) (v^2 ρpbSI / const2)^(-1/γ) / (A v) (** collapse time for cylindrical geometry, s **)
LogPlot[T, {t, 0, time}, PlotRange → All]
LogPlot[n/100^3, {t, 0, time}, PlotRange → All]
Plot[SurfArea, {t, 0, time}, PlotRange → All]

6.244711205501618` *^28
9.944043548638679` *^-6
```







```
(******** ENERGY GAIN **********)

power = 1/4 n^2 σv QJ V; (** function for fusion power release, Watts **)

ThermalEnergy = NIntegrate[power, {t, 0, time}]; (** total fusion energy released from D+T reactions, and (n,α) in ⁶Li **)
energy = η * ThermalEnergy (** net electrical energy output, J **)
work = const2 ((A (h₀ - (time) * v))^(1-γ) - V₀^(1-γ))/(1 - γ) ;(** work done on fuel gas during adiabatic compression, J **)
6.009629767586353` *^11

(******** ENERGY LOSSES **********)

Eb = NIntegrate[1.5 × 10^-38 (z)^2 (n)^2 (T / keVperJ / e)^(1/2) * V, {t, 0, time}] (** electron Bremsstrahlung loss,
J (from Plasma Physics for Nuclear Fusion, Miyamoto, pg 7) **)

(** Rosseland opacity calculation for radiation loss **)
g_ff = 1; (** gaunt factor, conservative for these temperatures (probably closer to 2 or 3 in reality) **)
κ_ff = 3.68 × 10^18 g_ff (0.0025 n)/(6.022 × 10^23) 1/T^3.5 ;(** free-free opacity, m^2/kg **)
κ_es = 0.02 * 2; (** electron scattering opacity, m^2/kg **)
κ̄ = Mean[{κ_ff, κ_es}]; (** Rosseland mean opacity, m^2/kg **)
λ_R = 1/((0.0025 n)/(6.022 10^23) κ̄) ;
rad = NIntegrate[16/3 λ_R SurfArea σ T^4, {t, 0, time}] (** Radiation loss **)
(*****************************************************)

heat = A (h₀ - (time) v) * 3 num/(A (h₀ - (time) * v)) k const/(A (h₀ - (time) * v))^(γ-1) (** Energy lost heating fuel up to final temperature and density **)
deform = utm * volp (** energy lost deforming the projectile (integral of stress-strain curve times solid volume) **)
KE = (1 - η) 1/2 mp v^2 (** fraction of kinetic energy lost, 1-η is lost out the cooling stacks **)
(*conduct=NIntegrate[2 kPb A (const/(A(h₀-(t)*v)))^-290/L +2 √(A/π) π (h₀-(t)*v) kPb (const/(A(h₀-(t)*v)))^-290/L,{t,0,time}]; (** thermal conduction losses through Pb wall **)*)
conduct = NIntegrate[0.1 3^(3/2)/2 n k const/(A (h₀ - t * v))^(γ-1) (k const/((A (h₀-t*v))^(γ-1)) / m_e)^(1/2) SurfArea, {t, 0, time}] (** thermal conduction losses through plasma **)
joule = ρ₀ V₀ c_p (T₀ - 10) (** energy required to pre-heat fuel through joule heating/discharge **)

losses = KE + deform + Eb + heat + conduct + joule + rad (** net energy losses **)
5.7106980141407814` *^10
2.863722857079955` *^-38
1.495178932571494` *^8
13611.111111111113`
1.5` *^6
5.331217920044078` *^11
3.744703323330664` *^7
5.904172506834171` *^11
```

(******** NET ENERGY **********)

netenergy = energy - losses (** net energy produced, J **)

gain = energy / losses (** energy gain **)

$$\text{workFraction} = \text{Abs}\left[\frac{\text{work}}{\frac{1}{2}\,\text{mp}\,v^2}\right] * 100$$ (** percent of projectile kinetic energy used to adibatically compress fuel **)

$$\text{final}\rho = \rho_0 \frac{V_0}{(h_0 - (\text{time}) * v)\,A}$$ (** final mass density of fuel, kg/m^3 **)

$$h_f = (h_0 - (\text{time}) * v) * 1000$$ (** final height of cavity, mm **)

$$\text{Pfinal} = \frac{\text{const2}}{(A\,(h_0 - \text{time} * v))^{\gamma}} \Big/ 1000000$$ (** final pressure in MPa **)

1.054572607521814`*^10

1.01786148027181`

0.05791961089916113`

46329.90946285008`

0.5595645136132199`

566999.9999999906`

(******** DO WE HAVE IGNITION? **********)

$$\text{IgEnergy} = \frac{\text{energy}}{QJ}\,\text{Ech}$$ (** total α energy released from D+T reactions **)

$$\text{IgLoss} = \frac{Eb}{(1-\eta)} + \text{heat} + \text{conduct}$$ (** energy loss relevant to ignition **)

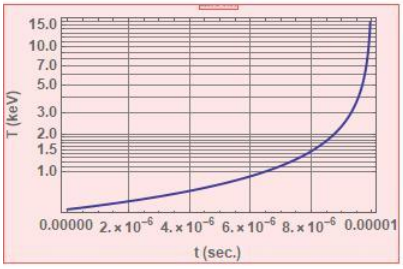
gain = IgEnergy / IgLoss (** if this is >1, then ignition has been achieved **)

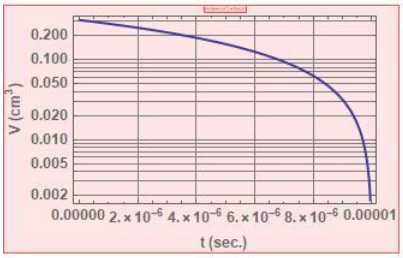
9.197346426914766`*^10

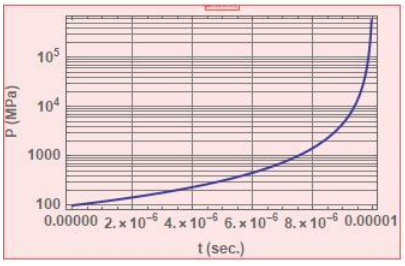
6.284496101333446`*^11

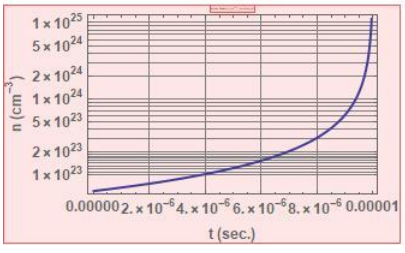
0.14634978331776308`

```
(******** PLOTS **********)

LogPlot[T, {t, 0, time}, PlotLabel → "Temperature [keV] vs Time [s]", PlotRange → All, LabelStyle → {Bold, Medium}, GridLines → Automatic, PlotStyle → {Thick},
 Frame → True, FrameLabel → {"t (sec.)", "T (keV)"}]
LogPlot[V * 100^3, {t, 0, time}, PlotLabel → "Cavity Volume [cm^3] vs Time [s]", PlotRange → All, FrameLabel → {"t (sec.)", "V (cm^3)"}, LabelStyle → {Bold, Medium},
 GridLines → Automatic, PlotStyle → {Thick}, Frame → True]
LogPlot[P / 1000000, {t, 0, time}, PlotLabel → "Pressure [MPa] vs Time [s]", PlotRange → All, FrameLabel → {"t (sec.)", "P (MPa)"}, LabelStyle → {Bold, Medium},
 GridLines → Automatic, PlotStyle → {Thick}, Frame → True]
LogPlot[n / 100^3, {t, 0, time}, PlotLabel → "Number Density [cm^-3] vs Time [s]", PlotRange → All, FrameLabel → {"t (sec.)", "n (cm^-3)"}, LabelStyle → {Bold, Medium},
 GridLines → Automatic, PlotStyle → {Thick}, Frame → True]
```









Appendix C, Fortran-77 modified source files for 1-D hydrodynamic code

The following is the modified Fortran 77 source code for the 1-D hydrodynamic model. Only the modified source files are included below. Each subroutine is labeled by filename so they can be easily distinguished. Modified sections of code are highlighted in yellow.

CLAW1.F

```
c
c    Modified version of claw1 for moving mesh method
c    Includes calls to inter1 and set1aux each time step
c
c
c     ============================================================
      subroutine claw1(maxmx,meqn,mwaves,mbc,mx,
     &   q,aux,dx,tstart,tend,dtv,cflv,nv,method,mthlim,
     &           work,mwork,info,bc1,rp1,src1,set1,inter1)
c     ============================================================
c
c  Solves a hyperbolic system of conservation laws in one space dimension
c  of the general form
c
c     capa * q_t + A q_x = psi
c
c  The "capacity function" capa(x) and source term psi are optional
c  (see below).
c
c  For a more complete description see the documentation in the directory
c  claw/doc,  especially note1.ps and note2.ps.
c
c  Sample driver programs and user-supplied subroutines are available.
c  see the the directory claw/clawpack/1d/example for one example, and
c  codes in claw/applications for more extensive examples.
c
c  ----------------------------------------------------------
c
c  The user must supply the following subroutines:
c
c    bc1, rp1        subroutines specifying the boundary conditions and
c                    Riemann solver.
c                    These are described in greater detail below.
c
c  In addition, if the equation contains source terms psi, then the user
c  must provide:
c
c    src1               subroutine that solves capa * q_t = psi
c                       over a single time step.
c
c  These routines must be declared EXTERNAL in the main program.
c  For description of the calling sequences, see below.
```

```
c
c
c
c  Description of parameters...
c  ----------------------------
c
c
c    maxmx is the maximum number of interior grid points in x,
c          and is used in declarations of the array q.
c
c    meqn is the number of equations in the system of
c         conservation laws.
c
c    mwaves is the number of waves that result from the
c           solution of each Riemann problem.  Often mwaves = meqn but
c           for some problems these may be different.
c
c    mbc is the number of "ghost cells" that must be added on to each
c       side of the domain to handle boundary conditions.  The cells
c       actually in the physical domain are labelled from 1 to mx in x.
c       The arrays are dimensioned actually indexed from 1-mbc to mx+mbc.
c       For the methods currently implemented, mbc = 2 should be used.
c       If the user implements another method that has a larger stencil and
c       hence requires more ghost cells, a larger value of mbc could be used.
c       q is extended from the physical domain to the ghost cells by the
c       user-supplied routine bc1.
c
c    mx is the number of grid cells in the x-direction, in the
c       physical domain.  In addition there are mbc grid cells
c       along each edge of the grid that are used for boundary
c       conditions.
c       Must have mx .le. maxmx
c
c    q(1-mbc:maxmx+mbc, meqn)
c        On input:  initial data at time tstart.
c        On output: final solution at time tend.
c        q(i,m) = value of mth component in the i'th cell.
c        Values within the physical domain are in q(i,m)
c                for i = 1,2,...,mx
c        mbc extra cells on each end are needed for boundary conditions
c        as specified in the routine bc1.
c
c    aux(1-mbc:maxmx+mbc, maux)
c        Array of auxiliary variables that are used in specifying the
problem.
c        If method(7) = 0 then there are no auxiliary variables and aux
c                         can be a dummy variable.
c        If method(7) = maux > 0 then there are maux auxiliary variables
c                         and aux must be dimensioned as above.
c
c        Capacity functions are one particular form of auxiliary variable.
c        These arise in some applications, e.g. variable coefficients in
c        advection or acoustics problems.
c        See Clawpack Note # 5 for examples.
c
c        If method(6) = 0 then there is no capacity function.
c        If method(6) = mcapa > 0  then there is a capacity function and
```

```
c            capa(i), the "capacity" of the i'th cell, is assumed to be
c            stored in aux(i,mcapa).
c            In this case we require method(7).ge.mcapa.
c
c    dx = grid spacing in x.
c         (for a computation in ax <= x <= bx,  set dx = (bx-ax)/mx.)
c
c    tstart = initial time.
c
c    tend = Desired final time (on input).
c         = Actual time reached (on output).
c
c    dtv(1:5) = array of values related to the time step:
c                (Note: method(1)=1 indicates variable size time steps)
c         dtv(1) = value of dt to be used in all steps if method(1) = 0
c                = value of dt to use in first step if method(1) = 1
c         dtv(2) = unused if method(1) = 0.
c                = maximum dt allowed if method(1) = 1.
c         dtv(3) = smallest dt used (on output)
c         dtv(4) = largest dt used (on output)
c         dtv(5) = dt used in last step (on output)
c
c    cflv(1:4) = array of values related to Courant number:
c         cflv(1) = maximum Courant number to be allowed.  With variable
c                   time steps the step is repeated if the Courant
c                   number is larger than this value.  With fixed time
c                   steps the routine aborts.  Usually cflv(1)=1.0
c                   should work.
c         cflv(2) = unused if method(1) = 0.
c                 = desired Courant number if method(1) = 1.
c                   Should be somewhat less than cflv(1), e.g. 0.9
c         cflv(3) = largest Courant number observed (on output).
c         cflv(4) = Courant number in last step (on output).
c
c    nv(1:2) = array of values related to the number of time steps:
c         nv(1) = unused if method(1) = 0
c               = maximum number of time steps allowed if method(1) = 1
c         nv(2) = number of time steps taken (on output).
c
c    method(1:7) = array of values specifying the numerical method to use
c         method(1) = 0 if fixed size time steps are to be taken.
c                       In this case, dt = dtv(1) in all steps.
c                   = 1 if variable time steps are to be used.
c                       In this case, dt = dtv(1) in the first step and
c                       thereafter the value cflv(2) is used to choose the
c                       next time step based on the maximum wave speed seen
c                       in the previous step.  Note that since this value
c                       comes from the previous step, the Courant number will
c                       not in general be exactly equal to the desired value
c                       If the actual Courant number in the next step is
c                       greater than 1, then this step is redone with a
c                       smaller dt.
c
c         method(2) = 1 if Godunov's method is to be used, with no 2nd order
c                       corrections.
c                   = 2 if second order correction terms are to be added,
with
```

