Computational techniques for simulation and design of a biological sample irradiation chamber

by

Eric Giunta

B.S. Kansas State University, 2019

A THESIS

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Alan Levin Department of Mechanical and Nuclear Engineering
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2022

Approved by:

Major Professor
Amir A. Bahadori

Copyright

Abstract

This study covers the computational efforts and theory performed to simulate a biological irradiation facility at Kansas State University's TRIGA Mark II Nuclear Reactor Facility, as well as model and optimize a path taken by a platform to uniformly irradiate a target sample. The work presented in here covers the steps taken to simulate the reactor as a radiation source, model a flask of water as a target, approximate the dose from a path taken through the beam, and optimize the path to reach uniform irradiation.

Previous reactor models were improved with updates to MCNP thermal cross-section libraries and converted to a general radiation source. This source was used to simulate the energy deposited in a voxelized flask of water, following a path in front of a reactor beamport in discrete steps, based on a series of parameters which set the velocity over different sections. The simulation output was used to construct a matrix with rows for each step function velocity and columns for the impact on each voxel, which has a parameter vector that minimizes the deviation in dose. Finally, simulation of a random position machine was performed to investigate how this process may change with the added constraint of flask gravity.

The algorithms applied were able to converge to a series of linear motions that brought 99.7% of voxel doses within 10% of the median dose for both random and nonrandom initial parameters. There is a relationship between the parameters defining the path and the expected gravity vector, which can be randomly sampled to obtain paths with the expected gravity. Incorporating dose uniformity with gravity constraints would require both linear and rotational motion. Doing so with current memory use would be outside of the application of the algorithms in this work.

Table of Contents

List of Figures

List of Tables

Acknowledgments

This work would not be possible if not for the work done by John Boyington in modeling the radiation flux output of the beamport, the entire CORPS team behind building and updating the reactor model, and the guidance of the reactor staff. I would also like to thank my major professor, Dr. Amir Bahadori, for his guidance and bringing me onto this project.

Dedication

I would like to dedicate this work to my cat. She spent countless hours keeping me in my chair and listening to me debug my code. I would also like to dedicate this work to my friends and family who were always in my corner.

Chapter 1

Introduction

Kansas State University's TRIGA Mark II Nuclear Reactor Facility provides faculty and students with the opportunity to perform reactor experiments. The design of these experiments requires some understanding of the reactor as a radiation source, particularly to be confident in the radiation field in the exterior of the beamports, which is vital to simulating the danger to bystanders. The ability to accurately predict the output of the reactor allows the researcher to ensure that (1) doses are delivered as desired and (2) applicable radiation protection requirements are met. To this end, countless hours of work has been dedicated to building models of the reactor so that experiments can be simulated prior to being performed[1;6;7].

The radiation source for the work covered in the following chapters is the KSU TRIGA Mark II Nuclear Reactor[1], which has no dedicated process for performing biological sample irradiation. The goal of the work presented in this paper is to derive and demonstrate a method of simulating a biological target being irradiated within a hypothetical chamber by a specific beamport of the reactor. The sub-goals are to assist in prototyping a chamber design and to solve for a target path that minimizes non-uniformity in dose. There are three main unknowns in the irradiation planning process. The first unknown is the shape and material composition of the chamber. The second unknown is the shape and material composition of the container surrounding the biological sample. The third unknown is the combination of movements through the beam that will result in a uniform dose in the sample without damaging any equipment. The methods described in the following chapters are intended to help resolve these unknowns. Each unknown sets limits on the final process, the first limitation is the ability to generalize the process to any combination of shielding and targets.

The first limit is that the process needs to be flexible enough to be applied independently of what chamber and target are used. Quantities such as dose to bystanders and the change in flux within the chamber need to be easily determined for any combination of chamber and target. Chapter 3 meets the constraint by converting a simulation of the reactor into a source term for a second simulation covering only the reactor exterior. Separating the simulation into the reactor and exterior is faster than simulating the entire reactor for any given setup and allows for quick prototyping. Through a combination of importing CAD files into MCNP6[8] and building macrobody approximations, accurate models of the intended chamber and target can be simulated in the expected radiation environment and the effects of these choices can be examined without building a prototype and taking physical measurements.

The second limit is that the code should replicate the target moving through the beam and generate a path if not already fully defined. This code is intended to test the uniformity in dose for a variety of paths, not just a single predefined path. Chapter 4 meets the constraint by defining a series of possible movements, simulating the target over discrete steps in distance along each movement, and treating the simulation results as instantaneous values of dose rate as a function of position for each voxel. The path is then defined as the target position as a function of time, and the dose for the path is the integral of dose rate over time. Simulating a set number of positions sets a limit on where the target can move through, but allows for variable speed. Additionally, doing so reduces the number of simulations required by moving the calculation into the realm of numerical methods.

The third constraint is that the code needs to be able to find an optimum path. The code is intended to find a path with minimal non-uniformity in dose, so there has to be a way of improving the path aside from an individual randomly guessing. Chapter 4 satisfies the constraint by treating the dose as a function of parameters that define the path. The velocity can be defined as a function of position and a series of constant parameters, so the deviation in dose can be treated as a function of these same parameters. If the deviation and derivative of deviation with respect to the parameters can be calculated, then an algorithm can be written to slowly modify the velocities to reduce the deviation.

With all three of these conditions satisfied, a user could take an idea for a chamber and target combination, place limits on the target positions, and approximate the best performance that combination can achieve. These steps define what is necessary to perform prototyping prior to constructing and irradiating a target.

Chapter 2

Background and Motivation

2.1 Previous Work in Sample Irradiation

The use of radiation transport software to simulate biological sample irradiation is not a new practice. In recent years, work has been published on the use of GEANT4 to model particle beams for cancer treatment and predict the dose profile. For instance, simulations have been done for both general particle accelerators on a water target[9] as well as small animal irradiation with a proton beam[10]. In both instances, the researchers demonstrated that the simulations could predict the dose as a function of depth and beam width, but teams did not address dose uniformity or how to move the beam to reach a uniform dose. Similar research has been done into the feasibility of delivering a uniform dose using a cyclotron[11]. The cyclotron simulation addressed dose uniformity by defocusing the beam, placing tungsten scattering foils, and varying the distance from the exit window to sample. The work demonstrated a method of broadening the irradiation field to cover a 35 mm diameter sample. Once again, the work demonstrates the feasibility of predicting dose with a simulation, but does not provide a means of optimizing the foil placement or target distance aside from trial and error. The primary motivation for the work described in the following chapters is devising a method to automate the optimization.

2.1.1 Previous Reactor Simulation

The further development of the radiation model of the reactor was based on the results of a previous study into characterizing the reactor[1]. In that work, a model of the reactor was used to define a tally of the neutron flux density exiting the reactor's northeast beam port as a function of energy, position, and angle. The tally defined a source term for neutron flux at the beam port. Measurements were conducted with Bonner spheres and activated gold foils to validate the model. Finally, various neutron spectrum reconstruction algorithms were applied to find a neutron energy-dependent flux spectrum exiting the beamport that minimized the difference between the simulated and experimental count rates. Significant discrepancies were observed during model validation, particularly between simulated and experimental Bonner sphere measurements (Fig. 2.1). The objective of the model improvements made in this work were to resolve the apparent discrepancy between simulated and experimental Bonner sphere measurements and to approximate the neutron spectrum exiting the reactor.

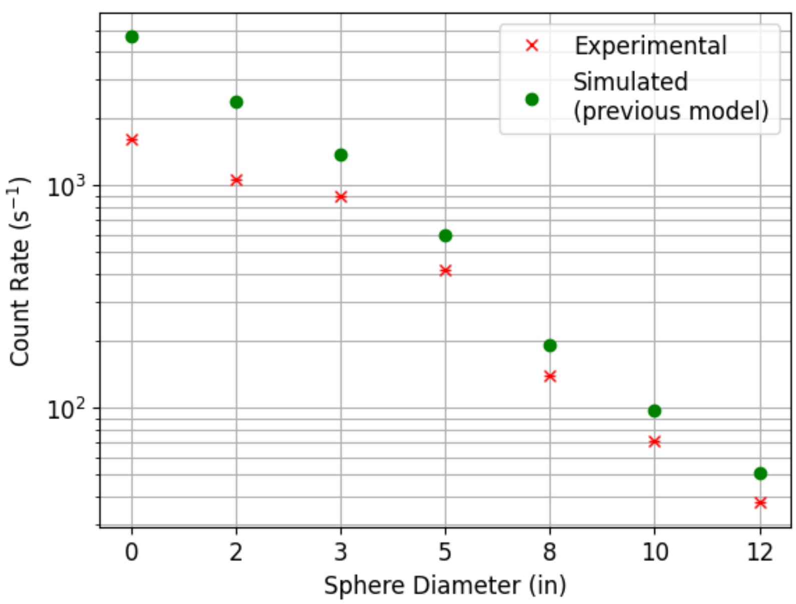

Figure 2.1: *A comparison between the experimental and simulated count rates reproduced from previous work*[1]

The issue with validating the neutron spectrum is that neutrons do not deposit ionizing radiation that could be measured. Neutrons are indirectly ionizing and radiation detection, in a general sense, relies on the particle producing electron-hole pairs via ionization. Neutrons are detected via reaction products, often the result of alpha particle production in a material containing ^{6}Li. However, neutron interaction probability is energy dependent and the cross-section for the (n, t) reaction outside the thermal range is small in most materials. According to data cataloged by the Japan Atomic Energy Agency, the (n, t) reaction cross-section in ^{6}Li at 0.025 eV is about 1 kb, the cross-section at 1 keV is about 5 b, and the cross-section at 1 MeV is about 0.3 b[2]. That means that a neutron at 1 MeV is close to 3,300 times less likely to create an alpha particle than a thermal neutron.

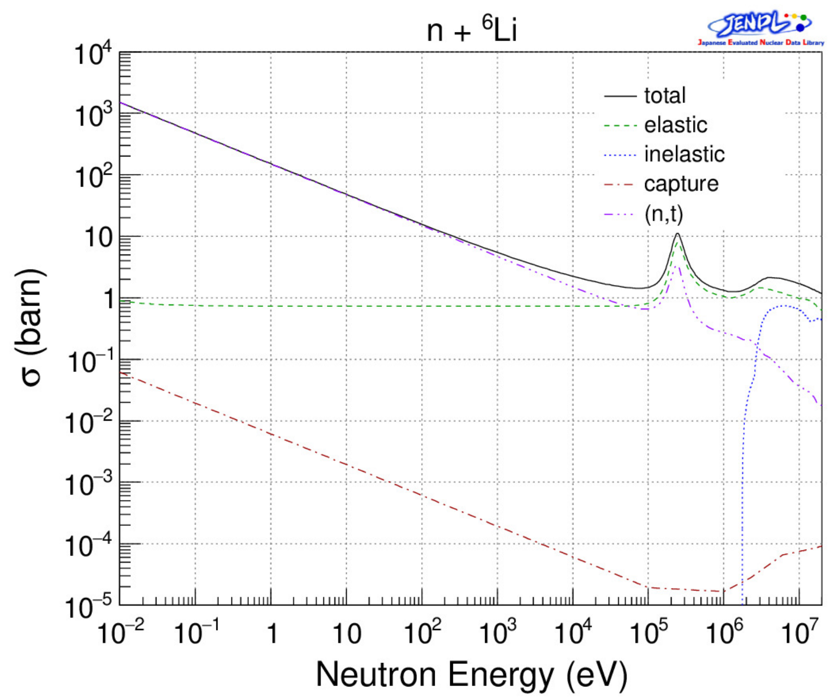

Figure 2.2: *Neutron interaction cross-section in ^{6}Li from JENDL-5*[2]

One method of performing neutron energy spectroscopy is to employ high-density polyethylene (HDPE) spheres placed around a thermal neutron detector. Due to the similarity in weight between neutrons and hydrogen atoms, the dominant interaction for neutrons in hydrogen rich materials is elastic scattering. High energy neutrons lose energy by scattering off of hydrogen in polyethylene molecules. Low energy neutrons are attenuated and higher energy neutrons are brought closer to thermal levels. The neutron energy range that interacts with the detector is then highly dependent on the diameter of the sphere. Between two spheres with different diameters, the larger sphere is able to bring higher energy neutrons down to thermal levels. The detector is not directly detecting higher energy neutrons, but the spheres are lowering neutron energies to the energy range that the detector is sensitive to. For each sphere diameter, there is an equation relating the probability of a neutron being detected to the energy of the particle exiting the reactor[3] (Fig. 2.3). By taking measurements with multiple spheres, called Bonner Spheres, a system of linear equations relating flux at each energy level to the counts with each sphere can be constructed. The solution to the underdetermined system of equations can then be approximated with some algorithm to minimize error between the simulated and experimental count rates.[12].

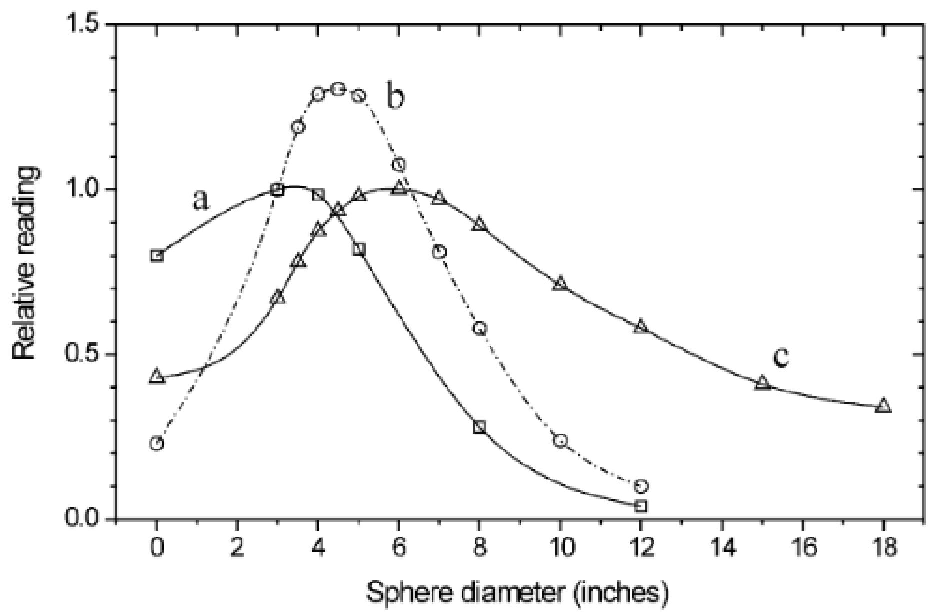

Figure 2.3: *Relative count rates for multiple sphere diameters and 3 very different neutron flux environments, denoted (a), (b), and (c)*[3]

2.2 Irradiation in Microgravity Environment

Chapter 5 of this work covers a method to plan a microgravity experiment using a Random Position Machine constructed by Kansas State University students. The future of space exploration is dependent on research into cell damage in microgravity environments. Low gravity has already been found to reduce bone mass[13]. Experiments performed by NASA in 2018 found that gamma ray dose of 0.1 Gy induced osteoclast cell fusion and dose above 0.5 Gy reduced cell fusion[14]. The effects were additive but it was found that microgravity had a more significant effect on bone mass loss than gamma ray dose. Research by the Genelab Animal Working group in 2019 found that mice exposed to 0.01 cGy/h in brain tissue for 21 days experienced changes in gene expression that lasted over 4 months[15]. The effect of both microgravity and irradiation was different from the effects of either factor individually. Research performed in 2020 found that similar additive effects applied to cell survival in lung tissue[16]. In general research has demonstrated time and time again that microgravity and irradiation both individually pose a risk to astronauts, but the combined effects of both are still an active area of research. Applying the Random Position Machine to the KSU TRIGA reactor would allow for researchers at Kansas State University to contribute to the field.

2.2.1 Random Position Machine Construction

The Random Position Machine used in this work was the result of a senior design project in spring 2021. The team was tasked with designing, developing, and building a Random Position Machine to enable research into the effects of micro-gravity and radiation at the cellular level. The device used Arduinos to send instructions to two step motors independently and rotate a target about two axes. The machine functions similarly to a two-axis clinostat, except the motors are changing paths randomly instead of following a constant loop. A method was devised to "randomly" generate paths.

The team proposed generating paths by starting with a shape in ϕ and θ with symmetry with respect to the coordinate axes to cancel out the gravity in the X and Y directions. They investigated instructing a single motor to follow a linear function (Eq. 2.3-2.4) or instructing both motors to follow sinusoidal waves (Eq. 2.1-2.2). Both types of equations would use a factor s to set the period for the path run time and factors a and b to control the magnitude of the positions.

$$\phi = a - a \times cos(s \times t) \quad (2.1)$$

$$\theta = b \times sin(s \times t) \quad (2.2)$$

$$\phi = a \times s \times t \quad (2.3)$$

$$\theta = 0 \quad (2.4)$$

The design team's plan was to generate a list of paths with known average gravity sorted by average gravity. Then, the code would randomly sample from the list to either increase or decrease the average gravity so that the final average gravity would be equal to the desired gravity. Doing so means that the same set of paths could be used for any desired gravity. The selected paths would be removed from the pool to sample from until a user supplied number of paths remains (n). For instance, for a desired gravity of 0.5g the following steps would be taken.

Algorithm 1 Path Selection Algorithm

```
Randomly sample a path with gravity g_i
Remove g_i from the paths to sample
if g_i > 0.5 then
    Randomly sample from the paths with g_j < 0.5
else
    Sample from the paths with g_j > 0.5
end if
Remove g_j from the paths to sample
while Time limit not reached do
    Average gravity, ḡ, is now the weighted sum of the gravity in the previous steps
    if ḡ is above 0.5 then
        Randomly sample from the paths with g_j < 0.5
    else
        Sample from the paths with g_j > 0.5
    end if
    if More than n paths remain then
        Remove g_j from the paths to sample
    end if
end while
```

The idea was that as paths are removed, the average gravity tends toward the desired value, but the series of paths performed are random. With a large set of paths from which to sample, the number of possible combined paths the machine can take can be considered pseudo-random. The paths are all designed to start and end at the same point, so no issues with switching between different paths are expected. There are multiple combinations of path parameters that give the same average gravity, so in this work the paths are instead sampled from a pool of paths which all have average gravity near the desired value. Doing so means that the average gravity converges to the desired value much faster than the previous solution, but the set of paths are specific to the desired gravity.

Another benefit of applying the irradiation to a target with gravity constraints was because it is an example of a system with multiple values to optimize. Individually optimizing for gravity and optimizing for dose uniformity is a completely different problem than optimizing for both at once. Analyzing the methods behind selecting paths for each optimization could give insight into how a path could be optimized for multiple goals simultaneously. The majority of this work is spent discussing how each individual optimization is done, but a discussion of how the methods relate would be an important first step in any future work aiming to perform both optimizations at once.

2.3 Computational Tools

There are multiple software and programming languages that have been used in this work that warrant a general description. Three types of tools were used. The first type of tool was radiation modeling software, which includes the software used to perform the radiation transport, MCNP6[8], and the software used to automate variance reduction for the transport, ADVANTG[17]. The second type of tool was neutron spectrum unfolding software, which includes the algorithms used to approximate the reactor flux, MAXED[18] and Gravel[19], as well as algorithms used as the basis for the path optimization code. The final type of tool was text file inputs, which include the language the basis of software was written in, Julia[20], as well as the native languages for the controllers used, a Velmex VXM controller[21] and an

Arduino.

2.3.1 MCNP6.2

MCNP6.2[8] was used in this work to perform simulations of neutrons exiting the reactor and intersecting a target, either a Bonner sphere or a model of a flask containing water. There are several aspects of MCNP6.2 as a code that are referenced in later chapters that should be explained in more detail.

MCNP6.2 allows for a radiation source to vary in energy, direction relative to an axis, position in space, and particle. The energy, direction, and position can be expressed as random variables sampled from a user supplied probability distribution. The distribution for a parameter can be dependent on the value of a single second parameter. The radiation source defined to approximate neutrons leaving the reactor varied in energy, angle from a vector perpendicular to the reactor wall, and radial position relative to the center of the beam. Only a single level of dependence can be directly handled by MCNP6.2, so separate simulation inputs were written for each radial bin.

MCNP6.2 geometry is defined using surfaces and cells. Surfaces describe boundaries in relative positions. Cells use the intersection and union of surfaces to fill regions with materials. There are three ways to define surfaces used in this work. The simplest methods are using planes and cylinders extending infinitely in one direction or a macrobody. Every surface has an equation defining the boundary where the equation is zero, so cells define regions as either positive or negative relative to the boundary equation (Fig. 2.4). A macrobody defines a closed boundary of a three dimensional polygon. Instead of defining regions as positive or negative to an equation, a macrobody defines a region as inside or outside the volume. A macrobody is equivalent to a group of surfaces with a simplified definition of the region inside the volume.

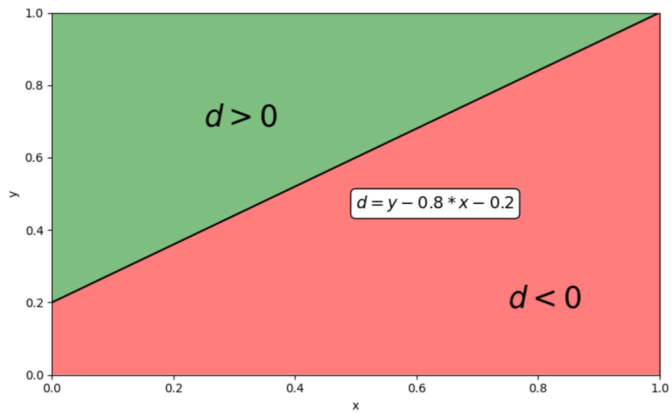

Figure 2.4: *Regions relative to a plane*

MCNP6.2 can import a CAD file using the embed card. Each part in the CAD file needs to be converted to a list of nodes. The nodes define a mesh boundary surrounding the part. MCNP6.2 assigns each part to a cell. A CAD file is the most accurate representation, but it also has the greatest risk of geometry errors. In this work, simulations with the CAD model experienced issues with particles becoming lost. MCNP6.2 reports a lost particle whenever a particle exits a cell and there is not another cell on the other side. The only way to correct this issue is by trying to locate where and why particles are getting lost, and then regenerating the mesh file with those features modified. In this work the issue was resolved by replacing the CAD model with an equivalent macrobody approximation. MCNP6.2 uses a Monte Carlo approach to simulation. The software contains references for the probabilities of different combinations of particles, energy, materials, and interactions and uses randomly generated values to randomly sample paths that the radiation can take. Tallies are radiation quantities that MCNP6.2 approximates. MCNP6.2 computes a tally, like particle flux on a plane, by following a large number of random paths and calculating the contribution from each particle. The variance in the tally approximation decreases as more particles contribute to the tally. In many instances, the combination of radiation source and geometry may limit how many particles even reach the region being tallied. An example would be if a radiation source was defined such that a large percentage of source particles start moving away from the tally region. MCNP6.2 reduces variance by assigning weights to particles to increase the probability that a particle makes a contribution at the expense of reducing the contribution, or weight, of the particle.

2.3.2 ADVANTG

ADVANTG[17] is a software designed to automate variance reduction for MCNP5[22] simulations. ADVANTG has two types of variance reduction used in this work. The first was weighting the source distributions. Suppose that for 50% of initial energies there is a 0% chance of the particle contributing to the tally. The probability of a particle being emitted at each energy can be modified so that every particle starts at an energy that can contribute. This would mean that twice as many particles would contribute to the tally, so the value would double. If the contribution of each particle were half as much, then the final tally value would return to the unweighted value. By increasing the probability of the event and then decreasing the weight of the particle, the tally is unbiased, but twice as many particles are contributing, which reduces the tally variance.

