COMPUTATIONAL REPRESENTATIONS FOR ELECTRON MICROBEAM MICRODOSIMETRIC QUANTITIES IN ONE MICRON DIAMETER SPHERICAL TARGETS

By

MICHAEL TODD BATDORF

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

MASTER OF SCIENCE IN COMPUTER SCIENCE

WASHINGTON STATE UNIVERSITY
School of Electrical Engineering and Computer Science

MAY 2005

To the Faculty of Washington State University:

The members of the Committee appointed to examine the thesis of MICHAEL TODD BATDORF find it satisfactory and recommend that it be accepted.

Chair

ACKNOWLEDGEMENTS

I would like to thank the faculty here at WSU Tri-Cities for making my dream possible. They took me under their wings and financially supported my work on PITS over the past two years. I hope this information contained in this thesis at least provides some return on their investment. I also would like to thank my friends and family for putting up with my frustrations over such a long stretch of time. It is the journey that is important, not the destination.

COMPUTATIONAL REPRESENTATIONS FOR ELECTRON MICROBEAM MICRODOSIMETRIC QUANTITIES IN ONE MICRON DIAMETER SPHERICAL TARGETS

Abstract

By Michael Todd Batdorf, M.S.
Washington State University
May 2005

Chair: John H. Miller

In this study, computational tools were developed to gather, process, and fit simulated data resulting from the Monte-Carlo simulation of low energy electron particle tracks in liquid water using the Positive Ion Track Simulation (PITS) code set. To gather the data, the simulation space was packed with one micron diameter spheres in concentric rings to exploit angular symmetry, three million tracks were generated for initial beam energies ranging from 20 keV to 80 keV in 5 keV increments on a 16 node PC cluster, and histograms for stochastic quantities were tabulated for each sphere and placed in an XML file for archiving. Computational routines wrapping a preexisting B-spline library were then created to approximate the probability distributions and summary statistics for any stochastic quantity over a domain of target location and beam energy. Because of their biological significance, three particular quantities were chosen to illustrate the methods under varying conditions: the *conditional mean of energy imparted* (μ_E), which is a summary statistic with a relatively simple nature; the *probability of a track reaching a target* (Φ), which is a summary statistic with large peaks in its data set; and the *energy* *deposited per unit tracklength of the primary particle* (Λ), which is a stochastic whose distributions are approximately lognormal. For each of these quantities, approximations were created at various levels of refinement, with each approximation being tested for its storage requirements and its accuracy against a test data set. A second method of approximation using lognormal probability distribution curves and the method of least squares was pursued on the distributions of Λ for a set of locations and beam energies. Goodness-of-fit testing showed that these distributions may appear to be lognormal but are not *statistically* lognormal. Finally, an investigation was undertaken to determine whether the representations developed for the one micron diameter spheres could be applied to larger targets by packing the targets with the spheres and somehow aggregating the results. No methods were found for the stochastic representations, and methods that were assumed to work for summary statistics were shown to be inefficient.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Introduction

1.1 Radiation and Charged Particle Tracks

Radiation is all around us, in the air we breathe, the water we drink, and the land we stand on. Technically, "radiation" in a given volume is comprised of particles that move through that volume. Some of these particles have mass, such as the proton or electron, while the others are massless, such as photons from light or gamma radiation. Contrary to popular opinion, radiation is not always harmful to living organisms. Its effectiveness at biological damage depends in large part on its type, strength, and how it interacts with cellular matter. For example, the photons comprising white light in sunlight are not what burn your skin when you stay out in the sun too long. Rather, it is the photons associated with the ultraviolet frequencies. Radiation that has the ability to *ionize* atoms can be particularly damaging to cellular matter. When an atom is ionized it is stripped of one or more of its electrons, thus creating a positive charge, and possibly giving the ejected electron(s) enough energy to themselves ionize other atoms. Any incident particle has the ability to ionize target atoms depending on the particle type, its current state, and the target in question.