```
c                       a flux limiter as specified by mthlim.
c                   = 3 if "third order" correction terms are to be added,
c                       based on my paper my paper "A high-resolution
c                       conservative algorithm for advection in
c                       incompressible flow".
c                       This is currently recommended only for problems
c                       with smooth solutions, using no limiter (mthlim = 0)
c
c         method(3)  is not used in one-dimension.
c
c         method(4) = 0 to suppress printing
c                   = 1 to print dt and Courant number every time step
c
c         method(5) = 0 if there is no source term psi.  In this case
c                       the subroutine src2 is never called so a dummy
c                       parameter can be given.
c                   = 1 if there is a source term.  In this case
c                       the subroutine src2 must be provided.
c
c         method(6) = 0 if there is no capacity function capa.
c                   = mcapa > 0 if there is a capacity function.  In this
case
c                       aux(i,mcapa) is the capacity of the i'th cell and you
c                       must also specify method(7) .ge. mcapa and set aux.
c
c         method(7) = 0 if there is no aux array used.
c                   = maux > 0  if there are maux auxiliary variables.
c
c
c         The recommended choice of methods for most problems is
c            method(1) = 1,  method(2) = 2.
c
c    mthlim(1:mwaves) = array of values specifying the flux limiter to be
used
c                       in each wave family mw.  Often the same value will be
used
c                       for each value of mw, but in some cases it may be
c                       desirable to use different limiters.  For example,
c                       for the Euler equations the superbee limiter might be
c                       used for the contact discontinuity (mw=2) while another
c                       limiter is used for the nonlinear waves.  Several
limiters
c                       are built in and others can be added by modifying the
c                       subroutine philim.
c
c        mthlim(mw) = 0 for no limiter
c                   = 1 for minmod
c                   = 2 for superbee
c                   = 3 for van Leer
c                   = 4 for monotonized centered
c
c
c    work(mwork) = double precision work array of length at least mwork
c
c    mwork = length of work array.  Must be at least
c               (maxmx + 2*mbc) * (2 + 3*meqn + mwaves + meqn*mwaves)
c            If mwork is too small then the program returns with info = 4
```

```
c            and prints the necessary value of mwork to unit 6.
c
c
c    info = output value yielding error information:
c         = 0 if normal return.
c         = 1 if mx.gt.maxmx   or  mbc.lt.2
c         = 2 if method(1)=0 and dt doesn't divide (tend - tstart).
c         = 3 if method(1)=1 and cflv(2) > cflv(1).
c         = 4 if mwork is too small.
c         = 11 if the code attempted to take too many time steps, n > nv(1).
c              This could only happen if method(1) = 1 (variable time steps).
c         = 12 if the method(1)=0 and the Courant number is greater than 1
c              in some time step.
c
c           Note: if info.ne.0, then tend is reset to the value of t actually
c           reached and q contains the value of the solution at this time.
c
c    User-supplied subroutines
c    -------------------------
c
c    bc1 = subroutine that specifies the boundary conditions.
c         This subroutine should extend the values of q from cells
c         1:mx to the mbc ghost cells along each edge of the domain.
c
c          The form of this subroutine is
c  ------------------------------------------------
c     subroutine bc1(maxmx,meqn,mbc,mx,q,aux,t)
c     implicit double precision (a-h,o-z)
c     dimension   q(1-mbc:maxmx+mbc, meqn)
c     dimension aux(1-mbc:maxmx+mbc, *)
c  ------------------------------------------------
c
c
c    rp1 = user-supplied subroutine that implements the Riemann solver
c
c          The form of this subroutine is
c  ------------------------------------------------
c     subroutine
rp1(maxmx,meqn,mwaves,mbc,mx,ql,qr,auxl,auxr,wave,s,amdq,apdq)
c     implicit double precision (a-h,o-z)
c     dimension   ql(1-mbc:maxmx+mbc, meqn)
c     dimension   qr(1-mbc:maxmx+mbc, meqn)
c     dimension auxl(1-mbc:maxmx+mbc, *)
c     dimension auxr(1-mbc:maxmx+mbc, *)
c     dimension wave(1-mbc:maxmx+mbc, meqn, mwaves)
c     dimension    s(1-mbc:maxmx+mbc, mwaves)
c     dimension amdq(1-mbc:maxmx+mbc, meqn)
c     dimension apdq(1-mbc:maxmx+mbc, meqn)
c  ------------------------------------------------
c
c         On input, ql contains the state vector at the left edge of each
cell
c                   qr contains the state vector at the right edge of each
cell
c                auxl contains auxiliary values at the left edge of each
cell
```

```
c                auxr contains auxiliary values at the right edge of each
cell
c
c         Note that the i'th Riemann problem has left state qr(i-1,:)
c                                             and right state ql(i,:)
c         In the standard clawpack routines, this Riemann solver is
c         called with ql=qr=q along this slice.  More flexibility is allowed
c         in case the user wishes to implement another solution method
c         that requires left and rate states at each interface.

c         If method(7)=maux > 0 then the auxiliary variables along this slice
c         are passed in using auxl and auxr.  Again, in the standard routines
c         auxl=auxr=aux in the call to rp1.
c
c          On output,
c              wave(i,m,mw) is the m'th component of the jump across
c                           wave number mw in the ith Riemann problem.
c              s(i,mw) is the wave speed of wave number mw in the
c                           ith Riemann problem.
c              amdq(i,m) = m'th component of A^- Delta q,
c              apdq(i,m) = m'th component of A^+ Delta q,
c                     the decomposition of the flux difference
c                         f(qr(i-1)) - f(ql(i))
c                     into leftgoing and rightgoing parts respectively.
c
c           It is assumed that each wave consists of a jump discontinuity
c           propagating at a single speed, as results, for example, from a
c           Roe approximate Riemann solver.  An entropy fix can be included
c           into the specification of amdq and apdq.
c
c    src1 = subroutine for the source terms that solves the equation
c               capa * q_t = psi
c           over time dt.
c
c           If method(5)=0 then the equation does not contain a source
c           term and this routine is never called.  A dummy argument can
c           be used with many compilers, or provide a dummy subroutine that
c           does nothing (such a subroutine can be found in
c           clawpack/1d/misc/src1xx.f)
c
c           The form of this subroutine is
c  ------------------------------------------------
c     subroutine src1(maxmx,meqn,mbc,mx,q,aux,t,dt,method)
c     implicit double precision (a-h,o-z)
c     dimension   q(1-mbc:maxmx+mbc, meqn)
c     dimension aux(1-mbc:maxmx+mbc, *)
c  ------------------------------------------------
c      If method(7)=0  or the auxiliary variables are not needed in this
solver,
c      then the latter dimension statement can be omitted, but aux should
c      still appear in the argument list.
c
c      On input, q(i,m) contains the data for solving the
c                source term equation.
c      On output, q(i,m) should have been replaced by the solution to
c                 the source term equation after a step of length dt.
```

```
c
c
c  NOTES:
c  ------
c
c  -- This code is written for clarity rather than efficiency in many
c     respects.
c
c  -- Most of this routine is concerned with checking for errors and
c     choosing time steps.  The main work of each time step is done in step1.
c
c  -- Strang splitting is used to handle the source term.  This works well
c     and can be used to achieve second order accuracy on smooth solutions
c     provided that the source terms are not "stiff".  If they are, meaning
c     that the time scale on which solutions to the ODE vary are much faster
c     than the time scales of the homogeneous conservation law, then to
c     obtain good results it may be necessary to take dx and dt small enough
c     that the rapid transients are well-resolved.
c
c  -- Note that with variable time steps, the value dtv(2)
c     may be used to limit the time step to some maximum value, independent
c     of the Courant number.  This may be chosen based on the source terms.
c
c
c
c =====================================================================
c
c  Copyright 1994 R. J. LeVeque
c
c  This software is made available for research and instructional use only.
c  You may copy and use this software without charge for these non-commercial
c  purposes, provided that the copyright notice and associated text is
c  reproduced on all copies.  For all other uses (including distribution of
c  modified versions), please contact the author at the address given below.
c
c  *** This software is made available "as is" without any assurance that it
c  *** will work for your purposes.  The software may in fact have defects,
so
c  *** use the software at your own risk.
c
c  ------------------------------------
c    CLAWPACK Version 2.1,  October, 1995
c    available from netlib@research.att.com in pdes/claw
c    Homepage: http://www.amath.washington.edu/~rjl/clawpack.html
c  ------------------------------------
c    Author:  Randall J. LeVeque
c             Applied Mathematics
c             Box 352420
c             University of Washington,
c             Seattle, WA 98195-2420
c             rjl@amath.washington.edu
c =====================================================================
c
c
c
c
c    ================================================================
```

```
c     Beginning of claw1 code
c     ========================================================================
c
      implicit double precision (a-h,o-z)
      dimension q(1-mbc:maxmx+mbc, meqn)
      dimension work(mwork)
      dimension mthlim(mwaves),method(7),dtv(5),cflv(4),nv(2)
      dimension aux(1-mbc:maxmx+mbc,method(7))
c
c
      info = 0
      t = tstart
      maxn = nv(1)
      dt = dtv(1)   !# initial dt
      cflmax = 0.d0
      dtmin = dt
      dtmax = dt
      nv(2) = 0
c
c     # check for errors in data:
c
      if (mx .gt. maxmx) then
       info = 1
       go to 900
       endif
c
      if (method(1) .eq. 0) then
c        # fixed size time steps.  Compute the number of steps:
       maxn = (tend - tstart + 1d-10) / dt
       if (dabs(maxn*dt - (tend-tstart)) .gt. 1d-8) then
c            # dt doesn't divide time interval integer number of times
          info = 2
          go to 900
          endif
       endif
c
      if (method(1).eq.1 .and. cflv(1).gt.1.d0) then
       info = 3
       go to 900
       endif
c
c     # partition work array into pieces for passing into step1:
      i0f = 1
      i0wave = i0f + (maxmx + 2*mbc) * meqn
      i0s = i0wave + (maxmx + 2*mbc) * meqn * mwaves
      i0dtdx = i0s + (maxmx + 2*mbc) * mwaves
      i0qwork = i0dtdx + (maxmx + 2*mbc)
      i0amdq = i0qwork + (maxmx + 2*mbc) * meqn
      i0apdq = i0amdq + (maxmx + 2*mbc) * meqn
      i0dtdx = i0apdq + (maxmx + 2*mbc) * meqn
      i0end = i0dtdx + (maxmx + 2*mbc) - 1
c
      if (mwork .lt. i0end) then
       write(6,*) 'mwork must be increased to ',i0end
       info = 4
       go to 900
       endif
```

```
c
c     -----------
c     # main loop
c     -----------
c
      if (maxn.eq.0) go to 900
      do 100 n=1,maxn
       told = t
       if (told+dt .gt. tend) dt = tend - told
       if (method(1).eq.1) then
c           # save old q in case we need to retake step with smaller dt:
          call copyq1(maxmx,meqn,mbc,mx,q,work(i0qwork))
          endif
c
   40    continue
       dt2 = dt / 2.d0
       thalf = t + dt2  !# midpoint in time for Strang splitting
       t = told + dt    !# time at end of step
c
c       ------------------------------------------------------------------
c       # main steps in algorithm, using Strang splitting for source term:
c       ------------------------------------------------------------------
c
c
c       # determine the interface speed needed for mesh movement
        call inter1(maxmx,meqn,mbc,mx,q,q,aux,aux)
c
c       # extend data from grid to bordering boundary cells:
        call bc1(maxmx,meqn,mbc,mx,q,aux,told,dt)
c
c       # set the auxiliary array based on mesh movement
        call set1(maxmx,mbc,mx,aux,dx,dt,method)
c
       if (method(5).eq.1) then
c           # with source term:   use Strang splitting
           call src1(maxmx,meqn,mbc,mx,q,aux,told,dx,dt2,method)
           endif
c
c       # take a step on the homogeneous conservation law:
        call step1(maxmx,meqn,mwaves,mbc,mx,q,aux,dx,dt,
     &             method,mthlim,cfl,work(i0f),work(i0wave),
     &             work(i0s),work(i0amdq),work(i0apdq),work(i0dtdx),
     &             rp1)
       if (method(5).eq.1) then
c            # source terms over second half time step:
           call src1(maxmx,meqn,mbc,mx,q,aux,thalf,dx,dt,method)
           endif
c
c       ------------------------------------------------------------------
c
        if (method(4) .eq. 1) write(6,601) n,cfl,dt,t
  601   format('CLAW1... Step',i4,
     &                  '   Courant number =',f6.3,'  dt =',d12.4,
     &                  '  t =',d12.4)
c
        if (method(1) .eq. 1) then
c          # choose new time step if variable time step
```