A second variance reduction technique used by ADVANTG is weight windows. Weight windows define a grid in physical space with a weight assigned to each voxel in the mesh. When a particle changes voxels it either enters a voxel with higher weight or a voxel with lower weight. If the weight increases, then the particle is split into multiple particles with lower contribution. If the weight decreases then the particle has a chance of being killed but the contribution increases. Suppose two voxels have weights of one and two. If a particle moves from weight one to weight two, then the particle is split into two particles with half weight each. If the particle moves from weight two to weight one, then the particle has a 50% chance of being killed but the weight is doubled. This is done so that the time spent simulating particles moving specific directions is increased and time spent simulating particles moving away decreases. This is not to be confused with cell importance, which corresponds to cells in the geometry. In MCNP6 each cell is assigned a weight for each particle type used, which behaves similarly to a weight window mesh. The difference is that cell importance is uniform throughout the cell, and the weights are applied to the entire geometry. A weight window mesh assigns weights to a mesh with strict bounds in each direction. Particles moving through a weight window mesh are split or killed when crossing a mesh boundary, not a cell boundary. The two should not be used at the same time, with the exception of setting cell importance to zero to create a graveyard.

ADVANTG uses two inputs; an MCNP5 script and a text file describing the variance reduction parameters. The parameters include a description of the weight window mesh. The MCNP5 script is used to define the source, a tally to reduce variance for, and to sample materials in the weight window mesh. For each voxel in the weight window mesh, ray tracing is used with the MCNP5 file to determine the material. ADVANTG bases the weight window optimization off of the ray traced geometry, not the MCNP5 input file directly. The geometry used by ADVANTG is limited by weight window mesh resolution. If the weight window mesh is too coarse, then elements of the geometry are not being considered in the optimization. In this work the grid spacing in the weight window mesh was based on both capturing changes in the neutron flux as well as capturing details of the geometry.

An important distinction is that ADVANTG uses MCNP5, not MCNP6, to generate the geometry and source definition. This means that the input file given to ADVANTG is not necessarily the same input file used by the weight windows. In this work this meant that the thermal scattering cross-sections could not be applied to the weight window optimization. However, the geometry was not changed. The primary reason for using ADVANTG in this work was due to the size of the full reactor model being much larger than the tally region. This was not changed by the thermal scattering, so ADVANTG was expected to still be able to perform the variance reduction required.

2.3.3 Spectral Unfolding Algorithms

A more detailed discussion of the spectral unfolding algorithm used in this work is covered in John Boyington's thesis on spectral unfolding for the KSU reactor[1]. A brief discussion of the algorithms follows to cover the general theory behind using these algorithms. The basic structure of the regression is using count rates for a series of detectors (N_k) with an expected randomly distributed error (ϵ_k), an energy dependent flux spectrum (ϕ), and a response function for the counts in each detector k as a function of particle energy i ($R_{k,i}$) (Eq. 2.5).

$$N_k + \epsilon_k = \Sigma_i R_{ki} \phi_i \tag{2.5}$$

The basis of the Gravel algorithm comes from the Sand II algorithm introduced in 1967 by McElroy et. al.[23]. The Sand II algorithm is based on using foil activation to minimize differences between measured and calculated responses. In the following equations, the response in count rate of each detector, i, over the energy bins, j, is denoted R_{ij}^k. The estimated flux spectrum in each iteration, k, and energy bin, j, is denoted ϕ_j^k. The measured count rate for detector, i, is denoted N_i, and the standard deviation in the measured count rate for detector, i, is denoted σ_i.

The algorithm starts with folding a default spectrum and the calculated response functions to find the expected count rate over each energy bin, j, and detector, i, which is compared to the measured response to find a correction factor for each detector, C_i (Eq. 2.6).

$$C_i^k = \frac{N_i}{\Sigma_{j'} {W_{ij'}}^k} \tag{2.6}$$

Next, a weighting factor is found for each detector and energy bin. The corrected factor for each energy bin is calculated by folding the flux with the response function in the energy bin, and then normalizing by the flux folded with the response function summed over every energy bin (Eq. 2.7).

$${W_{ij}}^k = \frac{R_{ij} {\phi_j}^k}{\Sigma_{j'} R_{ij'} {\phi_{j'}}^k} \tag{2.7}$$

A new total correction factor is then calculated as the weighted sum of the logarithm of the detector correction factors. The flux for the current iteration is then multiplied by the exponential of the total correction factor (Eq. 2.8). This is continued until some user-defined stopping point. In this work the iteration stopped after either the change in flux was below a threshold, a maximum number of iterations was reached, or a set amount of time passed.

$$\phi_j{}^{k+1} = \phi_j{}^k \exp\left(\frac{\Sigma_i W_{ij}{}^k \log\left(C_i^k\right)}{\Sigma_i W_{ij}{}^k}\right) \tag{2.8}$$

The algorithm used in this work, Gravel[19], is a popular modification of the Sand II algorithm. In the Gravel algorithm, the weighting factors include a term to account for measurement error (Eq. 2.9). This term is added to increase the weight of detector readings as variance decreases. For a reading with high variance, the difference between the measured and calculated count rates may be due to uncertainty in the measured response, not necessarily an inaccuracy in the flux spectrum.

$$W_{ij}{}^k = \frac{R_{ij}\phi_j{}^k}{\Sigma_{j'} R_{ij'}\phi_{j'}{}^k}\frac{N_i{}^2}{\sigma_i{}^2} \tag{2.9}$$

The second algorithm used in this work, MAXED, employs a maximum-entropy method for neutron spectral unfolding. The following derivations are presented in more detail in the 1998 technical report by Reginatto and Goldhagen[18]. MAXED starts by setting a constraint on equation 2.5, where Ω is a user-selected limit on the allowable deviation in the measured and calculated responses (Eq. 2.10).

$$\Omega = \Sigma_i \frac{\epsilon_i{}^2}{\sigma_i{}^2} \tag{2.10}$$

MAXED defines entropy (S) as a value associated with the deviation between the default and solution spectra. MAXED searches for a maximum entropy solution that is similar to the default spectrum (Eq. 2.11). The solution is associated with the Lagrangian in equation 2.12 with λ_i and μ as Lagrangian multipliers. Reginatto and Goldhagen found that the solution is equal to the maximization of equation 2.13.

$$S = -\Sigma_j \left[\phi_j \ln\left(\frac{\phi_j}{{\phi_j}^{DEF}}\right) + {\phi_j}^{DEF} - \phi_j\right] \tag{2.11}$$

$$L(\phi_j, \epsilon_i, \lambda_i, \mu) = S - \Sigma\lambda_i\left[\Sigma R_{ij}\phi_j - N_i - \epsilon_i\right] - \mu\left[\Sigma\left(\frac{{\epsilon_i}^2}{{\sigma_i}^2}\right) - \Omega\right] \tag{2.12}$$

$$Z = \Sigma\left[{\phi_j}^{DEF} exp\left(-\Sigma_i[\lambda_i R_{ij}]\right)\right] - \left[\Omega\Sigma_i(\lambda_i\sigma_i)^2\right]^{1/2} - \Sigma_i N_i\lambda_i \tag{2.13}$$

The results of maximizing Z can then be substituted back into equations for ϕ and ϵ to calculate solution flux and calculated response error (Eq. 2.14-2.15).

$$\phi_j = {\phi_j}^{DEF}\exp\left(-\Sigma_i\lambda_i R_{ij}\right) \tag{2.14}$$

$$\epsilon_i = \frac{\lambda_i{\sigma_i}^2}{2}\left(\frac{4\Omega}{\Sigma_j\left(\lambda_j\sigma_j\right)^2}\right)^{1/2} \tag{2.15}$$

The benefit of MAXED is that it avoids solutions with drastically different shapes from the default spectrum. However, that means that the default spectrum is assumed to be close to the true spectrum. Without proper prior information, the result should not be considered trustworthy.

2.3.4 Multivariate Newton's Method

The algorithms used for spectral unfolding in this work were based on solving for the count rate in seven detectors as a linear combination of the flux in 252 energy bins, or system of equations with seven equations and 252 unknowns. The system of equations for dose in the cells are based on solving for the dose in thousands of voxels as linear combinations of dozens of unknowns. However, these equations do not have a constraint like MAXED that the solution be similar to the starting point. Thus, the system can be optimized with a multivariate application of Newton's method instead.

The basic premise of Newton's method is to approximate a function by the Taylor series (Eq. 2.16), and use the ratio of the function to the derivative as an approximate distance to a root of the function (Eq. 2.17). The estimate in the current iteration then moves toward the approximate root for the next iteration (Eq. 2.18). In this work the change in parameter was reduced by a learning rate (l).

$$f(x+\delta) \approx f(x) + f'(x) \times \delta + \frac{1}{2} f''(x) \times \delta^2 + ... \tag{2.16}$$

$$\delta \approx -\frac{f(x)}{f'(x)} \tag{2.17}$$

$$x^{k+1} = x^k - l \times \frac{f(x)}{f'(x)} \tag{2.18}$$

In the equations for dose in each voxel, each unknown was proportional to a time duration of the beam at a series of positions. Because the time duration must be non-negative, a limit was placed on the change per iteration. At the end of each iteration, negative values were set to zero. For a small maximum change in value, the solution does not change significantly if negative values are set to zero. Thus, iterations that reduce the deviation are less likely to move to a point with higher deviation than the starting point.

2.3.5 Julia

Julia is intended to be a general purpose language for use in high performance applications that would otherwise use Python, Matlab, R, or C. Julia was created with the plan of combining the versatility of Python, the statistical ability of R, the linear algebra techniques in Matlab, and the compilation speed of C. In benchmarks performed by Julia's development team, Julia performs at speeds comparable to C (Fig. 2.5)[4]. These benchmarks are not using perfectly optimized algorithms, but they do demonstrate that default Julia outperforms

Python[4].

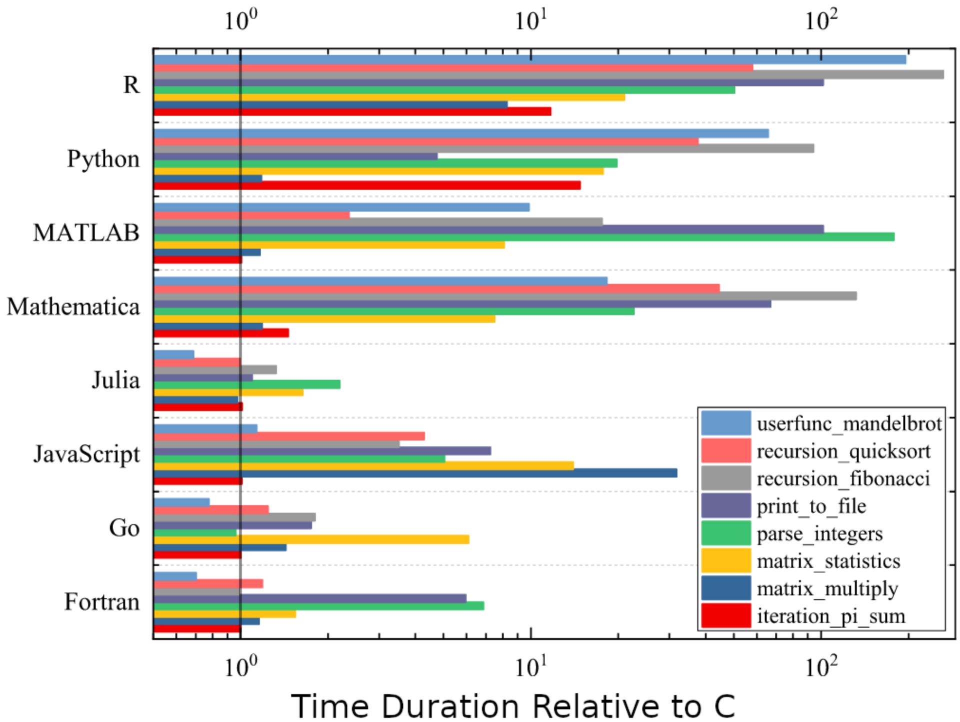

Figure 2.5: *Microbenchmarks published by Julia's team*[4]*, comparing performance with general algorithms to C as a baseline.*

One difference between Python and Julia is the compiler. Julia uses a "Just In Time" compiler to achieve speedups on par with C and Fortran, while Python is a scripted language. The difference is that a compiled language requires scripts to be written, but once it compiles, it is generally faster. A scripted language like Python compiles as it receives lines of code, so it is slower but able to return the results of single lines of code instantly. Julia can run in a terminal using the REPL (read-evaluate-print-loop) console to run interactively like a Python session, but in the background, the code is compiled so subsequent runs are faster[24].

2.3.6 Stepper Motor Controller Input Scripts

One distinction made in this work is the difference between an idealized path and the path a controller would provide. Idealized paths have perfect precision, infinite acceleration, and no limits on speed. For both the biological sample movement and the Random Position

Machine, the analysis started with the speeds and positions following a function of time that was treated as the idealized path. In both cases, code was written to calculate velocities over a series of positions with equal steps in time. Then code was developed to interpret this idealized path into the form read by the controller and then simulate the path with the restrictions imposed by the stepper motors. Then the new path was used to calculate the doses or gravity vector.

The limitations placed on the biological sample movement were primarily based on how the VXM controller interpreted continuous movement. The sample irradiation involved motors starting and stopping to change directions. Over the small distances that the motors were moving, there was the risk of the motor not reaching the intended speed. Instead of assuming instantaneous acceleration, the code had to apply a constant acceleration and calculate distances required to slow down.

The limitations placed on the Random Position Machine were based on the positional accuracy of the motors used. The stepper motors used were accurate to 0.9°, but the paths that the code was designed to follow had no limitation on precision. The instructions were sent to the Arduino as a series of vectors containing the motor positions, speeds, and distances between steps. If the motor positions were rounded, then the motors may be given commands to move zero steps, which could cause the motors to continuously start and stop suddenly. The solution was to smooth the path out. In this work, the path was smoothed out converting to a linear approximation. Instead of recording the position at every time step, the code was modified to only record positions within some tolerance of a multiple of 0.9°. A restriction was also placed on the time difference between recorded positions. This limited the size of the input to the Arduino.

Chapter 3

Radiation Source Simulation

3.1 Reactor Simulation

Prior to modeling the dose in the target, it is necessary to validate the radiation source definition. Substantial work has already been put into writing and validating an MCNP6 model of the university reactor, specifically validating the neutron flux spectrum exiting the north-east beamport[1]. To update from the model used previously, improvements to the KSU Reactor model were implemented in four main stages. First, updates were made to the model of the beam port exit. Next, improvements to the core control rods were applied. Third, libraries allowing for better methods of representing thermal scattering were employed. Finally, the new resulting Bonner sphere response functions and simulated neutron energy-dependent flux density spectrum were used as input into an algorithm to produce a neutron energy-dependent flux density spectrum that minimizes differences between the new simulated and experimental Bonner sphere measurements.

3.1.1 Computational Tools

The primary particles of interest were neutrons produced in the core exiting through the collimator. Photon transport was performed, but the predominant source of photons exiting the beamport was neutron interactions outside the core. The accuracy of the photon transport was dependent on accuracy in the neutron transport, hence the focus on neutrons. MCNP6.2[8] was chosen to perform the transport simulations. The only change to neutron physics was decreasing the maximum energy limit to 20 MeV. Since dose is of such importance, thermal neutrons and the associated scattering were simulated by applying additional material options. The previous models of the reactor used ENDF/B-VII[25], however, the manual written for the release of ENDF/B-VIII[25] clarified that newer library was designed to better model neutron scattering with hydrogen and the updated library provided additional thermal scattering options. The model incorporates a polyethylene collimator to direct neutrons toward the polyethylene sphere; so these changes were believed to be advantageous to the model. Simulations with both options supported the decision to change libraries (Fig. 3.1).

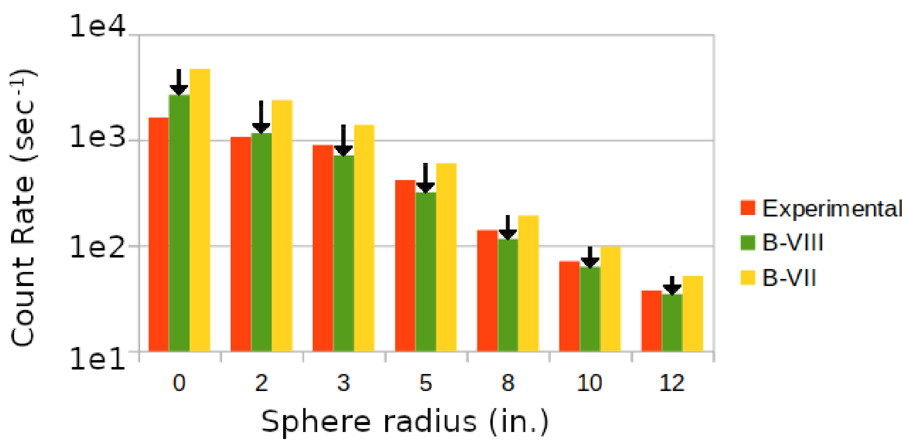

Figure 3.1: *Improvement in count rate due to switch from ENDF-BVII to ENDF-BVIII denoted by arrows*

There were two tallies computed to perform the Bonner sphere spectroscopy. The full reactor model tallies the neutron and photon current passing through radial regions at the beam port exit. The exterior model uses equal probability source energy bins and tallies flux multiplied by cross-section in the detector and binned by source energy bin. The current in the full reactor model is partitioned into bins by energy and angle from a parallel beam. These are additionally partitioned further into radial regions to define the current within different portions of the collimator. The tally provides a source term that varies in energy, angle, and radial position. The tallies in the detector provide a response function that measures the counts due to source neutrons in each energy bin. These tallies give response functions as a function of energy summed over the angular and radial positions.

The tally region is much smaller than the full reactor model, so variance reduction techniques were used to improve tally confidence. Early simulations indicated that computation time was expended simulating neutrons that had negligible probability of reaching the tally surface area. To quickly create weight windows and source biasing, the Automated Variance Reduction Generator software (ADVANTG)[17] was used and sections of the reactor were removed to reduce the physical extent simulated.

Within the reactor core, the ADVANTG weight window grid was selected to be finer in X, Y, and Z over the extent of the fuel to capture the complex geometry closely and be coarser in X along the beamport due to the simpler geometry in comparison to the core (Fig. 3.2). Along the beamport, the weight window grid was selected to be finer in X where the beamport changes diameter and where a shield was placed to reduce photon flux, and coarser in the surrounding concrete (Fig. 3.3).

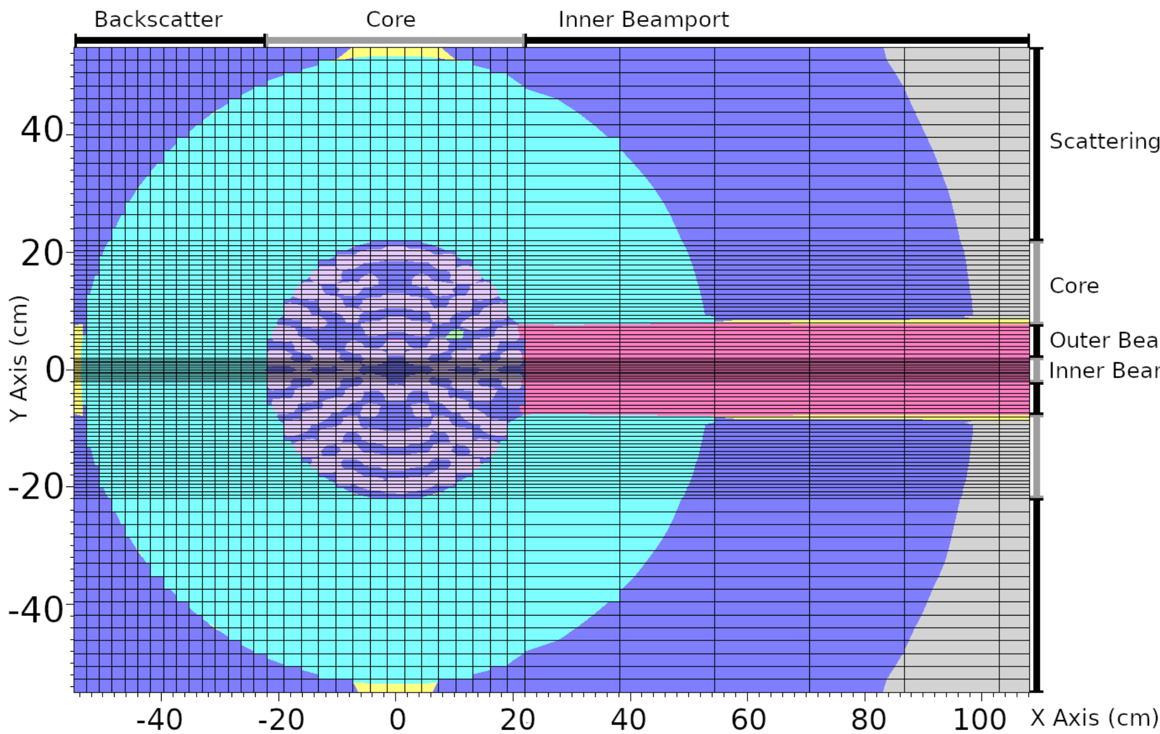

Figure 3.2: *Weight window grid overlaid on ADVANTG material diagram of core*

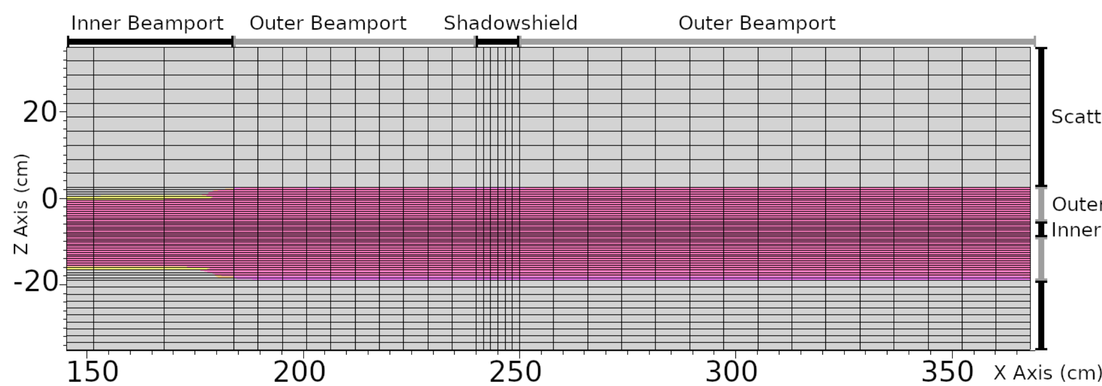

Figure 3.3: *Weight window grid overlaid on ADVANTG material diagram of beamport*

The grid was selected to be the most coarse in Y and Z in the surrounding concrete, finer in over the 11.43 cm by 11.43 cm square enveloping the beamport cross-section, and finest over the 4 cm by 4 cm square in the center of the beamport with the highest neutron flux (Fig. 3.2-3.3). The flux was tallied at the end of the beamport in radial sections ranging from 2 cm to 10 cm at the outer radius of the collimator. Tallies of the flux integrated over energy and angle in each radial region at the beamport show that the flux in the radial regions under 2 cm radius is ten times higher than the flux at a 5 cm radius and over one hundred times higher than the flux at the radial region furthest from the center-line (Fig. 3.4).

