Automating Radiation Damage Studies of Materials Irradiated by High-Energy Protons Using Multiphysics Simulations

by

Gabrielle J. Ledoux

Submitted to the Department of Nuclear Science and Engineering
in partial fulfillment of the requirements for the degree of

Bachelor of Science in Nuclear Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2016

Author .....
Department of Nuclear Science and Engineering
May 23, 2016

Certified by .....
Michael P. Short
Assistant Professor of Nuclear Science and Engineering
Thesis Supervisor

Accepted by .....
Michael P. Short
Assistant Professor of Nuclear Science and Engineering
Chairman, NSE Committee for Undergraduates

Automating Radiation Damage Studies of Materials Irradiated by High-Energy Protons Using Multiphysics Simulations

by

Gabrielle J. Ledoux

Submitted to the Department of Nuclear Science and Engineering
on May 23, 2016, in partial fulfillment of the
requirements for the degree of
Bachelor of Science in Nuclear Science and Engineering

Abstract

Understanding radiation and corrosion damage in nuclear materials has become increasingly important in reactor design considerations. However, running irradiation damage studies in nuclear reactors is expensive and time-consuming. Thus, accurate, quick simulations have become more attractive to researchers studying alternative materials in nuclear reactors. This thesis investigates the possibility of automating irradiation damage studies using ion stopping range simulations coupled with heat generation simulations to find the change in temperature across a sample. The range simulations generate 1D slabs with different thicknesses and bombards them with high-energy proton beams. The slabs are automatically sorted, a meshed geometry is created, and the recoil energy information is entered into a multiphysics Finite-Element solver. Ultimately, the optimal beam current for which the temperature gradient across a coolant-sample geometry is less than 5 K is predicted. This thesis examines the possibility of automating the entire simulation process so that many materials and slab thicknesses can be tested for resistance to temperature change (and thus implying specifics of radiation damage effects).

Thesis Supervisor: Michael P. Short
Title: Assistant Professor of Nuclear Science and Engineering

Acknowledgments

There are so many people who have been influential during my time at MIT. I can't possibly name them all in one section, but there are a few people in particular I'd like to thank:

Professor Michael Short: Thank you so much for the opportunities you have given me these past years. You inspire me through your work and compassion for your students. UROPing in your lab and the exciting, open-ended projects you have given me have enhanced my MIT experience tremendously. You have made a profound impact on my love for nuclear materials, research, and finding ways to positively change the world around me through hard work and passion. I am forever grateful for the knowledge you have imparted on me, and for always believing in me (even when I found it hard to believe in myself).

Heather Barry: Thanks for listening to all of my panicking moments in your office and talking to me about anything. I don't think people appreciate enough how much work you do for the students of this department, and your friendship has meant so much to me.

Lena and Vince: Thanks for being there through all of the craziness of the past 4 years. From the late night studying, stressing out about everything Course 22 related, crying from frustrating projects, and celebrating after long nights of work with a drink at the Asgard, I can't think of two better people I would have done it with.

Contents

List of Figures

Chapter 1

Introduction

When designing nuclear reactor components, factors such as reactor efficiency, lifetime, and worker safety are important features to consider. To optimize these features, a variety of engineering strategies can be used. The use of alternative materials in reactor design is an innovative approach to mitigate physical damage to reactor internals. Radiation damage is of particular interest in materials studies because of its negative effects on material ductility and hardness through the creation of defects and creep.

Nuclear materials studies have been crucial to the development of reactor technology and weapons since World War II [10]. Radiation damage-specific studies became more widespread as nuclear reactors became a real possibility of energy generation in the US [10]. Many studies in the 1970s, mainly by Guyette, Gittus, and Boresi, began to analyze the effects of creep in nuclear reactor cladding under various stresses, including radiation damage [8] [6] [3]. These studies, amongst others on various types of radiation damage, serve as the experimental basis for future nuclear materials studies.

However, radiation damage studies can be expensive and time-consuming, and recent software has made it easier to solve heat transfer equations in materials and thus better understand the effects of radiation damage. Before widespread computer use, experimentation was the only method of understanding the elusive nature of radiation damage in materials. Recent nuclear materials studies frequently utilize simulations. Two notable examples are Zarkadoula, who studied high-energy radiation damage in Zirconia in 2014, and Gunay,

who utilized Monte Carlo methods to understand the neutronic performance of fuels [14] [7]. The continued utility of nuclear materials studies, coupled with more advanced simulation software, has resulted in an accelerated analysis of alternative materials in nuclear reactors.

This thesis explores the possibility of automating radiation damage studies and quantifying these effects in materials. The power of simulations can potentially allow a user to enter a particular geometry of materials, simulate damage from an irradiation beam through the geometry, and then calculate temperature changes (and other thermodynamic properties) in the geometry. In this thesis, radiation damage is quantified through temperature gradient across the sample (see Section 2.5). Using scripts to automate the interactions between these programs would allow for rapid radiation damage studies for a specific reactor component geometry and materials.

Given different beam energies, sample geometries, and irradiation temperatures, it is possible to run automated beam heating simulations to quantify the effects of radiation and corrosion damage for specific nuclear reactor design applications. This thesis will explore the possibility of using automated simulations to better understand limitations of ion beam energies that irradiated samples can take without increasing the temperature gradient and thus damage to the nuclear material.

Chapter 2

Background

The software used in this thesis solves heat transfer problems that would otherwise be time-consuming *to solve by hand. Understanding * basic heat transfer properties as well as corrosion and radiation damage mechanisms in nuclear materials *will help the user of this simulation software *to understand the physics behind the programming.

2.1 Walkthrough

One of the benefits of simulation work is that it allows the user to set specific inputs and interpret output data without requiring the user to perform tedious calculations. Automation scripts can execute these calculations rapidly and sort the results, allowing the user to quickly perform tests.

This section will go through the basics of radiation damage in materials and the resulting temperature increase, as well as explain the basics of the Finite Element Method, a method used by simulations to solve complex functions piece-wise. Understanding ion interactions in matter, and more specifically the stopping power and range of ions in matter, allows the reader to visualize the difficulty of quantifying radiation damage in materials.

As a particle beam from an accelerator penetrates a material, the energy lost in collisions slows down the particles and heats up the material. Finding the range and types of interactions a particle can experience in matter allows for calculation of the temperature change in the material. To do this, types of heating must be factored in to the heat gener-

ation equation.

The intricacies of calculating the temperature change in a sample (even in a single particle interaction) make it clear why simulations are used for radiation damage studies. Methods such as the Finite Element Method (FEM) allow a user to enter a physics problem with heat transfer boundary conditions and solve for the desired variables. However, in order to interpret the results and processes in a simulation project, the basic physics used in the simulation should be understood.

2.2 Ion Interactions in Matter

In any material, there are a variety of atomic-level interactions occurring. Modeling these interactions helps us deduce more about heat transfer properties in nuclear materials.

2.2.1 Cross Section

By knowing the scope of which particles can interact with matter, energy deposition can be calculated. The cross section of a reaction is the cross-sectional "area" which it occupies. The unit of cross section is area, and the relationship between cross section and linear attenuation coefficient of a nuclear reaction is:

$$\mu = \sigma_i * \frac{\rho * N_a}{A}$$

where N_a is Avogadro's number, ρ is the mass density of the medium and A is the atomic weight of the medium [12]. The nuclear cross section allows one to understand the physical space that an interaction will likely take place in a medium. In other words, cross section can be interpreted as a probability of interaction over a certain path. When trying to quantify damage from irradiation to a material, the scope of the area of which the particle will interact with the medium helps describe the physical changes that will occur in the medium [12]. Many factors can effect the cross section of an incident particle: its energy and direction being the most prominent, along with the properties of the medium in which the particle is interacting. In the following sections, cross section is often used to better understand the nuclear and charged particle interactions.

The scattering cross section gives the probability that a particle will scatter in a medium [2]. This helps quantify the probability that a recoil atom from an interaction will have a certain amount of energy. When analyzing propagation of a particle interacting with matter, the scattering cross section is used.

2.2.2 Neutron Interactions

Because the neutron is uncharged, it can have purely elastic and inelastic collisions with other nuclei and ions.

Elastic Collisions

A nuclear elastic collision occurs when some energy of a neutron is imparted onto the particle it collides with, and there is no loss of kinetic energy in the system. The probability that a nucleus collides with another particle in a material can be analysed using the total scattering cross section, σ_s in terms of the solid angle Ω that the neutron is incident to the target at:

$$\sigma_s(E_i) = \int \sigma_s(E_i, \Omega) d\Omega$$

However, this does not give a complete understanding of the interactions between a particle and the medium it is incident on. In irradiation studies, the recoil energy (T) of the atom that is struck gives information on how a material changes. Thus, T must be found first to further analyse the probabilistic energy loss.
Using mechanics principles, the energy transferred from the neutron to the particle can be derived in terms of the incident energy and angle of recoil. The results below were derived by Was in [2].

$$T = \frac{\gamma}{2} E_i (1 - cos(\phi))$$

where T is recoil energy, E_i is the initial neutron energy, and γ is

$$\frac{4mM}{(M+m)^2}$$

M,m are masses of the neutron and particle. A full derivation can be found in [2].
The minimum recoil energy occurs at 0 degrees (the neutron misses the particle), and the maximum at π (the neutron causes the particle to exactly backscatter). The distribution in energy by angle is shown in Figure 2-1.

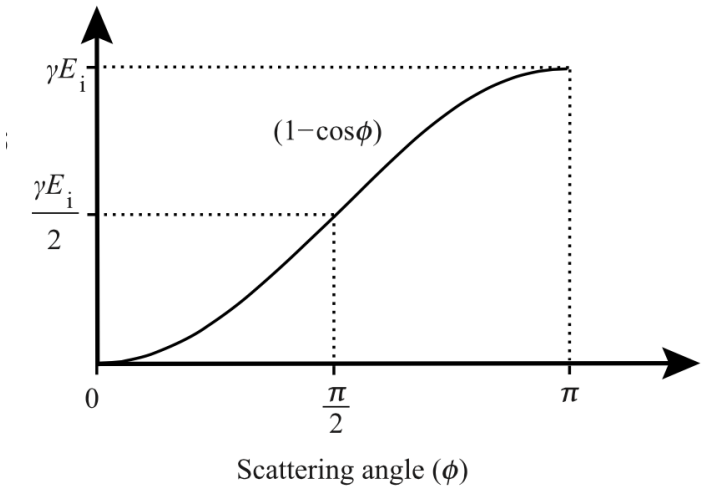

Figure 2-1: Recoil energy as a function of scattering angle of a neutron in a material. [2]

With information on T, the probability that this T is imparted on an atom can be found. By representing the probability of a collision occurring in the range (ϕ, Ω) as $\sigma_s(E_i, \phi)d\Omega$, which is equal to $\sigma_s(E_i, T)dT$, and solving using geometric properties of a sphere (see Was 1.4), the total elastic scattering cross section is:

$$\sigma_s(E_i, T) = 2\pi \int \sigma_s(E_i, \phi) sin\phi d\phi$$

Now, to simplify most calculations, scattering can be assumed to be isotropic (independent of scattering angle). The scattering cross section is

$$\sigma_s(E_i) = 4\pi\sigma_s(E_i, \phi)$$

which is independent of T. This means that the probability of an atom with an energy E_i scatters in a material does not depend on the recoil energy.

Inelastic Collisions

Inelastic collisions are collisions that result in a loss of kinetic energy in the system. The lost energy is transferred into excitation of the target nucleus [2]. The neutron is absorbed by the nucleus, and a γ-ray and neutron are emitted from the nucleus. A full derivation can be found in [2]. It is important to know the kinetic energy loss of the collision*, because this will impact the temperature of the material that contains the incident particle and nucleus. Since kinetic energy is not conserved, the equations in the previous section representing the scattering cross section are not completely true. Thus, when analyzing inelastic collisions, only the total energy is important [2]. Was provides a derivation for the scattering cross section in an inelastic collision based on the Q_j, the reaction energy, which includes the kinetic energy lost in the collision. The result is:

$$\sigma_s(E_i, Q_j, T) = \frac{\sigma(E_i, Q_j)}{\gamma E_i (1 + \frac{Q_j}{E_i}\frac{A+1}{A})^{1/2}}$$

where Q_j is the gamma decay energy at a certain resonance, which has to do with the excitation the particle experiences when the system experiences a loss of kinetic energy.

This calculation is just one of the many formulas that can be used at different models and different particle energies. Table 2-2 shows the different cross sections that can be determined with different models.

There are many criteria and complex cross section equations that make it difficult to predict the energy loss of a particle in matter. Because of this, it is nearly impossible to determine the energy loss by hand.

2.2.3 Ions and Matter

While nuclear reactions do not involve charged interactions, ionic interactions involve complicated interactions between the nuclei and electron clouds of two ions. While hard to quantify these interactions exactly, they can be explained using interatomic potentials [2].