```
            if (cfl .gt. 0.d0) then
            dt = dmin1(dtv(2), dt * cflv(2)/cfl)
            dtmin = dmin1(dt,dtmin)
            dtmax = dmax1(dt,dtmax)
            else
            dt = dtv(2)
            endif
          endif
c
c       # check to see if the Courant number was too large:
c
       if (cfl .le. cflv(1)) then
c               # accept this step
             cflmax = dmax1(cfl,cflmax)
            else
c               # reject this step
            t = told
              call copyq1(maxmx,meqn,mbc,mx,work(i0qwork),q)
c
            if (method(4) .eq. 1) then
               write(6,602)
  602          format('CLAW1 rejecting step... ',
     &                        'Courant number too large')
               endif
              if (method(1).eq.1) then
c                  # if variable dt, go back and take a smaller step
                go to 40
                else
c                   # if fixed dt, give up and return
                  cflmax = dmax1(cfl,cflmax)
                  go to 900
                endif
             endif
c
c       # see if we are done:
       nv(2) = nv(2) + 1
       if (t .ge. tend) go to 900
c
  100    continue
c
  900  continue
c
c      # return information
c
       if (method(1).eq.1 .and. t.lt.tend .and. nv(2) .eq. maxn) then
c         # too many timesteps
        info = 11
        endif
c
       if (method(1).eq.0 .and. cflmax .gt. cflv(1)) then
c         # Courant number too large with fixed dt
        info = 12
        endif
       tend = t
       cflv(3) = cflmax
       cflv(4) = cfl
       dtv(3) = dtmin
```

```
        dtv(4) = dtmax
        dtv(5) = dt
        return
        end



      program driver
c
c  Sample driver routine for claw1
c  This program solves the 1D Euler equations on a moving mesh
c  for a gamma-law gas with two gases separated by an interface
c
c  The motion of the interface is determined in each step by the
c  routine inter1 which solves the Riemann problem at this interface.
c  This routine is called in each time step by the modified version of claw1.
c
c  The mesh movement tracks both this interface and the piston bounding
c  the tube on the right.
c
c  The initial conditions are set in the routine ic
c
c  The Riemann problem is solved in rp1eumm
c
c  The boundary conditions are set in bc1pist
c
c  The solution is output by the routine out1eu
c  This routine is called nout times at equal time increments.
c
c  The aux array contains:
c     aux(i,1) = length of i'th cell in physical space divided by dx
c     aux(i,2) = velocity of i'th cell interface in next time step
c     aux(i,3) = cell center of i'th cell in physical space
c     aux(i,4) = gamma in i'th cell
c  These are updated in each time step in the routine set1aux, called from
c  the modified version of claw1.
c
c
c  Authors: Riccardo Fazio and Randall J. LeVeque
c  Version of March, 1998 --  CLAWPACK Version 3.0
c
c
c
      implicit double precision (a-h,o-z)
      external bc1pres,rp1,src1,set1aux,ic,inter1
      parameter (maxmx = 5000)
      parameter (meqn = 3)
      parameter (maux = 4)
      parameter (mcapa = 1)
      parameter (mwaves = 3)
      parameter (mbc = 2)
      parameter (mwork = 131040)
c
      dimension q(1-mbc:maxmx+mbc, meqn)
      dimension aux(1-mbc:maxmx+mbc, maux)
      dimension work(mwork)
      dimension x(1-mbc:maxmx+mbc)
```

```
      dimension mthlim(mwaves),method(7),dtv(5),cflv(4),nv(2)
      real*8 L
      common /param/  GAMMA1,GAMMA2
      common /arpa/ L,dLdt,S,dSdt
      common /pcond/ am,pw,peps,pomega
      common /comrad/ ndim
c
C       GAMMA1 = 1.4d0
C       GAMMA2 = 2.8d0

C           Gamma for monatomic gas (plasma) on both sides of boundary**
Lucas Beveridge, Feb. 2019
            GAMMA1 = 1.66d0
            GAMMA2 = 1.66d0
      ndim = 3
c
      open(10,file='fort.info',status='unknown',form='formatted')
      open(11,file='fort.nplot',status='unknown',form='formatted')
      open(20,file='fort.x',status='unknown',form='formatted')
c
C       write(6,*) 'input dx'
C       read(5,*) dx
C       write(6,*) 'input dt, tend, nout'
C       read(5,*) dt,tend,nout
C       write(6,*) 'input method(1), method(2)'
C       read(5,*) method(1), method(2)
      write(6,*) 'input mthlim(1:3)'
      read(5,*) (mthlim(mw), mw=1,mwaves)
C       write(6,*) 'input am, pw, peps, pomega'
C       read(5,*) am, pw, peps, pomega
      dx = 0.001
        dt = 1D-12
        tend = 0.00000995
        nout = 1000
        method(1) = 1
        method(2) = 2
C       mthlim(mw)=1
C       mw=1
C       mwaves=1
        am = 0.000000314
        pw = 100000
        peps = 1
        pomega = 10
c
      t0 = 0.d0
      cflv(1) = 1.0d0  !# largest cfl to allow before rejecting step
      cflv(2) = 0.9d0  !# desired cfl
      dtv(1) = dt    !# dt to try in first step
      dtv(2) = 1.d6  !# no explicit limit on dt
      nv(1) = 500000    !# maximum number of time steps allowed
      method(4) = 0  !# don't print dt and cfl every time step
      method(5) = 0   !# with source terms
      method(6) = mcapa   !# with a capa array
      method(7) = maux   !# and aux array
c
c     # set grid and initial conditions
c
```

```
c     # domain is  0 <= x <= 1:
      ax = 0.d0
      bx = 1.d0
      mx= (bx-ax + 1d-12)/dx
c
      do 10 i=1,mx
       x(i) = ax + (i-0.5d0) * dx
       write(20,1020) x(i)
 1020    format(e16.6)
   10    continue
c
      do 50 i=1-mbc,mx+mbc
         aux(i,1) = 1.0d0
         aux(i,2) = 0.0d0
         aux(i,3) = 0.0d0
         IF (I.LT.MX/2+1) THEN
            AUX(I,4) = GAMMA1
            ELSE IF (I.GE.MX/2+1) THEN
c            ELSE
            AUX(I,4) = GAMMA2
         ENDIF
   50 continue
c
      call ic(maxmx,meqn,mbc,mx,x,dx,q,aux)
c
c     # output initial data to unit 100
      call out1eu(maxmx,meqn,mbc,mx,x,q,0.d0,100)
c
      dtout = (tend - t0)/dfloat(nout)
      do 100 n=1,nout
       tstart = t0 + (n-1) * dtout
       tend = t0 + n*dtout
c
c        # main call to claw1.......................
c
         call claw1(maxmx,meqn,mwaves,mbc,mx,
     &      q,aux,dx,tstart,tend,dtv,cflv,nv,method,mthlim,
     &           work,mwork,info,bc1pres,rp1,src1,set1aux,inter1)
c
         dtv(1) = dtv(5)  !# set dt to start next call
c
         call out1eu(maxmx,meqn,mbc,mx,x,q,tend,100+n)
c
       write(10,1010) tend,info,dtv(3),dtv(4),dtv(5),
     &             cflv(2),cflv(3),nv(2)
 1010    format('tend =',d15.4,/,
     &       'info =',i5,/,'smallest dt =',d15.4,/,'largest dt =',
     &       d15.4,/,'last dt =',d15.4,/,'largest cfl =',
     &         d15.4,/,'last cfl =',d15.4,/,'steps taken =',i4,/)
c
       write(6,6001) nv(2),tend,info
 6001    format('CLAW1 returning after ',i3,' steps,  t =',e10.5,
     &             '  info =',i3,/)
c
       if (info .ne. 0) then
          write(6,*) 'ERROR in claw1... info =', info
          go to 200
```

```
          endif
  100    continue
c
  200 continue
      write(11,1101) nout
c
 1101 format(i5)
      stop
      end
```

STEP1.F

```
c
c
c =====================================================================
      subroutine step1(maxmx,meqn,mwaves,mbc,mx,q,aux,dx,dt,
     &              method,mthlim,cfl,f,wave,s,amdq,apdq,dtdx,rp1)
c =====================================================================
c
c     # Take one time step, updating q.
c
c     method(1) = 1   ==>  Godunov method
c     method(1) = 2   ==>  Slope limiter method
c     mthlim(p)  controls what limiter is used in the pth family
c
c
c     amdq, apdq, wave, s, and f are used locally:
c
c     amdq(1-mbc:maxmx+mbc, meqn) = left-going flux-differences
c     apdq(1-mbc:maxmx+mbc, meqn) = right-going flux-differences
c        e.g. amdq(i,m) = m'th component of A^- \Delta q from i'th Riemann
c                         problem (between cells i-1 and i).
c
c     wave(1-mbc:maxmx+mbc, meqn, mwaves) = waves from solution of
c                                           Riemann problems,
c           wave(i,m,mw) = mth component of jump in q across
c                          wave in family mw in Riemann problem between
c                          states i-1 and i.
c
c     s(1-mbc:maxmx+mbc, mwaves) = wave speeds,
c           s(i,mw) = speed of wave in family mw in Riemann problem between
c                     states i-1 and i.
c
c     f(1-mbc:maxmx+mbc, meqn) = correction fluxes for second order method
c           f(i,m) = mth component of flux at left edge of ith cell
c     ------------------------------------------------------------------
c
      implicit double precision (a-h,o-z)
      dimension    q(1-mbc:maxmx+mbc, meqn)
      dimension  aux(1-mbc:maxmx+mbc, *)
      dimension    f(1-mbc:maxmx+mbc, meqn)
      dimension    s(1-mbc:maxmx+mbc, mwaves)
      dimension wave(1-mbc:maxmx+mbc, meqn, mwaves)
      dimension amdq(1-mbc:maxmx+mbc, meqn)
      dimension apdq(1-mbc:maxmx+mbc, meqn)
```

```
      dimension dtdx(1-mbc:maxmx+mbc)
      dimension method(7),mthlim(mwaves)
      logical limit
c     ==========================
c       # CONSTANTS FOR FUSION SOURCE TERMS: (Lucas B)
c       pi, radius, boltzmann const, molec. wgt of fuel,
c     avogadros number, alpha heat Q (J), electron mass (kg),
c     number of protons, electron charge [C], keV/J conversion
      pii = 3.14159d0
        Rds = 0.005d0
        kB = 1.38E+23
        Mwgt = 0.0025
        Na = 6.022E23
        Qch = 0.5607E-12
        me = 0.910E-30
      z = 1
        ech = 0.1602E-18
        kevperJ = 6.24E15

      dt2 = dt/2.d0
      press = 101000.d0
c     ==========================
c     # check if any limiters are used:
      limit = .false.
      do 5 mw=1,mwaves
       if (mthlim(mw) .gt. 0) limit = .true.
   5     continue
c
      mcapa = method(6)
      do 10 i=1-mbc,mx+mbc
       if (mcapa.gt.0) then
           if (aux(i,mcapa) .le. 0.d0) then
            write(6,*) 'Error -- capa must be positive'
            stop
            endif
             dtdx(i) = dt / (dx*aux(i,mcapa))
          else
             dtdx(i) = dt/dx
          endif
   10  continue
c
c
c
c     # solve Riemann problem at each interface
c     ----------------------------------------
c
      call rp1(maxmx,meqn,mwaves,mbc,mx,q,q,aux,aux,wave,s,amdq,apdq)
c
c     # Modify q for Godunov update:
c     # Note this may not correspond to a conservative flux-differencing
c     # for equations not in conservation form.  It is conservative if
c     # amdq + apdq = f(q(i)) - f(q(i-1)).

c ========================== BEGIN LUCAS B MODIFICATIONS
====================
c     # update q by differencing correction fluxes
```