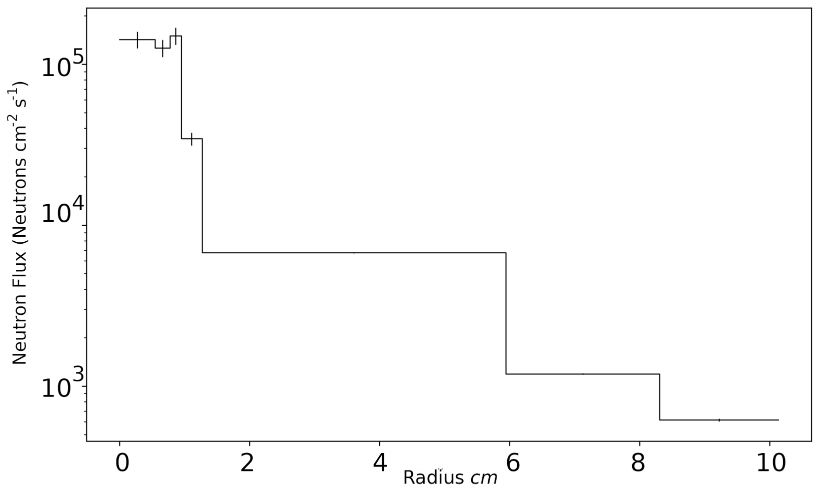

Figure 3.4: *Flux integrated over angle and energy for each radial region*

ADVANTG is only able to generate a weight window mesh for particles with a tally and an explicitly defined source. The neutrons produced in the core of the reactor had an explicit source definition. Photons were produced throughout the core by neutron interactions. The ADVANTG simulation used to generate the neutron weight window did not have an explicit photon definition, so a photon weight window mesh can't be defined with just the neutron source. To define an explicit photon source, the photon production rate would have to be tallied over a mesh covering the reactor. The photon production rate at any given point is dependent on the neutron flux. One method of generating the photon source would be to generate a weight window mesh that maximizes neutron flux in the beamport using ADVANTG, then tally the photon production rate in voxels covering the beamport and voxels covering the core. Then this new set of voxels could be used to define a volumetric source of photons. Both neutron and photon source terms could be merged into a single simulation, so that an ADVANTG simulation could be run that generates a weight window mesh for both particles. This weight window mesh would assume that photons were being produced by volumetric sources, not neutron interactions. This assumes that the initial photon source was correct, which assumes that the tally used to generate the photon source was sufficient. The tally used to define the photon source would have had to physically contain every region of the reactor with significant photon production, and also tally the production rate with low variance. This means that the photon tally exiting the beamport would accumulate the error of every previous step. In this work, the photon flux exiting the reactor was treated as being low enough that a lead shield could attenuate the flux to a point that the contribution of dose in the target from photons leaving the reactor would be several orders of magnitude lower than the dose due to neutrons. The modification of the neutron beamport source to a mixed particle source is left to future work.

3.1.2 Beam Port Geometry

Originally, the KSU Reactor model was separated into two regions to reduce required simulation time (Fig. 3.5), as the physical extent of the model, coupled with the localized nature of the beam port exit, resulted in inefficient performance when executed as a single region. None of the changes in the Bonner sphere or target would impact the beam prior to leaving the reactor, so it was not necessary to include the entire reactor for every simulation. Here, the region encompassing the reactor core and inner beam port was used to create a source term for a second region containing the outer beam port and the detector position.

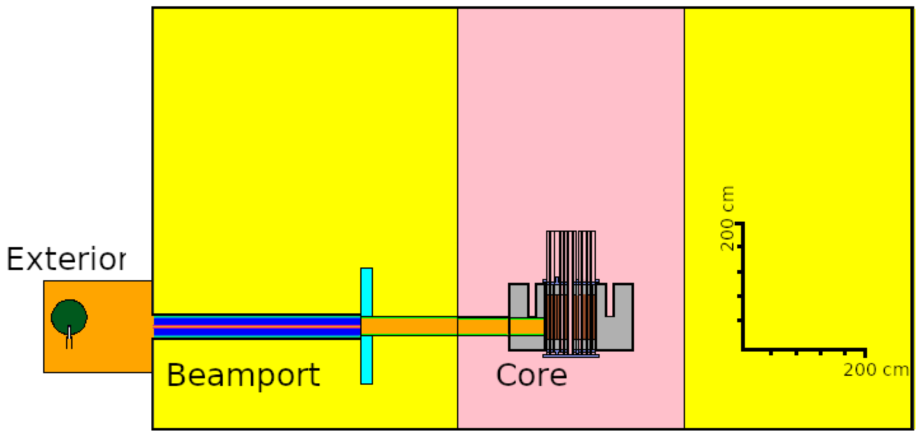

Figure 3.5: *Total model divided by reactor and exterior models*

Two simplifications in the original outer beam port model were investigated. The first change was made to the reactor wall at the beam port exit. Previously, the region was treated as a small gap in the collimator, but otherwise with no shielding past the beamport exit (Fig. 3.6). In reality, the beamport terminates into a rectangular recess in the wall. The recess was included in the model to allow for neutrons to intersect the edges of the recess, which may be a significant source of neutron scatter. Furthermore, an optional lead shield that can be moved to limit the photons exiting the beam port is present (Fig. 3.7). The shield was included as a means of limiting the photons leaving the beamport.

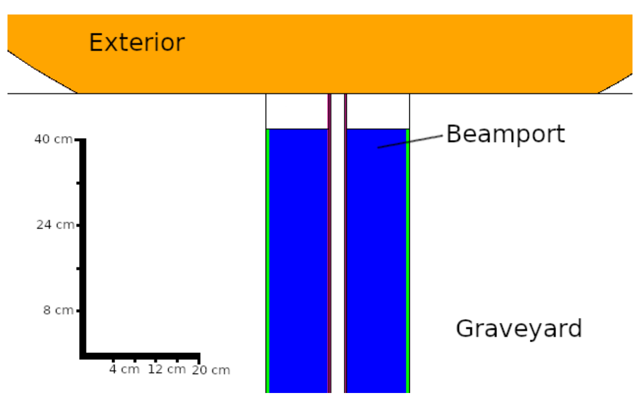

Figure 3.6: *Original exit region without lead shielding or surrounding concrete*

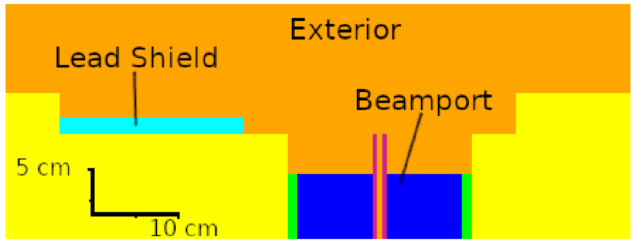

Figure 3.7: *Exit region expanded to include borated polyethylene collimator (blue) with nearby lead shield (cyan)*

Finally, the area outside the reactor was changed. In the previous model, once a neutron moves past the detector it enters the graveyard and is killed (Fig. 3.8). However, killing particles that move past the detector eliminates the possibility of a neutron scattering off nearby walls and floor. Moreover, the surroundings are necessary to simulate exposure to bystanders. A floor plan of the reactor was used to locate nearby walls and floor that might scatter neutrons back toward the detector and properly include these in the model to better represent the surroundings of the beam port exit (Fig. 3.9).

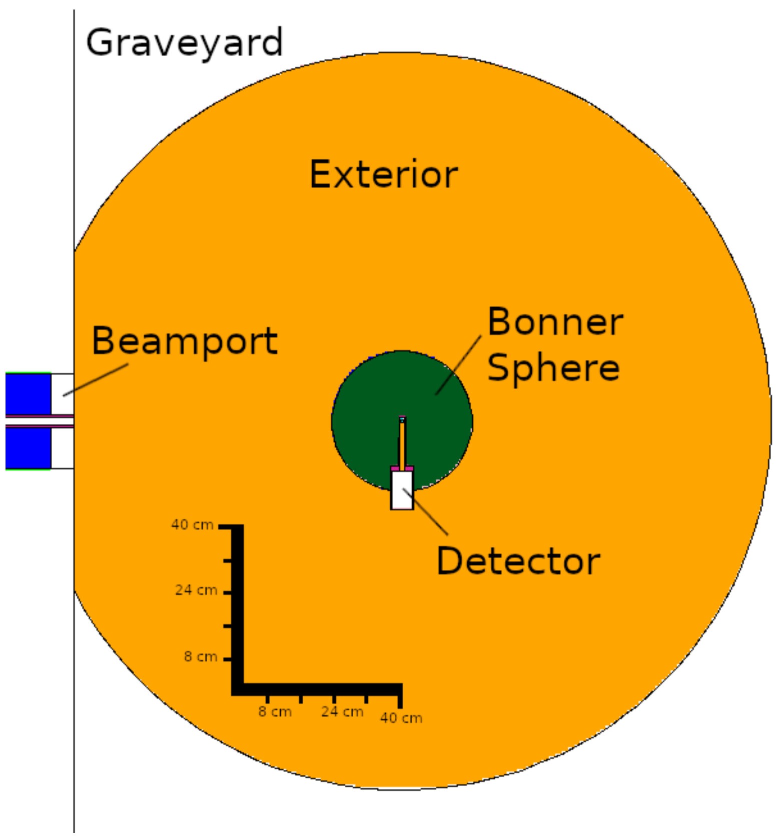

Figure 3.8: *Original exterior region without concrete surroundings*

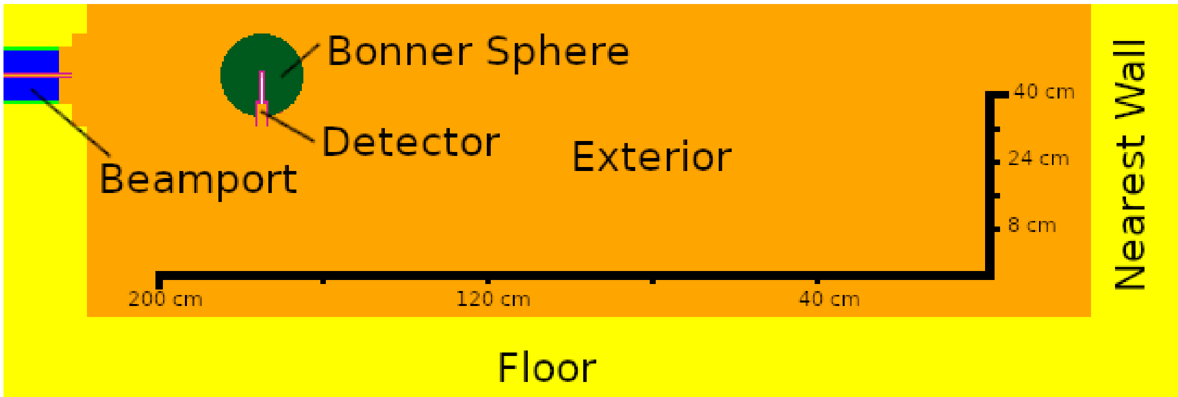

Figure 3.9: *Exterior region expanded to include nearby walls (yellow) and Bonner sphere (red)*

In general the exterior changes were designed to 1) account for scattering in the exterior simulation and 2) generalize the simulation geometry to be able to tally irradiation to equipment surrounding the detector.

3.1.3 Core Geometry

Updates were also made to improve the accuracy of the reactor core model. Understanding of the core geometry has improved substantially since the previous work[1].

- The control rod dimensions have been defined much more accurately,
- Several of the fuel rods are different models with different material compositions, and
- Updates to the ENDF cross-section library provided new and updated thermal scattering cross-sections to be applied to the surrounding water, concrete, and steel.

The first update made was to the control rods. The control rod positions were adjusted to better match the actual positions used to conduct the original Bonner sphere measurements. Ideally, the results could be abstracted to any combination of control rod positions that give the same power level. However, the beamport in question is adjacent to both the safety and regulating control rods, so the flux exiting the beamport is dependent on the position of those rods (Fig. 3.10). Additionally, every rod was previously modeled with the same axial dimensions and material compositions. In reality, the control rods have varying axial and radial dimensions, and they contain either B_4C powder or borated graphite poison. The corresponding material definitions were updated accordingly. The accuracy of the beam port tally is heavily dependent on the accuracy of the control rod geometry due to the proximity of two control rods to the beamport entrance.

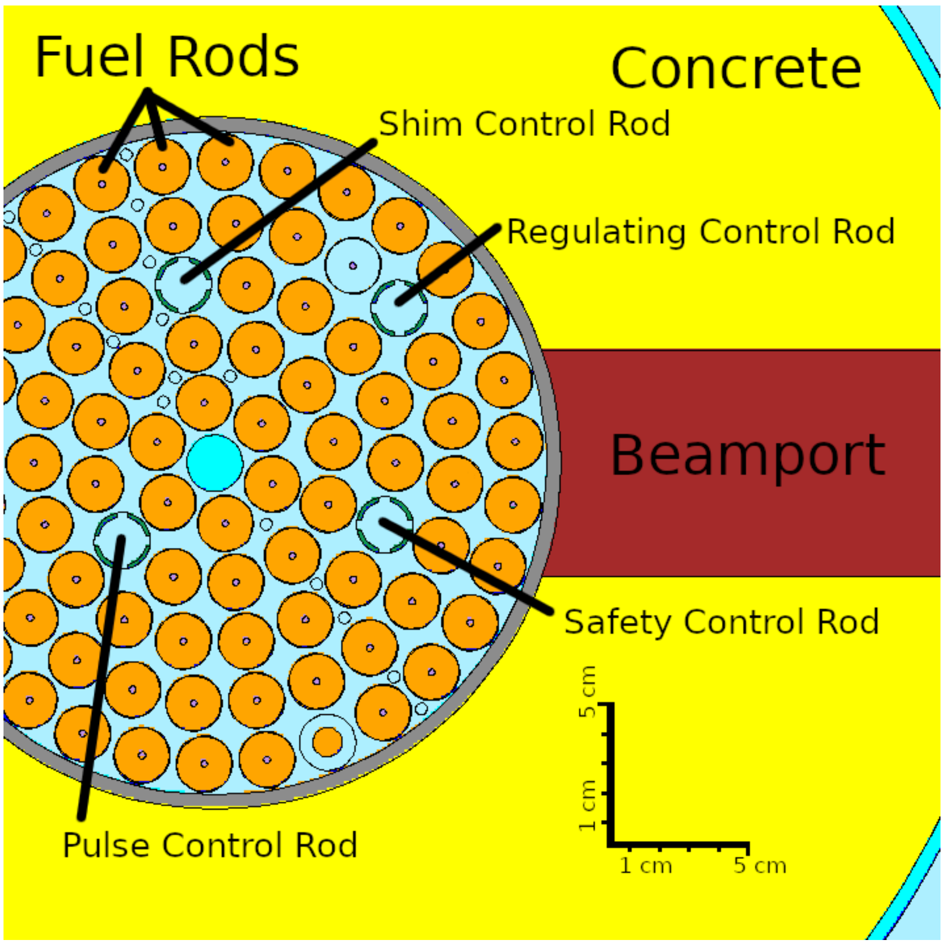

Figure 3.10: *Diagram of relative locations of control rod casing (green) to the beamport (red)*

Next, the fuel rod definitions were improved. Many of the fuel rods were correctly modeled, but several fuel rods were newer models (Fig. 3.12). In particular, these new fuel rods contain a molybdenum disk and various dimensional differences. Additionally, it was learned that the fuel rods may have up to 0.5 in gap between the attenuating graphite and the top end cap of the rod. The size of the gap is unknown and required sensitivity testing

(Fig. 3.11).

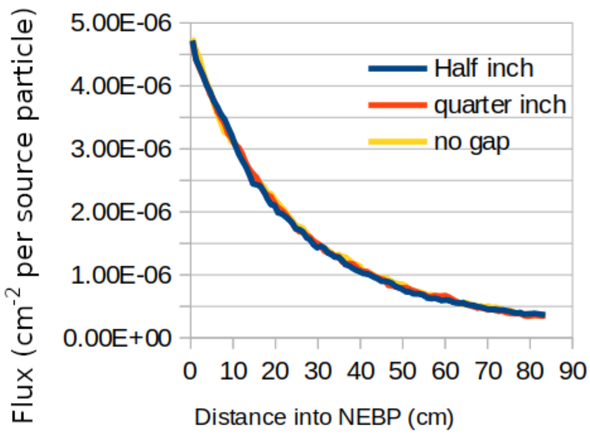

Figure 3.11: *Neutron flux in beamport in slices moving away from the core*

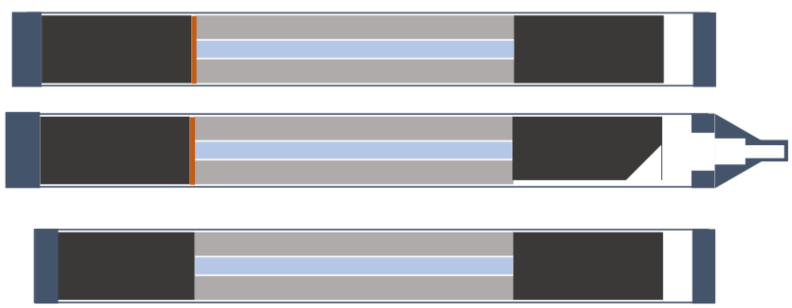

Figure 3.12: *Examples of different rod geometries with casing (dark blue), graphite (black), zirconium (light blue), molybdenum (orange), and fuel (grey)*

In general, the core modifications were done to improve the accuracy in the fuel rod and control rod geometry as well as verify that the control rod placement was correct.

3.1.4 Fuel Burnup Prediction

Aside from the dimensions of the core, another simplification made was to treat the fuel as if it were fresh. In actuality, because the reactor had been run considerably both prior to and since those experiments were performed, that simplification has become more inaccurate as time has passed. The difficulty of predicting the current fuel makeup was that the reactor use varied in time and power. The KSU Reactor is primarily a research reactor, so it is operated at a variety of power levels. Based on the need, it may be maintained at a constant power for an extended period of time or it may only be kept at a given power for minutes at a time. The only way to reproduce the composition of the modern day rod fuel would be to start with a simulation of fresh fuel and use the power and duration records to simulate the change in composition over every day of use.

The burnup calculations would be very computationally intensive, so it is necessary to consider the effect of this correction to the tallied quantities. The reactor beamport is designed to reduce the photon component of the beam and produce a collimated neutron beam, so the important question is how burnup would impact the resulting neutron beam. Burnup reduces the fissile components in the fuel, which reduces the neutron production rate around each fuel rod for any given control rod position. The population of neutrons entering the beamport is related to the neutron production rate in the reactor core. The power level of the reactor is proportional to the total neutron production rate in the reactor core. However the production rate varies throughout the core. If most of the reactor had accumulated significant burnup and then new fuel were added, the neutron production around the new fuel would be much higher than the neutron production around the old fuel. So there are instances in which the burnup would be crucial to simulate. For this work, an approximation of the burnup calculations was performed and included for the sake of accuracy.

3.1.5 Thermal Scattering

Fidelity of model physics was also refined. Neutrons with kinetic energies in the thermal region have the greatest influence on the response with no sphere used (Fig. 3.13). This suggests that the overestimate of experimental response by simulation might be a result of thermal neutron scattering models, since thermal scattering functions had previously only been applied to graphite, light water, and reactor fuel[1]. The beamport is a steel pipe with a borated high density polyethylene collimator surrounding a smaller aluminum pipe, and the Bonner spheres are composed of high density polyethylene. The pipe, collimator, and Bonner spheres all contain elements that have thermal scattering adjustments available. In particular, the functions of the collimator and Bonner spheres are based on interaction with hydrogen, which makes the thermal scattering correction vital to simulating the experiment accurately.

It was straightforward to incorporate the thermal scattering function for aluminum and steel because their thermal scattering cross-sections were for solid metal. Properly accounting for thermal scattering in concrete is more challenging because water is a substantial constituent in concrete but the water is not a liquid. The thermal scattering data for water in ENDF-VII and ENDF-VIII are for liquid water, so the water in concrete does not match the state of water that the thermal scattering data is based on. Previous research suggested instead using thermal scattering cross-sections from hydrogen in solid materials[26]. The same research found that ice and methylene were most appropriate surrogates for modeling thermal neutron scattering on hydrogen in concrete, and so both were tested for the present work.

An update in cross section version also permitted more detailed modeling of the reactor graphite. With the update from ENDF B-VII to ENDF B-VIII, options for nuclear graphite, an artificial graphite that is common in reactors, were made available in both 10% and 30% porous varieties. The graphite in the core was more accurately modeled as 30% porous artificial graphite as opposed to natural crystalline graphite as it was in previous models[27].

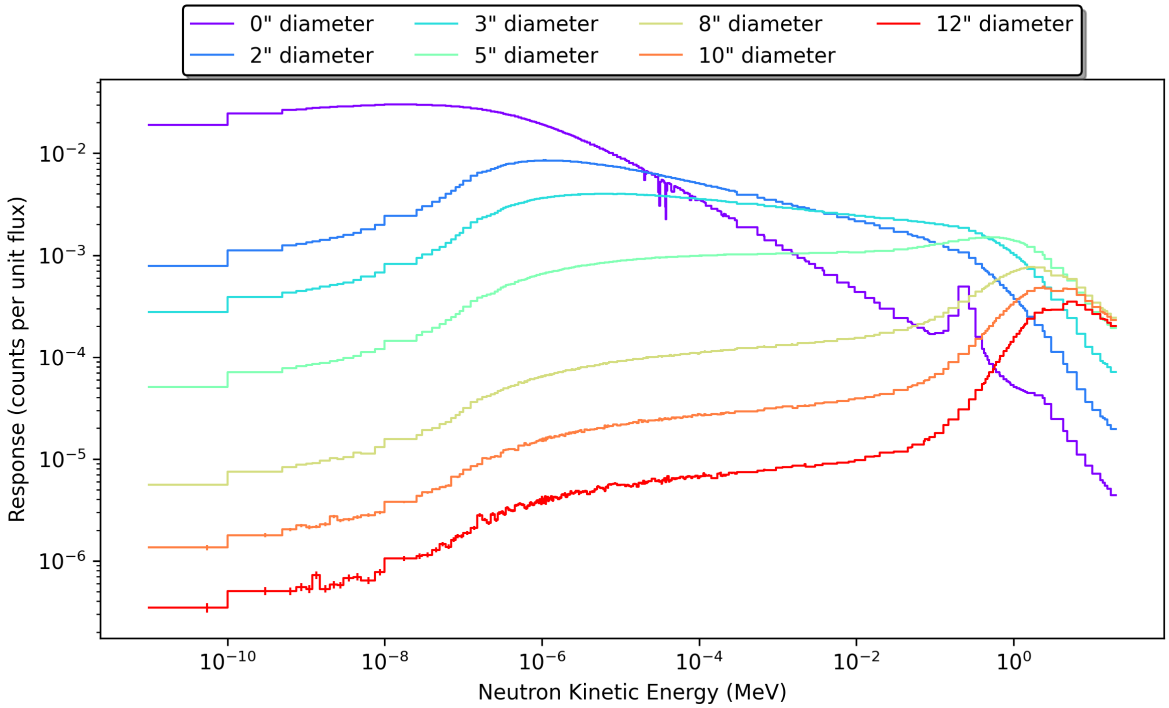

Figure 3.13: *Bonner sphere response functions for flux exiting the beamport to counts in Bonner spheres*

In the absence of other materials, the response function for the bare sphere simulation would be strongly dependent on the (n,t) cross-section presented in the background chapter (Fig. 2.2). Both the bare sphere response function and the (n,t) cross-section for ^{6}Li have a peak at 0.1 b. The most significant deviation between the shape of the response function and cross-section curve is the resonance between 10 eV to 100 eV. One likely cause is that the detector was using a lithium-iodide crystal. The total interaction cross-section for ^{127}I has a series of resonant peaks in that region (Fig. 3.14).