Figure 2-2: Recoil energy as a function of incident angle.[2]

Type of collision	Energy transfer and energy transfer cross section	Equation in text
Elastic scattering	$T = \frac{\gamma}{2} E_{\mathrm{i}} (1 - \cos\phi)$	(1.13)
	$\sigma_{\mathrm{s}}(E_{\mathrm{i}}, T) = \frac{\sigma_{\mathrm{s}}(E_{\mathrm{i}})}{\gamma E_{\mathrm{i}}}$	(1.21)
Inelastic scattering	$T(E_{\mathrm{i}}, Q_j, \phi) = \frac{\gamma}{2} E_{\mathrm{i}} - \frac{\gamma}{2} \left[E_{\mathrm{i}} \left(E_{\mathrm{i}} + Q_{\mathrm{i}} \frac{A+1}{A} \right) \right]^{1/2} \cos\phi + \frac{Q_j}{A+1}$	(1.27)
	resonance region	
	$\sigma_{\mathrm{s},j}(E_{\mathrm{i}}, Q_j, T) = \frac{\sigma_{\mathrm{s},j}(E_{\mathrm{i}}, Q_j)}{\gamma E_{\mathrm{i}} \left(1 + \frac{Q_j}{E_{\mathrm{i}}} \frac{l+A}{A} \right)^{1/2}}$	(1.30)
	unresolved resonance region	
	$\sigma_{\mathrm{is}}(E_{\mathrm{i}}, T) = \sigma_{\mathrm{is}}(E_{\mathrm{i}}) \int_0^{E'^{\max}_{\mathrm{m}}} \frac{f(E_{\mathrm{i}}, E'_{\mathrm{m}})}{4 \frac{1}{A+1} (E_{\mathrm{i}} E'_{\mathrm{m}})^{1/2}} \mathrm{d}E'_{\mathrm{m}}$	(1.31)
(n, 2n)	$T = \frac{A}{A-1} \frac{\eta_1}{\eta_2} E''_{\mathrm{m}} + \frac{A-1}{A} \bar{T}_\ell - 2 \left(\frac{\eta_1}{\eta_2} \right)^{1/2} (\bar{T}_\ell E''_{\mathrm{m}})^{1/2} \cos\phi$	(1.39)
	$\sigma_{\mathrm{n,2n}}(E_{\mathrm{i}}, T) = \int_0^{E_{\mathrm{i}}-U} \frac{E'_{\mathrm{m}}}{I(E_{\mathrm{i}})} \mathrm{e}^{-E'_{\mathrm{m}}/E_{\mathrm{D}}} \times \int_0^{E_{\mathrm{i}}-U-E'_{\mathrm{m}}} \frac{E''_{\mathrm{m}}}{I(E_{\mathrm{i}}, E'_{\mathrm{m}})} \mathrm{e}^{-E''_{\mathrm{m}}/E_{\mathrm{D}}} \mathrm{d}E'_{\mathrm{m}} \mathrm{d}E''_{\mathrm{m}}$	(1.40)
(n, γ)	$\bar{T} \cong \frac{E_\gamma^2}{4(M+m)c^2}$	(1.42)
	$\sigma_{\mathrm{n},\gamma}(E_{\mathrm{i}}) = \sigma_0 \sqrt{\frac{E_0}{E_{\mathrm{i}}}} \left\{ \frac{1}{[(E_{\mathrm{i}} - E_0)/(\Gamma/2)]^2 + 1} \right\}$	(1.44)

Interatomic Potential

In order to understand radiation interactions with matter, the ways in which the energy of an atom changes as it collides with other atoms in materials must be understood. Without this understanding, it is impossible to know the perturbations and change in materials properties that ensue when an ion beam interacts with matter. Although atoms are neutral, they have positive and negative components, and these interactions can be quantified with potential functions. The Coulomb equation describes the potential V(r) between two like charges, where ϵ is the unit charge and r is the interatomic distance:

$$V(r) = \frac{\epsilon^2}{r}$$

This equation is valid at very small distances between particles. When analyzing the in-

teractions between two whole atoms, this equation does not suffice since it can only compare two like charges. While it is nearly impossible to quantify the intricate interactions between two atoms, various potential functions can define this to various degrees of accuracy (see Was for examples such as the hard-sphere approximation). Depending on the separation, Figure 2-3 shows the interatomic potential between two atoms, where r_e is the "nearest neighbor" spacing the crystal, typically in the nanometer range.

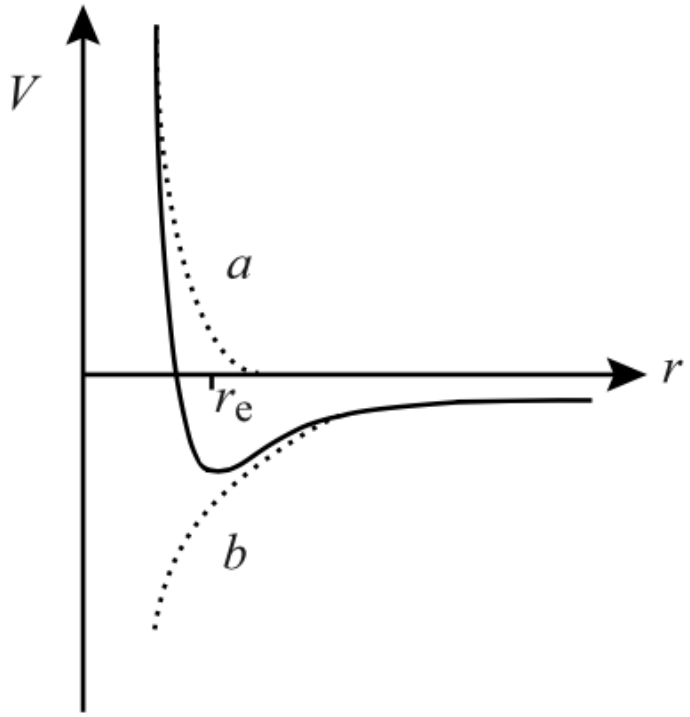

Figure 2-3: Interatomic potential as a function of distance [2]. At low separations, repulsion between the charges in two atoms dominate. As the distance increases, this repulsion affects the interaction less.

From this figure, two general areas can be identified. At larger distances, the repulsion of like charges dominates. For example, as two electron clouds become so close to each other than they overlap, they will begin to repel. Some electrons move to higher energy levels, so the potential increases at short distances [2]. This can be described by the formula

$$V(r) = Aexp\frac{-r}{B}$$

Where A, B can be determined by various elastic properties of the system. At smaller

distances, however, the electrons in the cloud also interact with nuclear forces. The screened Coulomb potential describes this interaction, where a is the adjusted radius of the interatomic separation in relation to the radius of the Hydrogen atom:

$$V(r) = \frac{Z_1 Z_2 \epsilon^2}{r} * exp\frac{-r}{a}$$

By summing the above equations, the interatomic potential between two interacting atoms can be estimated [2]. More detail can be found in Was. In practice, understanding the interatomic interactions in an irradiated material is very difficult to understand fully, which is why simulation software can help simplify calculations. In particular, this model fails at intermediate distances (where r is smaller than .1 nm but larger than .01 nm) that describe radiation damage [2]. More complex models attempt to solve the interactions at these distances, but are nearly impossible to solve without adequate simulation software. There are many potential function models that can be analyzed to better understand ion interactions in matter. They are summarized in Was in Table 1.3, reproduced in Figure 2-4.

Potential	**Equation for $V(r) =$**	**Range of applicability**	**Definitions**	**Eq. in text**
Hard-sphere	$0 \quad \text{for} \quad r > r_0$ $\infty \quad \text{for} \quad r < r_0$	$10^{-1} < T < 10^3$ eV	$r_0 =$ Size of atom	(1.46)
Born–Mayer	$V(r) = A\exp(-r/B)$	$10^{-1} < T < 10^3$ eV $a_0 < r \leq r_e$	A, B determined from elastic moduli	(1.47)
Simple Coulomb	$\frac{Z_1 Z_2 \epsilon^2}{r}$	Light ions of high energy $r \ll a_0$		(1.48)
Screened Coulomb	$\left(\frac{Z_1 Z_2 \epsilon^2}{r}\right) \exp(-r/a)$	Light ions $R < a_0$	$a_0 =$ Bohr radius $a =$ Screening radius	(1.49)
Brinkman I	$\frac{Z^2\epsilon^2}{r} e^{(-r/a)} \left(1 - \frac{r}{2a}\right)$	$r < a$	$a \cong \quad a_0/Z^{1/3}$	(1.51)
Brinkman II	$\frac{AZ_1 Z_2 \epsilon^2 \exp(-Br)}{1 - \exp(-Ar)}$	$Z > 25$ $r < 0.7 r_e$	$A = \frac{0.95 \times 10^{-6}}{a_0} Z_{\text{eff}}^{7/6}$ $B = Z_{\text{eff}}^{1/3}/Ca_0$ $C \cong 1.5$	(1.52)
Firsov	$\frac{Z_1 Z_2 \epsilon^2}{r} \chi \left[\left(Z_1^{1/2} + Z_2^{1/2}\right)^{2/3} \frac{r}{a}\right]$	$r \leq a_0$	χ is screening function	(1.56)
TFD two-center	$\frac{Z^2\epsilon^2}{r} \chi\left(Z^{1/3}\frac{r}{a}\right) - \alpha Z + \bar{\Lambda}$	$r < r_b(3a_0)$	$r_b =$ Radius at which the electron cloud density vanishes	(1.57)
Inverse square	$\frac{2E_r}{e}(Z_1 Z_2)^{5/6}\left(\frac{a_0}{r}\right)^2$	$a/2 < r < 5a$	$E_R =$ Rydberg energy $= 13.6$eV	(1.59)

Figure 2-4: Potential function models and ranges [2]. These ranges depend on the types of interactions, energy levels, and the assumptions made about the particle's behavior.

2.2.4 Ionization Energy Loss

In the previous sections, the types of interactions with ions in matter and the resulting energy loss were discussed. The energy loss is primarily due to electronic (e), nuclear elastic (n),and radiation (r) components of the This energy loss. The energy loss per unit length is summarized by:

$$(-\frac{dE}{dx})_{total} = (-\frac{dE}{dx})_n + (-\frac{dE}{dx})_e + (-\frac{dE}{dx})_r$$

However, the radiation component is generally very small so it is neglected in most calculations. Yet it is difficult to solve this equation with potential functions and scattering cross section equations.

The previous sections describe the type of interactions an ion or neutron can experience in matter. The energy loss a particle experiences has been generally described. The stopping power and range of a particle provide more information on energy loss and when a particle will lose all of its kinetic energy.

2.2.5 Stopping Power

To understand the preliminary software used for this simulation, it is important to know basic nomenclature and concepts relating to ions interaction with matter. Ionization energy is not enough to fully describe the energy loss a particle experiences: there must be better ways to find out how far a particle can travel in matter. Stopping Power is the "force" acting upon a charged particle in matter that slows it down. When an ion is incident on matter, the stopping power (-dE/dS) increases until the particle comes to rest [12]. The stopping power is a greater force in particles of generally larger masses, and these particles often slow down almost to rest before experience another collision.

Stopping power can be considered collisional stopping power, radiative stopping power, or electronic stopping power. Collision stopping power is the stopping power that results from from Coulombic interactions with matter and depends largely on the speed and charge (z) of the particle. For proton and other heavy charged particle energies between 2 and 10

MeV, the stopping power is:

$$(-\frac{dE}{ds})_{coll} = \rho z^2 \frac{Z}{A} f(I, \beta)$$

where Z/A is the ratio of the incident materials' atomic number to atomic mass, f(I,β) is a complex function describing the behavior of the specific particle, and β is the ratio of the velocity to the speed of light [12]. Radiative stopping power is the stopping power produced from bremsstrahlung radiation, or the loss of kinetic energy of a particle by photon emission and electron collisions. It is much more difficult to describe the radiative behavior of a particle in a medium. The formula for radiative stopping power is often expressed as:

$$(-\frac{dE}{ds})_{rad} = \rho \frac{N_a}{A} (E + m_e c^2) Z^2 F(E, Z)$$

where F(E,Z) is a complex function depending on the incident material and the particle's incident energy.

While difficult to understand the complexities of these formulas, it is important for this thesis to understand the behavior at high energies and Z. As kinetic energy E increases, the ratio of the radiative to collision stopping power is proportional to ZE. This means that radiative stopping power becomes more important as Z and E increase. Figure [**?**] shows the stopping power vs particle energy for different incident particles.

Electronic stopping power is the stopping power associated with electrons slowing down nuclear fragments. A full derivation of the estimate of electronic stopping power can be found in Was. Figure [**?**] contains the full solution to the nuclear and electronic stopping powers and their associated ranges (discussed in the following section).

How far a particle penetrates a material depends greatly on the stopping power.

2.2.6 Range

Stopping power provides the mean energy loss over a certain length and thus the range that an incident particle travels in a material can be calculated. If one assumes that the stopping power is the averaged energy loss over the particle's path (in reality, the stopping power is not constantly increasing or decreasing over a range), then the range can be expressed as:

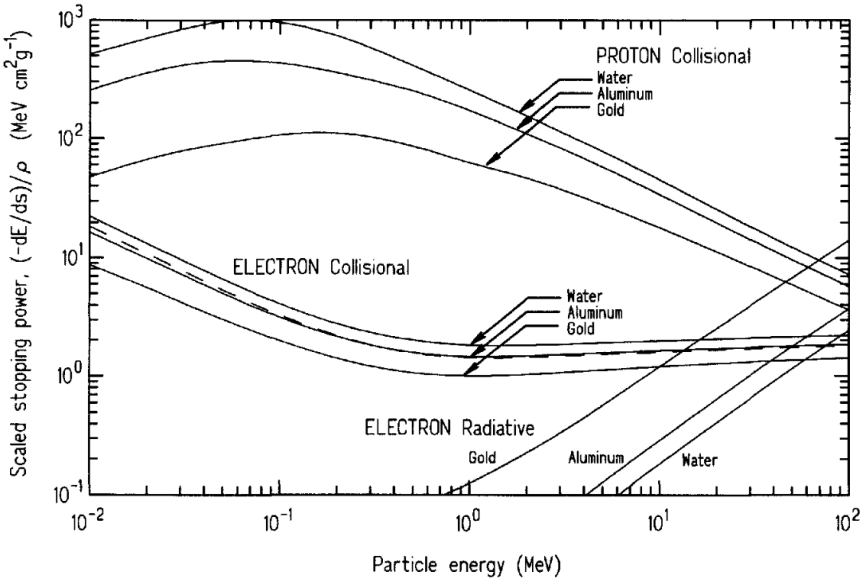

Figure 2-5: The stopping power for different elements. Note the regions where different types of cross section dominate [12]

$$R = \int_0^{E_0} \frac{dE}{(-dE/ds)_{tot}}$$

where $(-dE/ds)_{tot}$ is the total stopping power (radiative, electronic and collisional) and E_0 is the initial energy of the particle [12]. For proton simulations, the electronic and collisional are dominant. There are many ways to approximate the range based on the energy of the particle. One common way is to use the *continuous slowing down approximation* (CSDA) approximation, which assumes that the stopping power is infinite when the energy is 0, and decreases to a known final value linearly. From this approximation, graphs such as the one in Figure 2-7 can be created.