Photons are the massless particles that transport electromagnetic energy. They can range from the very low energy photons associated with low frequency radio waves to the very energetic photons associated with high frequency gamma waves. Up to an energy of about 10 MeV, photons interacting with matter lose energy by either ejecting an orbital electron via the photoelectric effect, transferring energy to an orbital electron via Compton scattering, or by producing an electron-positron pair in a process known as pair production, which is rare under most biological irradiation circumstances. In the photoelectric effect, the entire target atom absorbs the photon and then ejects an electron with an energy of the photon minus the electron's binding energy. In light elements this process is dominant for photons up to 40 keV. In Compton scattering, the photon's energy is transferred directly to an orbital electron in the target atom rather than to the entire atom. Upon photon absorption, the electron is ejected and a scattered photon is emitted, with only a fraction of the incident photon energy being transferred to the ejected electron. Compton scattering is dominant for photons from 40 keV to 10 MeV [ICRU1983, pg. 7].

Neutrons are chargeless subatomic particles that ionize matter by transferring energy to the protons in hydrogen-bearing molecules and causing them to eject, or recoil. These recoiled protons then ionize atoms within the material because of their positive charge. Neutrons, therefore, can play a significant role in ionizing biological matter because of its high hydrogen content, and in particular high water content.

Unlike neutrons and photons, electrons and ions have a charge, and they interact with target atoms via the Coulomb force. Electrons are subatomic particles each having a charge of -1, while ions can range from +1 charged subatomic protons to large complex positively charged molecules. In the case of these "charged particles", their ability to ionize depends largely on their velocity, charge, and proximity to the target. Slower higher charged particles moving closer to the target have a greater chance of causing an energy transference to the target.

In the act of ionizing matter, particles in radiation eject electrons which can themselves be ionizing charged particles if a sufficient level of energy is transferred in the collision, typically on the order of 100 eV [ICRU1983, pg.11]. Otherwise, the ionized electron does not have enough energy to break the ionization threshold of further atoms and ends its life in a thermal death. Many generations of ionizing electrons can be produced from the same parent electron. In fact, a large fraction of the ionizations that take place from the track of a charged particle are made up of these children electrons. The first generation electrons then themselves act as irradiating charged particles, producing further generations. All electrons produced by ionization are known as *delta rays*.

A primary charged particle's entire history of collisions, along with its ionized offspring and their collision histories, is known as a "track". Collisions can be either elastic, where the total kinetic energy of colliding particles is preserved, or inelastic, where energy is deposited to the target atom and an ionization or an atomic excitation is produced. In the case of an atomic excitation, the energy is absorbed by electrons moving to a higher energy level. Though the ratio of elastic events to inelastic events along a charged particle track is typically on the order of 10 to 1, virtually all of a primary particle's initial kinetic energy is expended through inelastic events [ICRU1983, pg. 11]. For example, throughout the life of a proton with an initial energy of 1.5 MeV, 20% of the its initial kinetic energy is expended by overcoming the binding energy of the electrons it ionizes, 60% of the kinetic energy is transferred to the kinetic energy of the first generation of electrons released from ionization, and 20% of the energy is transferred to the target medium as atomic excitations [ICRU1983, pg. 11]. Surprisingly, the ratio of excitation to ionization energy losses in a track is independent of the nature and energy of the primary particle.

Figure 1-1 shows a sample segment from a 2 MeV helium ion particle track. Each sphere in the figure represents an energy depositing event, while the sphere's color and size both represent the magnitude of the event. The primary particle's trajectory is straight because its relatively heavy mass prevents it from being scattered by the collisions it makes with the medium. These collisions are in the form of ionizations and excitations with energies ranging from 10 eV to 35 eV in the figure. The secondary electrons produced are low enough in energy that they almost immediately relinquish their energy in the form of heat after being produced. This thin "fuzz" of delta rays is denoted by the small aqua spheres in the figure.