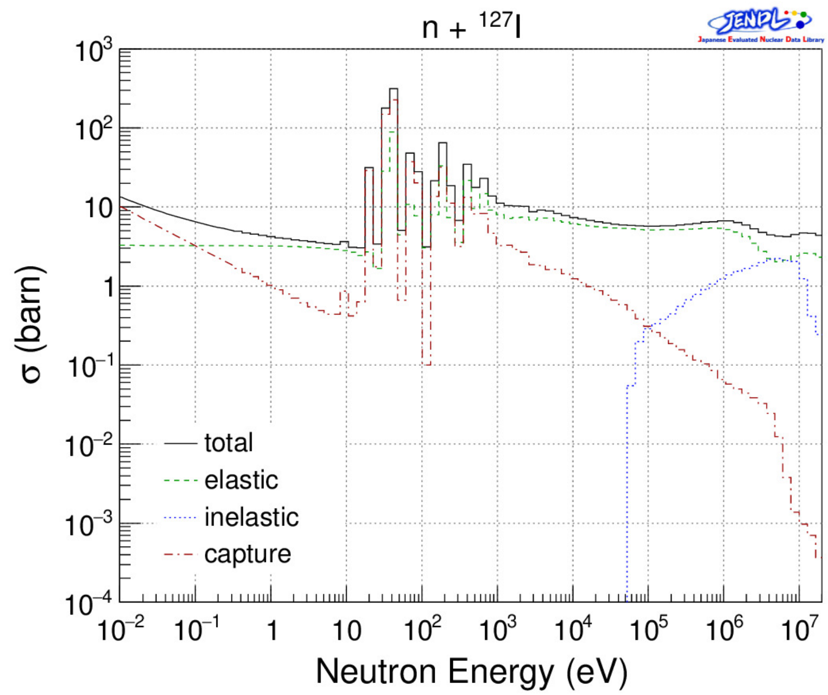

Figure 3.14: *Neutron interaction cross-section in ^{127}I from JENDL-5*[2]

3.1.6 Spectrum Unfolding

The implemented changes resulted in a model geometry that was improved with respect to previous modeling work. However, a method to adjust simulated neutron energy-dependent flux density spectra using relevant experimental measurements was desirable to ensure the greatest accuracy for dosimetric simulations. To incorporate the experimental data, the algorithms used in the previous characterization were repeated with the updated source and response functions[1].

There are far more unknowns than equations, so the flux cannot be directly solved and algorithms are needed that can create a reasonable solution. These algorithms use as input the Bonner sphere response functions, simulated neutron energy-dependent flux distribution, error in count rate, and experimental count rate and use the error between experimental and simulated count rates to iterate towards a more accurate energy spectrum[1]. The new spectrum is similar in shape to the MCNP solution, but with slight magnitude changes that minimize the error in count rate. The Gravel[19] and Maxed[18] algorithms were applied to find adjusted neutron energy spectrums that minimized the error between simulated and experimental Bonner sphere measurements. To determine if the algorithms converged, the change in error over the iterations was recorded to determine if the relative change was slowing down. In general it was noticed that the algorithms did not experience significant change past approximately one hundred iterations.

3.2 Reactor Simulation Results

One consistent trend in the simulations was small statistical uncertainty. Error in the simulations comprises both error due to the model and statistical uncertainty due to the nature of Monte Carlo calculations. To keep the statistical uncertainty small, the simulations were continued until the tallies converged. This left model error as the predominant source of error. Ultimately, while the changes to the reactor exterior better represent actual surroundings, they did not have a significant impact on the simulated Bonner sphere counts (Fig. 3.15). This indicates that the differences are likely due to the reactor core model.

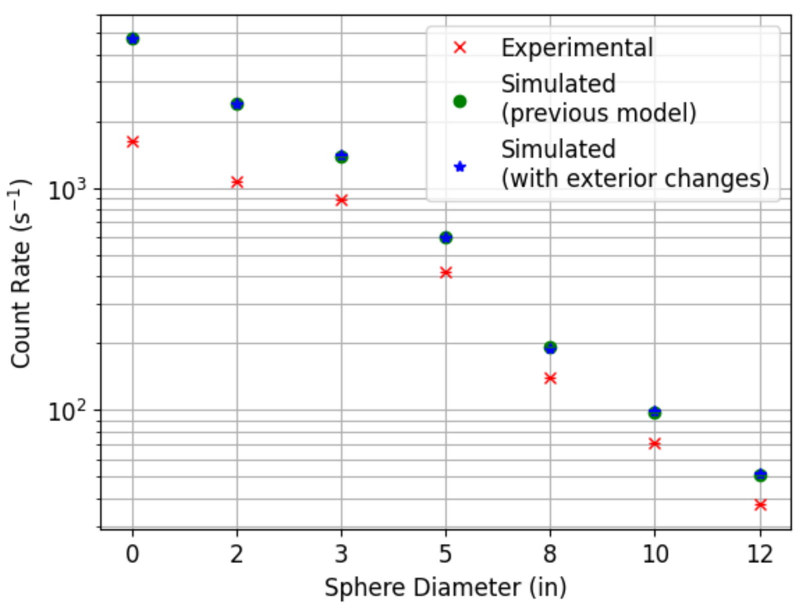

Figure 3.15: *Predicted count rate with exterior changes*

Next, the simulated k_{eff} was produced using the k-code functionality of MCNP. Measurements were conducted with the KSU reactor at criticality, so one would expect MCNP to predict a k_{eff} of unity. Following improvements to the core model and application of additional thermal scattering functions, simulated k_{eff} was reduced from 1.09 to 1.07. With these changes, the simulated count rates were observed to be much closer to the experimental count rates (Fig. 3.16). The largest discrepancy was still observed in the case without a Bonner sphere, which is impacted substantially by the thermal neutron population.

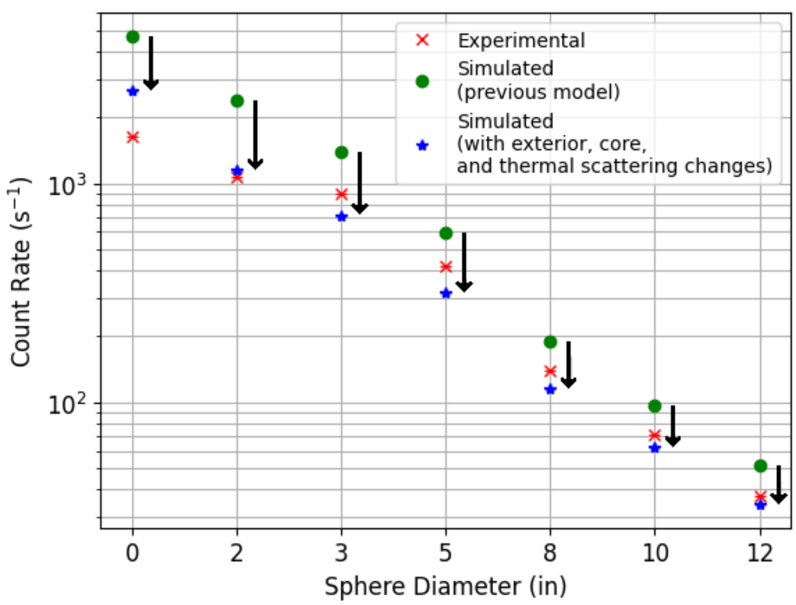

Figure 3.16: *Improvements in count rate with core changes denoted with arrows*

The Gravel[19] and MAXED[18] algorithms were applied to generate new neutron energy-dependent flux distributions by reducing the discrepancy between simulated and experimental count rate following the core model changes. Following convergence, the two distributions were plotted against the original to compare (Fig. 3.17). As expected, the energy-dependent flux density in the thermal region decreased in order to reduce the bare sphere count rates. These spectra were used to repeat Bonner sphere count rate calculations (Fig. 3.18)). Comparing the results from the two spectra, it was found that the Gravel spectrum produced a much lower error.

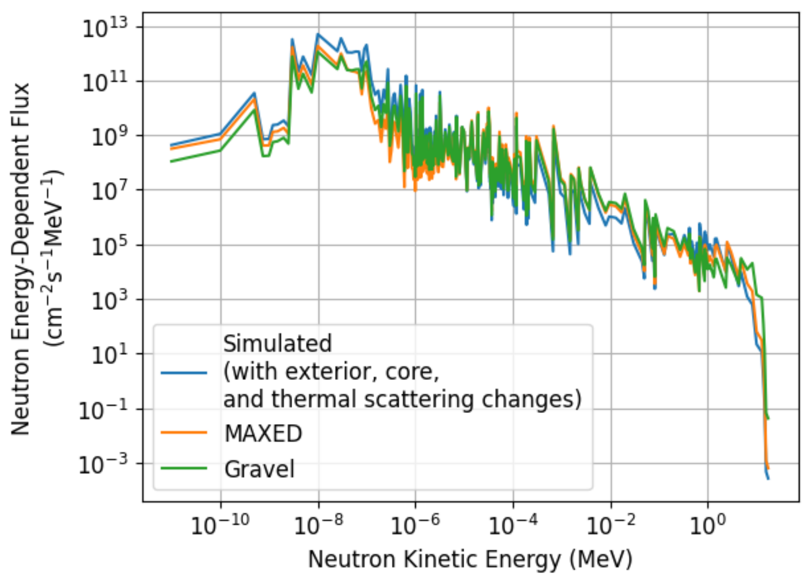

Figure 3.17: *Unfolded Spectra*

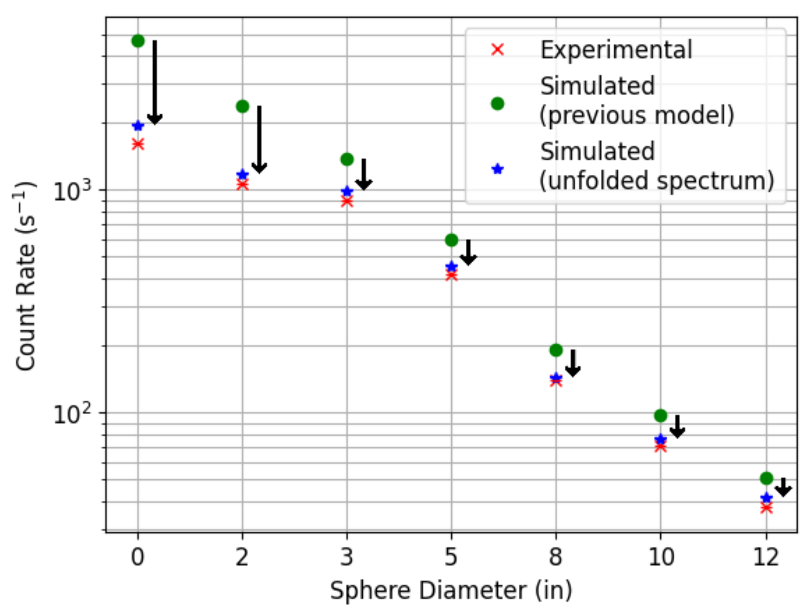

Figure 3.18: *Weighted Bonner sphere count rate, improvement from previous work denoted with arrows*

With these changes to the reactor and Bonner sphere models, the difference between the experimental and new simulated results are much smaller. This new model provides a more realistic solution for the radiation environment at the northeast beamport exit.

Chapter 4

Target and Path Simulation

4.1 Chamber Simulation

4.1.1 Theory

The next step in irradiation facility design was to simulate interactions between the neutron beam and a representative biological sample to determine the best method for obtaining uniform irradiation. The purpose of this section is to combine radiation transport modeling with biological sample rotation and translation modeling, to evaluate the uniformity in absorbed dose experienced by cells in a cell culture flask, with a goal of reducing the range of absorbed dose delivered to all cells relative to the mean absorbed dose.

Several challenges prevented model simplification. The cell sample was contained in a plastic flask, and the lid of the flask had a significant impact on the dose profile, so the container had to be defined in detail alongside the target. The complexity of the geometry also prevented a simple response function approach from being used. Additionally, the beam size is slightly smaller than the thinnest side of the flask, so irradiation could not be achieved by a single pass over each side. Instead, a more complicated series of movements is required.

The problem can be separated into simulating the radiation and simulating the flask movements. Simulating the target irradiation requires 1) representing the geometry accurately in a radiation transport simulation, 2) determining what simplifications can be made to reduce simulation time without sacrificing accuracy, and 3) storing the results in a way that can be referenced easily. Simulating the flask movement requires 1) a means of representing the movements taken by the flask as a function of continuous parameters, 2) code to reference and interpolate data from the radiation transport based on coordinate positions, 3) a process to combine a path and the dose data to find a quantifiable evaluation parameter, and 4) steps to optimize the process based on the evaluation parameter. In this work, the first step was performed using MCNP6[8] to tally dose on a grid in the sample, and the second step was done using Julia[20] to calculate the dose over the path to evaluate a set of movements based on the range of doses in the sample. Julia was chosen over other languages due to the speedup observed in performing matrix operations. Early iterations of the dose interpolation used Python, but were slowed down by the large number of matrix calculations required. It was found that Julia was more equipped to handle calculations for this work.

The objective of the following work was to create a simulation pipeline that could 1) realistically model dose in the cell sample due to the reactor source for any sample position, 2) accurately simulate the movements the sample would make to irradiate different sections, and 3) produce a series of movements that bring dose over the voxelized flask within 10% of the median.

4.2 Radiation Transport Modeling

The flask considered was a Corning $25\,\mathrm{cm}^2$ cell culture canted neck flask with anti-tip skirt and a plug seal cap (Fig. 4.1). The flask was 3.78 in. long, 2.06 in. wide, and 1.03 in. tall with a 0.59 in. diameter neck. The materials were found from the product page for the flask and dimensions were taken from both CAD drawings distributed on the supplier website and measurements of an empty flask[28]. The flask body was simulated as $0.93\,\mathrm{g\,cm^{-3}}$ pure polyethylene and the cap was simulated as $1.06\,\mathrm{g\,cm^{-3}}$ pure polystyrene. The contents were treated as $1\,\mathrm{g\,cm^{-3}}$ homogeneous water. Since the source had small angular variation[1], it was assumed that the flask could be simulated without the beam intercepting the equipment selected to move the sample. The flask was treated as if the equipment moving it was either too far away to provide significant scatter or constructed from materials with small neutron total interaction cross-sections. The assumption that the equipment was not necessary to model was validated after specific equipment had been selected.

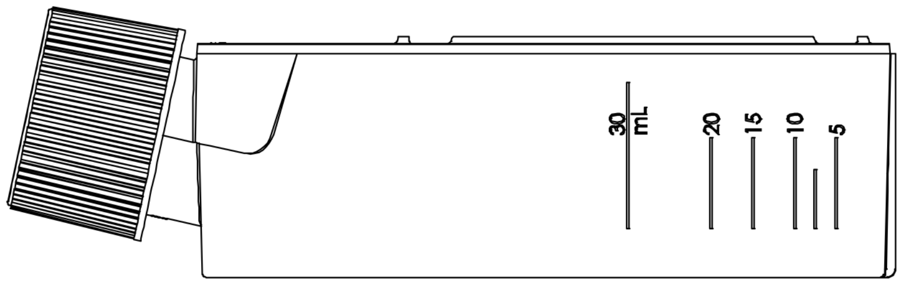

Figure 4.1: *Original CAD drawing of the flask.*

A CAD model of the flask was created using Solidworks[29], converted to an .inp file, and then imported into MCNP6[8] using the embed card. Initial simulations with the imported CAD model revealed that the neck and lid of the flask had a significant impact on the dose distribution where the beam encounters the top side. However, it was also found that many of the thin features had a negligible effect on the beam. A simplified macro-body structure that removed unnecessary features, such as a thin shell under the lid and rounded corners (Fig. 4.2), was implemented to reduce the complexity of the model and decrease overall simulation time.

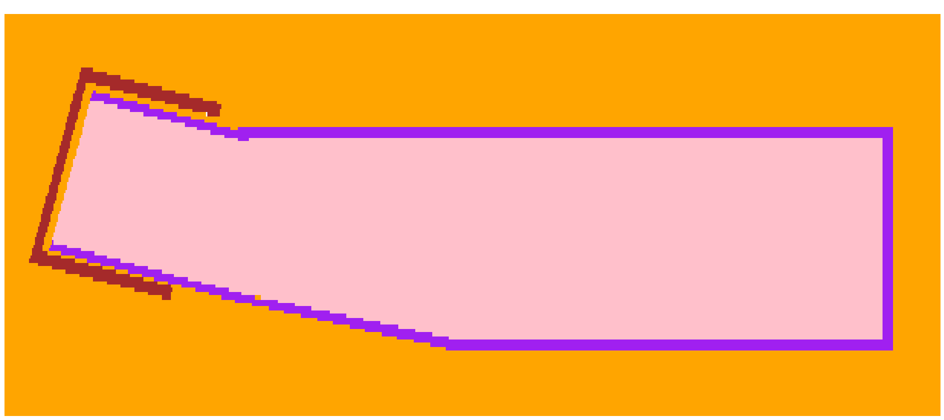

Figure 4.2: *Macro-body model of the sample in its flask with simplified lid (red) and flask body (purple) with sample region (pink).*

An early question in the target modeling was how to define the voxels for the dose tally. Voxels in a rectangular mesh have equal spacing, equal volume, and can be easily stored as an array mapping indices to coordinates. If the entire flask was filled then the target would no longer completely fill a rectangular mesh. The sample data could not be stored in an array without either voxels not having equal volume, not having equal spacing, or tallying flux outside of the sample (Fig. 4.3). Losing constant spacing or volume would mean that changes in indices would not be proportional to changes in coordinates. Additional calculations to map voxel locations would be required each time interpolation was performed, but every voxel is within the target. Voxels outside of the sample do not provide useful data, but the beam movements can be expressed as proportional changes in indices. In both cases, the data are more difficult to use and visualize. A rectangular mesh over the entire filled sample was employed to simplify coordinate translation, at the expense of masking indices outside the target (Fig. 4.3.c).

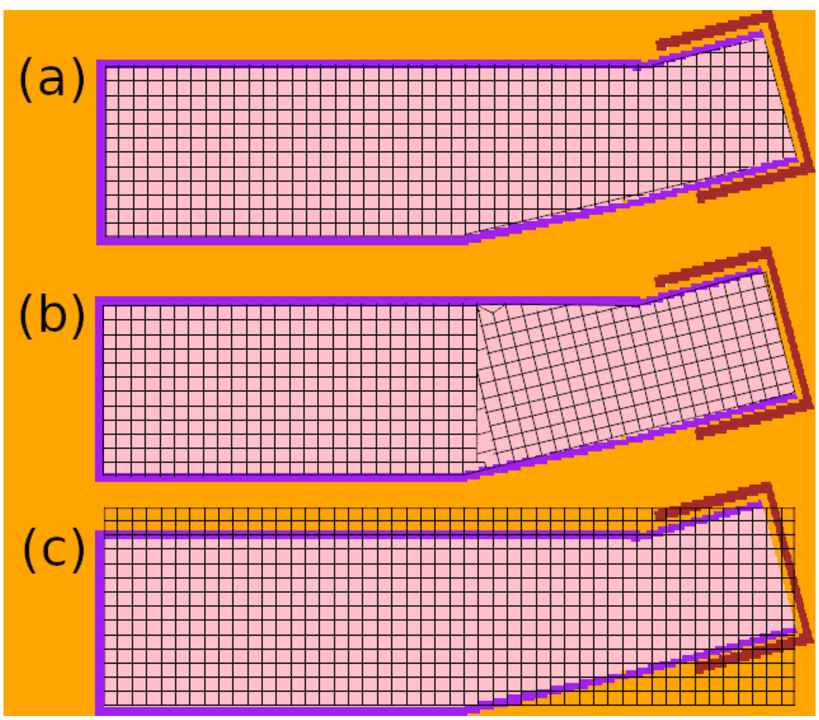

Figure 4.3: *Examples of mesh with (a) constant spacing and no unused voxels but varying volume, (b) constant volume and no unused voxels but varying spacing, and (c) constant spacing and volume, but unused voxels*

The MCNP6 input file was written to allow the flask to be freely rotated and translated through a region in front of the neutron beam. The flask was moved horizontally and vertically normal to the beam to irradiate specific portions of each side, and was rotated 90° at a time to expose each face to the beam. The source and geometry were replicated from the Bonner sphere simulations in chapter 3, replacing the Bonner spheres with the flask (Fig. 4.4).

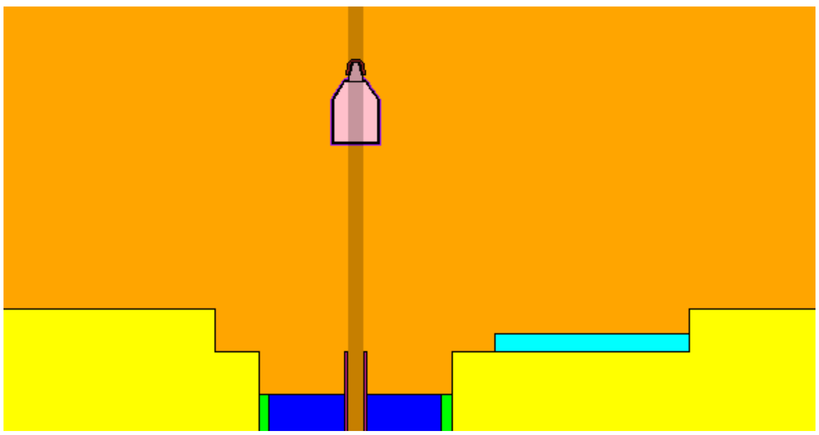

Figure 4.4: *MCNP6 model of flask positioned at (0,0), beam shaded for visualization of the beam diameter*

4.2.1 MCNP6 Settings

A major factor in simulation time was the grid spacing for voxels and the step size between simulations. Voxel size was gradually decreased as simulations were tested for time and storage required until a side length of 0.2 cm was chosen. The voxel mesh was defined as a grid with 14 grid points along the height, 26 grid points along the width, and 48 grid points along the depth. Smaller voxel side lengths were simulated, but it was decided that the additional time and storage requirements were not worth the improvement in resolution. The step sizes between simulations followed similar logic. The beam was approximately 2 cm in diameter, and a step size of 0.3 cm was small enough to capture changes in the shape of the beam with 0.2 cm voxel spacing.

The dose tally was achieved using an FMESH tally to calculate flux in each voxel, and FM cards were used to multiply by the total interaction cross-section and average energy deposited per interaction to produce average energy deposited. Two separate tallies were created for neutrons and photons. The FM card was set to use the cross-sections of the predominant material in each voxel independently. Using the predominant material meant that voxels outside the target containing only air would have much lower energy deposited and could be easily identified. MCNP default physics were not changed from Chapter 3, and ENDF-VIII[27] thermal scattering data were used for the relevant new materials, particularly to model hydrogen scattering in the cell sample and flask.