The CSDA equation for protons and electrons in matter is:

$$\rho R = 10^{a+bx+cx^2}$$

Where $x = log_{10}E$ and the constants a,b, and c are given based on the material that the particle is incident on. For fission fragments, the behavior is more complex. This is because the fission fragments can change charge as they propagate through a material. Also, fission

Type of interaction	Nuclear stopping power NS_n		Electronic stopping power NS_e	
High E Coulomb	$\frac{4N\pi Z^4 a_0^2 E_R^2}{E_i} \ln\left(\frac{a^2 E_i^2}{4a_0^2 E_R^2 Z^4}\right)$	(1.134)	$N\pi \frac{Z_1^2 Z_2 \varepsilon^4}{E_i} \frac{M}{m_e} \ln\left(\frac{\gamma_e E_i}{\bar{I}}\right)$	(1.165)
Low E	General expression: $\frac{8.462\times10^{-15} Z_1 Z_2 M_1 S_n(\in)}{(M_1+M_2)(Z_1^{0.23}+Z_2^{0.23})}$	(1.150)	$\left(\frac{dE}{dx}\right)_e = k' E_i^{1/2}$	(1.170)
	Inverse square: $\frac{\pi^2}{4} a^2 N E_a \gamma$	(1.158)	$k' = 3.83 \frac{Z_1^{7/6} Z_2}{M_1^{1/2}\left(Z_1^{2/3}+Z_2^{2/3}\right)^{3/2}}$.	
	Thomas–Fermi screening: $K \frac{Z_1 Z_2}{Z^{1/3}} \frac{M_1}{M_1+M_2}$	(1.162)	$\left(\frac{dE}{dx}\right)_e = kE^{1/2}$	(1.182)
	$Z^{1/3} = \left(Z_1^{2/3}+Z_2^{2/3}\right)^{1/2}$		$k = 8\sigma_e N \left(\frac{m_e}{M_1}\right)^{1/2}$	
	$K = 1.158\pi\varepsilon^2 a_0 = 2.8\times10^{-15}$ eV cm		valid for $0 < E\ (\text{keV}) < 37Z^{7/3}$	

Figure 2-6: Nuclear and Electronic stopping power for different ranges of energies. [2]

fragment size, especially when large, can have a large kinetic energy and take more collisions to slow down. Figure 2-7 shows the energy and range for different sized fission fragments.

These graphs, according to studies described in [12], can be modelled by:

$$\rho R = CE^{2/3}$$

Where C is a constant depending on the incident material's Z. However, this equation has an accuracy of without 10% of the actual range of the particle – an estimate that is often not precise enough when trying to find the range of a particle in matter.

2.3 Heat Transfer

In the previous sections, the physics of particle interactions in matter has been explored. There are many complexities involved in understanding a particle's behavior in matter. As

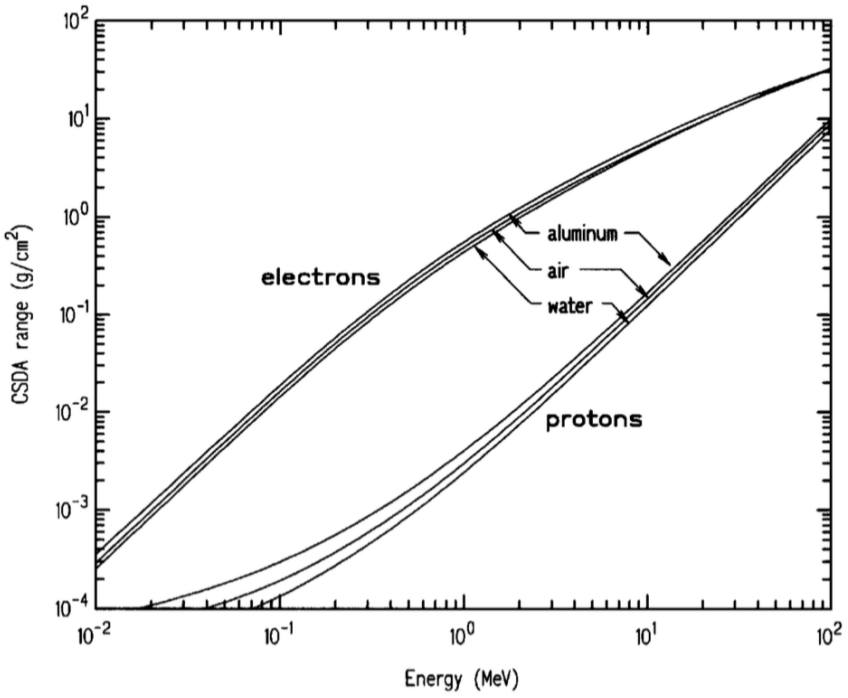

Figure 2-7: The CSDA range based on the incident particle energy for electrons and protons in aluminum, water, and air [12].

particles bombard a material, they affect the temperature of the material. In general, the heat equation that describes the temperature in a material over time is:

$$\frac{du}{dt} - \alpha \nabla^2 u = 0$$

Where u is any function. In heat transfer problems, often u is the temperature and α is the thermal diffusivity [4]. The following sections will discuss the three main modes of heat transfer: convection, conduction, and radiation.

2.3.1 Conduction, Convection, Radiation

In order to analyze heat transfer between two systems, it is useful to categorize the nature of the heat transfer through the physical mechanism that occurs. By identifying the types of heating occuring in a material, the interactions at the surface (the boundary conditions) can be found. This allows for the heat generation equation to be calculated.

Conduction

Conduction is a form of heat transfer where heat is transferred by colliding gas molecules [4]. The conduction heat transfer rate can be expressed as:

$$\dot{Q} = -kA\frac{dT}{dx}$$

Where $\dot{Q}$ is the heat transfer rate, k is the thermal conductivity, A is the cross-sectional area of the material, and dT/dx is the temperature gradient. This is also known as the Fourier Conduction Law, and can be used to describe solids, liquids, and gases [4].

Convection

Heat transfer associated with a moving fluid in contact with a solid is call convection. The heat transfer rate associated with convection is defined by Newton's Law of Cooling:

$$\dot{Q} = h_c A_s (T_s - T_\infty)$$

where h_c is the heat transfer coefficient for convection, T_s is the temperature of the surface of the solid, and T_∞ is the temperature of the fluid far away from the surface [4]. Generally, the convective heat transfer depends on the nature of the flow.

Radiation

Heat transfer by radiation implies heat transfer by some form of wave: generally, this is the heat transfer between a surface and a vacuum of which radiation can "pass through" [4]. This differs from conduction and convection: in conduction and convection, there has to be an interaction between fluids (gas, liquid) and/or a solid. Generally, heat transfer by radiation can be expressed by:

$$\dot{Q} = \varepsilon_1 A_1 (\sigma T_1^4 - \sigma T_2^4) = A_1 h_r (T_1 - T_2)$$

Where $h_r = 4\epsilon_1 \sigma T_m^4$ is the radiation heat transfer coefficient (ϵ is the emissivity and σ is the Stefan-Boltzmann radiation constant) [4].

2.3.2 Volumetric Heat Generation

Beyond knowing just the types of heat transfer, it is important to understand heat generation in a solid. This is often done using properties of conductive heat transfer. The types of heat transfer lead to a temperature gradient. In one dimension (i.e. no 2D surface area), this can be expressed as:

$$q'' = -k\frac{dT}{dx}$$

Where q" is the heat flux normal to the surface [5]. This is also known as Fourier's Law of heat conduction. Yet this only describes 1D flux – and in the real world, the flux is in multiple directions. The full derivation for the heat flux in multiple dimensions can be found in [5]. The general premise of the derivation lies in balancing the heat flux in the x,y, and z (or cylindrical, spherical, etc.) directions and defining q"' as the volumetric heat generation rate. The result is:

$$\bigtriangledown^2 T + \frac{q'''(\vec{r},t)}{k} = \frac{1}{\alpha}\frac{dT}{dt}$$

Where α is the thermal diffusivity and $\bigtriangledown^2$ is the Laplace operator [5]. The Laplace operator can be found based on the geometry of the volume.

The heat generation equation makes it clear that the temperature gradient in a sample is an important factor in heat transfer fundamentals. In a material being bombarded by particles, the temperature gradient is caused by a loss of kinetic energy in the incident particles and the resulting interactions with the atoms in the material. This is why in irradiation damage studies, understanding both heat transfer fundamentals and particle interactions in matter is helps related ion deposition to radiation damage.

There are many simplifications to the heat generation equation, such as the steady state simplification, that make calculations more easy. However, if one wants a more accurate calculation of the temperature change in a material due to particle irradiation, simulation methods such as the Finite-Element Method can be used.

2.4 Finite-Element Method

Finite-Element Method (FEM) frameworks assist in developing multiphysics simulations. This is incredibly powerful in reducing the computation time while incorporating many physical phenomena into a simulation. The FEM approximates partial differential equations by creating functions made up of smaller "shape functions" with determined coefficients [1]. FEM works by converting a function to its "weak form", which essentially makes the differential equation easier to solve. To create the weak form of a function, there are multiple steps that must be taken. Fig ?? shows a conversion of a complicated function to its weak form.

1. $-\nabla \cdot k\nabla u + \boldsymbol{\beta} \cdot \nabla u = f$
2. $-\nabla \cdot k\nabla u + \boldsymbol{\beta} \cdot \nabla u - f = 0$
3. $-\psi\left(\nabla \cdot k\nabla u\right) + \psi\left(\boldsymbol{\beta} \cdot \nabla u\right) - \psi f = 0$
4. $-\int_{\Omega} \psi\left(\nabla \cdot k\nabla u\right) + \int_{\Omega} \psi\left(\boldsymbol{\beta} \cdot \nabla u\right) - \int_{\Omega} \psi f = 0$
5. $\int_{\Omega} \nabla\psi \cdot k\nabla u - \int_{\partial\Omega} \psi\left(k\nabla u \cdot \hat{\boldsymbol{n}}\right) + \int_{\Omega} \psi\left(\boldsymbol{\beta} \cdot \nabla u\right) - \int_{\Omega} \psi f = 0$
6. $\underbrace{\left(\nabla\psi, k\nabla u\right)}_{\text{Kernel}} - \underbrace{\langle\psi, k\nabla u \cdot \hat{\boldsymbol{n}}\rangle}_{\text{BC}} + \underbrace{\left(\psi, \boldsymbol{\beta} \cdot \nabla u\right)}_{\text{Kernel}} - \underbrace{\left(\psi, f\right)}_{\text{Kernel}} = 0$

Figure 2-8: Steps to convert a functon to its weak form [1].

The steps involved are:

1. Write down the PDE
2. Rearrange so all of the terms are on the left side of the equation.
3. Choose a test function (ψ) and multiply the original equation by it.
4. Integrate the test function over the range Ω
5. Integrate the test function over the range, which creates kernels and boundary conditions.
6. Analyze this new equation using a FEM framework.

FEM framework is essential in approximating complicated heat transfer differential equa-

tions. The power of FEM lies in its ability to take complicated partial differential equations, break them down into kernels and boundary conditions, and approximate the solution over time with very little error.

One benefit of FEM over other analysis methods, such as the Finite-Difference Method, is that it creates continuous functions over the domain of the function. Most software that uses FEM uses Newton's method to approximate over an interval [9]. This helps approximate the value of a function by taking its derivative, "guessing" a solution, and finds the difference between the guessed root and the function over its derivative. While this can be accurate, new methods have been used in programs such as MOOSE. MOOSE uses the Jacobian Free Newton-Krylov Method, which better approximates nonlinear equations [9]. While the details of these improved methods are not examined in this thesis, it is important to note the improvements that continue to be made to FEM and other analysis methods. As these methods become more and more accurate, the reliability of simulation software increases.

2.4.1 Solving the Heat Equation

In a system with many components, it is incredibly difficult to solve complicated heat transfer functions. However, there are a few ways one can define the system in order to be solved. The use of *kernels*, or small pieces of physics, helps simplify the calculations. Kernels can also be the differential equations that represent the physics of the problem [1]. Boundary conditions are known states of a system at certain boundaries. This can be a function or a value, such as the temperature on a surface. In the case of heat generation, the boundary conditions are a residual sometimes coupled with a Jacobian [1]. In this thesis, the boundary conditions are based on conduction and convection heat transfer at the interface between the sample and the coolant, as well as the coolant and the air.