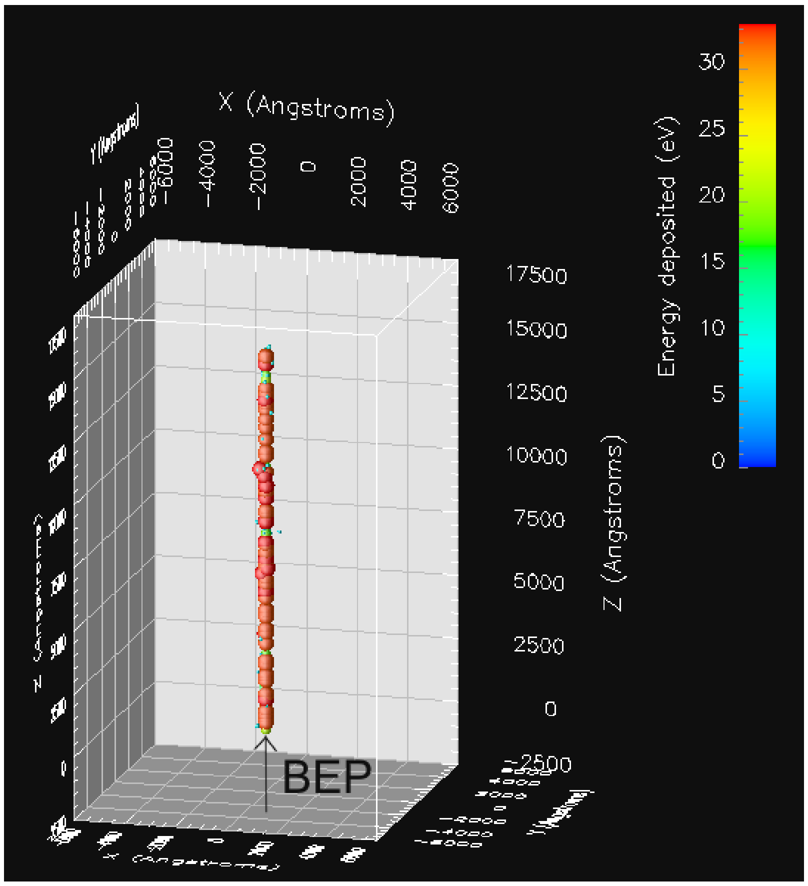

Figure 1-1: A 2 MeV helium ion particle track segment.

Notice how different the 25 keV electron track in Figure 1-2 is compared with the heavy ion track segment in Figure 1-1. The mass of an electron is incredibly small compared with an ion, so it is easily deflected in its collisions with target atoms, especially as its kinetic energy decreases. The beam entry point is on the right-hand side of the figure and the track endpoint is on the left-hand side. Notice how the magnitude of energy depositions increases as the primary electron nears the track endpoint. As mentioned earlier, this is because a slower moving electron has more influence on its surroundings.