An array of neutron dose and photon dose in each voxel of the sample was created for each flask position over a series of discrete steps. The full list of positions was treated as a series of discrete time steps of a continuous path. The simplest initial path was to move the beam horizontally along each face (Fig. 4.5.a) starting and stopping outside the sample. As previously mentioned, the beam width was slightly smaller than the depth of the flask, so three passes were made along each side to gather the center of the beam with the top, middle, and bottom of each face. Initial results indicated that method lacked the ability to increase dose on the outermost layers without increasing dose in the interior, so a second path was defined with the peripheral of the beam overlapping the edges of each face of the flask to slowly increase dose on the edges (Fig. 4.5.b). The paths through the interior controlled the dose in the interior and the paths along the edges could bring the edge dose up to the interior dose levels. These paths provide a list of positions that the beam can take, however, this does not mean that the path the beam follows uses every position. Sections can be skipped, i.e spend zero seconds at a position.

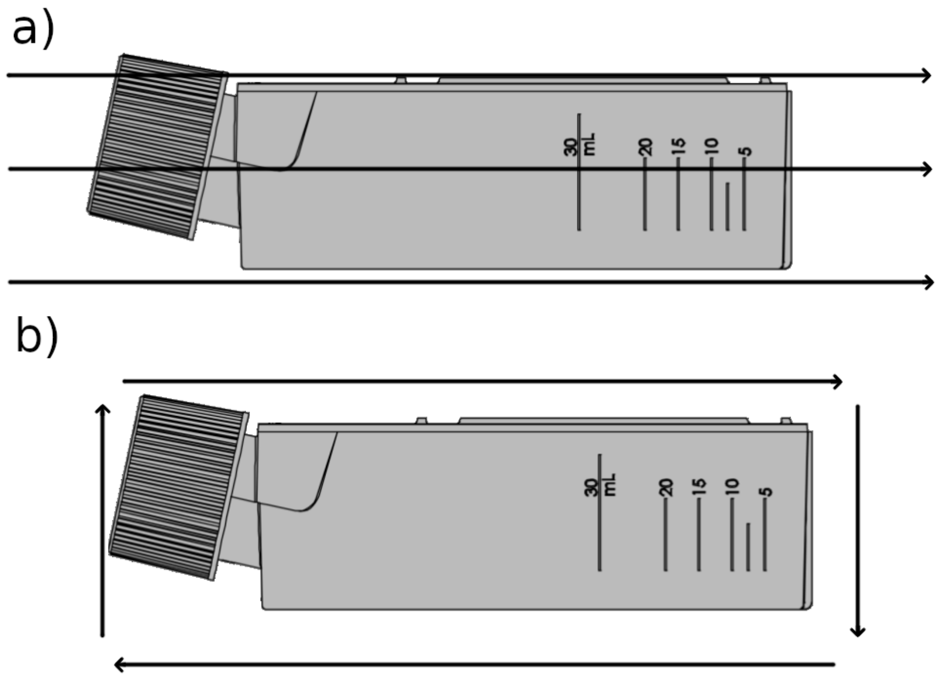

Figure 4.5: *Beam movements a) along the left face interior and b) along the left face edges.*

The simulations defined a list of positions and the corresponding dose profile. Dose profiles for positions between the predefined positions would have to be solved with interpolation. Error due to interpolation was expected to decrease as step size decreased. However, the size of the steps in distance is limited by the available file storage.

4.3 Dose Database Application

There were three main issues that needed to be resolved before accurately predicting dose and finding the optimal path. The first issue was how to separate the dose in the target from the dose in the flask and surroundings. Since the neck is angled, the mesh grid includes sections of the flask and lid below the neck and above the body. The second issue was modeling realistic movements, since the sample's movements are subject to the equipment limitations. Considering realistic movements added a series of additional movements, and velocity considerations that had to be included in the model to represent realistic irradiation. The third issue was how to approach optimizing a path with no set equation or limits. The only limit on the path was keeping the position within the simulated path and keeping speed and acceleration feasible. The relative speeds were not restricted.

4.3.1 Material Separation

As discussed previously, the array of doses contains more than just dose in the target material. The voxels containing air have much lower interaction cross-section, so the dose in air voxels are several orders of magnitude lower than in the target. The more complicated step is separating the flask from the contents. More importantly was removing the lid. The flask body and lid have densities and hydrogen contents similar enough to the target for the dose to be indistinguishable (Fig. 4.6). The flask body was within one voxel of the target, so it would impact only the outer layer. The lid was surrounded by air, meaning that most of the voxels containing the lid were a mix of material with high neutron interaction cross-section and materials with low neutron interaction cross-section which caused the dose to vary significantly.

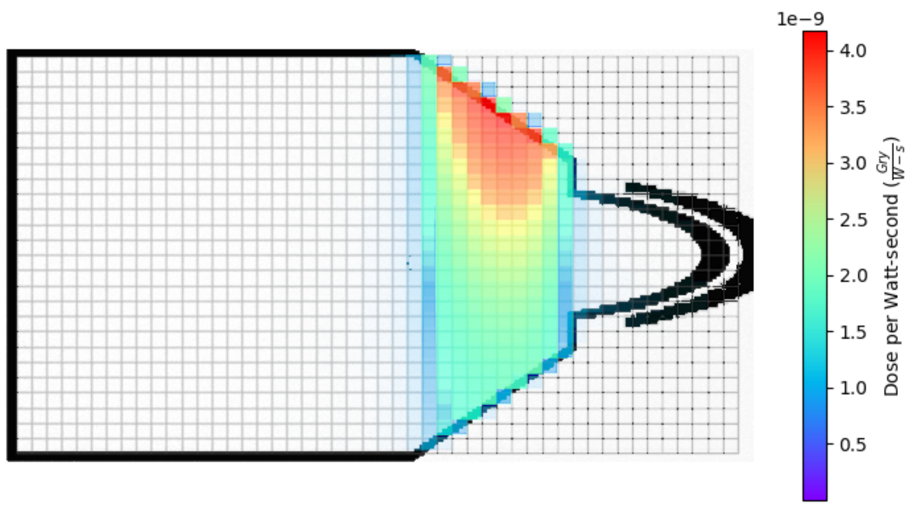

Figure 4.6: *Flask diagram under a heatmap of dose, ranging linearly from low dose at blue to maximum dose at red*

In that example, the highest dose was occurring on the boundary of the target, within a voxel containing the flask. To separate the inside of the target from the flask shell, the equations used to define the MCNP6 model were used to generate an array populated by values indicating the cells located inside the target. Generating a matrix to identify cells to exclude was limited by the resolution of the voxels, because angled sections can not be represented in a rectangular array. By removing these voxels, the analysis is limited to only dose in the target.

4.3.2 Equipment Specification

Initially, the paths were treated as if the flask could instantaneously move between positions along each side and the speed could change instantaneously to any value. In reality, the sample will be moved by a motor with velocity and acceleration limits. Additionally, the assumption that equipment would not impact the particle flux in the target needed to be verified. The hardware limitations were based on replicating movement on Velmex motorized stages following input from a controller. Equipment options were researched from Velmex, THORLABS, Precision Instruments (PI), and Optics Focus. The criteria for the system was that the platform could move at least 10 cm vertically and horizontally and fully rotate in place around the vertical axis. The expected weight of the target is under 50 g, so the weight limit is not expected to be a significant factor. The final factor is cost. With the exception of Precision Instruments, every part considered had a quote listed. Precision Instruments creates custom parts, so the cost was assumed to be higher than a prebuilt part. Instead of leaving table cells empty, unknown values in the following tables are denoted with a question mark.

Company	Movement Types	Resolution	Maximum Speed	Weight Limit	Total Cost
THORLABS[30]	X or Z	2 µm	$3\,\mathrm{cm\,s^{-1}}$	5 kg	$2280.45 x 2
PI[31]	X or Z	3 µm	$8\,\mathrm{cm\,s^{-1}}$	15 kg	? x 2
Optics Focus[32]	XYZ	1.56 µm	$2\,\mathrm{cm\,s^{-1}}$	10 kg	$3071.00 x 1
Velmex[33]	X or Z	2.5 µm	$3\,\mathrm{cm\,s^{-1}}$	4.5 kg	$701.50 x 2

Table 4.1: *Motorized Linear Platforms.*

With the exception of Optics Focus, the cheapest option for a platform that moves in the X and Z axes was a linear stage mounted vertically on a horizontal linear stage. For Optics

Focus it was cheaper to use a 3-axis linear motion device. All four options have resolutions on the scale of 1.5 µm to 3 µm and all four have weight limits exceeding the weight of the target. The cheapest listed options were the Optics Focus 3-axis device and the use of two Velmex 1-axis stages. Both meet the weight and size requirements.

Company	Resolution	Maximum Speed	Weight Limit	Total Cost
THORLABS[34]	820 µrad	$50\,°\,s^{-1}$	50 kg	$2914.92
PI[35]	17.45 µrad	$200\,°\,s^{-1}$	10 kg	?
Optics Focus[36]	175 µrad	$50\,°\,s^{-1}$	30 kg	$280.00
Velmex[37;38]	175 µrad	$40.2\,°\,s^{-1}$	2.3 kg	$1219.00

Table 4.2: *Motorized Rotation Platforms.*

All four companies offered similar rotational platforms. THORLABS offered the highest weight limit but also the worst resolution. Precision Instruments offered the best resolution and the highest speed. The rotation needed for this work is 90° increments but there was no requirement for rotation velocity or acceleration. The platforms offered by Optics Focus and Velmex were the cheapest and the resolution and speeds were within the requirements. The additional speed and accuracy that Precision Instruments offers were not assumed to be worth the additional cost.

Company	Controllers Required	Total Cost
THORLABS[39]	1	$3177.01
PI[40]	3	?
Optics Focus[41]	1	$1084.00-$1321.00
Velmex[42]	1	$2015.00

Table 4.3: *Available Controllers.*

The final limit was the controller. The motor would require a controller to be connected to input the movement commands. The primary distinction was the number of available motors per controller. With the exception of Precision Instruments, every company offered a controller that could control multiple axes at once. Precision Instruments had a single controller listed available for stepper motors that could control only one axis at a time. Optics Focus offered a controller with 3-axes for $1084.00. Technically the linear platform could move in three dimensions, so a four-axis controller could be used instead for $1321.00.

In total the estimate for THORLABS came out to $10,653.00, the Optics Focus estimate was $4435 or $4672 for the additional axis control, and the Velmex estimate was $4637. Precision Estimates did not list prices and their representative did not have a suggestion. The cheapest options were Velmex or Optics Focus, which were within $200. The additional costs for wires, screws, mounts, and shipping is likely higher than this difference, so a direct comparison can not be made without obtaining a detailed quote. The most significant difference was that both companies were contacted with a description of the requirements and only Velmex offered a solution. This work required a product that could offer accurate velocity control. Optics Focus responded that their products were not designed for accurate velocity control and did not elaborate further. Velmex provided specifications for a design and a quote, so Velmex was chosen as the equipment supplier.

Two x-slide stages (Fig. 4.7.b, Fig. 4.7.c) were used to allow for horizontal and vertical motion normal to the beam and a B59 stage was used for rotational motion (Fig. 4.7.a). The flask was raised above the rotational platform (Fig. 4.7.d) and to the right of the vertical stage (Fig. 4.7.e) to keep the beam from intersecting the motors, shown in pink. In this configuration the flask is held above the platform with a neutron-insensitive platform, depicted in the figure as an aluminum frame. The rotational platform is stabilized with a gusset mounted to the vertical stage. The combination of platforms were chosen because they offered the range of motion needed at controllable speeds. The application needed a small range of movement as well as speed variability. Alternatives had the correct ranges of movement but were designed to slowly position a sample with very low tolerances.

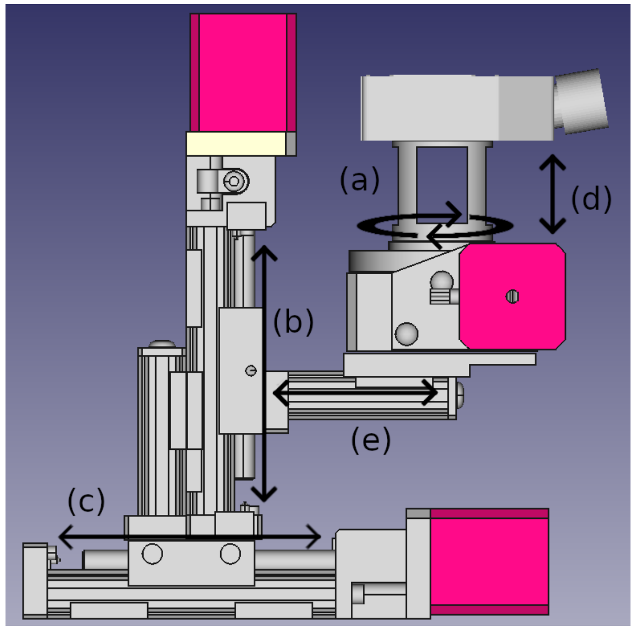

Figure 4.7: *Moving platform with flask. (a) platform rotation. (b) vertical motion. (c) horizontal motion. (d) vertical clearance. (e) horizontal clearance*

Finally, MCNP6 simulations were executed to determine if the presence of the stages would change the neutron energy distribution in the sample. The target position that brought the beam closest to the rotational motor was positioning the beam below the target. The reactor source term was collimated to the point that the equipment could complete its full range of motion without entering a region higher than 1% the peak beam intensity if the proper clearance was created (Fig. 4.8). The vertical and horizontal clearance required was proportional to the beam size. An uncollimated beam would require a larger gap between the flask and nearby motors.

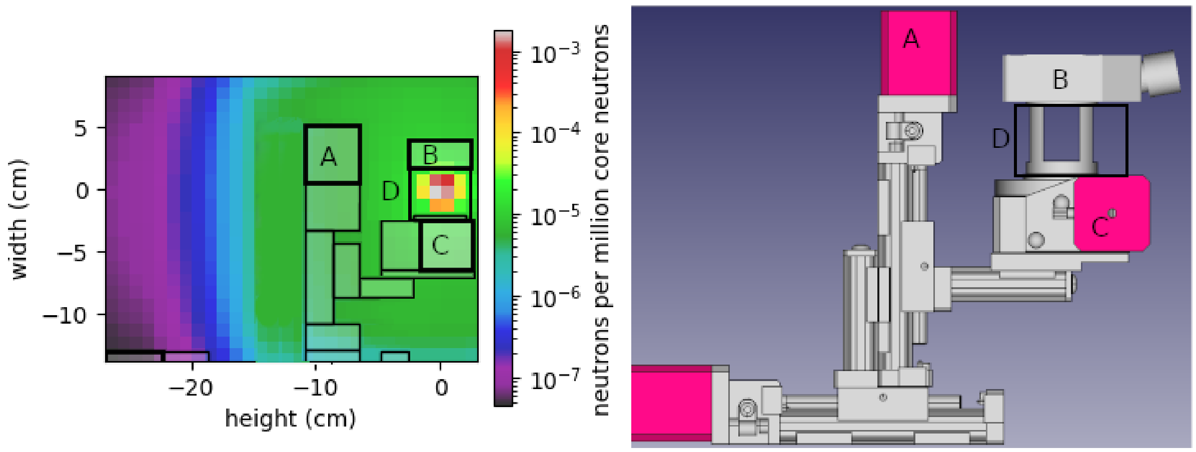

Figure 4.8: *Neutron current integrated over energy along the CAD diagram in the same configuration.*

4.3.3 Controller Simulation

By far, the most significant effect from considering the controller was the addition of dose from starting and stopping each movement. The path could be separated into two types of movements: intentional irradiation and transitional. There was a specific series of movements done to intersect specific portions of the flask with the beam that caused intentional dose, defined in the previous section (Fig. 4.5). However, doing so does not include transitional movements from the end of a path to the start of the next.

To replicate the movement of the stages, the idealized path was converted to directions in the controller's syntax. Then, to replicate the controller response, a second script was created to read the input and replicate how the controller interprets each command. Each movement is separated into a speed definition and a distance. The motors then move at a constant acceleration until the speed is reached or a stopping point is reached. The controller movement could then be translated back into a series of positions, which has an associated dose profile. These steps defined the simulation pipeline from ideal path to controller path to dose. Converting back to a list of positions allowed the code for dose interpolation to be used for both idealized paths and the controller responses.

The most important component of the controller simulation was accounting for stopping points. For a constant acceleration, the controller has a maximum speed that the motor can safely accelerate to almost instantaneously. Below this velocity, the acceleration behaves like a step function. When the next movement ends with a complete stop, the controller calculates the distance required to slow down from the current speed to a stopping point safely. If possible, the motor will accelerate to the input speed and then decelerate fully. If there is not sufficient distance, it will approach the highest speed it can decelerate from and then fully decelerate. Instead of assuming velocity with instantaneous acceleration, the velocity follows trapezoidal patterns based on proximity to the next point (Fig. 4.9). The acceleration pattern becomes more important the quicker the platform is expected to change speeds. This means however, that if the continuous movement does not change speed drastically, then it can be treated as having instantaneous acceleration.

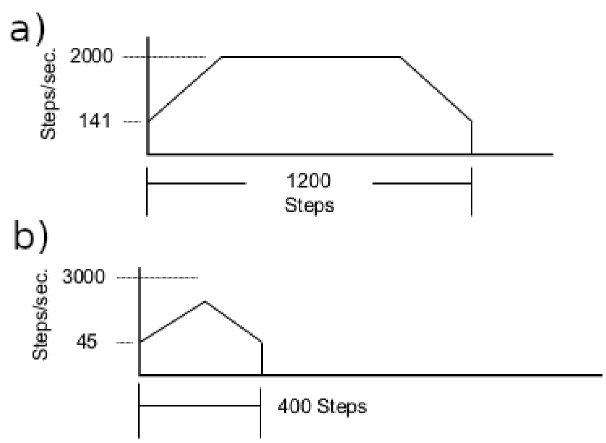

Figure 4.9: *Example of platform reaching full speed (a) and failing to reach full speed (b)*[5]

In Figure 4.9.(a), the motor is instructed to accelerate at $8000\,\mathrm{steps\,s^{-2}}$ up to $2000\,\mathrm{steps\,s^{-1}}$ and stop at $1200\,\mathrm{steps}$. At that acceleration, the motor is able to instantaneously accelerate up to $141\,\mathrm{steps\,s^{-1}}$. The motor has enough distance to fully accelerate and fully decelerate, so it accelerates up to $2000\,\mathrm{steps\,s^{-1}}$, maintains the speed, and then decelerates to a stop. In figure 4.9.(b), the motor is instructed to accelerate at $4000\,\mathrm{steps\,s^{-2}}$ up to $3000\,\mathrm{steps\,s^{-1}}$ and stop at $400\,\mathrm{steps}$. It does not have enough distance to reach the desired speed, so the motor accelerates the first $200\,\mathrm{steps}$ up to $1266\,\mathrm{steps\,s^{-1}}$ and then decelerates for the remaining 200 steps.

By default, the controller treats each movement as having a stopping point; this can be overwritten if a series of movements do not include a change in direction or a complete stop. The paths through the interior and edges could be read in as a series of velocity changes with a single continuous movement, but many of the transitional movements required two dimensional movement, which requires starting and stopping motors. The primary difference between the ideal path and the controller response was the time spent making these movements. The starting and stopping points were moved to the edges of the platforms ranges, so the beam would be as far from the target as possible when changing direction (Fig. 4.10).

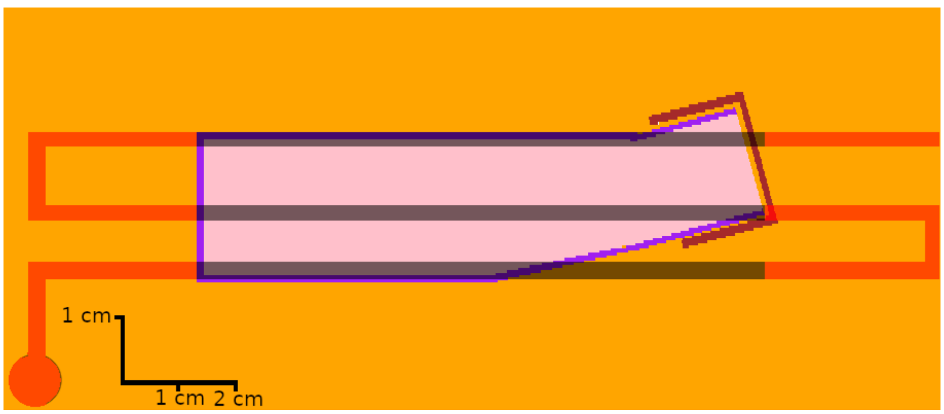

Figure 4.10: *Path followed from start (circle) along starting and stopping points (red) and continuous movements (black)*

A similar path is taken for the movements along the edges. Each movement starts at one side of the movement range, moves to the opposite side, then moves along the corners to be in position for the next movement. By doing so,dose imparted during time spent moving between intentional movements is minimized.

4.3.4 Dose Distribution Optimization

For this irradiation, the objective was to produce a path for which the dose in every voxel would be predicted to be within 10% of the average dose. There were two ways this problem was approached. First, the problem was approached as a numerical analysis problem. Second, the problem was approached as a linear algebra problem. In both cases, the path started out as a constant velocity over the entire duration and was modified to a series of step functions.

The first approach was to view the dose as a parametric integral. The dose distribution in column i and row j was treated as a cell in a matrix ($D_{i,j}$). For each simulated position, the tally in each voxel was the instantaneous values of the dose rate with respect to time ($\dot{D}_{i,j}$) as a function of position.

$$D_{i,j} = \int_0^T \dot{D}_{i,j}(\boldsymbol{P}(t))dt \tag{4.1}$$

The path was defined as a function of time ($\boldsymbol{P}(t)$) (Eq. 4.3). This meant that the dose rate could be treated as a function of time and integrated with respect to time to calculate the final dose distribution (Eq. 4.1). The speed at any point was defined as a series of step functions ($\boldsymbol{V}(t)$) (Eq. 4.2). In order to allow for sections of the path to be skipped, a maximum velocity was placed. For each face there were seven sections, three through the interior and four along the edges, and each section was split into subsections with constant velocity. If the velocity in every subsection of a section was at or above the maximum velocity, then the code interpreted that section as being skipped.

$$\boldsymbol{V}(t) = \left\{ \begin{array}{ll} \boldsymbol{V_0}, & t < t_0 \\ \boldsymbol{V_1}, & t_0 < t < t_1 \\ \ldots & \\ \boldsymbol{V_n}, & t_{n-1} < t < t_n \end{array} \right\} \tag{4.2}$$

$$\boldsymbol{P}(t) = \int_0^t \boldsymbol{V}(u)du \tag{4.3}$$

The integral was evaluated by creating a list of positions and interpolating the dose at each step of a series of discrete steps and summed to calculate the Riemann sum. The maximum velocity and step size in time were chosen such that the largest movement (0.15 cm) was smaller than the cell length (0.3 cm). The optimization for a given path was performed by evaluating the integral and tuning the velocity magnitudes, and new paths were defined by manually adding new step functions at points where the beam encounters the voxels with the highest or lowest dose. These equations did not require the dose rate to have a closed form equation, so the dose arrays could be shifted by rows and columns to approximate the beam moving small distances between points. Taking the weighted sum of the dose at two points does not necessarily preserve the shape of the dose profile. If the points are too far apart then the average creates a bimodal distribution, but if the entire distribution is shifted then the shape is preserved (Fig. 4.11).