2.5 Effects of Radiation Damage in Materials

Ion interactions in matter and the resulting heat generated can change a material's properties. Three changes a material experiences are hardening, increased brittleness, and thermal creep [2]. While this project does not quantify this damage, these changes happen in

irradiated materials which serves as the motivation for this study. For full quantitative descriptions of the materials property changes that result from irradiation, see Was chapters 12-14 [2].

Irradiated metals experience increased hardening due to the defects and dislocations that are created in the material as a result of atom displacements [2]. When ions collide with atoms in a material, some of the resultant energy is converted to heat. This excited other ions. When their energy is high enough, they can break from the uniform lattice and form defects [2]. While this leads to an increase in yield strength, the ductility of the metal decreases. In stress-strain analysis, this means a material can better resist deformation when a compressive force is applied. However, by losing ductility, it is less able to absorb an applied load [2].

By losing some ductility, an irradiated material also becomes more brittle. Increased brittleness is due to the same phenomena as increased hardness: the defects that arise as a result of radiation of a material and the resulting temperature increase. However, when a material becomes more brittle, it will generally fracture instead of absorb forces acting upon it [2]. In stress-strain analysis, this means that when a stress is applied to a material, it will have very little elastic deformation before it cracks.

Creep results from time-dependent deformation from an applied stress [2]. Creep in particular is very temperature dependent, because temperature changes in a material over a sustained period of time change its material properties. When a material experiences radiation damage, the material heats up, atoms vibrate and collisions cause them to change formation [2]. Over sustained periods of time, this causes a material to be reshaped and creep occurs. Thus, when studying radiation damage in materials, it is useful to know the temperature gradient across the sample as a result of the ion interactions.

The simulation method described in this thesis aims to automate radiation damage studies by finding the temperature gradient across an irradiated sample and finding geometries that minimize this temperature gradient. It serves as a first step to quantitatively understanding the effects of radiation damage on materials using automated simulations.

Chapter 3

Methodology

This thesis uses multiphysics simulations with specific boundary conditions coupled with radiation damage information taken from range data in particle simulation software. This section will explain the basics behind each program used (SRIM, Cubit, and MOOSE) and the overall structure of the program.

3.1 SRIM

SRIM, Stopping Range of Ions in Matter, is used to simulate the stopping range of ions in different materials. SRIM allows the user to simulate the irradiation damage of a beam in layers of materials. Once this profile is found, it can be imported into a multiphysics simulation to find heat generation in the sample. For this project, SRIM will be used to remove conditions for which the proton beam did not fully penetrate the sample and to record information on the interactions of the proton beam in the sample and coolant. SRIM ultimately provides the range of ions in matter and the resulting irradiation damage by using monte-carlo simulations.

For this thesis, the SRIM files were run and analyzed in three steps, shown in 3.1. The first three files are related to the SRIM input and automation files. The second step is the data acquisition step, with output text files of data. The third step sorts range data and eliminates range files that do not fit the user-defined criteria (described later).

The next sections will go in depth into the use of SRIM, how it was automated, and

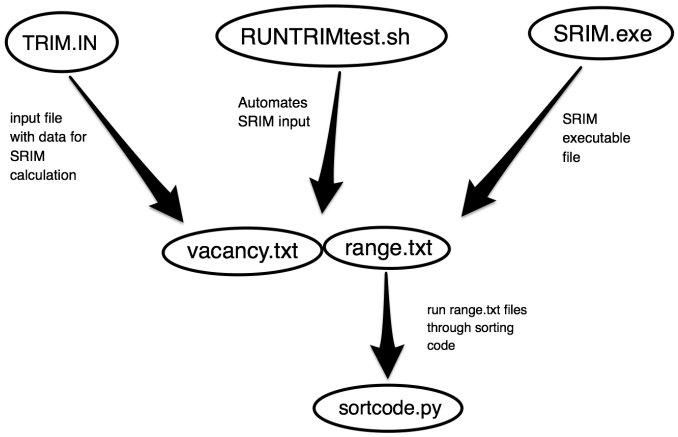

Figure 3-1: Relevant SRIM programs and their order of use for this project.

criteria for "sortcode.py".

3.1.1 SRIM Interface

SRIM provides a variety of input options for the user. Ions of energies from 10eV-2GeV can be simulated. SRIM lets the user choose the type of ion to irradiate, the number of ions to test, and the energy of these ions. SRIM also allows the user to create a 1-dimensional slab that represents a cross-section of the desired geometry. Multiple layers and materials can be used. Figure 3.1.1 shows the opening window for SRIM when the user runs SRIM.exe and the possible inputs.

For this study, 25,000 protons were sampled for each automated SRIM run. SRIM allows the user to indicate what models for ion displacement are. Based on the satisfactory results from a prior investigator, the "Kinchin-Pease" method was chosen [11]. The output window for a SRIM calculation in progress is shown in Figure 3.1.1. There are many options of what output images you can view. For example, this screen shows the ions moving in the XY direction. Information about the layer materials and widths (in Angstroms) can also be viewed. The right side of the screen shows various radiation damage properties in the material.

The data output in SRIM.exe is saved to two files: range.txt and vacancy.txt. The range file gives ions and their penetration distance in the material. The vacancy file gives some of the radiation damage information.

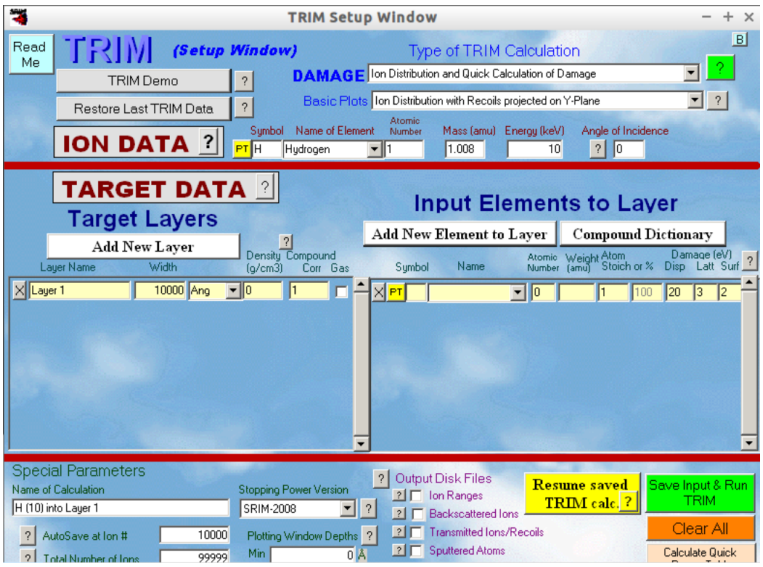

Figure 3-2: Input dialogue when SRIM.exe is run. The incident particle type, layer thicknesses, layer materials, damage analysis methods, and number of ions to sample can be specified in this window.

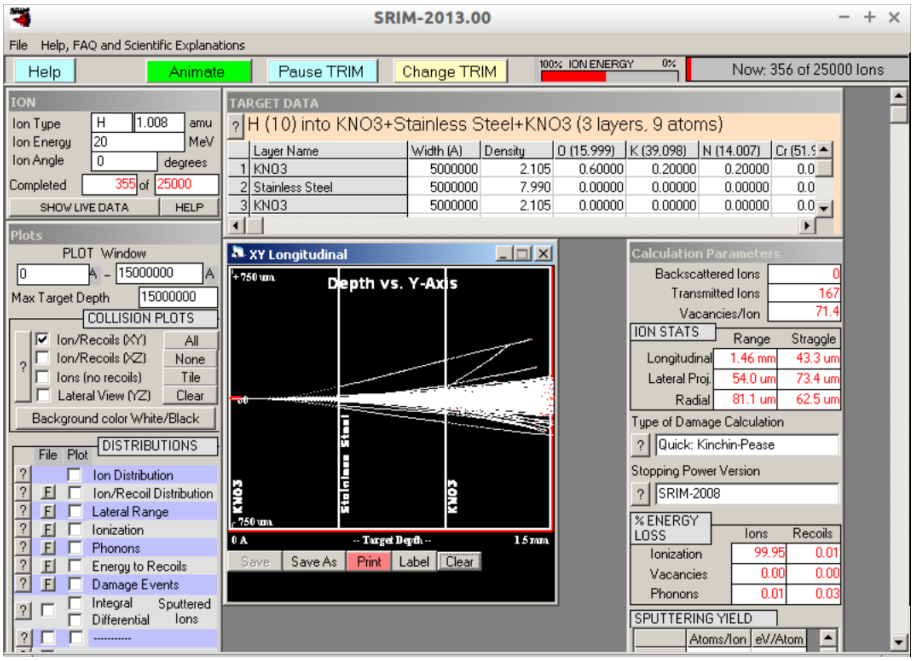

Figure 3-3: Output screen when SRIM is running.

3.1.2 SRIM Automation

SRIM.exe does not allow for explicit automation. However, for this project, a variety of sample and coolant materials and thicknesses was to be tested. Inputting all of this information by hand is time-consuming and difficult to replicate, so automation code was written

using the SRIM-provided TRIM.IN input file and RUNTRIMtest.sh created for this project.

TRIM.IN is the input file that stores all of the information the user enters in SRIM.exe. This includes the energy of the incident particles and the layer thicknesses. For this project, stainless steel was used as the "sample" and KNO_3 was used as the coolant. Thus the intermediate file TRIM-Script-sample-SS316.IN was chosen to contain the variables SAMPTHICK, COOLTHICK, and ENERGY. RUNTRIMtest.sh is a bash file that automates the replacing of the three variables in TRIM-Script-sample-SS316.IN and renaming this file TRIM.IN. Then, RUNTRIMtest.sh iterates through different values for these variables and runs SRIM (wine is an Unix program used to allow a user to run a Windows program). Figure 3.1.2 shows the source code of the bash file RUNTRIMtest.sh.

```
#!/bin/bash

for energies in 20000 30000 40000  ## energies in keV
do
    for j in 5000000 10000000 15000000  20000000 25000000 30000000  ##sample thickness in A
    do
        for k in 5000000 10000000 15000000  20000000 25000000 30000000  ##coolant thickness in A
        do
            sed "s/<BEAMENERGY>/$energies/"  TRIM-script-SS316-KNO3.IN > TRIM1.IN
            sed "s/<SAMPTHICK>/$j/" TRIM1.IN > TRIM2.IN
            sed "s/<COOLTHICK>/$k/" TRIM2.IN > TRIM.IN
            wine TRIM.exe
            touch vacancy_files_SS316_KNO3/VAC-SS316COOLANT-CTHICK-"$k"-SAMPLETHICK-"$j"-E-"$energies".txt
            cp VACANCY.txt vacancy_files_SS316_KNO3/VAC-SS316COOLANT-CTHICK-"$k"-SAMPLETHICK-"$j"-E-"$energies".txt
            touch range_files_SS316_KNO3/RANGE-SS316COOLANT-CTHICK-"$k"-SAMPLETHICK-"$j"-E-"$energies".txt
            cp RANGE.txt range_files_SS316_KNO3/RANGE-SS316COOLANT-CTHICK-"$k"-SAMPLETHICK-"$j"-E-"$energies".txt
        done
    done
done
```

Figure 3-4: Bash file to automate TRIM.IN file creation.

For this particular project, high energy protons were used, ranging from 20 to 40 MeV. Sample and coolant thicknesses ranged from 0.5 mm to 3 mm, which led to 108 possible iterations. For future investigations, more variables could be added to the SRIM automation code to allow more variation in the TRIM.IN input code.

3.1.3 Choosing SRIM Files

After successfully running 108 iterations of SRIM, files had to be sorted based on certain criteria. The output RANGE.txt file stores the range of ions in the layers. There are many ways the files could be sorted. For this project, files were sorted according to whether or not they penetrated the Stainless Steel sample. One purpose of this project is to simplify the process of simulating high-energy particles (radiation) damage in materials for different

geometries. If a beam did not successfully penetrate the sample of interest, then it was not analyzed.

To visualize this, see Figures 3.1.3 and 3.1.3. Figure 3.1.3 shows different 30 MeV protons penetrating .5mm layers. All of the protons pass through the stainless steal sample. However, the 20 MeV protons in 3.1.3 barely penetrate the stainless steel sample (if at all). Runs for which ions did not fully penetrate the stainless steel sample were removed from the further analysis.

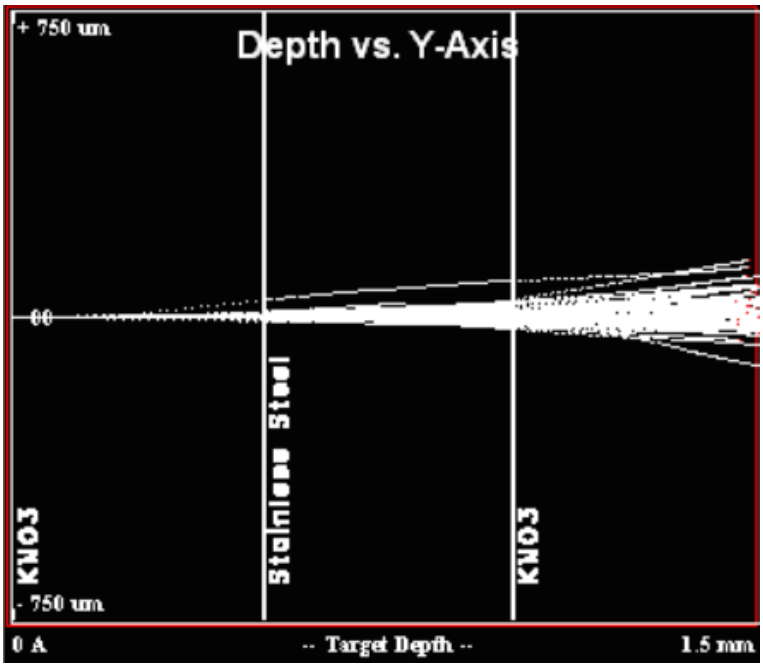

Figure 3-5: Energy deposition for a 20 MeV proton.