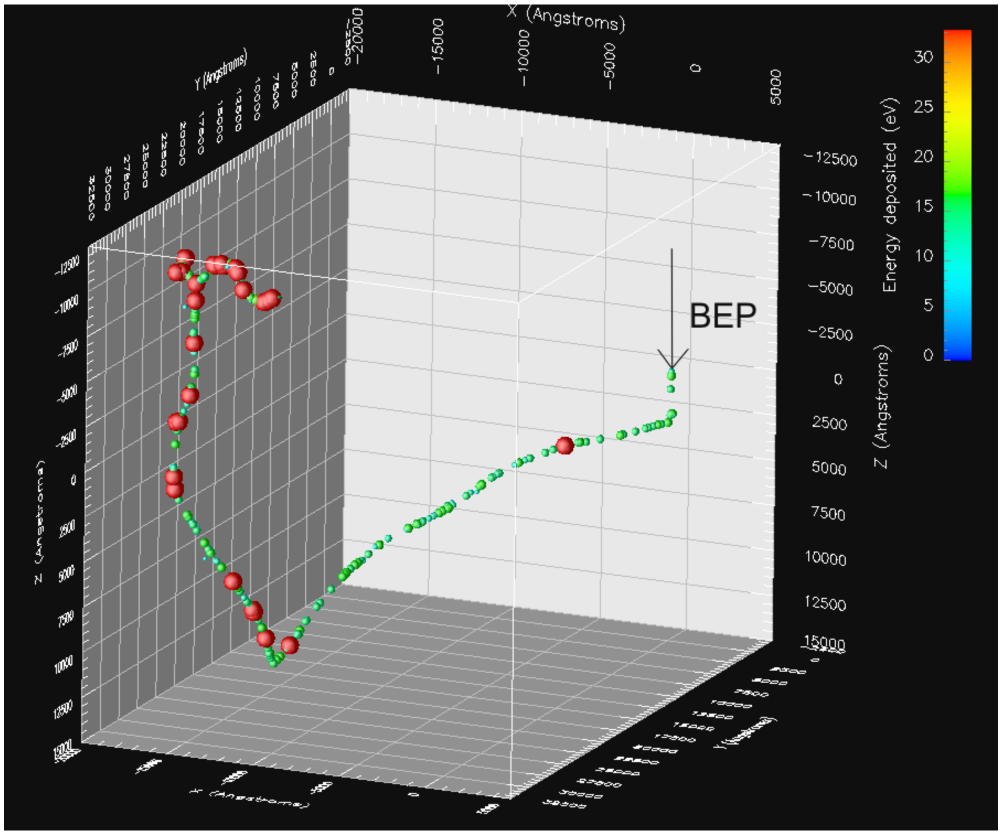

Figure 1-2: A 25 keV electron particle track segment.

1.2 Radiation Quality and Microdosimetry

A crude measure of a radiation's ability to inflict damage within a given volume of biological material is to calculate the total "dose" of radiation delivered to the volume. Dose is the energy deposited by the radiation per unit mass of material. The dose delivered, however, only gives part of the picture. The "quality" of the radiation, or *how* the radiation's energy is distributed within the volume, has been shown to be important as well [ICRU1970, pg. 2]. This involves the number of energy depositions, along with their magnitude and spatial distribution. Two differing radiations delivering the same dose to a biological target may result in two completely different levels of damage in the target, owing to the spatial extent of their energy depositions.

Analyzing the structure of each and every radiation track within a target to assess biological damage, however, is impractical. A more manageable way of relating track structures to targets is to study the statistical nature of the tracks in various sized volumes. A picture can then be constructed of how various ionizing radiations relate with target size and location. This is the aim of *microdosimetry*. Microdosimetry seeks to systematically analyze the microscopic distribution of energy and other quantities of interest in irradiated matter.

Quantities that are generated in a fixed volume from a single track are considered *stochastic*. Their value in a volume varies from one track to the next amongst the tracks that actually reach that volume. The events along the track are governed by well-defined laws in Physics, but there is an element of randomness built into charged particle tracks that makes their event locations and magnitudes uncertain from one track to the next. A mathematical tool called a *probability density distribution* can be built for a fixed volume and stochastic by running many tracks through the volume and collecting the results. This tool provides a description of the stochastic quantity and can predict the chances of a future track yielding a certain range of values for that quantity. Assuming a fixed volume in the experiment space, examples of stochastic quantities include E, the total energy deposited by the track, L, the total path length of the track, and N, the number of ionization and excitation events within the volume.

Quantities summarizing the behavior of multiple tracks traveling through a fixed volume, on the other hand, are non-stochastic in nature, if a sufficient number of tracks travel through the volume. These quantities are known as *summary statistics*. Their value approaches a limit within a volume as the number of tracks reaching the volume approaches infinity. Examples of summary statistics include the mean, variance, mode, and median of a data set.

Figure 1-3 shows a hypothetical electron charged particle track moving through a circular target volume. The primary particle track is represented by the thick curved line, while the delta rays are represented by the thinner lines. Inelastic collisions are denoted by a large dot if an ionization has occurred and by a small dot if an excitation has occurred. Most of the energy transferred in an ionization goes into the kinetic energy of the released secondary electron, but some is deposited to the medium. The total energy deposited by the primary track and its delta rays can be found by adding up all of the individual energy deposits at the inelastic interaction points occurring in the volume. A distribution of the energy deposited eventually emerges as many more tracks move through the volume.