```
c     Lucas Beveridge March 2019: update fluxes with source terms, talk about
in dissertation
c     ==========================================
c
      do 40 i=1,mx+1
         do 40 m=1,meqn
             if (m .eq. 3) then
c    ========================
c     # computing expressions for fusion sources and losses
c             # fluid density at cell i
            rho = q(i,1)
c             # fluid speed at cell i
            u = q(i,2)/q(i,1)
c             # pressure at cell i
            press = GAMMAM1*(q(i,3) - 0.5d0*rho*u**2)

c           contact surface area of mesh cell
            Ac = 2.d0*pii*dx*Rds+2.d0*pii*Rds**2
c     Temperature [K]
            Tk = (press*Mwgt)/(rho*Na*kB)
c     Temperature [keV]
            Tkev = Tk/(11624525)
c           D+T cross section <sigma*v>
            sv = (0.368E-11*Tkev**(-2/3)*exp(-19.94*Tkev**(-1/3)))/(100**3)
c     number density of fuel atoms [1/m**3]
            ndens = (rho*Na)/Mgwt
c     volume of mesh cell [m**3]
            vol = pii*(Rds**2.d0)*dx
c    ========================
c     energy source and loss terms
c     Heat conduction losses
            Ec = -(0.1*(3**(3/2)/2)*ndens*kB*Tk*((kB*Tk/me)**0.5)*Ac*dt/vol)
c     alpha heating rate from fusion reactions
            Ealpha = 0.25d0*(ndens**2)*sv*Qch
c     Bremmstrahlung loss
            Ebremms = -dt*0.15E-
39*(z**2)*(ndens**2)*((Tkev/kevperJ)/ech)**0.5
c     Bolztmann distribution heating
            Eboltz = -3.d0*press
            energ_tot = Ealpha-Ebremms-Ec-Eboltz
c    ============================
c              # add source terms to energy equation

                  q(i,m) = q(i,m) - dtdx(i)*apdq(i,m)
                  q(i-1,m) = q(i-1,m) - dtdx(i-1)*amdq(i,m)
                  q(i,m) = q(i,m) + (Ealpha-Ebremms-Eboltz)*0
            else
                  q(i,m) = q(i,m) - dtdx(i)*apdq(i,m)
                  q(i-1,m) = q(i-1,m) - dtdx(i-1)*amdq(i,m)
            endif
c ============================ END LUCAS B MODIFICATIONS
====================
   40       continue

c
c     # compute maximum wave speed:
      cfl = 0.d0
```

```
      do 50 mw=1,mwaves
         do 45 i=1,mx+1
c          # if s>0 use dtdx(i) to compute CFL,
c          # if s<0 use dtdx(i-1) to compute CFL:
           cfl = dmax1(cfl, dtdx(i)*s(i,mw), -dtdx(i-1)*s(i,mw))
   45      continue
   50    continue
c
      if (method(2) .eq. 1) go to 900
c
c     # compute correction fluxes for second order q_{xx} terms:
c     -------------------------------------------------------
c
      do 100 m = 1, meqn
            do 100 i = 1-mbc, mx+mbc
               f(i,m) = 0.d0
  100          continue
c
c      # apply limiter to waves:
      if (limit) call limiter(maxmx,meqn,mwaves,mbc,mx,wave,s,mthlim)
c
      do 120 i=1,mx+1
       do 120 m=1,meqn
          do 110 mw=1,mwaves
             dtdxave = 0.5d0 * (dtdx(i-1) + dtdx(i))
             f(i,m) = f(i,m) + 0.5d0 * dabs(s(i,mw))
     &               * (1.d0 - dabs(s(i,mw))*dtdxave) * wave(i,m,mw)
c
c              # third order corrections:
c              # (still experimental... works well for smooth solutions
c              # with no limiters but not well with limiters so far.
c

               if (method(2).lt.3) go to 110
               if (s(i,mw) .gt. 0.d0) then
                   dq2 = wave(i,m,mw) - wave(i-1,m,mw)
                 else
                   dq2 = wave(i+1,m,mw) - wave(i,m,mw)
                 endif
               f(i,m) = f(i,m) - s(i,mw)/6.d0 *
     &                     (1.d0 - (s(i,mw)*dtdxave)**2) * dq2

  110          continue
  120       continue
c
c
  140 continue

c
      do 150 m=1,meqn
       do 150 i=1,mx
            q(i,m) = q(i,m) - dtdx(i) * (f(i+1,m) - f(i,m))


  150       continue
c
  900 continue
```

```
      return
      end
```

SRC1RAD.F,

This is the source-term subroutine. Due to a bug in the code, it was not ultimately used, but it was modified for the project, and so is included here for completeness.

```
c   dx added to arguments, and fusion source terms, Lucas B, 2/23/2019.
c
c     source terms for Bremmstrahlung, free-stream heat conduction,
c     internal heat, and alpha heating. all units in MKS Lucas B, 2/23/2019
c
c =========================================================
      subroutine src1(maxmx,meqn,mbc,mx,q,aux,t,dx,dt,method)
c =========================================================
      implicit real*8(a-h,o-z)
      dimension q(1-mbc:maxmx+mbc, meqn)
c
      parameter (maxmode = 2002)
      dimension qstar(-1:maxmode, 3)
      dimension x(-1:maxmode)
      dimension method(7)
      dimension aux(1-mbc:maxmx+mbc,method(7))
      common /comrad/ ndim
      common /param/  GAMMA1,GAMMA2
c
c     # source terms for radial symmetry
c     # 2-step Runge-Kutta method
c
c       # CONSTANTS FOR FUSION SOURCE TERMS: (Lucas B)
c       pi, radius, boltzmann const, molec. wgt of fuel,
c     avogadros number, alpha heat Q (J), electron mass (kg),
c     number of protons, electron charge [C], keV/J conversion
      pii = 3.14159d0
        Rds = 0.0025d0
        kB = 0.13806E+24
        Mwgt = 0.025d0
        Na = 0.6022E+24
        Qch = 0.5607E-12
        me = 0.9101E-30
      z = 1.d0
        ech = 0.1602E-18
        kevperJ = 0.6242E+16

      dt2 = dt/2.d0
      press = 0.d0
      do 10 i=1-mbc,mx+mbc
         IF (I.LT.MX/2+1) THEN
            GAMMA = GAMMA1
            ELSE IF (I.GE.MX/2+1) THEN
                    GAMMA = GAMMA2
```

```
                    ELSE
         ENDIF
         GAMMAM1 = GAMMA-1.0d0
         x(i) = aux(i,3)
        rho = q(i,1)
        u = q(i,2)/q(i,1)
        press = GAMMAM1*(q(i,3) - 0.5d0*rho*u**2)
c     =========================
c    # computing expressions for fusion sources and losses
c       contact surface area of mesh cell
         Ac = 2.d0*pii*dx*Rds+2.d0*pii*Rds**2
c     Temperature [K]
        Tk = (press*Mwgt)/(rho*Na*kB)
c     Temperature [keV]
        Tkev = Tk/(11624525.d0)
c       D+T cross section <sigma*v>
         sv = (0.368E-11*Tkev**(-2.d0/3.d0)*exp(-19.94d0*Tkev**(-1.d0/3.d0)))
         &                                                    /(100.d0**3)
c     number density of fuel atoms [1/m**3]
        ndens = (rho*Na)/Mgwt
c     volume of mesh cell [m**3]
        vol = pii*(Rds**2.d0)*dx
c     =========================
c     energy source and loss terms
c     Heat conduction losses
        Ec = -
(0.1d0*(3.d0**(3.d0/2.d0)/2.d0)*ndens*kB*Tk*((kB*Tk/me)**0.5d0)*
         &
                  Ac*dt/vol)
c     alpha heating rate from fusion reactions
        Ealpha = 0.25d0*(ndens**2)*sv*Qch
c     Bremmstrahlung loss
       Ebremms = -dt*0.15E-39*(z**2)*(ndens**2)*((Tkev/kevperJ)/ech)**0.5
c     Bolztmann distribution heating
        Eboltz = -3.d0*press

        qstar(i,1) = q(i,1) - (dt2*(ndim-1)/dabs(x(i)) * q(i,2))
        qstar(i,2) = q(i,2) - (dt2*(ndim-1)/dabs(x(i)) *
      &                        (rho*u**2))
        qstar(i,3) = q(i,3) - (dt2*(ndim-1)/dabs(x(i)) *
      &               u*(q(i,3) + press)) + (Ec + Ealpha + Ebremms +
         &
       Eboltz)*0
   10    continue
c
       do 20 i=1-mbc,mx+mbc
          IF (I.LT.MX/2+1) THEN
             GAMMA = GAMMA1
             ELSE IF (I.GE.MX/2+1) THEN
                     GAMMA = GAMMA2
                     ELSE
         ENDIF
         GAMMAM1 = GAMMA-1.0d0
        rho = qstar(i,1)
        u = qstar(i,2)/qstar(i,1)
        press = GAMMAM1*(qstar(i,3) - 0.5d0*rho*u**2)
        q(i,1) = q(i,1) - (dt*(ndim-1)/dabs(x(i)) * qstar(i,2))
```

```
        q(i,2) = q(i,2) - (dt*(ndim-1)/dabs(x(i)) *
     &                        (rho*u**2))
        q(i,3) = q(i,3) - (dt*(ndim-1)/dabs(x(i)) *
     &                 u*(qstar(i,3) + press)) + (Ec + Ealpha + Ebremms +
        &
     Eboltz)*0
  20    continue
     return
     end
```

Appendix D, Matlab script for computing fusion gain

```
% Code for computing total fusion gain, and losses
% Lucas Beveridge, March 2019

% CONSTANTS FOR FUSION SOURCE TERMS: (Lucas B)
% pi, radius, boltzmann const, molec. wgt of fuel,
% avogadros number, alpha heat Q (J), electron mass (kg),
% number of protons, electron charge [C], keV/J conversion
clear;clc
Rds = 0.0025;
kB = 1.3806E-23;
Mwgt = 0.025;
Na = 0.6022E+24;
Qch = 0.5607E-12;
Qtot = 3.58E-12;
me = 0.9101E-30;
z = 1;
ech = 0.1602E-18;
kevperJ = 0.6242E+16;

t2=0;
t_arr(1)=0;

nplot = readv('fort.nplot');

for n = 0:nplot
    try
        x = readv('fort.x');
        n1 = 100+n;
        [t,mx,meqn,piston,interface,data] = readq1(n1);
        x2=x;
        x = x*piston;
        rho = data(:,1);
        u = data(:,2);
        p = data(:,3);

        press_scalar = mean(p);
        press(n+1) = press_scalar;

        rho_scalar = mean(rho);
        if rho_scalar < 1e6
            rho_4plot(n+1) = rho_scalar;
        else
            rho_4plot(n+1) = rho_4plot(n);
        end

        %  temp = (Mwgt*p./(rho*Na*kB))/11624525;
        temp = 0.125*(p./rho)/11624525;

        time(n+1)=t;
        dt=t-t2;
        dx=abs(x-x2);

        %%%%%% COMPUTE SOURCES %%%%%%
        %    contact surface area of mesh cell
```

```
        Ac = 2*pi*dx*Rds+2*pi*Rds^2;
        %    Temperature [K]
        num=(Mwgt.*p);
        den=(Na*kB.*rho);
        Tk = 0.125*p./rho;
        %    Temperature [keV]
        Tkev = Tk/(11624525);
        %    D+T cross section <sigma*v>
        sv = ((3.668E-12).*Tkev.^(-2/3).*exp(-19.94.*Tkev.^(-1/3)))/(100^3);
        %    number density of fuel atoms [1/m**3]
        ndens = (rho.*Na)/Mwgt;
        %    volume of mesh cell [m**3]
        vol = pi*(Rds^2)*dx;
        %    ========================
        %    energy source and loss terms
        %    Heat conduction losses
        Ec = -(0.1*(3^(3/2)/2)*kB*Tk.*((kB*Tk/me).^0.5).*Ac).*ndens; %
Theoretical absolute maximum electron conduction loss (free-streaming limit)
        %    alpha heating rate from fusion reactions
        Ealpha = 0.25.*sv.*Qtot.*vol.*(ndens.^2);
        %    Bremmstrahlung loss
        Ebremms = -(0.15E-
39*(z^2).*((Tkev./kevperJ)/ech).^0.5).*vol.*(ndens.^2);
        %    Bolztmann distribution heating
        Eboltz = -3*vol.*p;

        E_tot(:,n+1) = Ealpha+Ec+Ebremms;
        E_tot_some_cond(:,n+1) = Ealpha+0.025*Ec+Ebremms;
        E_tot_more_cond(:,n+1) = Ealpha+0.1*Ec+Ebremms;
        E_tot_no_cond(:,n+1) = Ealpha+Ebremms;
        disp(n)
        t2=t;
        t_arr(n+1)=t;
    catch
        sprintf('\n %d is bad time step at t=%d ',n,t)
    end
end

for i=1:1001
    Evst(i)=sum(E_tot(:,i)); % sum up all energies across mesh at each time
step
    Evst_nocond(i)=sum(E_tot_no_cond(:,i)); % sum up all energies across mesh
at each time step without conduction loss
    Evst_somecond(i)=sum(E_tot_some_cond(:,i)); % sum up all energies across
mesh at each time step with some conduction loss
    Evst_morecond(i)=sum(E_tot_more_cond(:,i)); % sum up all energies across
mesh at each time step with more conduction loss
end
plot(t_arr,Evst,t_arr,Evst_morecond,t_arr,Evst_somecond,t_arr,Evst_nocond)
title("Net power vs time, with amplification");
xlabel("Time (s.)");
ylabel("Net power (Watt)");
legend("Net power (worst case)","Net power (10% of worst case)", "Net power
(2.5% of worst case)","Net power (best case, no conduction
loss)",'location','north');
```

```
E_grand_tot=sum(sum(E_tot*dt)) % Integrate powers from components to get net
energy released
E_grand_tot_some_cond=sum(sum(E_tot_some_cond*dt)) % Integrate powers from
components to get net energy released best case
E_grand_tot_more_cond=sum(sum(E_tot_more_cond*dt)) % Integrate powers from
components to get net energy released best case
E_grand_tot_no_cond=sum(sum(E_tot_no_cond*dt)) % Integrate powers from
components to get net energy released best case

function vector = readv(fname)
 %
 % read a vector of data from file called fname
 %
fid = fopen(fname);
vector = fscanf(fid,'%g',inf);
status = fclose(fid);

function [t,mx,meqn,piston,interface,data] = readq1(n)
%
% read a 1d data file fort.n
%
 fname = ['fort.',num2str(n)];
 fid = fopen(fname);
 [info,count] = fscanf(fid,'%g %d %d %g %g',5);
 t = info(1);
 mx = info(2);
 meqn = info(3);
 piston = info(4);
 interface = info(5);
 data = fscanf(fid,'%g',[meqn,inf]);
 data = data';
 status = fclose(fid);
```

Appendix E, Matlab script for generating plots from 1-D code

This is a modified function provided with CLAWPACK for reading and plotting data from the output files that code produces. This was modified to show the temperature in keV.