To sort the data, sortcode.py was written. This python script loaded the data from RANGE.txt, made the range data into readable numbers, and compared this to the sample and coolant thicknesses. The RANGE.txt file has three columns: the depth of the layers, how many ions of the 25,000 tested were stopped in that layer, and the recoil distribution (recoil distribution not important for this analysis). If there were no ions that stopped after halfway through the sample, then the file should be ignored for later analysis. To actually do this, a counter in sortcode.py increased if there were ions that stopped after the halfway point in the sample. If there are not enough depths after the halfway depth that contained ions, the RANGE.txt file is deleted.

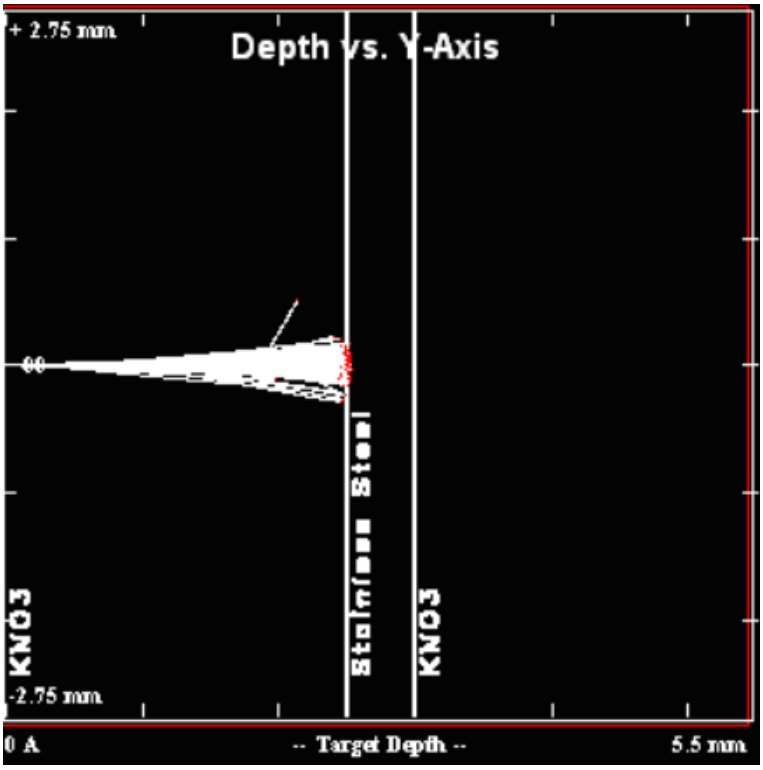

Figure 3-6: Energy deposition for a 20 MeV proton with high thicknesses.

3.2 Cubit

Cubit is a program developed by Sandia National Labs that serves as a geometry and mesh generator. Cubit was used to represent the geometries used in SRIM for use in MOOSE. Similar to the SRIM automation techniques, a bash file was written that reads the RANGE.txt file titles to gather the coolant and sample thicknesses, then created a 10 mm by 10 mm square with the corresponding coolant or sample thickness to be imported into MOOSE.

3.2.1 Cubit Automation

The Cubit automation used for this simulation is similar to that written for SRIM automation and can be found in Appendix B. Cubit files are ".jou" (journal) files. The Cubit GUI allows for the physical creation of the mesh, just as any CAD program works. Like TRIM.IN, a cubit journal file can be coded and imported into Cubit. The automation script reads the remaining RANGE.txt files that exist after the sorting algorithm removes files, finds their coolant and sample thicknesses, then replaces a journal file with these properties.

Then, the file is meshed. These files are then stored in a separate folder.

3.3 MOOSE

MOOSE is a program developed by Idaho National Laboratory that stands for Multiphysics Object-Oriented Simulation Environment. MOOSE is a finite-element framework that assists in developing multiphysics simulations. To use MOOSE, a meshed geometry is first imported, and a material block (where the materials properties are declared) is created. Kernels contain the partial differential equations (PDEs) used to solve the physics problems. The software allows users to visualize results in real-time for applications in reactor core design and other technology. The interface between the different MOOSE inputs is shown in Figure [**?**].

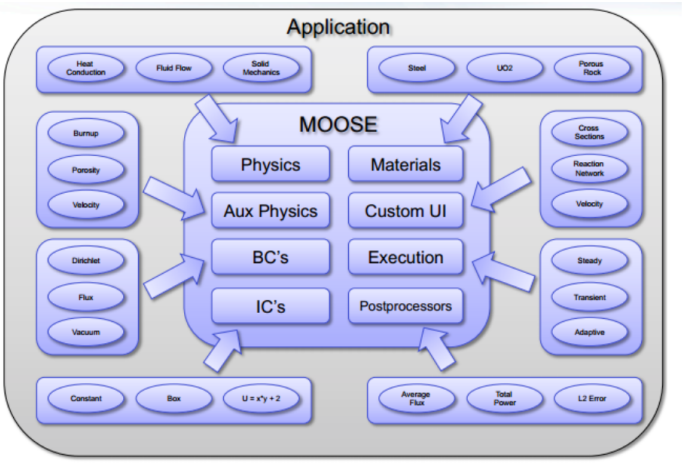

Figure 3-7: The general structure of a MOOSE-based program [1].

The program, for the purposes of this project, allows the user to solve problems in heat conduction. From here, information on the heat conduction in the sample and coolant can be found. The MOOSE software greatly decreases the time it takes to solve these complicated heat transfer scenarios. The beam energy and radius for the irradiation beam is defined in the MOOSE file, and the output gives the temperature distribution across a specific geometry.

3.3.1 Interface between programs

By coupling SRIM and MOOSE, the user should be able to define sample geometries and materials, import this Cubit file, and generate a profile of the temperature gradient through the sample. Figure 3-8 shows the basic inputs and outputs of each of these programs.

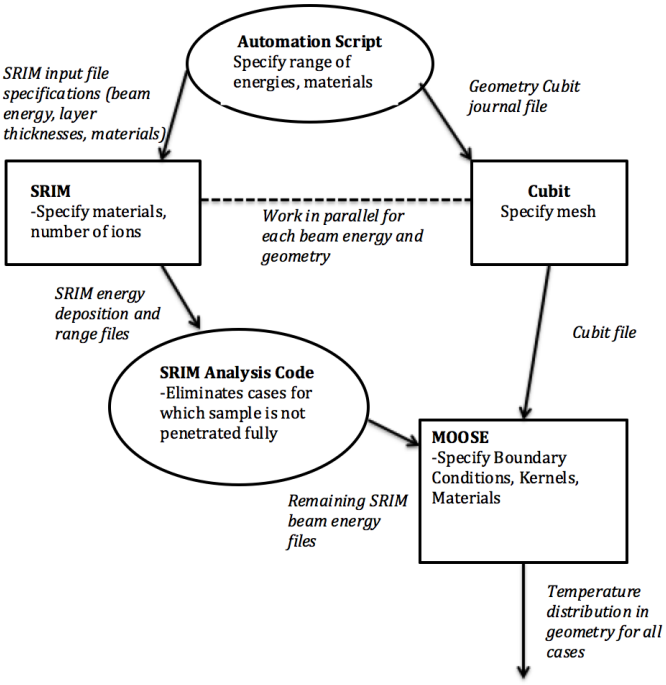

Figure 3-8: Chart of the connections between SRIM, Cubit, and MOOSE.

Most of the MOOSE inputs had been defined by Jamie Sahote, a previous investigator for the project [**?**]. This includes materials property inputs, boundary conditions, and kernels (Mike– should I just cite his thesis here? I can also go into all of the detail in this thesis, and cite his work, but it's already written in his thesis. I want to focus a bit more on the actual automation part of MOOSE). The outline of the Sahote code (automated for this thesis) is shown in Figure 3-9.

3.3.2 Previous MOOSE work

Prior work on this simulation was done by Jamie Sahote [11]. The boundary conditions, kernel information, materials properties, and logic behind these choices can be found in his thesis [11]. Sahote's code takes the information from SRIM, makes it into a 3D beam, and

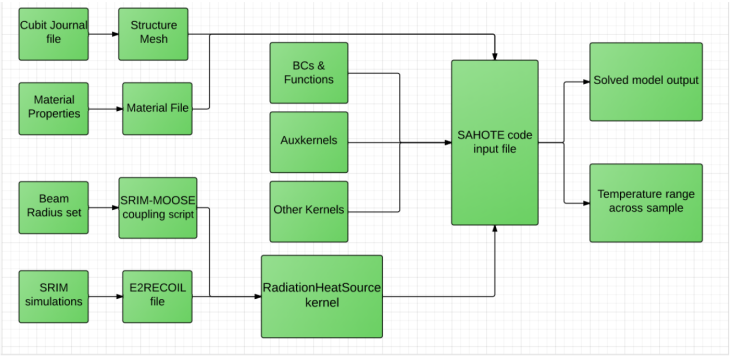

Figure 3-9: The general structure of a MOOSE-based program [11].

is then used for analysis in MOOSE.

For this simulation, the heat generation equation described in Section 2.3.2 is solved in MOOSE. The boundary conditions are the coolant boundary conditions that relates the heat transfer through the KNO_3 coolant to its heat transfer coefficient as well as heat transfer boundary conditions that result from the air around the coolant and sample [11].

Sahote also included the materials needed for the simulation in this thesis. This included heat transfer properties and the density of materials, including 316 stainless steel and KNO_3. The values were confirmed to be correct for this analysis.

3.3.3 MOOSE Automation

MOOSE automation was similar to that of SRIM automation, only more components had to be included. At the time that this thesis was written, no large-scale MOOSE automation simulations had been published or proven to work effectively. Because of this, it was unclear whether or not MOOSE and its gui, Peacock, have the capabilities to support the simulation of 60 files in succession.

There are many challenges in coupling SRIM and Cubit to MOOSE. One challenge is the vast amount of files that exist and matching the corresponding SRIM range and ionization files to Cubit files. This was done in a bash script by determining the coolant thickness, the sample thickness, and the energy of each range file in SRIM and matching it to the corresponding ionization file. For simplicity, the final shell script that found the ionization

files also created the mesh for Cubit. This file was not saved and was automatically imported into the MOOSE run file, SahoteTestLocal.i. See Appendix C for this file.

The ion energy file was then automatically entered into the 2D to 3D file (or SRIM to MOOSE) and the entire program was ready to run. It is important to note here that this *only* works for the specific materials and geometry used in this thesis.

Chapter 4

Results

4.1 Simulation Results

The automation of the SRIM and Cubit code worked, however, the automation of the MOOSE code was unsuccessful. Discussion of what the potential errors in the MOOSE code were can be found in the Discussion section.

The runtime of the SRIM code for 25,000 ions was on average 5 minutes per simulation. After running the SRIM automation and sorting, fewer files remained from the original 108. On first iterations of the SRIM sorting algorithm, only files of a particular energy were analyzed. This issue was never resolved, it sorted correctly with range files of the same energies. In total, 68 files passed the python tests.

4.2 Optimal Current for 5K Temperature Gradient

Although the calculations for temperature gradient across the sample due to a particular current could not be calculated for MOOSE, estimations for the heat generation in the sample and thus the temperature gradient was found using stopping power and range to find the heat generated, and then relating this to the temperature gradient across the sample through conduction. From 2.2.5, the stopping power is split into different components.

SRIM provides the user with the collisional and radiative stopping power (in MeV/m) in the IONIZATION.txt and E2RECOIL.txt output files. The range of the particles is shown in RANGE.txt, and the heat generation is the product of the stopping power, range, and current divided by the charge of a proton. The current range was 10^{-9} mA to .1 mA in accordance with the current chosen by Jamie Sahote for his original MOOSE code [11].

Results for the normalized number of ions per distance and the resulting temperature gradient as a function of current for 20 MeV, 30 MeV, and 40 MeV protons for KNO_3 thickness of 0.5 mm and SS316 thickness of 1.5 mm are shown in Figures 4-1 and 4-2, respectively.

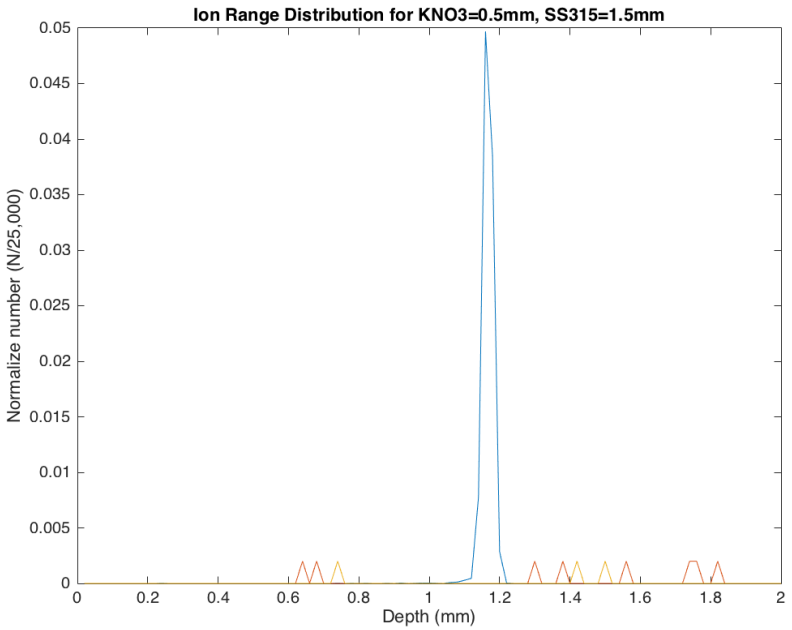

Figure 4-1: The normalized number of particles per distance in sample. Note that the lower number of particles in the 30 MeV and 40 MeV beams is due to most of the particles not stopping in the sample.