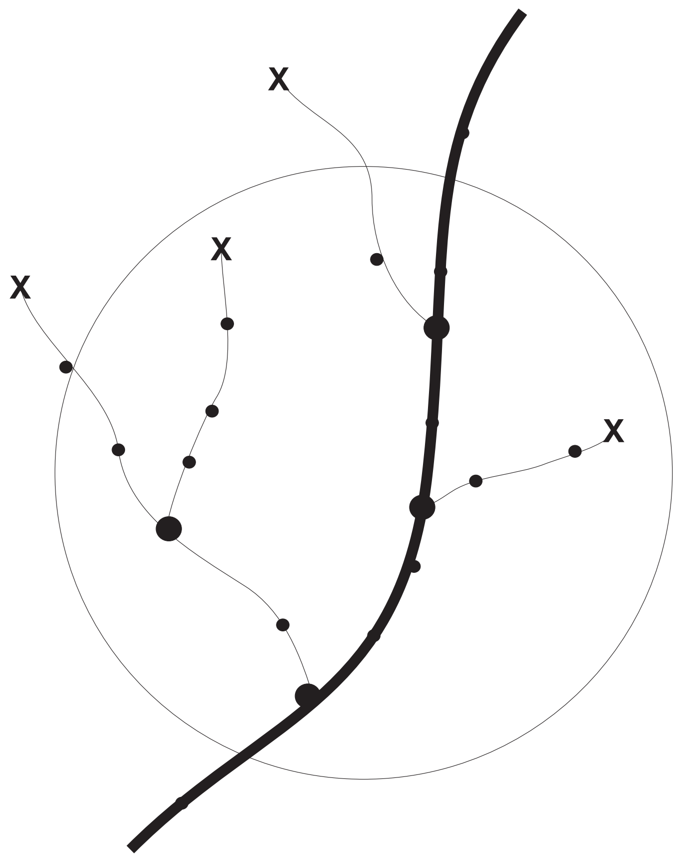

Figure 1-3: A sample track moving through a fixed volume.

1.3 Simulating Particle Tracks with PITS

A physical experiment in microdosimetry is carried out by injecting a collimated microbeam of charged particles or high energy photons into a medium and positioning detectors within the experiment space to measure pertinent quantities. Unfortunately, these experiments are often difficult to carry out and offer little hope in capturing the detailed history of the particle tracks. Sometimes a physical experiment is necessary, such as when a measurable biological response is sought, or when theoretical models must be verified. Otherwise, charged particle tracks are *simulated* through a medium of interest, such as liquid water, using Monte Carlo techniques. Simulation offers the ability to record a detailed history of interactions over all of the tracks, run as many tracks as time will allow, and change the simulation parameters and scoring geometry at will. These abilities open the door to studying statistical track characteristics and microdosimetry, where a multitude of tracks are necessary to build up the proper statistics.

In Monte Carlo charged particle track simulations, the particle in question is given an initial energy and "injected" into a medium in the simulation space. The medium is any chemical whose interaction with the charged particle needs to be studied, provided the Monte Carlo code can model the physics behind how the charged particle interacts with it. In many circumstances the medium is water, since biological tissue is composed primarily of water. Using a table of experimentally calculated interaction probabilities between the particle and medium—known as cross sections—as well as a pseudo-random number generator, the simulation code calculates the entire history of the particle's interactions until the particle's energy falls below a predefined limit. The interaction histories of all delta rays are followed as well, until their energies too fall below the same predefined limit. For each inelastic collision, a particle's interaction history typically includes the spatial coordinates of the interaction, an identifier of whether the collision was an ionization or an excitation, the distance the particle has traveled since its last interaction, and the energy deposited in the interaction.

An individual event in this interaction history is completely dependent on events that preceded it. In other words, the events within a single track are *correlated*. For multiple tracks the events from one track may be correlated with the events from another track depending upon the assumptions made by the simulation code. Most of the time the assumption is that tracks are independent of one another, since the fluence rate in microbeams is typically very low, thus limiting inter-track electromagnetic interactions. This simplifies the analysis of microdosimetric quantities by enabling a study of their *per track* behavior using a large number of tracks. It also allows the study of their multiple track behavior through a technique in the field of statistics known as *convolution*.

The Monte Carlo simulation code used in the current study is called PITS. PITS, which stands for Positive Ion Track Structure, simulates independent charged particle tracks as they travel through and interact with water. One version of the code uses the gas phase of water as the medium [Wilson, 1994], while an updated version uses the liquid phase [Wilson, 2004]. The PITS code set is comprised of a main driver module that handles the transport of positive ions, a module dedicated to the transport of electrons, and an optional "scoring" module. It is in the scoring module where the behavior of individual tracks or the microdosimetry of the tracks within fixed volumes can be analyzed. Running a PITS simulation involves writing the scoring module, compiling and linking the PITS code set, specifying the input parameters for the run, executing the run, and then analyzing the output data set. The input parameters to a simulation include (but are not limited to) the primary particle's initial kinetic energy, the particle's type, the particle's mass, and the number of particles in the simulation. The output consists of either the scoring module's results or the interaction history of the entire track.