```
function plot1eu(rholim,ulim,plim,tlim,xlim)
%
% plot solution to the Euler equations in 1d
% rholim, etc. are 2-vectors with desired limits of each variable,
%   density, velocity, pressure, temperature.
% calling plot1eu with no arguments causes plot routine to choose them
% automatically in each plot.
%

Rds = 0.0025;
kB = 1.3806E-23;
Mwgt = 0.025;
Na = 0.6022E+24;
Qch = 0.5607E-12;
Qtot = 3.58E-12;
me = 0.9101E-30;
z = 1;
ech = 0.1602E-18;
kevperJ = 0.6242E+16;

nplot = readv('fort.nplot');

for n = 0:nplot
    try
        clf
        x = readv('fort.x');
        if nargin<5
            xlim = [min(x),max(x)];
        end
        n1 = 100+n;
        [t,mx,meqn,piston,interface,data] = readq1(n1);

        rho = data(:,1);
        subplot(2,2,1)
        x = x*piston;
        plot(x,rho)
        if nargin>0
            axis([xlim,rholim(1),rholim(2)])
        end
        title(['Density [kg/m^3] at time t = ', num2str(t)])

        %u = data(:,2);

        Mwgt = 0.0025;
        Na = 6.022e23;
        kB = 1.38e-23;

        %  temp = (Mwgt*p./(rho*Na*kB))/11624525;
```

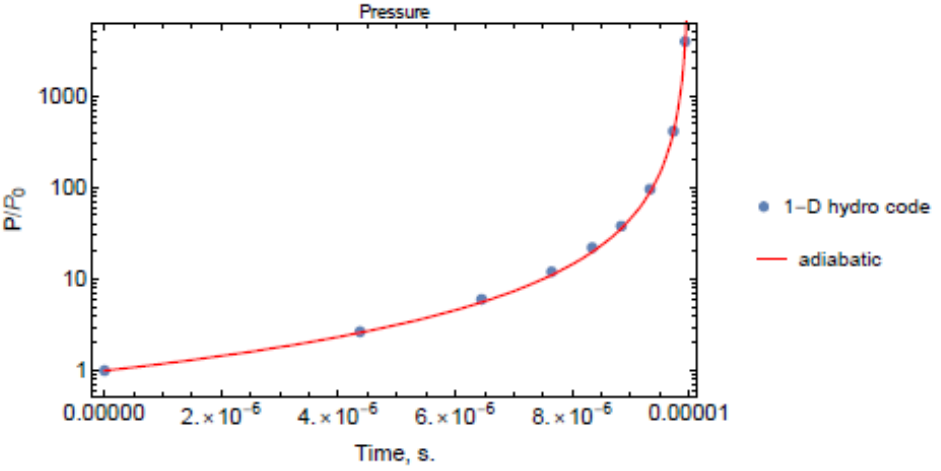
```
        p = data(:,3);
        temp = 0.125*(p./rho)/11624525*(1-1/260);
        sv = ((3.668E-12).*temp.^(-2/3).*exp(-19.94.*temp.^(-1/3)))/(100^3);
        ndens = (rho.*Na)/Mwgt;
        vol = pi*(Rds^2)*dx;

        Ebremms = -(0.15E-
39*(1^2).*((temp./kevperJ)/ech).^0.5).*vol.*(ndens.^2);
        Ec = -(0.1*(3^(3/2)/2)*kB*Tk.*((kB*Tk/me).^0.5).*Ac).*ndens; %
Theoretical absolute maximum electron conduction loss (free-streaming limit)
        Rea = 0.25.*sv.*Qtot.*vol.*(ndens.^2);
        tot=Rea+Ebremms+Ec;

        subplot(2,2,2)
        plot(x,Rea)
        if nargin>1
            axis([xlim,ulim(1),ulim(2)])
        end
        title(['Power [Watt] at time t = ', num2str(t)])

        subplot(2,2,3)
        plot(x,p)
        if nargin>2
            axis([xlim,plim(1),plim(2)])
        end
        title(['Pressure [Pa] at time t = ', num2str(t)])

        subplot(2,2,4)

        plot(x,temp)
        if nargin>3
            axis([xlim,tlim(1),tlim(2)])
        end
        title(['Temperature [keV] at time t = ', num2str(t)])

        if n<nplot
            %query
            temp_frm(n+1) = getframe;
            t=input('press enter')
        end
    catch
        sprintf('\n %d is bad time step at t=%d ',n,t)
    end
end

function vector = readv(fname)
 %
 % read a vector of data from file called fname
 %
fid = fopen(fname);
vector = fscanf(fid,'%g',inf);
status = fclose(fid);
```

This is a Mathematica script used to compare a simple adiabatic model of temperature and pressure to the 1-D hydro code.

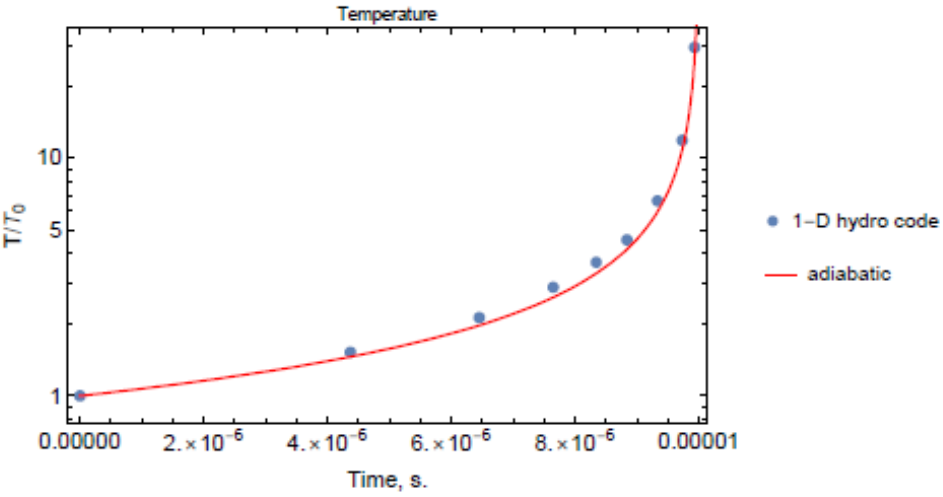
```
In[328]:= γ = 1.66;
          p0 = 7 × 10^9;
          T0 = 1.5;
          V0 = (0.0025^2 π) * 0.1;
          V1 = (0.0025^2 π) * 7 × 10^-4;
          p1 = p0 V0^γ / V1^γ
          T1 = T0 V0^(γ-1) / V1^(γ-1)
          v = 10 000; (** impact velocity, m/s **)
          times = {0, 4.369 × 10^-6, 6.45 × 10^-6, 7.646 × 10^-6,
             8.341 × 10^-6, 8.838 × 10^-6, 9.33 × 10^-6, 9.73 × 10^-6, 9.93 × 10^-6};
          simpress = {{0, 7 × 10^9 / p0}, {4.369 × 10^-6, 1.865 × 10^10 / p0}, {6.45 × 10^-6, 4.2 × 10^10 / p0},
             {7.646 × 10^-6, 8.4 × 10^10 / p0}, {8.341 × 10^-6, 1.54 × 10^11 / p0},
             {8.838 × 10^-6, 2.65 × 10^11 / p0}, {9.33 × 10^-6, 6.695 × 10^11 / p0},
             {9.73 × 10^-6, 2.87 × 10^12 / p0}, {9.93 × 10^-6, 2.735 × 10^13 / p0}};
          (**simtemp={1.5,2.29,3.2,4.3,5.475,6.8,9.935,17.858,43.92};**)
          simtemp = {{0, 1.5 / T0}, {4.369 × 10^-6, 2.29 / T0}, {6.45 × 10^-6, 3.2 / T0},
             {7.646 × 10^-6, 4.3 / T0}, {8.341 × 10^-6, 5.475 / T0}, {8.838 × 10^-6, 6.8 / T0},
             {9.33 × 10^-6, 9.935 / T0}, {9.73 × 10^-6, 17.858 / T0}, {9.93 × 10^-6, 43.92 / T0}};

          simplotpress =
            ListLogPlot[simpress, PlotLegends → {"1-D hydro code"}, PlotStyle → PointSize[0.0175]];
          simplottemp = ListLogPlot[simtemp, PlotLegends → {"1-D hydro code"},
             PlotStyle → PointSize[0.0175]];


          pressure = LogPlot[{V0^γ / ((0.0025^2 π) * (0.1 - v * t))^γ}, {t, 0, 100 × 10^-6},
             PlotRange → All, PlotStyle → Red, PlotLegends → {"adiabatic"}];
          Show[simplotpress, pressure, Frame → True, FrameStyle → 14,
           FrameLabel → {"Time, s.", "P/P0"}, PlotLabel → "Pressure", ImageSize → Large]


          temperature = LogPlot[{V0^(γ-1) / ((0.0025^2 π) * (0.1 - v * t))^(γ-1)}, {t, 0, 10 × 10^-6},
             PlotRange → All, PlotStyle → Red, PlotLegends → {"adiabatic"}];
          Show[simplottemp, temperature, Frame → True, FrameStyle → 14,
           FrameLabel → {"Time, s.", "T/T0"}, PlotLabel → "Temperature", ImageSize → Large]
Out[333]= 2.64384 × 10^13
```

Out[334]= 39.6576

Out[342]=



Out[344]=



Appendix G, MCNP6 input file for the Graphite shell “stepped ignition”

```
spark plug model staging, GRAPHITE
10      100     -0.2        -77 4 -5     imp:n=1
50   300  -0.1 -1 -88 90 imp:n=1
40   300  -0.1     -9 8 -7 #10 #50   imp:n=1
20   400  -11.34    -2 3 -6 #10 #40 #50 imp:n=1
30      0                  #10 #20 #40 #50   imp:n=0

7    pz   4
8    pz   0.3
9    cz   0.6
c -------------------
1       cz     0.4
2    cz   1
3    pz   0
4    pz   3.5
5    pz   4
6    pz   5.2
77   cz   0.1
c --------------------
88   pz   4.9
90   pz   4.3

sdef x=d1 y=d2 z=d3 erg=d4 cell 50
 si1 -0.6 0.6
 sp1 0 1
 si2 -0.6 0.6
 sp2 0 1
 si3 4 6.6
 sp3 0 1
 si4 h 1 20
```

```
 sp4 -4 -0.02 -1
nps 10000000
c
c Mesh Tallies, n-capture rate in cells
fmesh14:n geom=xyz origin=-5 -5 0
     imesh=5 iints=300
     jmesh=5 jints=300
     kmesh=10 kints=300
     out=ij enorm=no
fm14 -1.0 0 -2
c tally -2 is absorption per source neutron
print 110
c f2:n (4 5 1)
+f6 10 20
c 6LiD sparkplug inside lead cavity filled with DT
m100    3006.80c 1 1002.80c 1
m200 82000.42c 1
m300 1003.80c 1 1002.80c 1
m400 6000.80c 1
mt400 grph.29t spark plug model staging, GRAPHITE
```

Appendix H, MCNP6 input file for the LiD shell for “stepped ignition”

```
spark plug model staging, LiD
10      100     -0.2        -77 4 -5     imp:n=1
50   300  -0.1 -1 -88 90 imp:n=1
40   300  -0.1     -9 8 -7 #10 #50   imp:n=1
20   100  -0.2 -2 3 -6 #10 #40 #50 imp:n=1
30      0                  #10 #20 #40 #50   imp:n=0

7    pz   4
8    pz   0.3
9    cz   0.6
c -------------------
1       cz     0.4
2    cz   1
3    pz   0
4    pz   3.5
5    pz   4
6    pz   5.2
77   cz   0.1
c --------------------
88   pz   4.9
90   pz   4.3

sdef x=d1 y=d2 z=d3 erg=d4 cell 50
 si1 -0.6 0.6
 sp1 0 1
 si2 -0.6 0.6
 sp2 0 1
 si3 4 6.6
 sp3 0 1
 si4 h 1 20
```

```
 sp4 -4 -0.02 -1
nps 10000000
c
c Mesh Tallies, n-capture rate in cells
fmesh14:n geom=xyz origin=-5 -5 0
     imesh=5 iints=300
     jmesh=5 jints=300
     kmesh=10 kints=300
     out=ij enorm=no
fm14 -1.0 0 -2
c tally -2 is absorption per source neutron
print 110
c f2:n (4 5 1)
+f6 10 20
c 6LiD sparkplug inside lead cavity filled with DT
m100    3006.80c 1 1002.80c 1
m200 82000.42c 1
m300 1003.80c 1 1002.80c 1
m400 6000.80c 1
mt400 grph.29t
```

Appendix I, MCNP6 input file for the LiD pellet for Coaxial Geometry “stepped ignition”