Since the 30 MeV and 40 MeV have very little energy deposition in the sample, there is no optimal beam current that can be found. For the 20 Mev energy, the beam current that keeps dT/dx below 5 K is 0.0027 mA.

Results for the normalized number of ions per distance and the resulting temperature gradient as a function of current for 20 MeV, 30 MeV, and 40 MeV protons for KNO_3 thickness of 3.0 mm and SS316 thickness of 1.5 mm are shown in Figures 4-3 and 4-4, respectively.

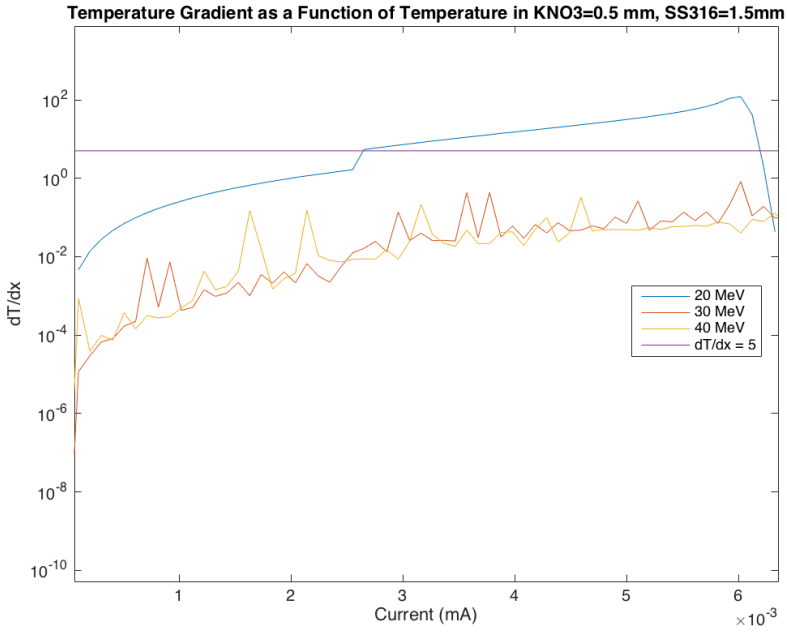

Figure 4-2: Temperature gradient as a result of particle beam current.

Now, with the increasing coolant thickness, there is greater particle deposition of the 30 MeV and 40 MeV protons. The optimal current to minimize the temperature gradient below 5 K for the 20 MeV, 30 MeV, and 40 MeV proton beams are .07 mA, .09 mA, and .2 mA, respectively.

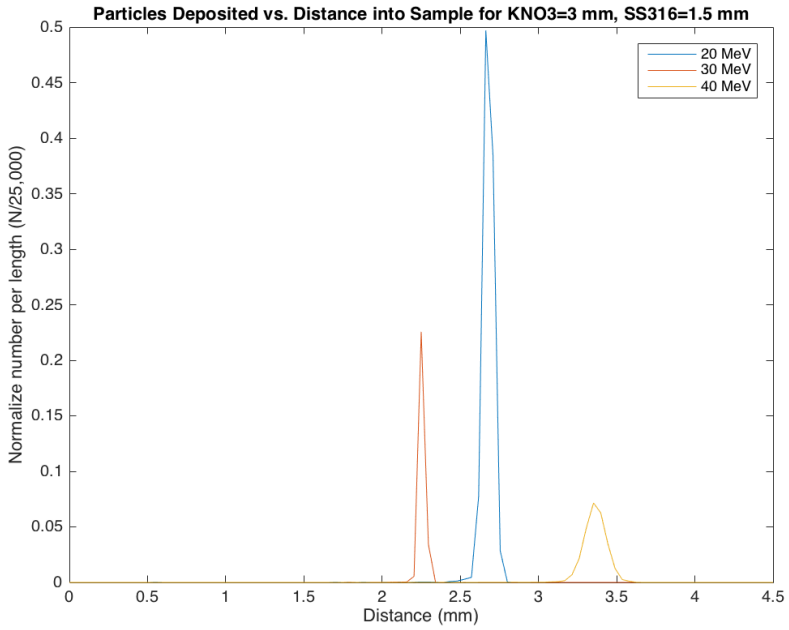

Figure 4-3: The normalized number of particles per distance in SRIM output of KNO_3 thickness of 3.0 mm and SS316 thickness of 1.5 mm.

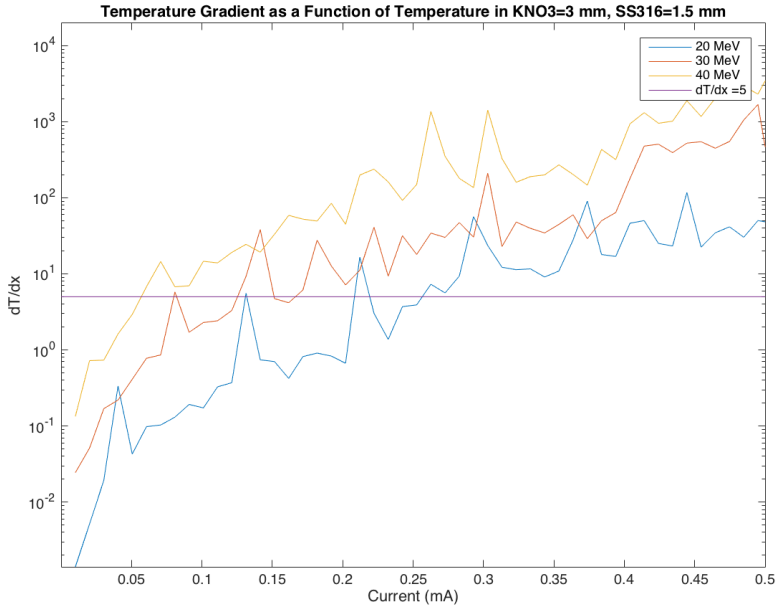

Figure 4-4: Temperature gradient as a result of particle beam current for KNO_3 thickness of 3.0 mm and SS316 thickness of 1.5 mm.

Chapter 5

Discussion

The results below analyze the success of automated simulation attempts in SRIM, Cubit, and MOOSE.

5.1 SRIM

The general trends and conclusions are below for each energy:

20 MeV protons: Very few of these configurations passed the tests. Generally, geometries with a coolant thickness greater than 1 mm and/or sample thicknesses greater than 1.5 mm did not contain ions that penetrated the sample beyond the halfway point.

30 MeV protons: Almost all of the SRIM files contained ions that penetrated more than halfway through the samples. Only the coolant thicknesses of 2.5 mm and above and sample thicknesses of 2.5 mm or above did not pass.

40 MeV protons: All of the 40 MeV protons penetrated at least halfway through the sample.

These results indicate that the energy of the protons is an important factor in whether or not the protons penetrate the sample. In future testing, protons with energies greater than 20 MeV proton should be used with thicker samples, but lower energy particles should

be used with geometries containing a thinner coolant layer. Smaller thicknesses could also be tested with lower energies.

Optimal Current Analysis

Due to the 25,000 proton simulations instead of 100,000, and the 100 bins for range calculations, the results were very noisy for current. However, the results indicate that an optimal proton beam current should be on the .1 mA scale, and that higher energy protons can have a higher threshold current. Also, the results indicate that for the sample and coolant thicknesses, a small coolant thickness allows for many higher-energy ions to pass through the material. Thus, for thinner geometries, greater than 20 MeV proton beams should be avoided.

5.2 Cubit Automation

The Cubit automation successfully created a file with the correct dimensions based on the remaining SRIM files that were sorted. Since the geometry and mesh were simple squares, the Cubit file generation step ran quickly.

5.3 MOOSE Automation

Entirely automating the MOOSE code proved unsuccessful due to a few different potential factors:

1) The MOOSE Framework GUI, Peacock, does not work well running multiple simulations. Trying to run multiple iterations of the same files with the Peacock GUI was unsuccessful. Peacock was developed through MOOSE and therefore might not have the capabilities that other visualization software has. Like SRIM, it is not possible to run two simulations in parallel– it must be done by manually selecting the number of threads to run. There is no easy way to edit the Peacock GUI within the scope of this

project.

2) There are *a lot* of files. The SRIM, Cubit, SRIM to MOOSE, Cubit to MOOSE, and MOOSE codes all occupy a lot of memory and take a non-negligible time to generate output files. The final shell file that combines SRIM, MOOSE, and Cubit is slow and could lead to a lag in the file creation for MOOSE input. Additionally, it would be impractical and take up even more memory to try and compile and run each simulation as a separate MOOSE directory.

3) MOOSE is very user-friendly. While this seems counter intuitive, MOOSE's user-friendliness, excellent documentation, and troubleshooting support from its creators makes it unnecessary to understand the intricacies of the input files. This is because MOOSE is designed for users who do not necessarily have a strong computer science background. SRIM, in comparison, has very little documentation but users have access to the basic input files. While learning the intricacies of every SRIM file without documentation is difficult, it allowed for more control over the input simulation properties.

Chapter 6

Conclusion and Future Work

6.1 General Conclusions

While the MOOSE automation script was ultimately unsuccessful, the SRIM and Cubit automation was successful. This could serve as the basis for future automated damage studies in materials. The SRIM sorting results indicate a strong dependence on irradiation beam energy when understanding the resulting damage in reactor materials. The optimal current for the 20-40 MeV samples was on the order of .01 mA, with an increasing optimal current with increasing beam energy. This indicates a strong relationship between coolant thickness, current, and heat generation in nuclear materials.

The SRIM automation outputs provided an estimate for the optimal beam current. Using an FEM solver such as MOOSE, however, would produce more accurate and readable results.

6.2 Future Work

The SRIM simulation ran satisfactorily, however, there is room for improvement. The program currently runs SRIM analysis then uses that geometry to generate a Cubit file. However, this is not practical for widespread applications. Ideally, a user should be able to input any Cubit geometry, and RUNTRIMtest.sh would read the thickness of the Cubit geometry layers into TRIM.IN. This would eliminate the manual entering of the thicknesses

into TRIM.IN the first time the simulation is run.

The second problem with the SRIM simulation code is that is it not widely accessible and usable. With more time, a better user interface could be created that does not require the user to understand and implement many different files. Since the origins of this thesis project, a new 3D monte-carlo radiation damage code has been developed by researchers at MIT [13]. This program, called IM3D, could be used in place of SRIM due to its easier-to-use framework. It also has the potential to eliminate the need for a Cubit input file into the program.

As MOOSE continues to develop, it will become more and more powerful as a simulation tool. Future investigators on this project should try alternative MOOSE automation scripts. Two suggested options are:

Directly input the SRIM output into the SRIM to MOOSE code. Instead of running the SRIM automation separately from the MOOSE automation, a potential solution could be to run one SRIM test run and input it directly into MOOSE. Sorting code would have to be built in to the MOOSE automation in this option.

Automate the creation of the MOOSE input files and save them each as different repositories. If each simulation was its own "program", many of the issues with the Peacock GUI could be solved. However, this would take up a significant amount of computer memory and take a long time to compile 70+ MOOSE-based programs.