1.4 Goal of the Study

PITS is a useful tool for analyzing the deposition of energy in biological structures, such as mammalian cells or nuclei. A cell, for example, is made up of separate parts that could possibly respond differently to ionizing radiation. Determining how the ionizing radiation is distributed amongst the parts could help to determine the cell's survivability to such radiation. PITS is a tool for modeling such problems. First a separate sub-volume is defined for each component of the cell and the cell is oriented in relation to the microbeam entry point. Then many ionizing particle tracks are generated that move through the cell's components and deposit energy. Finally, the scored simulation statistics gathered over all of the tracks are analyzed for each of the cell components. Miller, *et al*. [2000], used such an approach in their analysis of single-cell irradiation by an electron microbeam. Changing a simulation parameter, however, would amount to rewriting the scoring module to reflect the changes and then rerunning the simulation. Simply translating the target cell to a new location, for example, would require such a rerun.

Fortunately, it is possible to alter a parameter without having to rerun the simulation. The key is to *mathematically approximate* how the quantity of interest varies with a change in the parameter. Building this approximation, or *representation* as it will be called, involves running the simulation for many values of the parameter, gathering data points of how the quantity varies with the parameter, and then fitting the data with a smooth mathematical function. Afterward, the representation would give the quantity's value for *any* given value of the parameter within the parameter's interval, assuming all other simulation parameters are constant.

As an example of this process in action, consider the problem of having to rerun an simulation when a target is moved. To prevent the rerun, a mathematical representation would have to be built relating the quantity of interest with the coordinates of the target. The simulation space would be packed with identical units of the target volume and many tracks generated. The result would be a set of data points expressing the quantity of interest in terms of the target's location. A mathematically smooth representation of this data set would then be calculated using one of many possible approaches. The representation specifies how the quantity within the target varies as the target is moved *anywhere* over the simulation space. The following fundamental questions can be answered from this packed space approach.

- What is the probability of the track actually reaching a given target volume? This is denoted by the symbol Φ.
- If a particle track actually reaches a specified target volume, then what amount of energy is that track *expected* to deliver to the volume? This is known as the conditional mean of energy imparted and is denoted by the symbol μ_E.
- If a particle track actually reaches a specified target volume, then what is the probability that it will deposit energy per unit track length of the primary particle between λ and $\lambda + d\lambda$ within that volume? The symbol λ represents a particular value of the stochastic Λ, and the curve $f_\Lambda(\lambda)$ is its probability density function.

The major downside with such representations, however, is that only the parameter included in the representation can change. Other simulation parameters must remain constant. A representation can include multiple parameters by capturing data points of the quantity as a function of each parameter, but the problem still exists for all other parameters in the simulation. Also, a representation's complexity increases rapidly as the number of parameters is increased. It is therefore best to limit the number of parameters in the representation as much as possible.

Parameters such as target location and beam energy are very natural to include in the representations because their values are easily confined within a logical set of boundaries. Confining target shape and size, however, is another matter. There are an infinite number of ways to specify the geometry of a target. One idea for overcoming this limitation is to *pack* a larger target volume with representative target volumes whose microdosimetric behavior are known, and then calculate the larger volume's results from the representative target results. That way, the microdosimetry of *any* volume larger than the representative volume could be calculated.

This approach was the motivation for producing the 1 micron diameter sphere representations in this thesis. Through a set of preliminary calculations it was assumed that the method could be applied to at least the simpler case of non-stochastic quantities. From the outset, the stochastic quantities were known to be very difficult to handle. Their value in a given volume depends deeply on the track's path history, therefore introducing a dependency between volumes at different locations. Non-stochastic quantities, on the other hand, do not pertain to individual tracks, and so are not dependent on any individual path and its history. Working from this piece of knowledge as well as the preliminary calculations, the representations using 1 micron diameter spheres were developed and an example problem involving a cell nucleus was formulated to use the representations. Trying to solve the cell problem, however, led to the realization that the assumptions and preliminary calculations were flawed. A detailed discussion of these issues can be found in Section 4.