```
spark plug model, internal LiD, coax staging
10      100     -0.8        -77 4 -5     imp:n=1
40   300  -0.1     -9 8 -7 #10       imp:n=1
20   200  -11.34    -2 3 -6 #10 #40  imp:n=1
30      0                  #10 #20 #40    imp:n=0

7    pz   6.6
8    pz   4.9
9    cz   0.6
c ------------------
1       cz     0.5
2    cz   1
3    pz   4.6
4    pz   5.5
5    pz   6
6    pz   6.9
77   cz   0.1

sdef x=d1 y=d2 z=d3 erg=d4 cell 40
 si1 -0.6 0.6
 sp1 0 1
 si2 -0.6 0.6
 sp2 0 1
 si3 4.9 6.6
 sp3 0 1
 si4 h 1 20
 sp4 -4 -0.02 -1
nps 1000000
c
c Mesh Tallies, n-capture rate in cells
```

```
fmesh14:n geom=xyz origin=-5 -5 0
     imesh=5 iints=300
     jmesh=5 jints=300
     kmesh=10 kints=300
     out=ij enorm=no
fm14 -1.0 0 -2
c tally -2 is absorption per source neutron
print 110
c f2:n (4 5 1)
+f6 10 20
c 6LiD sparkplug inside lead cavity filled with DT
m100    3006.80c 1 1002.80c 1
m200 82000.42c 1
m300 1003.80c 1 1002.80c 1
```

Appendix J, MCNP6 input file for the LiD Pellet with external neutron beam

```
spark plug model
10      100     -0.8        -77 4 -5     imp:n=1
40   0         -9 8 -7 #10      imp:n=1
20   200  -2.7 -2 3 -6 #10 #40  imp:n=1
50   0         -80 81 -82 #10 #20 #40   imp:n=1
30      0                  #10 #20 #40 #50   imp:n=0

7    pz   6.6
8    pz   4.9
9    cz   0.6
c -------------------
1       cz     0.5
2    cz   1
3    pz   4.6
4    pz   5.5
5    pz   6
6    pz   6.9
77   cz   0.1
c -------------------
80   cz   5
81   pz   0
82   pz   7

sdef pos 1 0 5.75 axs 1 0 0 rad d1 vec -1 0 0 dir 1 erg=2.53-8
 si1 0 1
 sp1 -21 1
nps 10000000
c
c Mesh Tallies, n-capture rate in cells
fmesh14:n geom=xyz origin=-5 -5 0
```

```
     imesh=5 iints=300
     jmesh=5 jints=300
     kmesh=10 kints=300
     out=ij enorm=no
fm14 -1.0 0 -2
c tally -2 is absorption per source neutron
print 110
c f2:n (4 5 1)
+f6 10 20
c 6LiD sparkplug inside lead cavity filled with DT
m100    3006.80c 1 1002.80c 1
m200 13027.80c 1
m300 6000.80c 1
mt300 grph.29t
```

Appendix K, MCNP6 input files for the ^{10}B shell for a range of cavity heights

```
Boost model B, density corrected to conserve mass during
compression
10      0             -1 4 -5     imp:n=1
40   100  -10.07   -9 8 -7 #10       imp:n=1
20   100  -2.08     -2 3 -6 #10 #40  imp:n=1
30      0                  #10 #20 #40    imp:n=0

7    pz   5.2
8    pz   4.9
9    cz   0.6
c -------------------
1       cz     0.5
2    cz   10
3    pz   0
4    pz   5
5    pz   5.1
6    pz   10

sdef x=d1 y=d2 z=d3 erg=d4 cell 10
 si1 -0.5 0.5
 sp1 0 1
 si2 -0.5 0.5
 sp2 0 1
 si3 5 5.1
 sp3 0 1
 si4 h 1 20
 sp4 -4 -0.02 -1
nps 1000000
c
c Mesh Tallies, n-capture rate in cells
```

```
fmesh14:n geom=xyz origin=-5 -5 0
     imesh=5 iints=300
     jmesh=5 jints=300
     kmesh=10 kints=300
     out=ij enorm=no
fm14 -1.0 0 -2
c tally -2 is absorption per source neutron
print 110
f2:n (4 5 1)
+f6 40 20
c 6LiD shell inside U238
m100    5010.80c 1
m200 92238.80c 1


Boost model
10      0             -1 4 -5     imp:n=1
40   100  -6.58     -9 8 -7 #10      imp:n=1
20   100  -2.08     -2 3 -6 #10 #40  imp:n=1
30      0                  #10 #20 #40    imp:n=0

7    pz   5.6
8    pz   4.9
9    cz   0.6
c -------------------
1       cz     0.5
2    cz   10
3    pz   0
4    pz   5
5    pz   5.5
6    pz   10
```

```
sdef x=d1 y=d2 z=d3 erg=d4 cell 10
 si1 -0.5 0.5
 sp1 0 1
 si2 -0.5 0.5
 sp2 0 1
 si3 5 5.5
 sp3 0 1
 si4 h 1 20
 sp4 -4 -0.02 -1
nps 1000000
c
c Mesh Tallies, n-capture rate in cells
fmesh14:n geom=xyz origin=-5 -5 0
     imesh=5 iints=300
     jmesh=5 jints=300
     kmesh=10 kints=300
     out=ij enorm=no
fm14 -1.0 0 -2
c tally -2 is absorption per source neutron
print 110
f2:n (4 5 1)
+f6 40 20
c 6LiD shell inside U238
m100    5010.80c 1
m200 92238.80c 1


Boost model
10      0              -1 4 -5     imp:n=1
40   100  -4.59   -9 8 -7 #10   imp:n=1
20   100  -2.08     -2 3 -6 #10 #40  imp:n=1
30      0                   #10 #20 #40    imp:n=0
```

```
7    pz   6.1
8    pz   4.9
9    cz   0.6
c -------------------
1       cz     0.5
2    cz   10
3    pz   0
4    pz   5
5    pz   6
6    pz   10

sdef x=d1 y=d2 z=d3 erg=d4 cell 10
 si1 -0.5 0.5
 sp1 0 1
 si2 -0.5 0.5
 sp2 0 1
 si3 5 6
 sp3 0 1
 si4 h 1 20
 sp4 -4 -0.02 -1
nps 1000000
c
c Mesh Tallies, n-capture rate in cells
fmesh14:n geom=xyz origin=-5 -5 0
     imesh=5 iints=300
     jmesh=5 jints=300
     kmesh=10 kints=300
     out=ij enorm=no
fm14 -1.0 0 -2
c tally -2 is absorption per source neutron
print 110
```

```
f2:n (4 5 1)
+f6 40 20
c 6LiD shell inside U238
m100    5010.80c 1
m200 92238.80c 1


Boost model
10      0             -1 4 -5     imp:n=1
40   100  -3.53   -9 8 -7 #10   imp:n=1
20   100  -2.08     -2 3 -6 #10 #40  imp:n=1
30      0                  #10 #20 #40    imp:n=0

7    pz   6.6
8    pz   4.9
9    cz   0.6
c -------------------
1       cz     0.5
2    cz   10
3    pz   0
4    pz   5
5    pz   6.5
6    pz   10

sdef x=d1 y=d2 z=d3 erg=d4 cell 10
 si1 -0.5 0.5
 sp1 0 1
 si2 -0.5 0.5
 sp2 0 1
 si3 5 6.5
 sp3 0 1
 si4 h 1 20
```

```
 sp4 -4 -0.02 -1
nps 1000000
c
c Mesh Tallies, n-capture rate in cells
fmesh14:n geom=xyz origin=-5 -5 0
     imesh=5 iints=300
     jmesh=5 jints=300
     kmesh=10 kints=300
     out=ij enorm=no
fm14 -1.0 0 -2
c tally -2 is absorption per source neutron
print 110
f2:n (4 5 1)
+f6 40 20
c 6LiD shell inside U238
m100    5010.80c 1
m200 92238.80c 1


Boost model
10      0              -1 4 -5     imp:n=1
40   100  -2.41   -9 8 -7 #10   imp:n=1
20   100  -2.08     -2 3 -6 #10 #40  imp:n=1
30      0                   #10 #20 #40    imp:n=0

7    pz   7.6
8    pz   4.9
9    cz   0.6
c --------------------
1       cz     0.5
2    cz   10
3    pz   0
```

```
4    pz   5
5    pz   7.5
6    pz   10

sdef x=d1 y=d2 z=d3 erg=d4 cell 10
 si1 -0.5 0.5
 sp1 0 1
 si2 -0.5 0.5
 sp2 0 1
 si3 5 7.5
 sp3 0 1
 si4 h 1 20
 sp4 -4 -0.02 -1
nps 1000000
c
c Mesh Tallies, n-capture rate in cells
fmesh14:n geom=xyz origin=-5 -5 0
     imesh=5 iints=300
     jmesh=5 jints=300
     kmesh=10 kints=300
     out=ij enorm=no
fm14 -1.0 0 -2
c tally -2 is absorption per source neutron
print 110
f2:n (4 5 1)
+f6 40 20
c 6LiD shell inside U238
m100    5010.80c 1
m200 92238.80c 1


Boost model
```

```
10      0              -1 4 -5     imp:n=1
40   100  -2.08   -9 8 -7 #10   imp:n=1
20   100  -2.08     -2 3 -6 #10 #40  imp:n=1
30      0                   #10 #20 #40    imp:n=0

7    pz   8.1
8    pz   4.9
9    cz   0.6
c -------------------
1       cz     0.5
2    cz   10
3    pz   0
4    pz   5
5    pz   8
6    pz   10

sdef x=d1 y=d2 z=d3 erg=d4 cell 10
 si1 -0.5 0.5
 sp1 0 1
 si2 -0.5 0.5
 sp2 0 1
 si3 5 8
 sp3 0 1
 si4 h 1 20
 sp4 -4 -0.02 -1
nps 1000000
c
c Mesh Tallies, n-capture rate in cells
fmesh14:n geom=xyz origin=-5 -5 0
     imesh=5 iints=300
     jmesh=5 jints=300
     kmesh=10 kints=300
```

```
     out=ij enorm=no
fm14 -1.0 0 -2
c tally -2 is absorption per source neutron
print 110
f2:n (4 5 1)
+f6 40 20
c 6LiD shell inside U238
m100    5010.80c 1
m200 92238.80c 1
```

Appendix L, MCNP6 input files for the ^{6}Li metal shell for a range of cavity heights

```
Boost model Li, density corrected to conserve mass during
compression
10      0             -1 4 -5     imp:n=1
40   100  -2.42   -9 8 -7 #10   imp:n=1
20   100  -0.5 -2 3 -6 #10 #40  imp:n=1
30      0                  #10 #20 #40    imp:n=0

7    pz   5.2
8    pz   4.9
9    cz   0.6
c -------------------
1       cz     0.5
2    cz   10
3    pz   0
4    pz   5
5    pz   5.1
6    pz   10

sdef x=d1 y=d2 z=d3 erg=d4 cell 10
 si1 -0.5 0.5
 sp1 0 1
 si2 -0.5 0.5
 sp2 0 1
 si3 5 5.1
 sp3 0 1
 si4 h 1 20
 sp4 -4 -0.02 -1
nps 1000000
c
c Mesh Tallies, n-capture rate in cells
```

```
fmesh14:n geom=xyz origin=-5 -5 0
     imesh=5 iints=300
     jmesh=5 jints=300
     kmesh=10 kints=300
     out=ij enorm=no
fm14 -1.0 0 -2
c tally -2 is absorption per source neutron
print 110
f2:n (4 5 1)
+f6 40 20
c 6LiD shell inside U238
m100    3006.80c 1
m200 92238.80c 1


Boost model
10      0             -1 4 -5     imp:n=1
40   100  -1.58 -9 8 -7 #10     imp:n=1
20   100  -0.5 -2 3 -6 #10 #40  imp:n=1
30      0                  #10 #20 #40    imp:n=0

7    pz   5.6
8    pz   4.9
9    cz   0.6
c -------------------
1       cz     0.5
2    cz   10
3    pz   0
4    pz   5
5    pz   5.5
6    pz   10
```

```
sdef x=d1 y=d2 z=d3 erg=d4 cell 10
 si1 -0.5 0.5
 sp1 0 1
 si2 -0.5 0.5
 sp2 0 1
 si3 5 5.5
 sp3 0 1
 si4 h 1 20
 sp4 -4 -0.02 -1
nps 1000000
c
c Mesh Tallies, n-capture rate in cells
fmesh14:n geom=xyz origin=-5 -5 0
     imesh=5 iints=300
     jmesh=5 jints=300
     kmesh=10 kints=300
     out=ij enorm=no
fm14 -1.0 0 -2
c tally -2 is absorption per source neutron
print 110
f2:n (4 5 1)
+f6 40 20
c 6LiD shell inside U238
m100    3006.80c 1
m200 92238.80c 1