Appendix A

SRIM input files

A.1 SRIM Input File

==> SRIM-2013.00 This file controls TRIM Calculations.
Ion: Z1 , M1, Energy (keV), Angle,Number,Bragg Corr,AutoSave Number.
1 1.008 <BEAMENERGY> 0 10000 1 10000
Cascades(1=No;2=Full;3=Sputt;4-5=Ions;6-7=Neutrons), Random Number Seed, Reminders
1 0 0
Diskfiles (0=no,1=yes): Ranges, Backscatt, Transmit, Sputtered, Collisions(1=Ion;2=Ion+Recoils), Special EXYZ.txt file
0 0 0 0 0 0
Target material : Number of Elements & Layers
"H (10) into KNO3+Stainless Steel+KNO3 " 9 3

```
PlotType (0-5); Plot Depths: Xmin, Xmax(Ang.) [=0 0 for Viewing
   Full Target]
       0                              0          2.5E+07
Target Elements:     Z    Mass(amu)
Atom 1 = O =         8   15.999
Atom 2 = K =        19   39.098
Atom 3 = N =         7   14.007
Atom 4 = Cr =       24   51.996
Atom 5 = Fe =       26   55.847
Atom 6 = Ni =       28    58.69
Atom 7 = O =         8   15.999
Atom 8 = K =        19   39.098
Atom 9 = N =         7   14.007
Layer   Layer Name /                    Width Density      O(8)   K
   (19)    N(7)   Cr(24)   Fe(26)   Ni(28)    O(8)   K(19)    N(7)
Numb.   Description                     (Ang) (g/cm3)     Stoich
   Stoich  Stoich  Stoich  Stoich  Stoich  Stoich  Stoich  Stoich
 1       "KNO3"              <COOLTHICK>  2.105       .6       .2
    .2       0       0       0       0       0       0
 2       "Stainless Steel"             <SAMPTHICK>  7.99        0
          0       0      .08     .74     .18       0       0
          0
 3       "KNO3"              <COOLTHICK>  2.105       0       0
          0       0       0       0      .6       .2       .2
0  Target layer phases (0=Solid, 1=Gas)
0 0 0
Target Compound Corrections (Bragg)
 .9756683   1   1
Individual target atom displacement energies (eV)
      28      25      28      25      25      25      28      25
              28
Individual target atom lattice binding energies (eV)
```

3 3 3 3 3 3 3 3
3
Individual target atom surface binding energies (eV)
2 .93 2 4.12 4.34 4.46 2 .93
2
Stopping Power Version (1=2011, 0=2011)

A.2 SRIM Automation Code

```
#!/bin/bash
for energies in {10000..40000..5000}  ## energies
do
for j in {0.5..3..0.5}  ##sample thickness
do
for k in {0.5..3..0.5} ##coolant thickness
    do
        sed "s/<BEAMENERGY>/$energies/"  TRIM-Script-sample-SS316
           .IN >
        TRIM1.IN
        sed "s/<SAMPTHICK>/$j/"
        TRIM1.IN > TRIM2.IN
        sed "s/<COOLTHICK>/$k/" TRIM2.IN > TRIM.IN
            wine TRIM.exe
            touch vacancy_files/VAC-SS316COOLANT-CTHICK-"$k"-
               SAMPLETHICK-"$j"-E-"$energies".
            txt
            cp VACANCY.txt vacancy_files/VAC-SS316COOLANT-
            CTHICK-"$k"-SAMPLETHICK-"$j"-E-
            "$energies".txt
            touch range_files/RANGE-SS316COOLANT-CTHICK-"$k"-
               SAMPLETHICK-"$j"-E-"$energies"
            .txt
            cp RANGE.txt range_files/RANGE-SS316COOLANT-
            CTHICK-"$k"-SAMPLETHICK-"$j"-E-
            "$energies".txt
        done
    done
done
```

A.3 Example SRIM range output file

```
======= H (10) into KNO3+Stainless Steel+KNO3 =======
          SRIM-2013.00
=================================================

       ION and final RECOIL ATOM Distributions
  See  SRIM Outputs\TDATA.txt for calculation details
=================================================

-------------------------------------------------------------------------

     Recoil/Damage Calculations made with Kinchin-Pease Estimates
-------------------------------------------------------------------------


-------------------------------------------------------------------
   There are NO RECOILS unless a full TRIM calculation is made.
-------------------------------------------------------------------

See file : SRIM Outputs\TDATA.txt for details of calculation
 Ion     = H    Energy = 20000 keV
============= TARGET MATERIAL
    ======================================

Layer  1 : KNO3
Layer Width =      5.E+06 A   Layer # 1- Density = 6.269E22 atoms/
    cm3 = 2.105 g/cm3
   Layer # 1-  O = 60    Atomic Percent = 47.4 Mass Percent
   Layer # 1-  K = 20    Atomic Percent = 38.6 Mass Percent
   Layer # 1-  N = 20    Atomic Percent = 13.8 Mass Percent
Layer  2 : Stainless Steel
Layer Width =      5.E+06 A   Layer # 2- Density = 8.584E22 atoms/
    cm3 = 7.99 g/cm3
   Layer # 2- Cr = 8     Atomic Percent = 7.42 Mass Percent
   Layer # 2- Fe = 74    Atomic Percent = 73.7 Mass Percent
   Layer # 2- Ni = 18    Atomic Percent = 18.8 Mass Percent
```

Layer 3 : KNO3

Layer Width = 5.E+06 A Layer # 3– Density = 6.269E22 atoms/ cm3 = 2.105 g/cm3

Layer # 3– O = 60 Atomic Percent = 47.4 Mass Percent

Layer # 3– K = 20 Atomic Percent = 38.6 Mass Percent

Layer # 3– N = 20 Atomic Percent = 13.8 Mass Percent

Total Ions calculated =4999.98

Ion Average Range = 146.7E+05 A Straggling = 458.9E+03 A

Ion Lateral Range = 520.0E+03 A Straggling = 723.5E+03 A

Ion Radial Range = 824.3E+03 A Straggling = 621.0E+03 A

Transmitted Ions =2344; Backscattered Ions =

(These are not included in Skewne– and Kurtosis below.)

Range Skewne– = −009.6329 &= [−(X–Rp)^3]/[N*Straggle^3]

Range Kurtosis = 142.8629 &= [−(X–Rp)^4]/[N*Straggle^4]

Statistical definitions above are those used in VLSI implant modelling.

Table Distribution Units are >>> (Atoms/cm3) / (Atoms/cm2) <<<

DEPTH (Ang.)	H Ions	Recoil Distribution
150000.E+00	0.0000E+00	0.0000E+00
300000.E+00	0.0000E+00	0.0000E+00

450000.E+00	0.0000E+00	0.0000E+00
600000.E+00	0.0000E+00	0.0000E+00
750000.E+00	0.0000E+00	0.0000E+00
900000.E+00	0.0000E+00	0.0000E+00
105000.E+01	0.0000E+00	0.0000E+00
120000.E+01	0.0000E+00	0.0000E+00
135000.E+01	0.0000E+00	0.0000E+00
150000.E+01	0.0000E+00	0.0000E+00
165000.E+01	0.0000E+00	0.0000E+00
180000.E+01	0.0000E+00	0.0000E+00
195000.E+01	0.0000E+00	0.0000E+00
210000.E+01	0.0000E+00	0.0000E+00
225000.E+01	0.0000E+00	0.0000E+00
240000.E+01	0.0000E+00	0.0000E+00
255000.E+01	0.0000E+00	0.0000E+00
270000.E+01	0.0000E+00	0.0000E+00
285000.E+01	0.0000E+00	0.0000E+00
300000.E+01	0.0000E+00	0.0000E+00
315000.E+01	0.0000E+00	0.0000E+00
330000.E+01	0.0000E+00	0.0000E+00
345000.E+01	0.0000E+00	0.0000E+00
360000.E+01	0.0000E+00	0.0000E+00
375000.E+01	0.0000E+00	0.0000E+00
390000.E+01	0.0000E+00	0.0000E+00
405000.E+01	0.0000E+00	0.0000E+00
420000.E+01	0.0000E+00	0.0000E+00
435000.E+01	0.0000E+00	0.0000E+00
450000.E+01	0.0000E+00	0.0000E+00
465000.E+01	0.0000E+00	0.0000E+00
480000.E+01	0.0000E+00	0.0000E+00
495000.E+01	0.0000E+00	0.0000E+00
510000.E+01	1.3333E−01	0.0000E+00

525000.E+01	0.0000E+00	0.0000E+00
540000.E+01	0.0000E+00	0.0000E+00
555000.E+01	0.0000E+00	0.0000E+00
570000.E+01	0.0000E+00	0.0000E+00
585000.E+01	0.0000E+00	0.0000E+00
600000.E+01	0.0000E+00	0.0000E+00
615000.E+01	0.0000E+00	0.0000E+00
630000.E+01	0.0000E+00	0.0000E+00
645000.E+01	0.0000E+00	0.0000E+00
660000.E+01	0.0000E+00	0.0000E+00
675000.E+01	0.0000E+00	0.0000E+00
690000.E+01	0.0000E+00	0.0000E+00
705000.E+01	1.3333E−01	0.0000E+00
720000.E+01	0.0000E+00	0.0000E+00
735000.E+01	0.0000E+00	0.0000E+00
750000.E+01	0.0000E+00	0.0000E+00
765000.E+01	0.0000E+00	0.0000E+00
780000.E+01	0.0000E+00	0.0000E+00
795000.E+01	0.0000E+00	0.0000E+00
810000.E+01	0.0000E+00	0.0000E+00
825000.E+01	0.0000E+00	0.0000E+00
840000.E+01	0.0000E+00	0.0000E+00
855000.E+01	1.3333E−01	0.0000E+00
870000.E+01	0.0000E+00	0.0000E+00
885000.E+01	0.0000E+00	0.0000E+00
900000.E+01	0.0000E+00	0.0000E+00
915000.E+01	0.0000E+00	0.0000E+00
930000.E+01	0.0000E+00	0.0000E+00
945000.E+01	1.3333E−01	0.0000E+00
960000.E+01	0.0000E+00	0.0000E+00
975000.E+01	1.3333E−01	0.0000E+00
990000.E+01	2.6667E−01	0.0000E+00

100500.E+02	0.0000E+00	0.0000E+00
102000.E+02	0.0000E+00	0.0000E+00
103500.E+02	0.0000E+00	0.0000E+00
105000.E+02	1.3333E−01	0.0000E+00
106500.E+02	1.3333E−01	0.0000E+00
108000.E+02	0.0000E+00	0.0000E+00
109500.E+02	2.6667E−01	0.0000E+00
111000.E+02	0.0000E+00	0.0000E+00
112500.E+02	0.0000E+00	0.0000E+00
114000.E+02	0.0000E+00	0.0000E+00
115500.E+02	0.0000E+00	0.0000E+00
117000.E+02	0.0000E+00	0.0000E+00
118500.E+02	1.3333E−01	0.0000E+00
120000.E+02	0.0000E+00	0.0000E+00
121500.E+02	2.6667E−01	0.0000E+00
123000.E+02	1.3333E−01	0.0000E+00
124500.E+02	2.6667E−01	0.0000E+00
126000.E+02	0.0000E+00	0.0000E+00
127500.E+02	2.6667E−01	0.0000E+00
129000.E+02	1.3333E−01	0.0000E+00
130500.E+02	4.0000E−01	0.0000E+00
132000.E+02	1.3333E−01	0.0000E+00
133500.E+02	0.0000E+00	0.0000E+00
135000.E+02	4.0000E−01	0.0000E+00
136500.E+02	9.3334E−01	0.0000E+00
138000.E+02	5.3334E−01	0.0000E+00
139500.E+02	1.3333E+00	0.0000E+00
141000.E+02	4.4000E+00	0.0000E+00
142500.E+02	7.7334E+00	0.0000E+00
144000.E+02	1.9333E+01	0.0000E+00
145500.E+02	4.0533E+01	0.0000E+00
147000.E+02	6.8667E+01	0.0000E+00

148500.E+02 9.4000E+01 0.0000E+00

150000.E+02 1.1307E+02 0.0000E+00

A.4 SRIM Sorting Code

```
import glob
import os
from os import listdir


####Here, we have to search all files in the directory. I couldn'
    t find a more efficient way to do it in python, so for this
    program to work, you have to have this file in the same
    directory as the files you wish to analyze.####

files=glob.glob('./*.txt')

###We want to make three lists: depth (thickness of coolant +
    thickness of sample), number of ions, and the "okfiles", which
     is a list of the files that have ions that penetrate the
    sample. The next few lines read the RANGE.txt files for data
    and append it to the lists. ####

okfiles=[]
depth=[]
number=[]
count=0
numfiles=0
coolthick=''
sampthick=0
for file in files:
    with open(file, 'r') as f:
        for line in f:
            if '-----------  ----------  -------------' in line:
                for line in f:
```

```
                    #### here, we make the elements in the list
                        into integers for analysis ####
                    newline = line.split('  ')
                    stringdepth=newline[0]
                    stringdepthr=stringdepth.replace(".E+00","")
                    stringdepthrl=stringdepthr.replace(".E
                        +01","0")
                    stringdepthrp=stringdepthrl.replace(".E
                        +02","00")
                    depth.append(stringdepthrp)
                    number.append(newline[1])
    #### filenames are in the format of Coolant thick-number-
        Sample thick-number-energy-number. These lines read
        the filename to find the coolant and sample
        thicknesses. More improvement should be done here for
        when I make more complex arrangements, but for now,
        this is ok. ####
print 'Analysis of: ' + file
start='CTHICK-'
end='-SAMPLETHICK'
#does a search over the filename to find the coolant, then
    converts to an integer. Same for the sample thickness as
    well.
coolthick=file[file.find(start)+len(start):file.rfind(end)]
print 'Coolant thickness=' + coolthick
intcoolthick=int(coolthick)
start1='SAMPLETHICK-'
end1='-E'
sampthick=file[file.find(start1)+len(start1):file.rfind(end1)
    ]
print 'Sample thickness=' + sampthick
intsampthick=int(sampthick)
```

```
#For now, start and end sample is a bit redundant. With more
    complicated geometries, it will be more necessary. These
    are just the depths for which the sample starts and ends.
startsample=intcoolthick
endsample=intcoolthick + intsampthick
print 'Sample starts at ' + str(startsample)
print 'Sample ends at ' + str(endsample)
#SRIM works by taking the range of coolant+sample and
    dividing it by 100, thus giving 100 depths. Because of
    this, the start and end of a sample might not actually be
    depths. The lines below find the closest depth in the '
    depth' list to startsample and endsample and calls them
    SRIMstart and SRIMend
SRIMstart=min(depth, key=lambda x:abs(int(x)-startsample))
print 'closest element to start is ' + str(SRIMstart)
SRIMend=min(depth, key=lambda x:abs(int(x)-endsample))
print 'closest element to end is ' + str(SRIMend)

#find the range of indices for the sample.
samprange=abs(depth.index(SRIMstart) - depth.index(SRIMend))
#We look at the depths between the sample start and end, and
    see if there are 0 ions. If for all of the indices, we
    have 0 ions, then the ions do not penetrate the sample (
    count = index range). Otherwise, the file is OK and is
    added to okfiles. This can be modified later to throw out
    more files (if, say, they only penetrate 1/4 into the
    sample and not all the way through).
for depths in range(depth.index(SRIMstart), depth.index(
    SRIMstart)+ samprange ):
     if number[depths]=='0.0000E+00':
          count+=1
```

```
            print count
    if count>=(depth.index(SRIMend)- depth.index(SRIMstart)):
        print 'did not penetrate sample!'
    if count<(depth.index(SRIMend)- depth.index(SRIMstart)):
        print file
        numfiles+=1
        okfiles.append(file)
    #reset everything to 0 so it can be used for the next file.
    count=0
    depth=[]
    number=[]
#return a list of files that can now be used.
print okfiles
```