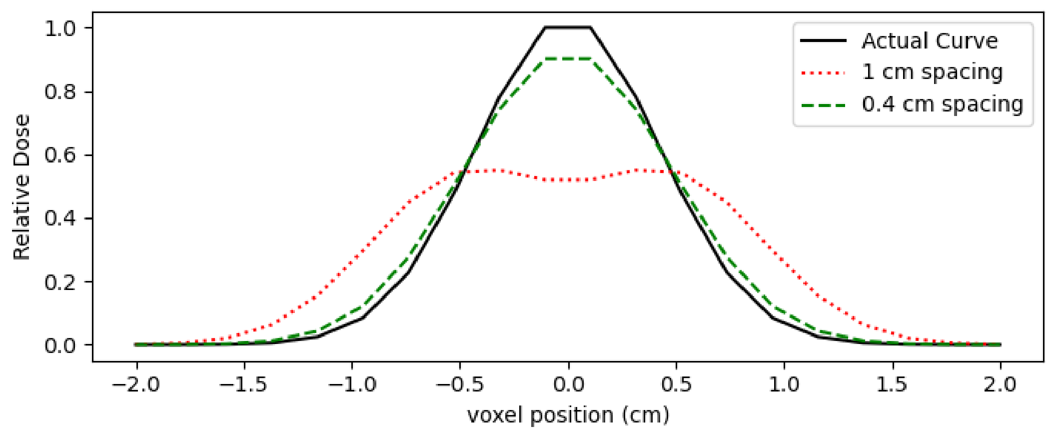

Figure 4.11: *Example of weighted sum of normal distributions with the same distribution in dose and voxel point as the simulated data*

The interpolation was separated into several steps. First, the two nearest simulated positions were located by index which gave two matrices on either side of the desired position (Fig. 4.12.(a)). Next, for each simulated position the matrix is shifted in rows and columns to gather the beam with the new beam position. Beam positions moved toward the center of the target face will have missing data that needs to be filled to complete the beam (Fig. 4.12.(b)). Then a new matrix was defined that gathered the center of the beam with the first row or column. This would then have either a quarter of the beam in a corner or half of the beam along an edge depending on which parts of the beam were missing. The new matrix can be reflected about the corner or edge to create a complete beam profile. Then the original shifted matrix and reflected matrix can be combined to fill missing data with the reflected estimate (Fig. 4.12.(c)).

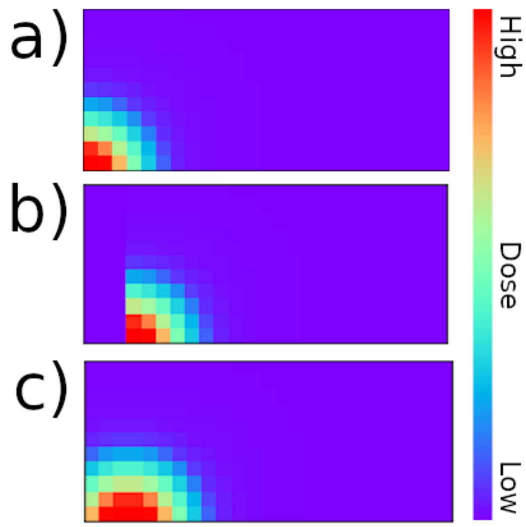

Figure 4.12: *Examples of a) the unshifted beam, b) a beam shifted toward the center with missing data, and c) the completed beam*

Reflecting the dose profile can be done with every point with the center of the beam within the target, because at least a quarter of the beam is within the target. This method was less accurate for positions where the beam is not normal to the intersecting face of the target. This includes the tapered sections, the neck, and the lid. The inaccuracy is due to two issues. First, the lid does not have a constant thickness. The walls of the flask are uniform thickness, so the beam passes through a constant thickness of plastic no matter where it intersects the target, keeping the shape nearly constant. Because the lid is not uniform thickness, the beam profile shape is not constant (Fig. 4.13). The beam has to pass through more plastic if it hits the sides of the lid than if it enters through the top of the lid, which distorts the shape of the dose profile.

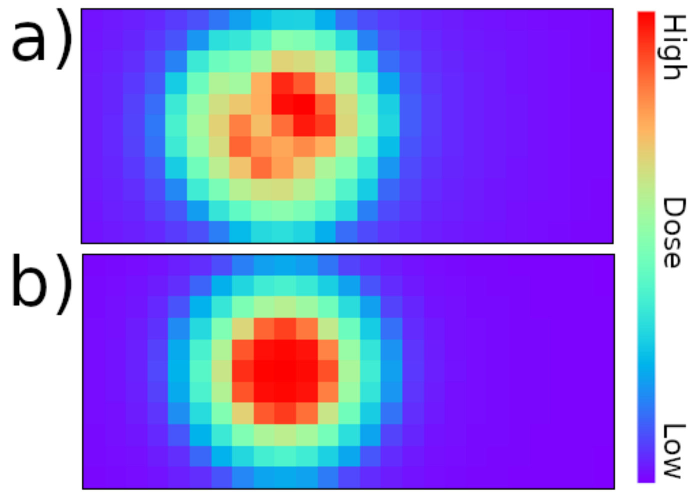

Figure 4.13: *Dose profile a) intersecting the lid and b) intersecting normal to a face*

The second issue is that the depth of the intersection of the beam and target isn't constant on angled faces. The interpolation assumes that the matrices only need to shift in two dimensions because the beam is entering normal to a flask face. However, in the angled sections, the entrance point is changing in three dimensions (Fig. 4.14). For sections where the angled surface is flat, the interpolation could be done in three dimensions because it is performed on a plane parallel to the flask. However, at the edges, this is no longer true and the beam is treated as entering either outside the flask or too deep within the flask.

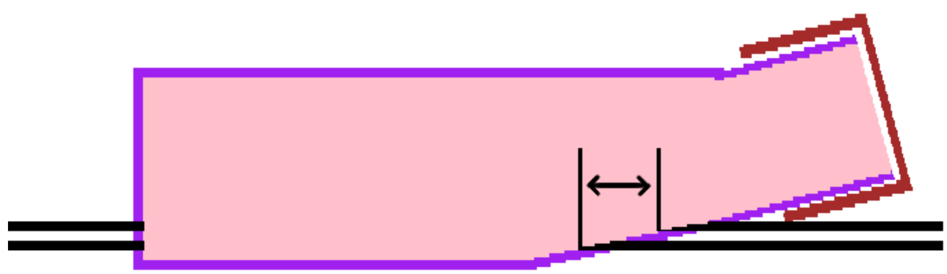

Figure 4.14: *The beams entering the lid side enter at different heights than the beams entering the opposite side.*

One method of performing interpolation on a beam would be by projecting a constant beam shape onto the flask. This would mean that the dose in a voxel would be calculated using the horizontal and vertical distance from the center of the beam and the depth from the entry point at the same horizontal and vertical distance. This would define dose as a function of radial distance from the center of the beam and a depth traveled in the target. This would only be accurate if the dose in a voxel were not dependent on the dose in adjacent voxels. Along the backside of the flask the beam enters the flask at a constant depth. The simulated dose profiles intersecting this side were compared by plotting the dose along a vertical line and horizontal line passing through the center of the beam, as well as comparing the dose attenuation with depth along the center of the beam (Fig. 4.15). The beam did not exit the flask at a constant depth which caused zero dose values at depths above 5.8 cm, but aside from this the distributions match very closely.

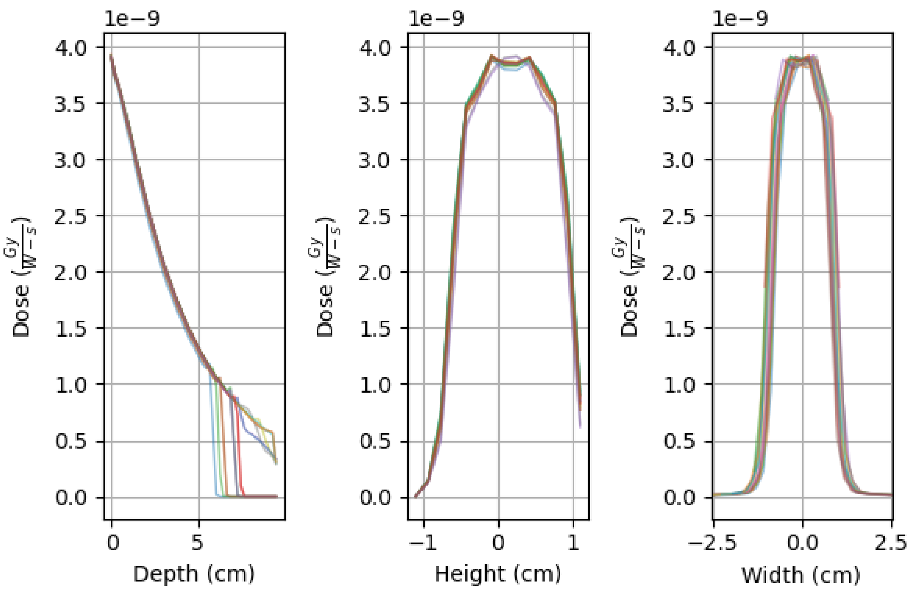

Figure 4.15: *The distribution in dose on the back side, Dose plotted horizontally and vertically through the center-line on the first layer, and dose varying with depth along the center of the beam.*

Along the front of the flask, the entry point was not constant. The beam either entered through the lid or through a face at approximately 45° from the beam. Dose was strictly decreasing with depth, so the depth the beam entered at was the same as the depth of the maximum dose. So the dose distributions vertical and horizontal to the center of the beam could be plotted using data on the outermost layer (Fig. 4.16). The dose in the first 0.2 cm layer was not a consistent shape between beam positions. This shows that in the presence of angled faces, the dose cannot be approximated with only radial position and depth into the target.

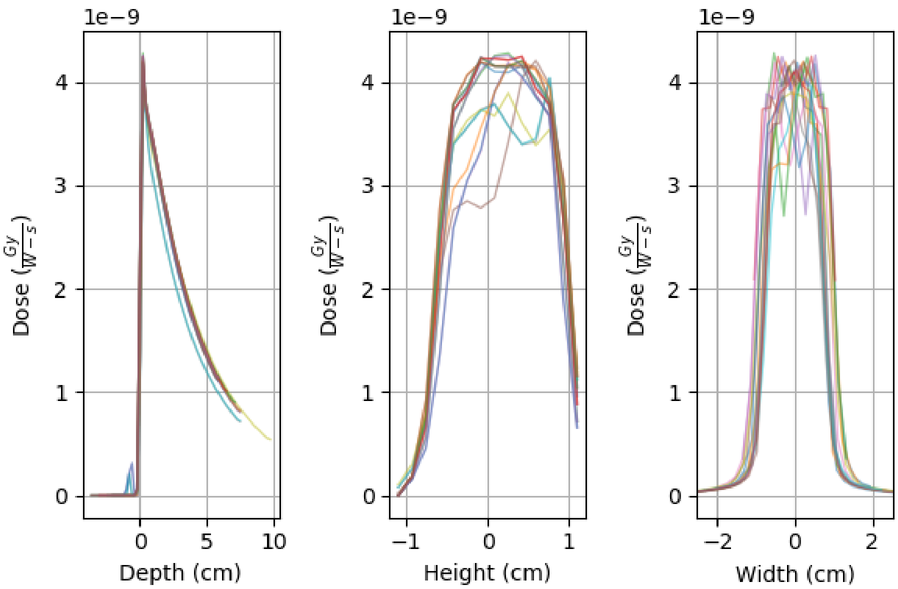

Figure 4.16: *The distribution in dose on the front side, Dose plotted horizontally and vertically through the center-line on the first layer, and dose varying with depth along the center of the beam.*

Overall, this approach was not appropriate for the shape of the flask. A more appropriate shape for this approach would be a rectangular target to avoid any issues with the beam hitting the target at an angle. Furthermore, the inaccuracy with edge cases may be reduced if the tally extended further outside the target into a similar material. The issue with edges is that there is no source of neutron scatter back into the flask from sides with air. If the transition is from water to a similar material, there is not a drastic change in interaction rate. If the dose profile does not change along the edges of the target, then the dose interpolation would be even more accurate.

The second approach was to view the dose as a system of linear equations. If the path is a series of step functions and dose between points is interpolated linearly, then the integral simplifies to a Riemann sum proportional to the total time spent irradiating the target. The interpolated positions were always between two points, so the dose rate ($\dot{D}_{i,j,k}$) at the interpolated point can be calculated as a weighted sum of the two nearest predefined positions (Eq. 4.5-4.6)

$$r_i = \sqrt{{x_i}^2 + {y_i}^2} \tag{4.4}$$

$$A = \frac{r(t) - r_0}{r_1 - r_0} \tag{4.5}$$

$$\dot{D}_{i,j,k}(x(t), y(t)) \approx (1 - A) \times \dot{D}_{i,j,k}(x_0, y_0) + A \times \dot{D}_{i,j,k}(x_1, y_1) \tag{4.6}$$

Each step function magnitude is associated with the time spent over a set of beam positions (Δt_n). For each step function (n) there is a dose matrix ($\dot{D}_{n_{i,j,k}}$), so the dose rate matrix ($\dot{D}_{i,j,k}$) for a cell in position (i,j,k) can be calculated as the sum of the contributions from each simulated position (Eq. 4.7). The time duration associated with each step function velocity can be calculated as the ratio of the magnitude of each step function to the distance traveled at that magnitude (Eq. 4.8).

$$\int_0^t \dot{D}_{i,j,k}(x(u), y(u))dt \approx \sum_{n=0}^{N} \Delta t_n \times \dot{D}_{n,i,j,k} \tag{4.7}$$

$$\sum_{n=0}^{N} t_n \times \dot{D}_{n,i,j,k} = \sum_{m=0}^{M} \frac{\Delta r_m}{V_m} \times \dot{D}_{m,i,j,k} \tag{4.8}$$

Each dose matrix is three dimensional and the dose rate for each voxel is the sum of the contributions from the N step functions. The system of equations can be expressed as a two dimensional matrix by assigning each step function with a row u and assigning each voxel index, (i,j,k), to a unique column v (Eq. 4.9). The equation for the relative dose matrix ($\boldsymbol{b}$) can be separated into a matrix of dose rates ($\boldsymbol{D}$) with rows for each beam position and columns for each voxel, and a vector of relative time duration ($\boldsymbol{t}$) with rows for each beam position (Eq. 4.11). The distance traveled over each step function (Δr_u) is a constant and the velocities are the only parameters changing, so the doses can be multiplied by distance so that the time vector is solely dependent on velocity.

$$\boldsymbol{D}[u,v] = \Delta r_u \times \dot{D}_v[u] \tag{4.9}$$

$$\boldsymbol{t}_i = \frac{1}{V_m} \tag{4.10}$$

$$\boldsymbol{b} = \boldsymbol{D} \cdot \boldsymbol{t} \tag{4.11}$$

The algorithm operated on a time duration, not the step function velocity, so the step function velocities were inverted and multiplied by 1 cm to construct a vector of relative time duration. The choice of 1 cm was because the scale of the geometry was on the scale of centimeters. There was not a constraint placed to disallow for a value of zero. A zero value was treated as if the section were skipped, i.e spending 0 s in the region associated with the velocity. The optimizing function ($\boldsymbol{f}$) was defined as the difference between the current median dose ($\bar{\boldsymbol{b}}$) and the dose in each voxel with the constraint that every time value is non-negative (Eq. 4.12).

$$\boldsymbol{f} = \boldsymbol{b} - \bar{\boldsymbol{b}} \tag{4.12}$$

For small changes in magnitude, this equation is linear with respect to each step function magnitude, so the Jacobian of the function ($\boldsymbol{J}$) is just the original dose matrix (Eq. 4.13). Using the squared difference would produce a nonlinear Jacobian, which would have to be either recalculated or approximated every iteration. In each iteration (m) of the algorithm, the optimization function was multiplied by the Jacobian to produce a vector of parameter changes that reduce the error in each voxel. The delta vector was multiplied by a learning rate (l) and subtracted from the previous time vector to produce the next guess (Eq. 4.14).

$$\boldsymbol{J}_{u,v} = \frac{\delta \boldsymbol{f}_v}{\delta \boldsymbol{t}_u} = \boldsymbol{D}_{u,v} \tag{4.13}$$

$$\boldsymbol{t}_{m+1} = \boldsymbol{t}_m - l \cdot \boldsymbol{J} \cdot \boldsymbol{f}_m \tag{4.14}$$

The dose scales linearly, so normalizing the magnitude vector does not change the dose distribution relative to the mean. After every iteration, the time vector was normalized to keep the scale of the parameters the same. The final result can be scaled to match the velocity and acceleration limits of the equipment. The equipment the limits were based on used step motors with a limit of $3.5\,\mathrm{cm\,s^{-2}}$ acceleration, so the final vector of velocities could be scaled to treat the acceleration as approximately instantaneous. This gave a new set of path coordinates within the feasible range of speeds that in theory reduces the total error. By precomputing the Jacobian, each path was evaluated by a single matrix multiplication instead of the sum of thousands of individually calculated arrays.

The controller movements were incorporated by calculating the dose for all of the transitional movements and storing them as a single array, which acted as an intercept term for the linear equations above. The additional controller movements connected the passes made through the flask by traveling along the edges of the hypothetical platform's range of motion. The controller was treated as if it had 2 in. of vertical range and 6 in. of horizontal range. This gave 0.72 cm of clearance horizontally on each side and 1.44 cm of clearance vertically on each side.

Prior to any modification, multiple random variations of the current path were all run through the optimization algorithm independently. Of the optimized paths, the results for the set of paths with the lowest deviation were isolated. The collection of paths gave a distribution for each path parameter. The original path parameters were random, but the new distributions only contained the parameters for the best performing paths. This visualized which parameters were consistently 0 for good performing paths, which parameters were consistently the greatest for each path, and which parameters had the greatest variation.

The final dose array for the best path was plotted as both heatmaps of each layer of voxels, as well as a histogram of the voxel doses. The histogram visualizes outliers in the dose distribution and the heatmap locates where the minimum and maximum doses occurred.

The modifications to the path sections were based on three factors. The first factor considered was the distribution in value for each parameter. The relative impact of the beam is not constant over the entire path, meaning that the impact of a path section may vary significantly over the subsections contained in it. If the parameters were constantly the smallest for every good performing path, then the best performing paths all skipped through that section. However, that does not mean that the entire section should be skipped. Often, parameter values were decreased by the presence of outlier voxels with the greatest dose rates. For illustration, parameter d8, one of a ten parameters used to control velocity for the beam following the edge of the bottom face of the flask, was selected as a parameter representative of a skipped section (Fig. 4.17).

The error was defined as three standard deviations (3σ) relative to the median to limit the impact of outlier voxels. The beam is not normal to the neck of the flask, which leads to the dose rate varying much more in the neck than the rest of the target. This created single voxel outliers that were not characteristic of the dose distribution as a whole. The median was used in place of the mean, and the standard deviation for the data fit to a normal curve was used in place of the difference between the maximum and minimum dose. Both changes were made to diminish the effect of outliers on the judgment of each path. An example of uncharacteristic outliers would be the outliers in figure 4.19. The x axis extends to 0.028 relative dose because there was at least one voxel with that dose. Using a median of 0.02 and three standard deviations of .002, the error would be approximately 10% of the median. Using a mean of 0.02 and maximum deviation of .008, the error would be approximately 40% of the mean.

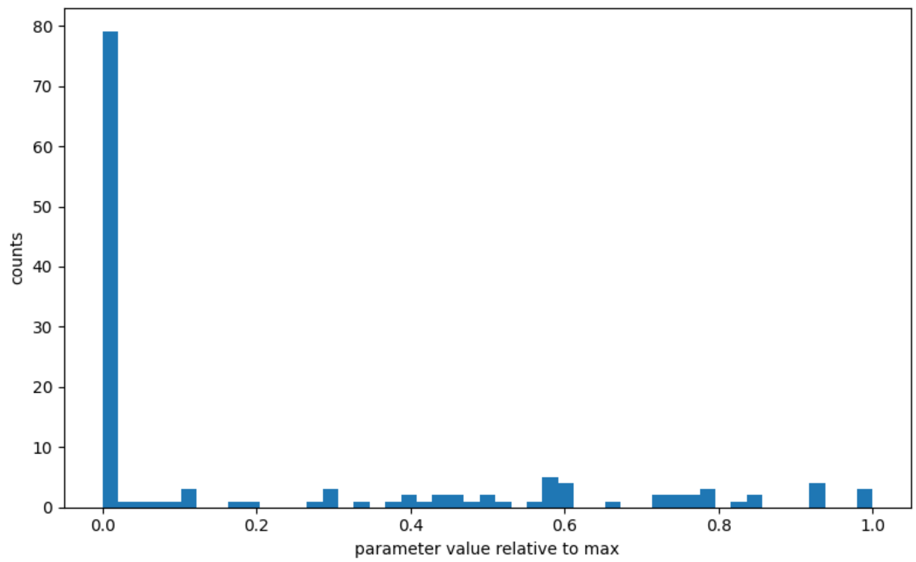

Figure 4.17: *Variation in parameter d8*

A large majority of optimized paths skipped this section or spent close to no time in it. This means that this parameter most likely only impacted a small section of the flask where some peak in dose occurred. Increasing the time spent in that section would further increase the variance. In this way, the distribution in parameters can be linked to the section of the flask most impacted by that section of the path (Fig. 4.18).

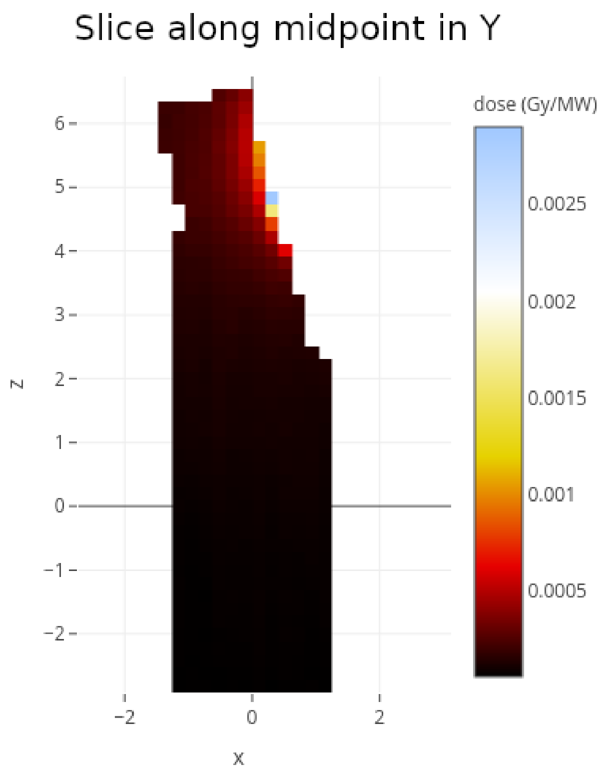

Figure 4.18: *Impact of parameter d8*

As expected, the parameter impacted a very small volume. The plotted dose for this parameter was concentrated in the base of the neck of the flask. The smaller thickness of this section caused the derivative of dose with respect to time to be much greater, which generally led to sections covering the volume to be skipped or passed quickly.

The second factor was the distribution in dose in the flask. Generally, the dose distribution was skewed to some degree, which shows if the variance was dominated by dose above the mean or dose below the mean (Fig. 4.19). In this work, the variance was dominated by voxels with low doses. Thus, the best paths increased dose to low dose voxel rather than decrease dose in high dose voxels. The dose distribution translates to volumes of the flask which contain the dose extremes, which once again translates to a section of the path. As previously mentioned, this often translated to volumes of the neck or edges which could be identified by the first factor as well.