Boost model
10      0              -1 4 -5     imp:n=1
40   100  -1.1   -9 8 -7 #10    imp:n=1
20   100  -0.5 -2 3 -6 #10 #40  imp:n=1
30      0                   #10 #20 #40    imp:n=0
```

```
7    pz   6.1
8    pz   4.9
9    cz   0.6
c -------------------
1       cz     0.5
2    cz   10
3    pz   0
4    pz   5
5    pz   6
6    pz   10

sdef x=d1 y=d2 z=d3 erg=d4 cell 10
 si1 -0.5 0.5
 sp1 0 1
 si2 -0.5 0.5
 sp2 0 1
 si3 5 6
 sp3 0 1
 si4 h 1 20
 sp4 -4 -0.02 -1
nps 1000000
c
c Mesh Tallies, n-capture rate in cells
fmesh14:n geom=xyz origin=-5 -5 0
     imesh=5 iints=300
     jmesh=5 jints=300
     kmesh=10 kints=300
     out=ij enorm=no
fm14 -1.0 0 -2
c tally -2 is absorption per source neutron
print 110
```

```
f2:n (4 5 1)
+f6 40 20
c 6LiD shell inside U238
m100    3006.80c 1
m200 92238.80c 1


Boost model
10      0             -1 4 -5     imp:n=1
40   100  -0.85   -9 8 -7 #10   imp:n=1
20   100  -0.5 -2 3 -6 #10 #40  imp:n=1
30      0                  #10 #20 #40    imp:n=0

7    pz   6.6
8    pz   4.9
9    cz   0.6
c -------------------
1       cz     0.5
2    cz   10
3    pz   0
4    pz   5
5    pz   6.5
6    pz   10

sdef x=d1 y=d2 z=d3 erg=d4 cell 10
 si1 -0.5 0.5
 sp1 0 1
 si2 -0.5 0.5
 sp2 0 1
 si3 5 6.5
 sp3 0 1
 si4 h 1 20
```

```
 sp4 -4 -0.02 -1
nps 1000000
c
c Mesh Tallies, n-capture rate in cells
fmesh14:n geom=xyz origin=-5 -5 0
     imesh=5 iints=300
     jmesh=5 jints=300
     kmesh=10 kints=300
     out=ij enorm=no
fm14 -1.0 0 -2
c tally -2 is absorption per source neutron
print 110
f2:n (4 5 1)
+f6 40 20
c 6LiD shell inside U238
m100    3006.80c 1
m200 92238.80c 1


Boost model
10      0              -1 4 -5     imp:n=1
40   100  -0.58   -9 8 -7 #10   imp:n=1
20   100  -0.5 -2 3 -6 #10 #40  imp:n=1
30      0                   #10 #20 #40    imp:n=0

7    pz   7.6
8    pz   4.9
9    cz   0.6
c --------------------
1       cz     0.5
2    cz   10
3    pz   0
```

```
4    pz   5
5    pz   7.5
6    pz   10

sdef x=d1 y=d2 z=d3 erg=d4 cell 10
 si1 -0.5 0.5
 sp1 0 1
 si2 -0.5 0.5
 sp2 0 1
 si3 5 7.5
 sp3 0 1
 si4 h 1 20
 sp4 -4 -0.02 -1
nps 1000000
c
c Mesh Tallies, n-capture rate in cells
fmesh14:n geom=xyz origin=-5 -5 0
     imesh=5 iints=300
     jmesh=5 jints=300
     kmesh=10 kints=300
     out=ij enorm=no
fm14 -1.0 0 -2
c tally -2 is absorption per source neutron
print 110
f2:n (4 5 1)
+f6 40 20
c 6LiD shell inside U238
m100    3006.80c 1
m200 92238.80c 1


Boost model
```

```
10      0              -1 4 -5     imp:n=1
40   100  -0.5   -9 8 -7 #10    imp:n=1
20   100  -0.5 -2 3 -6 #10 #40  imp:n=1
30      0                   #10 #20 #40    imp:n=0

7    pz   8.1
8    pz   4.9
9    cz   0.6
c -------------------
1       cz     0.5
2    cz   10
3    pz   0
4    pz   5
5    pz   8
6    pz   10

sdef x=d1 y=d2 z=d3 erg=d4 cell 10
 si1 -0.5 0.5
 sp1 0 1
 si2 -0.5 0.5
 sp2 0 1
 si3 5 8
 sp3 0 1
 si4 h 1 20
 sp4 -4 -0.02 -1
nps 1000000
c
c Mesh Tallies, n-capture rate in cells
fmesh14:n geom=xyz origin=-5 -5 0
     imesh=5 iints=300
     jmesh=5 jints=300
     kmesh=10 kints=300
```

```
     out=ij enorm=no
fm14 -1.0 0 -2
c tally -2 is absorption per source neutron
print 110
f2:n (4 5 1)
+f6 40 20
c 6LiD shell inside U238
m100    3006.80c 1
m200 92238.80c 1
```

Appendix M, MCNP6 input files for the ^{6}LiD shell for a range of cavity heights

```
Boost model LiD, density corrected to conserve mass during
compression
10      0             -1 4 -5     imp:n=1
40   100  -3.87   -9 8 -7 #10   imp:n=1
20   100  -0.8 -2 3 -6 #10 #40  imp:n=1
30      0                  #10 #20 #40    imp:n=0

7    pz   5.2
8    pz   4.9
9    cz   0.6
c -------------------
1       cz     0.5
2    cz   10
3    pz   0
4    pz   5
5    pz   5.1
6    pz   10

sdef x=d1 y=d2 z=d3 erg=d4 cell 10
 si1 -0.5 0.5
 sp1 0 1
 si2 -0.5 0.5
 sp2 0 1
 si3 5 5.1
 sp3 0 1
 si4 h 1 20
 sp4 -4 -0.02 -1
nps 1000000
c
c Mesh Tallies, n-capture rate in cells
```

```
fmesh14:n geom=xyz origin=-5 -5 0
     imesh=5 iints=300
     jmesh=5 jints=300
     kmesh=10 kints=300
     out=ij enorm=no
fm14 -1.0 0 -2
c tally -2 is absorption per source neutron
print 110
f2:n (4 5 1)
+f6 40 20
c 6LiD shell inside U238
m100    3006.80c 1 1002.80c 1
m200 92238.80c 1



Boost model
10      0             -1 4 -5     imp:n=1
40   100  -2.53 -9 8 -7 #10     imp:n=1
20   100  -0.8 -2 3 -6 #10 #40  imp:n=1
30      0                  #10 #20 #40    imp:n=0

7    pz   5.6
8    pz   4.9
9    cz   0.6
c -------------------
1       cz     0.5
2    cz   10
3    pz   0
4    pz   5
5    pz   5.5
6    pz   10
```

```
sdef x=d1 y=d2 z=d3 erg=d4 cell 10
 si1 -0.5 0.5
 sp1 0 1
 si2 -0.5 0.5
 sp2 0 1
 si3 5 5.5
 sp3 0 1
 si4 h 1 20
 sp4 -4 -0.02 -1
nps 1000000
c
c Mesh Tallies, n-capture rate in cells
fmesh14:n geom=xyz origin=-5 -5 0
     imesh=5 iints=300
     jmesh=5 jints=300
     kmesh=10 kints=300
     out=ij enorm=no
fm14 -1.0 0 -2
c tally -2 is absorption per source neutron
print 110
f2:n (4 5 1)
+f6 40 20
c 6LiD shell inside U238
m100    3006.80c 1 1002.80c 1
m200 92238.80c 1


Boost model
10      0              -1 4 -5     imp:n=1
40   100  -1.77   -9 8 -7 #10   imp:n=1
20   100  -0.8 -2 3 -6 #10 #40  imp:n=1
```

```
30      0                   #10 #20 #40    imp:n=0

7    pz   6.1
8    pz   4.9
9    cz   0.6
c --------------------
1       cz     0.5
2    cz   10
3    pz   0
4    pz   5
5    pz   6
6    pz   10

sdef x=d1 y=d2 z=d3 erg=d4 cell 10
 si1 -0.5 0.5
 sp1 0 1
 si2 -0.5 0.5
 sp2 0 1
 si3 5 6
 sp3 0 1
 si4 h 1 20
 sp4 -4 -0.02 -1
nps 1000000
c
c Mesh Tallies, n-capture rate in cells
fmesh14:n geom=xyz origin=-5 -5 0
     imesh=5 iints=300
     jmesh=5 jints=300
     kmesh=10 kints=300
     out=ij enorm=no
fm14 -1.0 0 -2
c tally -2 is absorption per source neutron
```

```
print 110
f2:n (4 5 1)
+f6 40 20
c 6LiD shell inside U238
m100    3006.80c 1 1002.80c 1
m200 92238.80c 1


Boost model
10      0             -1 4 -5     imp:n=1
40   100  -1.36   -9 8 -7 #10   imp:n=1
20   100  -0.8 -2 3 -6 #10 #40  imp:n=1
30      0                  #10 #20 #40    imp:n=0

7    pz   6.6
8    pz   4.9
9    cz   0.6
c -------------------
1       cz     0.5
2    cz   10
3    pz   0
4    pz   5
5    pz   6.5
6    pz   10

sdef x=d1 y=d2 z=d3 erg=d4 cell 10
 si1 -0.5 0.5
 sp1 0 1
 si2 -0.5 0.5
 sp2 0 1
 si3 5 6.5
 sp3 0 1
```

```
 si4 h 1 20
 sp4 -4 -0.02 -1
nps 1000000
c
c Mesh Tallies, n-capture rate in cells
fmesh14:n geom=xyz origin=-5 -5 0
     imesh=5 iints=300
     jmesh=5 jints=300
     kmesh=10 kints=300
     out=ij enorm=no
fm14 -1.0 0 -2
c tally -2 is absorption per source neutron
print 110
f2:n (4 5 1)
+f6 40 20
c 6LiD shell inside U238
m100    3006.80c 1 1002.80c 1
m200 92238.80c 1


Boost model
10      0             -1 4 -5     imp:n=1
40   100  -0.93   -9 8 -7 #10   imp:n=1
20   100  -0.8 -2 3 -6 #10 #40  imp:n=1
30      0                  #10 #20 #40    imp:n=0

7    pz   7.6
8    pz   4.9
9    cz   0.6
c --------------------
1       cz     0.5
2    cz   10
```

```
3    pz   0
4    pz   5
5    pz   7.5
6    pz   10

sdef x=d1 y=d2 z=d3 erg=d4 cell 10
 si1 -0.5 0.5
 sp1 0 1
 si2 -0.5 0.5
 sp2 0 1
 si3 5 7.5
 sp3 0 1
 si4 h 1 20
 sp4 -4 -0.02 -1
nps 1000000
c
c Mesh Tallies, n-capture rate in cells
fmesh14:n geom=xyz origin=-5 -5 0
     imesh=5 iints=300
     jmesh=5 jints=300
     kmesh=10 kints=300
     out=ij enorm=no
fm14 -1.0 0 -2
c tally -2 is absorption per source neutron
print 110
f2:n (4 5 1)
+f6 40 20
c 6LiD shell inside U238
m100    3006.80c 1 1002.80c 1
m200 92238.80c 1

Boost model
```

```
10      0              -1 4 -5     imp:n=1
40   100  -0.8   -9 8 -7 #10    imp:n=1
20   100  -0.8 -2 3 -6 #10 #40  imp:n=1
30      0                   #10 #20 #40    imp:n=0

7    pz   8.1
8    pz   4.9
9    cz   0.6
c -------------------
1       cz     0.5
2    cz   10
3    pz   0
4    pz   5
5    pz   8
6    pz   10

sdef x=d1 y=d2 z=d3 erg=d4 cell 10
 si1 -0.5 0.5
 sp1 0 1
 si2 -0.5 0.5
 sp2 0 1
 si3 5 8
 sp3 0 1
 si4 h 1 20
 sp4 -4 -0.02 -1
nps 1000000
c
c Mesh Tallies, n-capture rate in cells
fmesh14:n geom=xyz origin=-5 -5 0
     imesh=5 iints=300
     jmesh=5 jints=300
     kmesh=10 kints=300
```

```
     out=ij enorm=no
fm14 -1.0 0 -2
c tally -2 is absorption per source neutron
print 110
f2:n (4 5 1)
+f6 40 20
c 6LiD shell inside U238
m100    3006.80c 1 1002.80c 1
m200 92238.80c 1
```

Appendix N, MCNP6 input files for the ^{238}U shell for a range of cavity heights

```
Boost model

10      0              -1 4 -5     imp:n=1
20   100  -18  -2 3 -6 #10  imp:n=1
30      0                  #10 #20     imp:n=0


1       cz     0.5
2    cz   10
3    pz   0
4    pz   5
5    pz   5.1
6    pz   10

sdef x=d1 y=d2 z=d3 erg=d4 cell 10
 si1 -0.5 0.5
 sp1 0 1
 si2 -0.5 0.5
 sp2 0 1
 si3 5 5.1
 sp3 0 1
 si4 h 1 20
 sp4 -4 -0.02 -1
nps 1000000
c
c Mesh Tallies, n-capture rate in cells
fmesh14:n geom=xyz origin=-5 -5 0
     imesh=5 iints=300
     jmesh=5 jints=300
     kmesh=10 kints=300
     out=ij enorm=no
+fm14 -18 100 -8
```