Appendix B

Cubit Automation

B.1 Cubit Automation

```
#!/bin/bash

filepath=~/ledouxprojects/SaltyRad/SRIM-2013/thicknesses/*

for file in $filepath
do
    line1=$(tail -1 $file)
    line2=$(head -1 $file) #read first line
    echo $line1
    echo $line2
#       i=`echo $var | awk '{print $2}'`
#       j=`echo $var | awk '{print $1}'`
    #i = head -n "1" file | tail -n 1
    #j = head -n "0" file | tail -n 0
    #$i = $(sed -n "$1p" "$file")
   # $j = $(sed -n "$0p" "$file")
    sed "s/<SAMPTHICK>/$line1/" gabbystructure.jou >
       gabbystructure1.jou
```

```
    sed "s/<COOLTHICK>/$line2/" gabbystructure1.jou >
        finalstructure.jou
    touch CTHICK-"$line1" -SAMPLETHICK-"$line2" .jou
            cp finalstructure.jou > CTHICK-"$line1" -SAMPLETHICK
                -"$line2" .jou
done
```

B.2 Cubit Journal File

```
reset
brick x 10 y 10 z <COOLTHICK>
brick x 10 y 10 z <SAMPTHICK>
align Volume 2 surface 8 with surface 1
brick x 10 y 10 z <COOLTHICK>
align Volume 3 surface 14 with surface 7
mesh volume 1
mesh volume 2
mesh volume 3
```

Appendix C

MOOSE Input Files

C.1 SRIM to MOOSE coupling code

```
#!/bin/bash
tail -100 <ION> > Energy-List.txt
FILE="Energy-List.txt"
while read line; do
     range=$(echo $line | awk '{print ($1 / 10000000000)}')
     energy=$(echo $line | awk '{print $3}')
    string1=$range
    string2=$energy
done <$FILE
rm IonBeamEnergy.txt
touch IonBeamEnergy.txt
Rbeam=$(grep -r "beam_radius" ~/projects/sahote/problems/
   SahoteTestLocal.i | awk
'{print $3}')
echo "Beam radius is "$Rbeam;
echo "AXIS Z" >> IonBeamEnergy.txt
##need to present string "range" rather than as a column, as a
   row of number
seperated by a space
```

```
while read line; do
      range=$(echo $line | awk '{print (($1 / 10000000000) -
         0.0025)}')
      printf "%.5f " $range >> IonBeamEnergy.txt
done <$FILE
printf "\n" >> IonBeamEnergy.txt
echo "AXIS Y" >> IonBeamEnergy.txt
awk 'BEGIN{ for (i=-0.01; i <= 0.0105; i+=0.0005) printf("%.4f ",
     i); }' >>
IonBeamEnergy.txt
printf "\n" >> IonBeamEnergy.txt
echo "AXIS X" >> IonBeamEnergy.txt
awk 'BEGIN{ for (i=-0.01; i <= 0.0105; i+=0.0005) printf("%.4f ",
     i); }' >>
IonBeamEnergy.txt
printf "\n" >> IonBeamEnergy.txt
echo "DATA" >> IonBeamEnergy.txt
##require a for loop which iterates between the x and y
    coordinates in the fashion
of the piecewise multilinear data matrix input which states that
    if the coordinates
are within the radius (Rbeam) pulled in above from the input file
     pull in the SRIM
energies and if it is outside Rbeam, output 100 zeros
for x in {-20..20}
do
  for y in {-20..20}
  do
     circrad=$(echo "sqrt(($x/2000)^2 + ($y/2000)^2)" | bc -l)
     if (( $(echo "$Rbeam > $circrad" | bc -l) )); then
## Here, the current point is inside the beam, so dump the actual
     energies from
```

SRIM

```
        FILE="Energy-List.txt"
        while read line; do
          energy=$(echo $line | awk '{print $3}')
          echo "$energy" >> IonBeamEnergy.txt
        done <$FILE
      else
## Here, the current point is outside the beam, so just dump one
   hundred zeroes
for z in {1..100}
        do
          echo 0 >> IonBeamEnergy.txt
        done
fi done
done
mv IonBeamEnergy.txt /home/projects/sahote/problems
```

C.2 MOOSE Declaration file

```
[Mesh]
  type = FileMesh
  file = /home/projects/sahote/finalstructure.jou
[]

[Variables]
  [./Temperature]
    order = FIRST
    family = LAGRANGE
    initial_condition = 500
  [../]
  [./pressure]
    order = FIRST
    family = LAGRANGE
    block = 'Coolant'
    initial_condition = 1e5
  [../]
[]

[AuxVariables]
  [./HeatGenerationPerIon]
    order = FIRST
    family = LAGRANGE
    initial_condition = 0
  [../]
  [./HeatGeneration]
    order = FIRST
    family = LAGRANGE
    initial_condition = 0
  [../]
```

```
  [./HeatAux]
    order = CONSTANT
    family = MONOMIAL
    initial_condition = 1
  [../]
  [./VelocityAux]
    order = CONSTANT
    family = MONOMIAL
    initial_condition = 1
  [../]
[]

[Functions]
  [./HeatGenerationFunction]
    type = PiecewiseMultilinear
    data_file = /home/projects/sahote/problems/IonBeamEnergy.txt
  [../]
[]

[Kernels]
  [./Velocity-Field]
    type = Diffusion
    variable = pressure
    block = 'Coolant'
  [../]
  [./HeatConduction-Structure]
    type = MatPropDiffusion
    variable = Temperature
    block = 'Top_Structure Bottom_Structure'
    diffusivity = ThermalConductivity
  [../]
  [./HeatConduction-Coolant]
```

```
    type = MatPropDiffusion
    variable = Temperature
    block = 'Coolant'
    diffusivity = ThermalConductivity
  [../]
  [./Convection-Coolant]
    type = Convection
    variable = Temperature
    block = 'Coolant'
    pressure = pressure
  [../]
  [./HeatConduction-Pincers]
    type = MatPropDiffusion
    variable = Temperature
    block = 'Pincers'
    diffusivity = ThermalConductivity
  [../]
  [./HeatConduction-Sample]
    type = MatPropDiffusion
    variable = Temperature
    block = 'Sample'
    diffusivity = ThermalConductivity
  [../]
  [./RadiationHeatSource]
    type = RadiationHeatSource
    variable = Temperature
    PrimarySource = HeatGeneration
  [../]
[]

[AuxKernels]
  [./IonicHeatGenerationRate]
```

```
    type = FunctionAux
    variable = HeatGenerationPerIon
    function = HeatGenerationFunction
  [../]
  [./HeatGenerationRate]
    type = BeamHeating
    variable = HeatGeneration
    ionic_heating = HeatGenerationPerIon
    beam_current = 1e-6 # Specify your total beam current in Amps
    beam_radius = 0.003 # Specify your beam radius in metres
  [../]
  [./HeatAux]
    type = MaterialHeatAux
    variable = HeatAux
    block = 'Coolant'
    velocity = pressure
    material_type = KNO3
  [../]
  [./VelocityAux]
    type = VelocityAux
    variable = VelocityAux
    pressure = pressure
  [../]
[]

[Materials]
  [./Coolant]
    type = SahoteMaterial
    block = 1
    temperature = Temperature
    material_type = KNO3
  [../]
```

```
  [./Sample]
    type = SahoteMaterial
    block = 5
    temperature = Temperature
    material_type = 316SS
  [../]
  [./Coolant]
    type = SahoteMaterial
    block = 1
    temperature = Temperature
    material_type = KNO3
  [../]
[]

[BCs]
  [./Coolant_Entry]
    type = DirichletBC
    variable = pressure
    boundary = 'Coolant-Entry'
    value = 1e5          ## Enter inlet pressure in Pa
  [../]
  [./Coolant_Exit]
    type = DirichletBC
    variable = pressure
    boundary = 'Coolant-Exit'
    value = 9.98e4       ## Enter outlet pressure in Pa
  [../]
  [./StructureAir]
    type = CRUDCoolantNeumannBC
    variable = Temperature
    boundary = Structure-Air
    T_coolant = 300
```

```
    h_convection_coolant = 50
  [../]
  [./StructureCoolant]
    type = CoolantNeumannBC
    variable = Temperature
    boundary = Structure-Coolant
    T_coolant = Temperature
    heat_transfer_coefficient = HeatAux
  [../]
  [./CoolantSample]
    type = CoolantNeumannBC
    variable = Temperature
    boundary = Coolant-Pincer
    T_coolant = Temperature
    heat_transfer_coefficient = HeatAux
  [../]
  [./CoolantEntry]
    type = DirichletBC
    variable = Temperature
    boundary = Coolant-Entry
    value = 650
  [../]
[]

[VectorPostprocessors]
  [./SampleTemperature]
    type = LineValueSampler
    variable = Temperature
    num_points = 25
    start_point = '0 0 0.000250000011874363'
    end_point = '0 0 -0.000250000011874363'
    sort_by = id
```

```
  [../]
[]


[Preconditioning]
  [./smp]
    type = SMP
    full = true
  [../]
[]

[Executioner]
  # [./Adaptivity]
  # steps = 1
  # refine_fraction = 0.4
  # coarsen_fraction = 0.02
  # max_h_level = 3
  # error_estimator = KellyErrorEstimator
  # [../]
  type = Steady
  solve_type = PJFNK
  petsc_options_iname = '-pc_type -pc_hypre_type'
  petsc_options_value = 'hypre boomeramg'
  l_max_its = 20
  l_tol = 1e-3
  nl_rel_step_tol = 1e-50
  nl_rel_tol = 1e-4
[]

[Outputs]
  file_base = 1e-
  exodus = true
```

```
  csv = true
  [./Console]
    type = Console
    linear_residuals = true
    all_variable_norms = true
    perf_log = true
  [../]
[]
```

C.3 MOOSE Automation code

```
#!/bin/bash

filepath=~/projects/sahote/contrib/SRIM-2013/working_range_files
    /*
filepath2=~/projects/

for file in $filepath
do
        INPUT=$file
        tmp=${INPUT#*CTHICK-}
        COOL=${tmp%-SAMP*}

        tmp2=${INPUT#*SAMPLETHICK-}
        SAMP=${tmp2%-E-*}

        tmp3=${INPUT#*-E-}
        E=${tmp3%.txt*}

        filename2=~/projects/sahote/contrib/SRIM-2013/
            working_ion_files/
        ION-SS316COOLANT-CTHICK-$COOL-SAMPLETHICK
        -$SAMP-E-$E.txt

        sed "s/<ION>/$filename2/"
        ~/projects/sahote/conversion/code1.sh >
        ~/projects/sahote/conversion/code.sh

    sed "s/<SAMPTHICK>/$SAMP/" gabbystructure.jou >
        gabbystructure1.jou
```

```
    sed "s/<COOLTHICK>/$COOL/" gabbystructure1.jou >
        finalstructure.jou

        cd ~\projects\sahote\problems
        peacock -e SahoteTestLocal.i

done
```

Bibliography

[1] Moose framework wiki. electronic, 2016.

[2] *Fundamentals of Radiation Materials Science*. Springer, 2007.

[3] A. Boresi. Creep of metals under multiaxial states of stress. *Nuclear Engineering and Design*, 18:415–456, 1972.

[4] Ernest Cravalho. *Thermal-Fluids Engineering Course Notes*. Oxford University Press, Cambridge, Massachusetts, 2005.

[5] Michael Doster. Heat conduction, 2016.

[6] J.H. Gittus. Theoretical analysis of the strains produced in nuclear fuel cladding tubs by the presence of cracked cylindrical fuel pellets. *Nuclear Engineering and Design*, 18:69–82, 1972.

[7] M. Gunay. Assessment of the nuetronic performance in some alternative fluids in a fusion-fission hybrid reactor by using monte carlo method. *Annals of Nuclear Energy*, 60:93–97, 2013.

[8] M. Guyette. Cladding-strength analysis under the combined effect of creep and plasticity in fast-reacctor environments. *Nuclear Engineering and Design*, 18:53–68, 1972.

[9] Sterling Harper. Personal communication, May 17 2016.

[10] J. Howe. The beginning of nuclear materials: Studies of corrosion and cladding. *Journal of Nuclear Materials*, 100:36–48, 1981.

[11] Jamie Sahote. The investigation of the heat removal capabilities of an innovative proton irradiation test rig design, September 2014.

[12] J. Kenneth Shultis. *Fundamentals of Nuclear Science and Engineering*. Marcel Dekker, Inc., Kansas State University, 2002.

[13] Michael P. Short Yong Gang Li, Yang Yang. Im3d: A parallel monte carlo code for efficient simulations of primary radiation displacements and damage in 3d geometry. *Scientific Reports*, 5, 2015.

[14] E. Zarkedoula. High-energy radiation damage in zirconia: Modeling results. *Journal of Applied Physics*, 115:1–8, 2014.