Despite the disappointment with the packing problem, the other goals of the thesis remain the same. In general, the theme of the thesis is on the mathematical approximation of discrete simulation data. A major highlight of the paper, presented in Section 3.3, is on the creation of smooth mathematical representations of Φ, μ_E, and f_Λ using B-splines. The targets are 1 micron diameter spheres, and the simulation parameters in the representation are the beam energy, ranging from 20 keV to 80 keV, and the target location. A meticulous analysis of the fits for various levels of refinement is undertaken to determine the best overall fit given the fit's storage requirements. A second emphasis of the thesis is on modeling the f_Λ distribution data using lognormal probability density distributions. This is presented in Section 3.4. Assuming a constant location and beam energy, lognormal distribution curves with two parameters are fitted to the data using a variety of methods from the field of statistics. Based upon these fits, the data's goodness of fit to the lognormal distribution is determined.

2 Methods

2.1 PITS Monte Carlo Particle Track Simulator

2.1.1 Overview and Structure

As mentioned in the introduction, PITS uses Monte Carlo simulation techniques to produce the detailed interaction histories for independent charged particle tracks and their children. Once the simulation environment in initialized, the code determines the characteristics of each interaction in the track by utilizing a steady stream of random numbers, a table of interaction probabilities known as "cross sections", and the principles of particle collision physics. The simulation progresses until the path length of the primary particle has reached a user-defined limit, or until the particle's energy falls below a threshold of 10 eV. The delta rays are followed until their energy falls below the 10 eV threshold as well, whereupon they are assumed to dissipate the remainder of their energy as thermal heat to the medium. The entire interaction history for each track is saved in a table for further analysis.

PITS was originally developed to simulate short track segments of positive ions in the gas phase of water whose energy was assumed to be constant over the segment. The simulations in the current study use a modified version of PITS, where mono-energetic electrons are the primary particle, and their energy diminishes as the simulation progresses until they fall below the 10 eV threshold. Further, the study uses the updated version of PITS that includes liquid water as the interaction medium.

A diagram of the PITS code structure for this study is shown in Figure 2-1. Module **pits.f** is the main driver for the entire simulation. It first reads in two files and stores their data in memory. The first file, **inpits.dat**, contains data that defines the simulation, such as the number of tracks to be simulated, the initial energy of the primary particle, the starting coordinate of the particle microbeam, the maximum angle of the beam, and the seed for the random number generator. The second input file, **crossec.dat**, contains the cross-section tables for the interaction probabilities between electrons and liquid water. After reading in the files, tracks of the primary particle and its delta rays are computed. All electron transport is calculated by module **eltran.f**, but if the primary particle is an ion then its transport is calculated directly by pits.f. After the computation of a track, the entire interaction history for the primary electron and its delta rays is stored in the file **ioncoord.dat**, and any post-processing of the track is handled within **scor.f**, using **scor_out.dat** as the output file for the simulation results. A setting in inpits.dat allows one to switch between scoring and creating the ioncoord.dat file. New tracks are computed and scored until the maximum number defined by the user has been achieved.