```
c tally -8 is fission energy per source neutron
print 110
+f6 20
c
m100    92238  -100


Boost model
10      0             -1 4 -5     imp:n=1
20   100  -18  -2 3 -6 #10  imp:n=1
30      0                  #10 #20     imp:n=0

1       cz     0.5
2    cz   10
3    pz   0
4    pz   5
5    pz   5.5
6    pz   10

sdef x=d1 y=d2 z=d3 erg=d4 cell 10
 si1 -0.5 0.5
 sp1 0 1
 si2 -0.5 0.5
 sp2 0 1
 si3 5 5.5
 sp3 0 1
 si4 h 1 20
 sp4 -4 -0.02 -1
nps 10000000
c
c Mesh Tallies, n-capture rate in cells
fmesh14:n geom=xyz origin=-5 -5 0
```

```
     imesh=5 iints=300
     jmesh=5 jints=300
     kmesh=10 kints=300
     out=ij enorm=no
+fm14 -18 100 -8
print 110
+f6 20
c
m100    92238  -100


Boost model
10      0              -1 4 -5     imp:n=1
20   100  -18  -2 3 -6 #10  imp:n=1
30      0                  #10 #20     imp:n=0

1       cz     0.5
2    cz   10
3    pz   0
4    pz   5
5    pz   6
6    pz   10

sdef x=d1 y=d2 z=d3 erg=d4 cell 10
 si1 -0.5 0.5
 sp1 0 1
 si2 -0.5 0.5
 sp2 0 1
 si3 5 6
 sp3 0 1
 si4 h 1 20
 sp4 -4 -0.02 -1
```

```
nps 10000000
c
c Mesh Tallies, n-capture rate in cells
fmesh14:n geom=xyz origin=-5 -5 0
     imesh=5 iints=300
     jmesh=5 jints=300
     kmesh=10 kints=300
     out=ij enorm=no
+fm14 -18 100 -8
print 110
+f6 20
c
m100    92238  -100


Boost model
10      0             -1 4 -5     imp:n=1
20   100  -18  -2 3 -6 #10  imp:n=1
30      0                  #10 #20     imp:n=0

1       cz     0.5
2    cz   10
3    pz   0
4    pz   5
5    pz   6.5
6    pz   7.4

sdef x=d1 y=d2 z=d3 erg=d4 cell 10
 si1 -0.5 0.5
 sp1 0 1
 si2 -0.5 0.5
 sp2 0 1
```

```
 si3 5 6.5
 sp3 0 1
 si4 h 1 20
 sp4 -4 -0.02 -1
nps 10000000
c
c Mesh Tallies, n-capture rate in cells
c fmesh14:n geom=xyz origin=-5 -5 0
c     imesh=5 iints=300
c     jmesh=5 jints=300
c     kmesh=10 kints=300
c     out=ij enorm=no
c +fm14 -18 100 -8
c tally -8 is fission energy per source neutron
print 110
+f6 20
c
m100    92238  -100


Boost model
10      0             -1 4 -5     imp:n=1
20   100  -18  -2 3 -6 #10  imp:n=1
30      0                  #10 #20     imp:n=0

1       cz     0.5
2    cz   10
3    pz   0
4    pz   5
5    pz   7
6    pz   10
```

```
sdef x=d1 y=d2 z=d3 erg=d4 cell 10
 si1 -0.5 0.5
 sp1 0 1
 si2 -0.5 0.5
 sp2 0 1
 si3 5 7
 sp3 0 1
 si4 h 1 20
 sp4 -4 -0.02 -1
nps 10000000
c
c Mesh Tallies, n-capture rate in cells
fmesh14:n geom=xyz origin=-5 -5 0
     imesh=5 iints=300
     jmesh=5 jints=300
     kmesh=10 kints=300
     out=ij enorm=no
+fm14 -18 100 -8
print 110
+f6 20
c
m100    92238  -100


Boost model
10      0             -1 4 -5     imp:n=1
20   100  -18  -2 3 -6 #10  imp:n=1
30      0                  #10 #20     imp:n=0

1       cz     0.5
2    cz   10
3    pz   0
```

```
4    pz   5
5    pz   7.5
6    pz   10

sdef x=d1 y=d2 z=d3 erg=d4 cell 10
 si1 -0.5 0.5
 sp1 0 1
 si2 -0.5 0.5
 sp2 0 1
 si3 5 7.5
 sp3 0 1
 si4 h 1 20
 sp4 -4 -0.02 -1
nps 10000000
c
c Mesh Tallies, n-capture rate in cells
fmesh14:n geom=xyz origin=-5 -5 0
     imesh=5 iints=300
     jmesh=5 jints=300
     kmesh=10 kints=300
     out=ij enorm=no
+fm14 -18 100 -8
print 110
+f6 20
c
m100    92238  -100


Boost model
10      0              -1 4 -5     imp:n=1
20   100  -18  -2 3 -6 #10  imp:n=1
30      0                  #10 #20     imp:n=0
```

```
1       cz     0.5
2    cz   10
3    pz   0
4    pz   5
5    pz   8
6    pz   10

sdef x=d1 y=d2 z=d3 erg=d4 cell 10
 si1 -0.5 0.5
 sp1 0 1
 si2 -0.5 0.5
 sp2 0 1
 si3 5 8
 sp3 0 1
 si4 h 1 20
 sp4 -4 -0.02 -1
nps 10000000
c
c Mesh Tallies, n-capture rate in cells
fmesh14:n geom=xyz origin=-5 -5 0
     imesh=5 iints=300
     jmesh=5 jints=300
     kmesh=10 kints=300
     out=ij enorm=no
+fm14 -18 100 -8
print 110
+f6 20
c
m100    92238  -100
```

Appendix O, MCNP6 input files for the ^{238}U and LiD shell for a range of cavity heights

```
Boost model LiD + U238, density corrected to conserve mass
during compression
10      0             -1 4 -5     imp:n=1
40   100  -3.87   -9 8 -7 #10   imp:n=1
20   200  -18  -2 3 -6 #10 #40  imp:n=1
30      0                  #10 #20 #40    imp:n=0

7    pz   5.2
8    pz   4.9
9    cz   0.6
c -------------------
1       cz     0.5
2    cz   10
3    pz   0
4    pz   5
5    pz   5.1
6    pz   10

sdef x=d1 y=d2 z=d3 erg=d4 cell 10
 si1 -0.5 0.5
 sp1 0 1
 si2 -0.5 0.5
 sp2 0 1
 si3 5 5.1
 sp3 0 1
 si4 h 1 20
 sp4 -4 -0.02 -1
nps 1000000
c
c Mesh Tallies, n-capture rate in cells
```

```
fmesh14:n geom=xyz origin=-5 -5 0
     imesh=5 iints=300
     jmesh=5 jints=300
     kmesh=10 kints=300
     out=ij enorm=no
fm14 -1.0 0 -2
c tally -2 is absorption per source neutron
print 110
f2:n (4 5 1)
+f6 40 20
c 6LiD shell inside U238
m100    3006.80c 1 1002.80c 1
m200 92238.80c 1


Boost model
10      0             -1 4 -5     imp:n=1
40   100  -2.53 -9 8 -7 #10     imp:n=1
20   200  -18  -2 3 -6 #10 #40  imp:n=1
30      0                  #10 #20 #40    imp:n=0

7    pz   5.6
8    pz   4.9
9    cz   0.6
c --------------------
1       cz     0.5
2    cz   10
3    pz   0
4    pz   5
5    pz   5.5
6    pz   10
```

```
sdef x=d1 y=d2 z=d3 erg=d4 cell 10
 si1 -0.5 0.5
 sp1 0 1
 si2 -0.5 0.5
 sp2 0 1
 si3 5 5.5
 sp3 0 1
 si4 h 1 20
 sp4 -4 -0.02 -1
nps 1000000
c
c Mesh Tallies, n-capture rate in cells
fmesh14:n geom=xyz origin=-5 -5 0
     imesh=5 iints=300
     jmesh=5 jints=300
     kmesh=10 kints=300
     out=ij enorm=no
fm14 -1.0 0 -2
c tally -2 is absorption per source neutron
print 110
f2:n (4 5 1)
+f6 40 20
c 6LiD shell inside U238
m100    3006.80c 1 1002.80c 1
m200 92238.80c 1


Boost model
10      0              -1 4 -5     imp:n=1
40   100  -1.77   -9 8 -7 #10   imp:n=1
20   200  -18  -2 3 -6 #10 #40  imp:n=1
30      0                   #10 #20 #40    imp:n=0
```

```
7    pz   6.1
8    pz   4.9
9    cz   0.6
c -------------------
1       cz     0.5
2    cz   10
3    pz   0
4    pz   5
5    pz   6
6    pz   10

sdef x=d1 y=d2 z=d3 erg=d4 cell 10
 si1 -0.5 0.5
 sp1 0 1
 si2 -0.5 0.5
 sp2 0 1
 si3 5 6
 sp3 0 1
 si4 h 1 20
 sp4 -4 -0.02 -1
nps 1000000
c
c Mesh Tallies, n-capture rate in cells
fmesh14:n geom=xyz origin=-5 -5 0
     imesh=5 iints=300
     jmesh=5 jints=300
     kmesh=10 kints=300
     out=ij enorm=no
fm14 -1.0 0 -2
c tally -2 is absorption per source neutron
print 110
```

```
f2:n (4 5 1)
+f6 40 20
c 6LiD shell inside U238
m100    3006.80c 1 1002.80c 1
m200 92238.80c 1


Boost model
10      0             -1 4 -5     imp:n=1
40   100  -1.36   -9 8 -7 #10   imp:n=1
20   200  -18  -2 3 -6 #10 #40  imp:n=1
30      0                  #10 #20 #40    imp:n=0

7    pz   6.6
8    pz   4.9
9    cz   0.6
c -------------------
1       cz     0.5
2    cz   10
3    pz   0
4    pz   5
5    pz   6.5
6    pz   10

sdef x=d1 y=d2 z=d3 erg=d4 cell 10
 si1 -0.5 0.5
 sp1 0 1
 si2 -0.5 0.5
 sp2 0 1
 si3 5 6.5
 sp3 0 1
 si4 h 1 20
```

```
 sp4 -4 -0.02 -1
nps 1000000
c
c Mesh Tallies, n-capture rate in cells
fmesh14:n geom=xyz origin=-5 -5 0
     imesh=5 iints=300
     jmesh=5 jints=300
     kmesh=10 kints=300
     out=ij enorm=no
fm14 -1.0 0 -2
c tally -2 is absorption per source neutron
print 110
f2:n (4 5 1)
+f6 40 20
c 6LiD shell inside U238
m100    3006.80c 1 1002.80c 1
m200 92238.80c 1


Boost model
10      0              -1 4 -5     imp:n=1
40   100  -0.93   -9 8 -7 #10   imp:n=1
20   200  -18  -2 3 -6 #10 #40  imp:n=1
30      0                   #10 #20 #40    imp:n=0

7    pz   7.6
8    pz   4.9
9    cz   0.6
c --------------------
1       cz     0.5
2    cz   10
3    pz   0
```

```
4    pz   5
5    pz   7.5
6    pz   10

sdef x=d1 y=d2 z=d3 erg=d4 cell 10
 si1 -0.5 0.5
 sp1 0 1
 si2 -0.5 0.5
 sp2 0 1
 si3 5 7.5
 sp3 0 1
 si4 h 1 20
 sp4 -4 -0.02 -1
nps 1000000
c
c Mesh Tallies, n-capture rate in cells
fmesh14:n geom=xyz origin=-5 -5 0
     imesh=5 iints=300
     jmesh=5 jints=300
     kmesh=10 kints=300
     out=ij enorm=no
fm14 -1.0 0 -2
c tally -2 is absorption per source neutron
print 110
f2:n (4 5 1)
+f6 40 20
c 6LiD shell inside U238
m100    3006.80c 1 1002.80c 1
m200 92238.80c 1


Boost model
```

```
10      0              -1 4 -5     imp:n=1
40   100  -0.8   -9 8 -7 #10    imp:n=1
20   200  -18  -2 3 -6 #10 #40  imp:n=1
30      0                   #10 #20 #40    imp:n=0

7    pz   8.1
8    pz   4.9
9    cz   0.6
c -------------------
1       cz     0.5
2    cz   10
3    pz   0
4    pz   5
5    pz   8
6    pz   10

sdef x=d1 y=d2 z=d3 erg=d4 cell 10
 si1 -0.5 0.5
 sp1 0 1
 si2 -0.5 0.5
 sp2 0 1
 si3 5 8
 sp3 0 1
 si4 h 1 20
 sp4 -4 -0.02 -1
nps 1000000
c
c Mesh Tallies, n-capture rate in cells
fmesh14:n geom=xyz origin=-5 -5 0
     imesh=5 iints=300
     jmesh=5 jints=300
     kmesh=10 kints=300
```

```
     out=ij enorm=no
fm14 -1.0 0 -2
c tally -2 is absorption per source neutron
print 110
f2:n (4 5 1)
+f6 40 20
c 6LiD shell inside U238
m100    3006.80c 1 1002.80c 1
m200 92238.80c 1
```