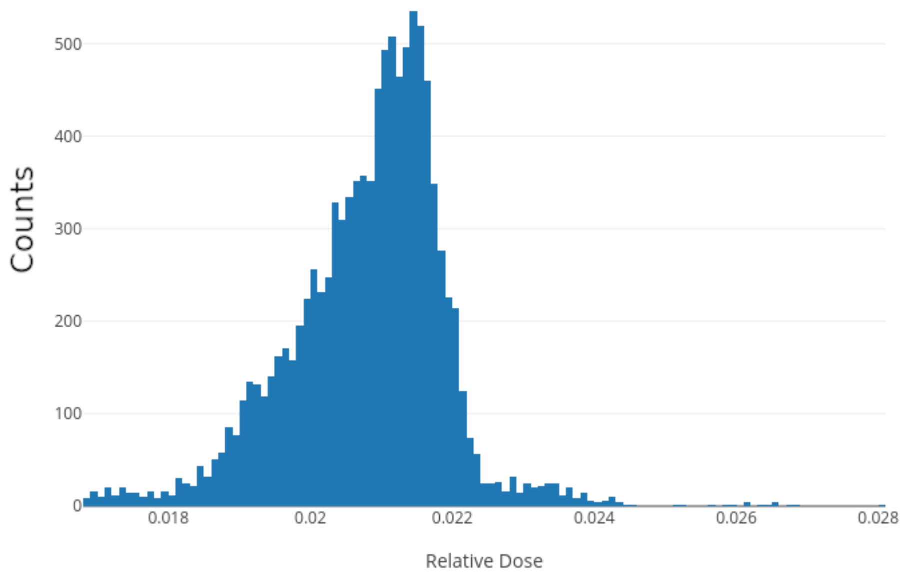

Figure 4.19: *Variation in Dose in the Final Dose Array*

The final factor was the physical dimensions of the flask. The thickness of the flask changes as the beam moves throughout its path, which changes the energy dependent flux in the voxels. For the beam intersecting the target normal to the left and right sides, the neutrons that intersect the center of the neck are higher energy on average than the neutrons intersecting the center of the body. Beam positions with thinner volumes to intersect have higher dose rates than the thicker volumes. This provides insight into where the step function velocities should change. By examining these three factors, the changes in path could be based on relating which identified parameters had the highest impact on sections with extremes in dose and how the changes in path related to the dimensions of the flask. This often meant splitting sections of the path with constant velocity into smaller subsections or shifting where velocities change to group path sections with similar impact and optimal speed.

4.4 Linear Path Simulation Results

For illustration, the optimization algorithm was applied to a path of 100 sections with the initial magnitudes randomly distributed, one path with uniform values, and the non-random path that was updated along with the path definitions (Fig. 4.20).

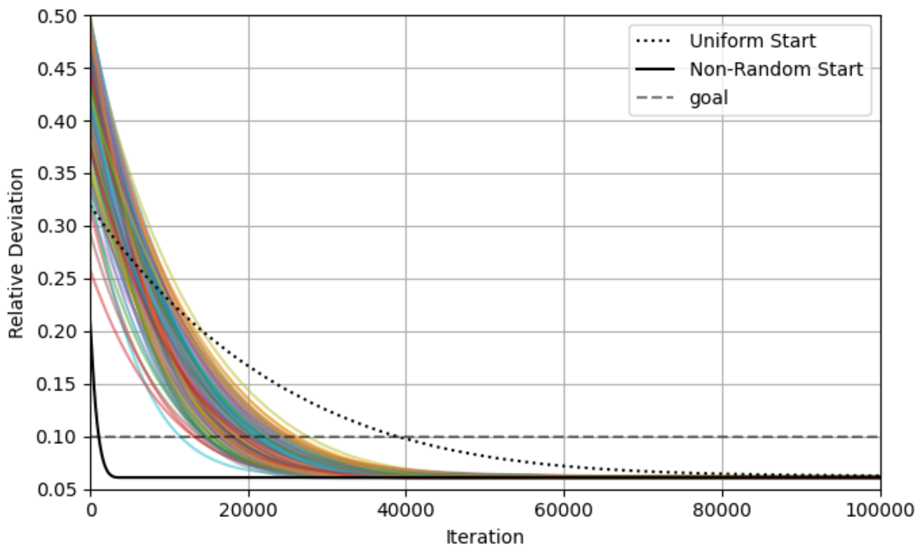

Figure 4.20: *Ratio of three standard deviations to the median dose for 100 random and two non-random initial guesses.*

The first conclusion is that the algorithm was able to converge below 10% for every random path generated, which demonstrates the versatility of the algorithm. A common issue with using Newton's method to minimize a function is the radius of convergence. For many functions there is a range of values that the algorithm converges from, and outside of that range the algorithm diverges. This behavior isn't present despite the differences in the initial parameters. Another less obvious conclusion is that the paths converge to a similar relative deviation, but the parameters are not converging to the same values. Thus, there are many local minima found with approximately the same error. One potential shortcoming of finding a solution with an iterative process is that the algorithm may be converging to a local minimum. The only way to prove that a solution is the global minimum is to either compute the error for each parameter combination on a fine grid or find some functional fit to the error and show that the function is minimized there. Either way there would be no point to using the optimizing algorithm. Instead of searching for a single unique solution, it is more practical to use this algorithm to find solutions that are below some predefined threshold in error.

Chapter 5

Extension To a Rotating Target with Additional Constraints

5.1 Random Position Machine Application

One possible extension of this work is repeating the analysis for a rotating target. One such example would be irradiating a biological sample being exposed to microgravity via a random position machine. The final portion of this work is the development of random paths for such a machine that produces a desired gravity vector and a discussion of how these restrictions impact the ability to simultaneously irradiate the target uniformly. This application was performed in three steps. The first step covered the mathematics behind generating paths for desired gravity. The second step covered the application of realistic movements. The third step covered an analysis of expected obstacles and results of initial dose simulation.

5.1.1 Path Generation

First, the limitations of the random position machine had to be determined and recorded. The machine used two stepper motors with an Arduino to provide input to each motor (Fig. 5.1). The coordinate system was defined such that the x axis was normal to the front of the machine, the y axis was normal to the sides, and the z axis was vertical. This meant that the outer motor rotated about the y axis and the inner motor rotated about the x axis relative to the flask. The motor positions were defined as angles θ and ϕ for the outer and inner motors respectively. The (0,0) position was defined as the point where the gravity vector is entirely in the z-direction. The objective was to define a general function for a path with multiple sets of parameters, and predict the average gravity vector relative to the flask. The parameters would then be sampled randomly to create a series of random paths that all are expected to give the same gravity relative to the flask.

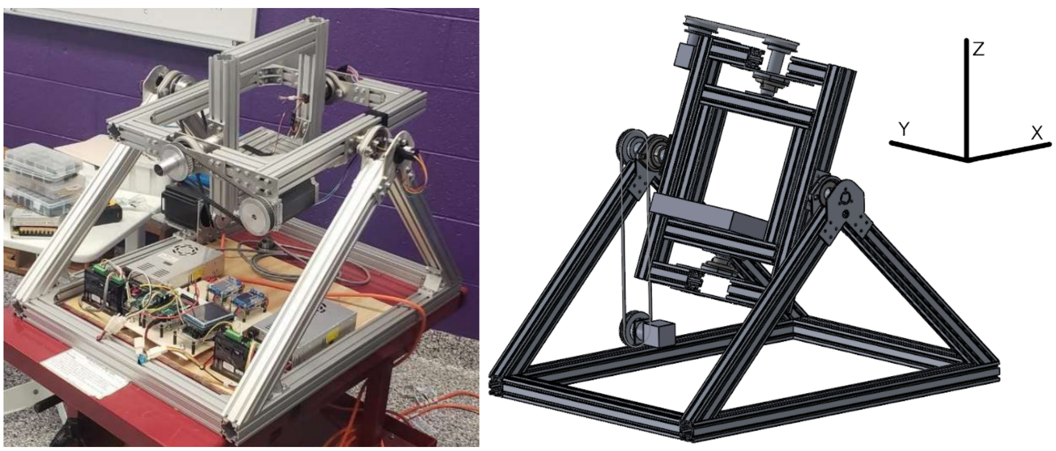

Figure 5.1: *Random position machine in (0,0) position and CAD model with coordinate system.*

The motors are capable of moving simultaneously and independently, so any continuously differentiable paths are valid options. However, the nature of the problem sets several constraints. The paths are intended to be closed loops, so the Equations were constrained to oscillate once over the intended path time for simplicity. The final gravity vectors in x and y must be near zero so the parametric Equation for angular position was constrained to be symmetric about both axes (Fig. 5.2). Finally, the Equations were limited to one parameter each. This was done to allow for easier analysis and visualization of results.

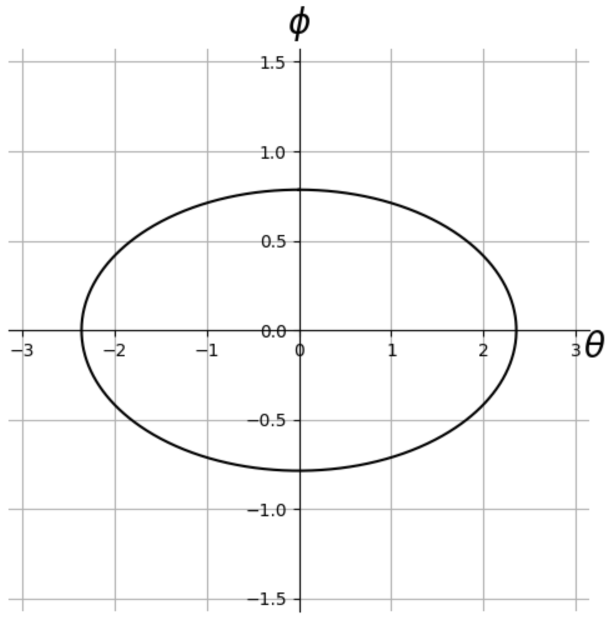

Figure 5.2: *Example of symmetric path*

The final Equations set the motor positions to follow sine and cosine waves amplified by products of π which was set to be the absolute limit on each angle. Instead of having one motor be in the 0 to 2π range and the second be in the $-\pi/2$ to $\pi/2$ range to translate motor position to spherical coordinates, both were kept in the $-\pi$ to π range for simplicity. If the flask were rotated about the Y axis by 135°, only the outer motor would need to move. In motor coordinates, the inner motor coordinate is constant and the outer motor coordinate is strictly increasing. In spherical coordinates, once the outer motor hits 90° the outer motor coordinate starts decreasing and the inner motor coordinate changes by 180°. This leads to discontinuity in the inner motor positions and a mismatch between the speed of the outer motor and the outer motor coordinate change (Fig. 5.3). The coordinate system has no impact on the results of the simulation, so spherical coordinates were avoided for simplicity.

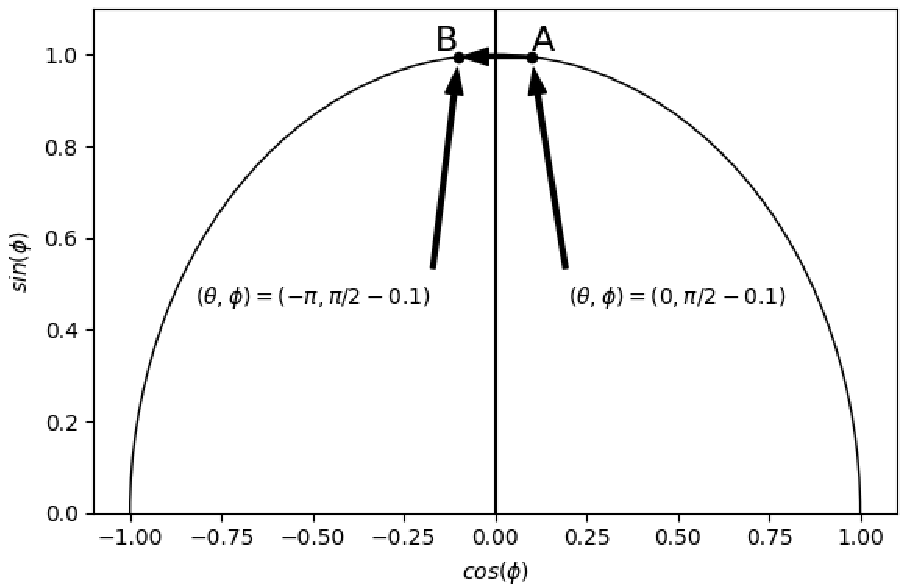

Figure 5.3: *Example of outer motor causing inner motor position to change*

The outer motor position (θ) was a function of the current time (t), the path time (T), and a parameter (a), and similarly the inner motor position was a function of the times and a parameter (b) (Eq. 5.1-5.2).

$$\theta(t, T, a) = a\pi \times sin\left(2\pi \times \frac{t}{T}\right) \tag{5.1}$$

$$\phi(t, T, b) = b\pi \times cos\left(2\pi \times \frac{t}{T}\right) \tag{5.2}$$

To determine the expected gravity for any combination, the Equations were used to generate a list of positions over a fine time grid. Multiple decreasing grid spacing were tested to verify that the gravity vector converged. For each position, the acceleration due to gravity was calculated relative to the flask (Eq. 5.3).

$$\vec{g}(\theta, \phi) = \left\langle -9.81 \times sin(\phi) \quad 9.81 \times sin(\theta)cos(\phi) \quad -9.81 \times cos(\theta)cos(\phi) \right\rangle \tag{5.3}$$

The gravity vectors were then summed and normalized by total time to calculate the approximated final gravity vector (Eq. 5.4). This assumes that the machine is moving slowly enough that the contribution of angular acceleration is negligible.

$$\vec{g} = \frac{1}{\Sigma_{i=0} t_i} \Sigma_{i=0} \left\langle g_{x,i} \quad g_{y,i} \quad g_{z,i} \right\rangle \times t_i \tag{5.4}$$

Plots of the approximated gravity vector by path parameters demonstrated that the gravity in the X and Y-directions were close to zero. Another more significant observation was that the gravity in the Z-direction changed radially with the path parameters (Fig. 5.4).

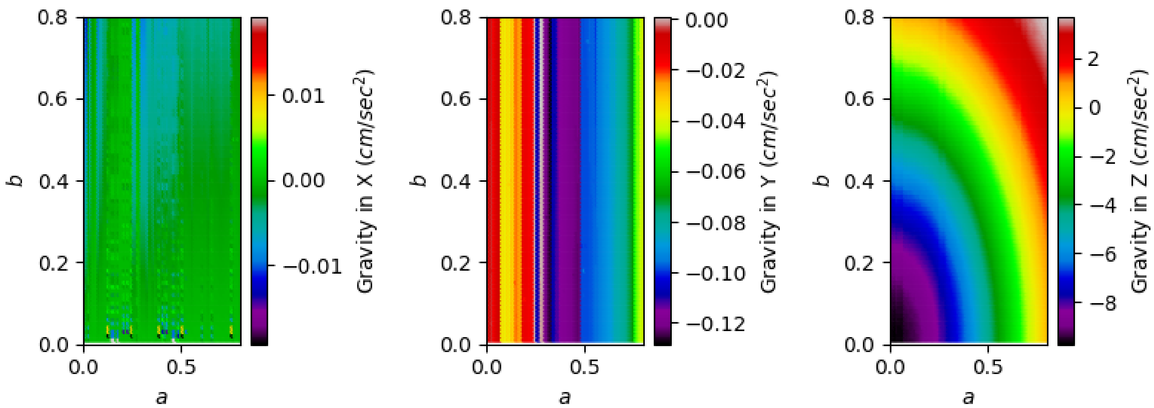

Figure 5.4: *Variation in predicted gravity magnitude with path parameters*

The gravity can be expressed as a function of the radial value $a^2 + b^2$ for any combination of a and b. The data can be fit with a cosine function, which means that an inverse function exists (Fig. 5.5). This means that for a desired gravity, there exists a unique radial value that can be solved. That radial value is the magnitude of the vector $< a, b >$. An angle can be randomly sampled to find random combinations of a and b that are expected to give the desired gravity.

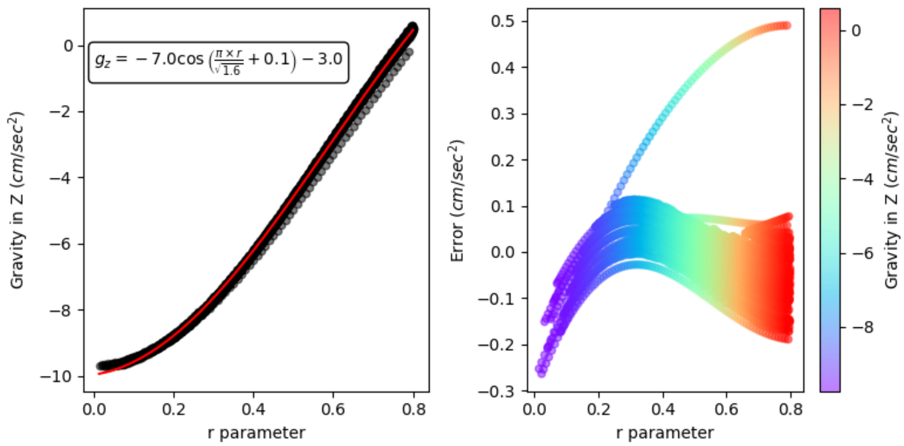

Figure 5.5: *Functional fit of gravity in z against the radial magnitude, and error between data and fit*

This functional fit could be improved even further in two ways. The first way would be to add a second cosine term to the regression. For a majority of the data, the error is following what appears to be a cosine function. The error could be fit to another function, and the combination of the functions would give a more accurate functional fit. The second way to improve the regression would be to modify the expression for r. There is a combination of a and b that give a clear deviation from the rest of the data. This suggests that the equation for r is missing a trend in the data. However, the goal of the functional fit is for the average gravity predicted to be close to zero and for the paths to be random. Each individual path may not be exactly correct, but on average the gravity converges to the correct value.

5.1.2 Machine Simulation

The gravity vectors found in the previous section assumed that the path could be perfectly followed. In reality, the input to the Arduino has memory limits and the motors have limited positional accuracy. The Arduino will follow a linear approximation to the ideal path (Fig. 5.6). The motors used in this work had a conversion of 1 step per 1.8 degrees angular movement[43]. This new path was created by following the ideal path and recording points that are within some ϵ of the rounded value in half-steps and are at least some δ in time since the last point recorded. By setting a time restriction, the size of the final path file is limited.

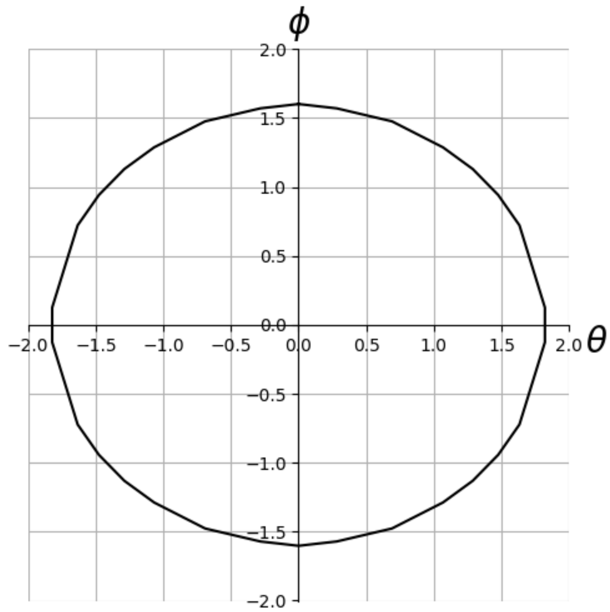

Figure 5.6: *Example of linear approximation of path*

Instead of numerical integration over time, it is more efficient to directly calculate the integral. The path could be expressed as a series of linear equations and substituted into the gravity equation to find the integral over individual sections. Then by using trigonometric identities, the average gravity over any time segment can be expressed in a closed form equation knowing only the start and end points of each motor.

The first step was to linearize the Equations for θ and ϕ for each linear segment (Eq. 5.5-5.6). The intercepts (b and d) and the slopes (a and c) were found by setting the Equations to be at the starting point at t_0 and be at the end point at time t_1 with some known difference in time.

$$\theta(t) = a \times t + b \tag{5.5}$$

$$\phi(t) = c \times t + d \tag{5.6}$$

Next the gravity in each direction was obtained via substitution. The gravity in x only required the substitution for ϕ (Eq. 5.7-5.8).

$$\bar{g_x}(t_0, t_1) = \int_{t_0}^{t_1} -9.81 \times sin(\phi(t))dt \tag{5.7}$$

$$\bar{g_x}(t_0, t_1) = -9.81 \times \int_{t_0}^{t_1} sin(c \times t + d)dt \tag{5.8}$$

Applying the chain rule to the integral in Equation 5.8 leads to the expression in Equation 5.9. The terms in the *cosine* are just the expression for ϕ at the starting and ending point, so the gravity in x integrated over a linear segment can be expressed as a function of the cosine of the starting and ending ϕ angles (Eq. 5.10).

$$\bar{g_x}(t_0, t_1) = -9.81 \times \left(\frac{cos(c \times t + d)}{c}\right)\Bigg|_{t_0}^{t_1} \tag{5.9}$$

$$\bar{g_x}(t_0, t_1) = -9.81 \times \left(\frac{cos(\phi_1) - cos(\phi_0)}{c}\right) \tag{5.10}$$

A similar approach was repeated for the gravity in y with the exception that both ϕ and θ are substituted (Eq. 5.11). After substituting for the angles, the product of the *sine* and *cosine* terms was converted to a sum of *sine* terms (Eq. 5.12).

$$\bar{g_y}(t_0, t_1) = \int_{t_0}^{t_1} 9.81 \times sin(\theta(t))cos(\phi(t))dt \tag{5.11}$$

$$\bar{g_y}(t_0, t_1) = 9.81 \times \int_{t_0}^{t_1} \frac{1}{2}(sin(at + b + ct + d) + sin(at + b - ct - d))dt \tag{5.12}$$

Applying the chain rule to Equation 5.12 leads to Equation 5.13. The terms in the *cosine* terms are just the expression for ϕ and θ at the starting and ending points, so the gravity in y integrated over a linear segment can be expressed as a function of the starting and ending ϕ angles, the starting and ending θ angles, and the slope in angles over the linear segment (Eq. 5.14).

$$\bar{g_y}(t_0,t_1) = \frac{-9.81}{2}\left(\frac{cos(at+b+ct+d)}{a+c} + \frac{cos(at+b-ct-d)}{a-c}\right)\Bigg|_{t_0}^{t_1} \tag{5.13}$$

$$\bar{g_y}(t_0,t_1) = \frac{-9.81}{2}\left(\frac{cos(\theta_1+\phi_1)-cos(\theta_0+\phi_0)}{a+c} + \frac{cos(\theta_1-\phi_1)-cos(\theta_0-\phi_0)}{a-c}\right) \tag{5.14}$$

The Equation for gravity in z was derived similarly to the gravity in y, first both ϕ and θ are substituted (Eq. 5.15). After substituting for the angles, the product of the *cosine* terms was converted to a sum of *cosine* terms (Eq. 5.16).

$$\bar{g_z}(t_0,t_1) = \int_{t_0}^{t_1} -9.81 \times cos(\theta(t))cos(\phi(t))dt \tag{5.15}$$

$$\bar{g_z}(t_0,t_1) = -9.81 \times \int_{t_0}^{t_1} \frac{1}{2}(cos(at+b+ct+d)+cos(at+b-ct-d))dt \tag{5.16}$$

Applying the chain rule to Equation 5.16 leads to Equation 5.17. The terms in the *sine* terms are just the expression for ϕ and θ at the starting and ending points, so the gravity in z integrated over a linear segment can be expressed as a function of the starting and ending ϕ angles, the starting and ending θ angles, and the slope in angles over the linear segment (Eq. 5.18).

$$\bar{g_z}(t_0,t_1) = \frac{-9.81}{2}\left(\frac{sin(at+b+ct+d)}{a+c} + \frac{sin(at+b-ct-d)}{a-c}\right)\Bigg|_{t_0}^{t_1} \tag{5.17}$$

$$\bar{g_z}(t_0,t_1) = \frac{-9.81}{2}\left(\frac{sin(\theta_1+\phi_1)-sin(\theta_0+\phi_0)}{a+c} + \frac{sin(\theta_1-\phi_1)-sin(\theta_0-\phi_0)}{a-c}\right) \tag{5.18}$$

After modifying the code to account for the Arduino limitations and incorporating the updated gravity Equations, the functional fit for gravity to parameter magnitude was repeated. It was found that the regression still fit the updated data with slightly more variation (Fig. 5.7-5.8).

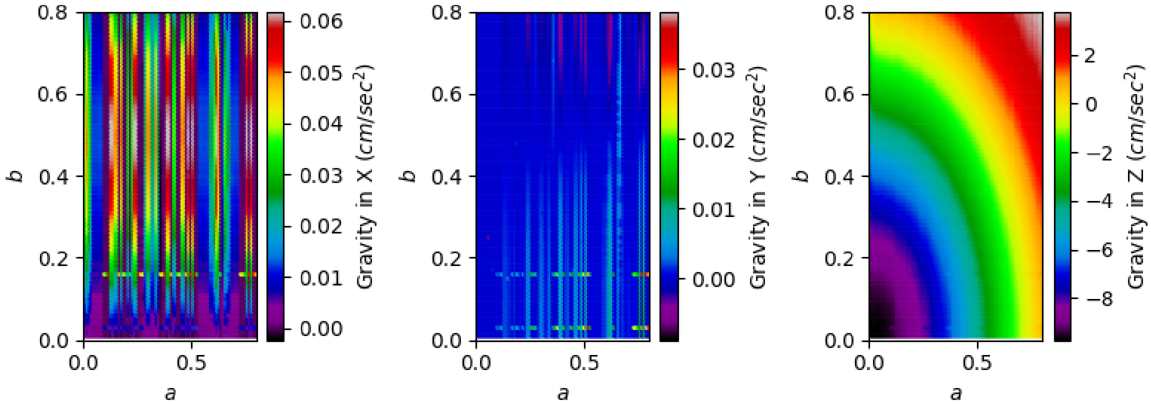

Figure 5.7: *Example of variation in gravity with motor path*

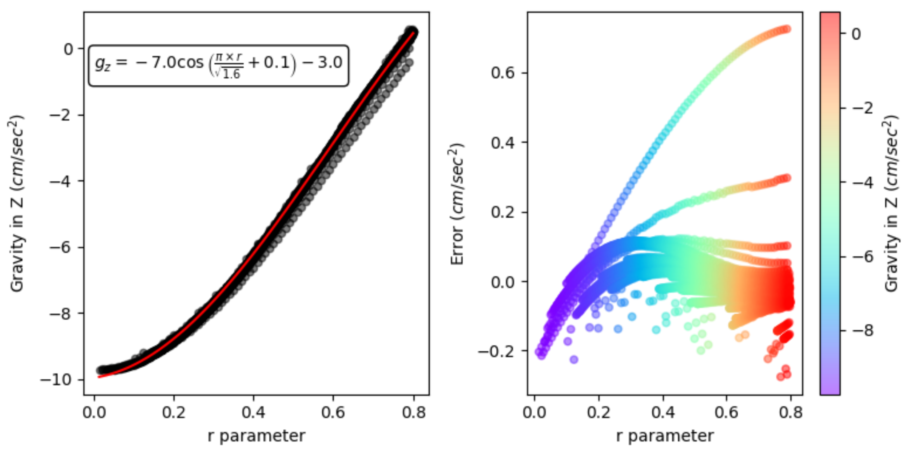

Figure 5.8: *Previous model fit over motor path data, and updated error*

5.1.3 Sample Irradiation Constraints

The first step for testing the feasibility of repeating the dose optimization is building the radiation transport geometry. Using a CAD model of the rotation machine, an aluminum frame was replicated in MCNP6[8] as if the same flask used in the previous chapter were positioned in the center of the inner ring of the rotation machine (Fig. 5.9). By embedding the rotating regions within nested universes, the rotations of both motors could be applied to the sample. Similar to the linear path, the flask was simulated in a series of positions that served as a database for interpolation. For this work, the flask was simulated over 20 grid points per motor, for a total of 400 points (Fig. 5.10). This produced a set of graphs for each position that, similar to the linear path case, could be used to approximate the dose profile for the paths that are solved for gravity.

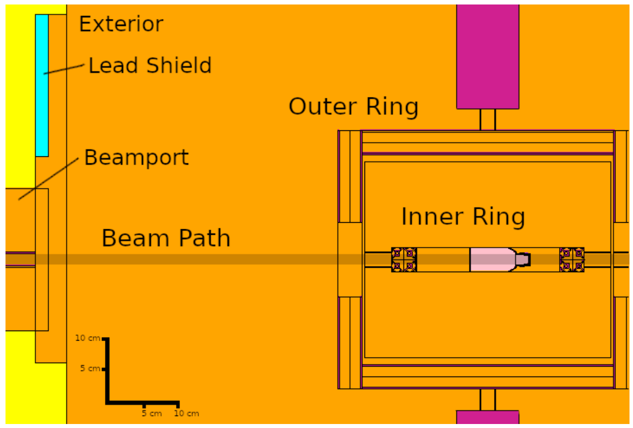

Figure 5.9: *Flask positioned in the inner ring with the beam path shaded.*

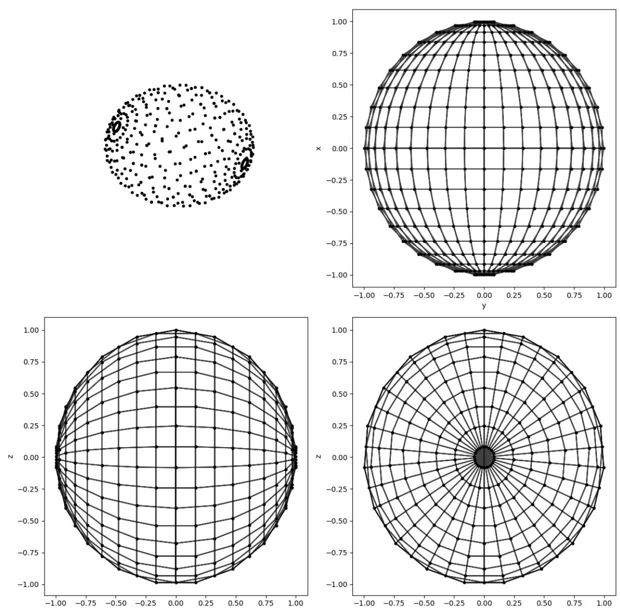

Figure 5.10: *Direction of gravity vector over the simulated positions, with unit radius.*

5.2 Rotating Path Simulation Results

There are two main issues posed by trying to solve for both gravity and dose in this setup. The most obvious is that the beam is always passing through the center of the flask.

This creates a situation where the dose in the center of the flask will always increase faster than dose at the edges. This is one reason why the linear paths were initially chosen instead of rotating paths. Rotating the flask defines a series of positions that enter the flask in different places, but all pass through the center of rotation. What this means is that linear movements could be included to shift the center of rotation. In the linear path the movements following the edges were added to define a beam position that passed through the low dose voxels without increasing the dose in the high dose voxels. Using both rotational and linear movements allows for the same process to be done with voxels in the interior. The logic is similar to trying to irradiate a tumor without damaging nearby organs[44]. For any single beam position the voxels between the beam entry and voxel of interest will experience an increase in dose. By using multiple orientations, the additional dose is distributed over the rest of the flask while still irradiating a voxel of interest over the entire path.

The second issue comes from the fact that gravity is imposing a constraint on the angle between the beam and the faces of the flask. Suppose that the default orientation of the flask was treated as being horizontal. For gravity near full magnitude, any path that gives the correct gravity will also spend a significant proportion of time with the flask near horizontal. The gravity constraint places a constraint on the proportion of time with the flask at any given angle. If the beam is always intersecting the flask, then the time spent irradiating at an angle is equal to the time spent of the flask at that angle. In the horizontal position, the beam is limited to entering the flask on the front, back, or sides. In order for the beam to enter on the top or bottom, the flask must be near vertical. For any time spent irradiating the top or bottom, a significantly longer amount of time is spent irradiating the other faces. The benefit of using rotations is the ability to control where the beam enters the flask, but that control is restricted if the proportional time spent at each angle is limited. A similar issue arises with near zero gravity. The time spent irradiating the front and back is much lower than time spent irradiating the top and bottom. Once again, that issue becomes less significant if linear motion is added. Incorporating linear motion allows the researcher to divide the time with the flask at any angle into time spent irradiating and time spent not irradiating. By moving the target in and out of the beam, the total time spent still follows the gravity constraint, but the proportion of time irradiating can be changed. Doing so does have the caveat that a large proportion of the path time may be spent not irradiating, and the linear movement may exert a force on the flask..

Both of these issues can be mitigated by including linear motion as well as rotational motion. Using both linear and rotational positioning doesn't take into account how many unique positions need to be simulated. The setup for the path assumes that the linear position is not previously known for any given rotational position. At a maximum that would mean that for every rotation orientation, the flask would be simulated at every position where the center of the beam hits the flask. Let us assume that the problem is simplified to the center of the beam being either horizontal or vertical to the center of rotation up to the maximum distance that the beam and flask could intersect. For the flask used in this work, if the beam were simulated up to 6.9 cm away, then for any orientation the beam could be approximated as approaching and moving away from the center of rotation out to the maximum distance (Fig. 5.11). For the linear path simulations, it was found that a step size of 0.3 cm was small enough to capture changes in the voxel doses. Suppose as well that the angular grid was selected to follow 2.5 degree steps. At 2.5 degrees, a point 6.9 cm away moves approximately 0.3 cm. For each rotation one would need simulations at 89 unique positions. To cover the full angular grid one would need 144 angles for each motor and 20,736 unique angular positions in total. Thus, one would need a total of 1,845,504 unique positions simulated. The simulation outputs required to calculate the dose distribution for a single position occupy approximately 15 MB, so in total the MCNP6 simulations would produce over 27 TB of text files. The numpy[45] representation of a single position requires on average 560 KB for both neutrons and photons which means that the dataset requires slightly over 1 TB in numpy files. There are libraries and techniques designed to handle data on that scale, but those libraries and techniques were not necessary for this study. As the paths grow more complex the matrices needed to express them increase in size quickly.

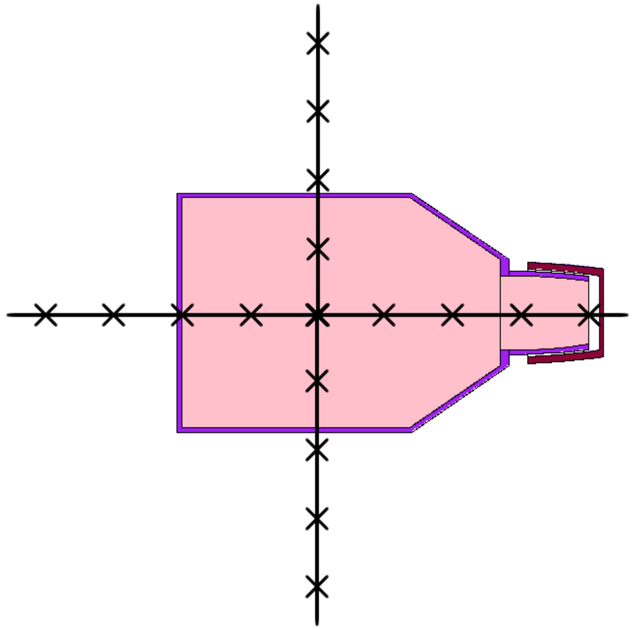

Figure 5.11: *Example of points simulating passing through the center up to maximum distance.*

If the target or radiation source could be changed, then there is a second simpler solution. The need for linear movement is largely due to the relative sizes of the beam and target. If the beam does not naturally intersect a section of the target, then the beam must be moved so that it does. The flask supplier lists a maximum volume of $60\,\text{cm}^3$. The same volume would then be contained in a cube with 3.91 cm side lengths. For the same linear and rotational step sizes, the cube would require 49 linear positions per angle and 6,694 angular positions. This would require approximately 184 GB in numpy files, about 80% less memory for simulating the same volume of target. Additionally, the source used in this work was collimated. If a collimator was not used, then the beam size would increase. A larger beam would pose a higher risk of irradiating equipment, but there is certainly an option to modify the collimator.

Chapter 6

Conclusions

6.1 Discussion of Results

The goal of this work was to outline the steps necessary to design a biological irradiation chamber, predict the dose profile for a planned irradiation, and calculate a path that minimizes non-uniformity in dose in the target. This work builds upon previous research into simulating the KSU reactor, and provides a summary of how a database of dose profiles can be generated and applied more generally to approximating a moving target.

The most significant contribution this work makes to the study of biological irradiation is the modeling of moving a target to correct dose non-uniformity. Previous work into simulating dose due to a moving beam relative to the target was primarily for visualizing the effects of proton therapy, and did not establish any algorithm to plan a beam path or optimize the path the beam takes. Previous work into beam irradiation optimization was focused on widening the beam to cover a stationary target and place foils to control the radiation field. The dose in the target could be controlled by relative distance between the beam and target and the thickness of foils placed to scatter radiation. An optimal combination was found by simulating different combinations of distance and foils, but the process was not automated. This work converts the database approach for irradiation approximation to a series of equations that can be integrated and differentiated. This establishes a process to optimize the path the beam takes for any feasible combination of radiation source, target, and sample chamber.

6.2 Future Work

There are two areas of improvement on this work that future work could build on. The first would be building the linear movement setup and measuring the dose in a physical experiment. Firstly, the controllers and platforms would have to be built and the controller simulation done in this study would have to be validated. Secondly, some way of performing a voxel dose measurement would be necessary. Finally, multiple paths would have to be simulated and performed physically to compare the simulated dose against the true dose. For a known path, a response function for the reactor flux spectrum and each voxel dose could be constructed. The reactor flux is approximated by minimizing error between simulated and true count rates from several previous experiments. These response functions would be additional equations that the reactor flux could be approximated with.

The second area of improvement is application to a rotating flask. Firstly, the code and steps taken in this work would need to be modified to handle larger datasets. The number of simulations is within the scale of models that can be optimized with parallel algorithms[46]. Doing so may require a more efficient means of running and storing the simulations and their results, or in the very least a series of job submission scripts that automate the simulation running and conversion to numpy file. Future work may even find that this method of building a dose database is not necessary. In theory there is a way to define a default dose profile and then modify it based on the flask orientation and beam position. This was investigated briefly during this work while testing interpolation schemes. It was found that the average beam shape was constant in the flask interior, but quickly changed shape near the edges. The second expected issue is that the platform setup designed in this work would not be ideal for moving the entire random position machine. There would be additional steps to find a heavier duty platform and motor, and then write code to simulate the platform movement.

Bibliography

[1] J. Boyington. Neutron flux characterization of the kansas state university triga mark ii's northeast beam port. Master's thesis, Kansas State University, 2019.

[2] Iwamoto, Osamu, Iwamoto, Nobuyuki, Shibata, Keiichi, Ichihara, Akira, Kunieda, Satoshi, Minato, Futoshi, and Nakayama, Shinsuke. Status of jendl. *EPJ Web Conf.*, 239:09002, 2020. doi: 10.1051/epjconf/202023909002. URL https://doi.org/10.1051/epjconf/202023909002.

[3] D.J Thomas and A.V Alevra. Bonner sphere spectrometers—a critical review. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 476(1):12–20, 2002. ISSN 0168-9002. doi: https://doi.org/10.1016/S0168-9002(01)01379-1. URL https://www.sciencedirect.com/science/article/pii/S0168900201013791. Int. Workshop on Neutron Field Spectrometry in Science, Technolog y and Radiation Protection.

[4] Kaifeng Gao, Gang Mei, Francesco Piccialli, Salvatore Cuomo, Jingzhi Tu, and Zenan Huo. Julia language in machine learning: Algorithms, applications, and open issues. *Computer Science Review*, 37:100254, 2020. ISSN 1574-0137. doi: https://doi.org/10.1016/j.cosrev.2020.100254. URL https://www.sciencedirect.com/science/article/pii/S157401372030071X.

[5] Velmex. Appnote 106: Acceleration units and start/stop speeds, 2020.

[6] S. Alshogeathri. Application of the reactivity method on ksu triga fuel. Master's thesis, Kansas State University, 2013.

[7] T. Ochs. *Advanced dual-sided microstructured semiconductor neutron detectors and instrumentation.* PhD thesis, Kansas State University, 2015.

[8] Christopher John Werner, Jeffrey S. Bull, et al. *MCNP6.2 Release Notes*, la-ur-18-20808 edition, 2018.

[9] Călin Mircea Rusu, Mihai Straticiuc, Mihaela Bacalum, and Mihai Radu. Monte carlo simulations of biological samples irradiation using a new setup at the 3 mv tandetrontm of ifin-hh. *Šaćira Mandal, Adlija Čaušević, Sabina Semiz, Plasma leptin concentrations and lipid*, page 278, 2021.

[10] Sonja Gerlach, Marco Pinto, et al. Beam characterization and feasibility study for a small animal irradiation platform at clinical proton therapy facilities. *Physics in Medicine & Biology*, 65(24):245045, 2020.

[11] Anna Baratto-Roldán, María del Carmen Jiménez-Ramos, et al. Feasibility study of a proton irradiation facility for radiobiological measurements at an 18 mev cyclotron. *Instruments*, 2(4), 2018. ISSN 2410-390X. doi: 10.3390/instruments2040026. URL https://www.mdpi.com/2410-390X/2/4/26.

[12] JM Gómez-Ros, R Bedogni, C Domingo, JS Eakins, N Roberts, and RJ Tanner. International comparison exercise on neutron spectra unfolding in bonner spheres spectrometry: problem description and preliminary analysis. *Radiation Protection Dosimetry*, 180(1-4):70–74, 2018.

[13] Jean D Sibonga. Spaceflight-induced bone loss: is there an osteoporosis risk? *Current osteoporosis reports*, 11(2):92–98, 2013.

[14] Ye Zhang, Srinivasan Shanmugarajan, Maria Moreno-Villanueva, Ryan Clanton, Larry H Rohde, Govindarajan T Ramesh, Jean D Sibonga, and Honglu Wu. Combined effects of simulated microgravity and radiation exposure on osteoclast cell fusion. In *COSPAR Scientific Assembly*, 2018.

[15] Eliah G Overbey, Amber M Paul, Willian A da Silveira, Candice GT Tahimic, Sigrid S Reinsch, Nathaniel Szewczyk, Seta Stanbouly, Charles Wang, Jonathan M Galazka, and Xiao Wen Mao. Mice exposed to combined chronic low-dose irradiation and modeled

microgravity develop long-term neurological sequelae. *International journal of molecular sciences*, 20(17):4094, 2019.

[16] Shaobo Tan, Weiwei Pei, Hao Huang, Guangming Zhou, and Wentao Hu. Additive effects of simulated microgravity and ionizing radiation in cell death, induction of ros and expression of rac2 in human bronchial epithelial cells. *npj Microgravity*, 6(1):1–6, 2020.

[17] Scott W. Mosher, Seth R. Johnson, et al. *ADVANTG – An Automated Variance Reduction Parameter Generator, Rev. 1*, ornl/tm-2013/416 edition, 2015.

[18] M Reginatto and P Goldhagen. Maxed, a computer code for maximum entropy deconvolution of multisphere neutron spectrometer data. *Health Physics*, 77(5):579–583, 1999.

[19] Marcel Reginatto, Burkhard Wiegel, Andreas Zimbal, and Frank Langner. *UMG 3.3, Analysis of data measured with spectrometers using unfolding techniques*, package nea-1665/03 edition, 2004.

[20] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017. URL https://doi.org/10.1137/141000671.

[21] Velmex. Vmx users manual extended, 2020.

[22] B Forrest et al. *MCNP5 development, verification, and performance*, 2003.

[23] S Berg and W N McElroy. *A computer-automated iterative method for neutron flux spectra determination by foil activation. volume ii. Sand II (spectrum analysis by neutron detectors ii) and associated codes. Technical Report, April 1966–July 1967.*, 1 1967. URL https://www.osti.gov/biblio/4171125.

[24] Jeffrey Perkel. Julia: come for the syntax, stay for the speed. *Nature*, 572:141–142, 08 2019. doi: 10.1038/d41586-019-02310-3.

[25] M.B. Chadwick, M. Herman, et al. Endf/b-vii.1 nuclear data for science and technology: Cross sections, covariances, fission product yields and decay data. *Nuclear Data Sheets*, 112(12):2887–2996, 2011.

[26] C. Wendorff. *Validation and benchmarking of the thermal neutron scattering law data files for neutron moderators using experimental data.* PhD thesis, Rensselaer Polytechnic Institute, 2018.

[27] D.A. Brown, M.B. Chadwick, et al. Endf/b-viii.0: The 8th major release of the nuclear reaction data library with cielo-project cross sections, new standards and thermal scattering data. *Nuclear Data Sheets*, pages 1–142, 2018.

[28] Corning 25 cm^2 cell culture canted neck flask, anti-tip skirt, plug seal cap. Product Description, 2019.

[29] Dassault Systèmes. Solidworks, 2020.

[30] THORLABS. Nrt100 manual, 2022.

[31] PI. Datasheet l-836, 2022.

[32] Optics-Focus. Moxyz-06-150-150-150 page, 2022.

[33] Velmex. Xn10 series manual, 2022.

[34] THORLABS. Hdr50 manual, 2019.

[35] PI. Datasheet l-611, 2021.

[36] Optics-Focus. Mor-60 page, 2022.

[37] Velmex. B5990ts specification page, 2022.

[38] Velmex. Nema type 17 single shaft stepper motor manual, 2015.

[39] THORLABS. Bsc203 manual, 2016.

[40] PI. Datasheet c-663.12, 2021.

[41] Optics-Focus. Moc-01-3-110, moc-0143-110 page, 2022.

[42] Velmex. Vxm manual, 2004.

[43] AutomationDirect. Stp-mtrh-34127d specifications, 2020.

[44] Koka K., Verma A., Dwarakanath BS., and Papineni RVL. Technological advancements in external beam radiation therapy (ebrt): An indispensable tool for cancer treatment. *Cancer Managment and Research*, 14:1421–1429, April 2022. doi: 10.2147/2FCMAR.S351744.

[45] Charles R. Harris, K. Jarrod Millman, et al. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

[46] Jens Dittrich and Jorge-Arnulfo Quiané-Ruiz. Efficient big data processing in hadoop mapreduce. *Proc. VLDB Endow.*, 5(12):2014–2015, aug 2012. ISSN 2150-8097. doi: 10.14778/2367502.2367562. URL https://doi.org/10.14778/2367502.2367562.