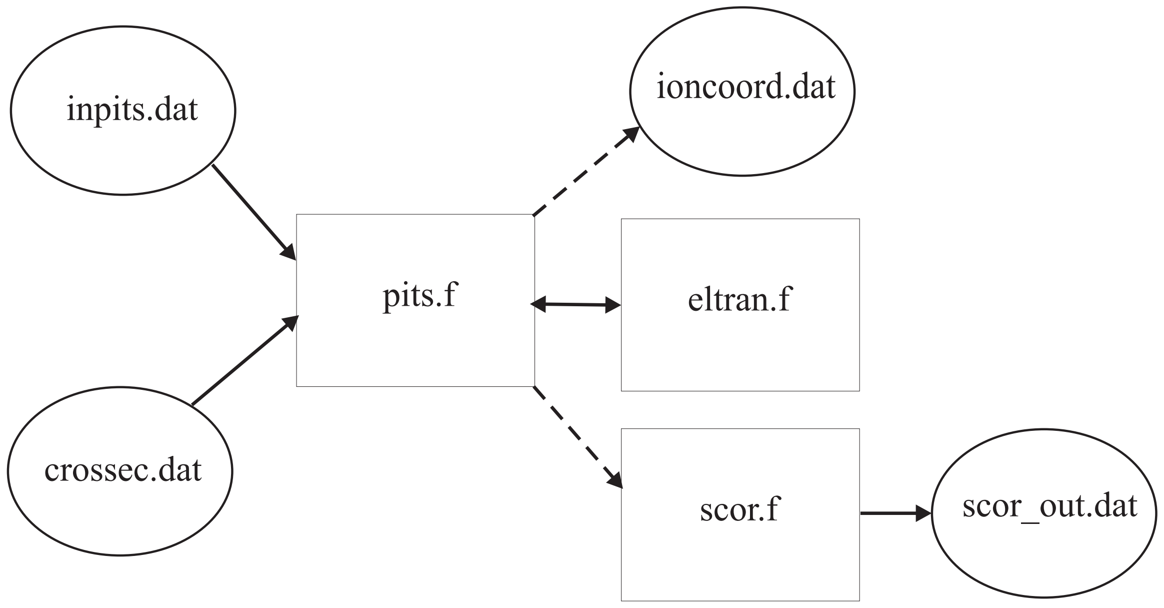

Figure 2-1: Diagram of the PITS code set. Each block represents a module in the set, whereas the data files are represented by ellipses. The arrows indicate the flow of data.

2.1.2 The track_info Array

For every track calculated, PITS stores the track's data in a two dimensional array that is freely shared between the PITS modules and the scoring routine module. This is how the scoring routine gets its information about the track. Overall, there are thirteen columns in the track_info array, one for each statistic that is recorded for a given track, and one row for every inelastic event and thermal death that is calculated. If scoring is not requested, the information contained in track_info is written to ioncoord.dat as it appears in the array. A description for each statistic is given in Table 2-1.

Column	PITS Variable	Description
1	GenNo	Generation of the particle creating the event. • GenNo = 1: The primary particle created the event • GenNo = 2: A child of the primary created the event • GenNo = n: An nth generation particle created the event • GenNo = 96, 97: Thermal death of a particle • GenNo = 99: Secondary dies immediately
2	Ityp	Designates the type of event: • 12-16: excitation • 19-23: ionization
3	deloss	Energy the electron loses during the event in eV
4	time	User-defined column. In this case, it holds the relative time of the event measured as the time it takes a 1eV electron to travel 1Å.
5	dedep	Energy deposited at the interaction in eV
6	w	Energy of the particle entering the interaction in eV
7	ds	Distance traveled by the particle from the previous interaction to this interaction in Å.
8	x	*x*, *y*, *z* coordinates defining the location of the event in Angstroms
9	y	
10	z	
11	cx	Direction cosines of the current particle along the x, y, and z axes. Together, they define the new direction of the particle after the event.
12	cy	
13	cz	

Table 2-1: Layout of the `track_info` array.

Table 2-2 contains the `track_info` array for an electron track segment generated by a 25 keV initial energy electron. The direction cosines were removed from the table for clarity. A graph of the generation numbers in Figure 2-2 illustrates visually how the events are linked to one another. In the figure, inelastic collisions are represented by circles—double circles for ionizations and a single circle for excitations. Thermal death events are denoted by boxes.

	Gen	Ityp	deloss	dedep	w	ds	x	y	z
1	1	21	6.11E+01	1.66E+01	2.32E+04	2.22E+02	8.92E+02	2.47E+03	1.53E+04
2	2	13	8.22E+00	8.22E+00	4.45E+01	1.54E+01	9.02E+02	2.48E+03	1.53E+04
3	2	19	2.22E+01	1.20E+01	3.63E+01	2.24E+01	9.10E+02	2.49E+03	1.54E+04
4	3	12	8.12E+00	8.12E+00	1.02E+01	6.94E+02	9.87E+02	2.40E+03	1.54E+04
5	96	0	2.12E+00	2.12E+00	2.12E+00	6.27E+00	9.91E+02	2.40E+03	1.54E+04
6	2	12	8.33E+00	8.33E+00	1.41E+01	3.57E+02	9.38E+02	2.50E+03	1.53E+04
7	96	0	5.78E+00	5.78E+00	5.78E+00	5.26E-01	9.38E+02	2.50E+03	1.53E+04
8	1	21	2.02E+01	1.66E+01	2.32E+04	2.51E+02	9.31E+02	2.54E+03	1.56E+04
9	99	0	3.61E+00	3.61E+00	3.61E+00	5.18E-01	9.32E+02	2.54E+03	1.56E+04

Table 2-2: Table of the events within a 25 keV electron track segment.

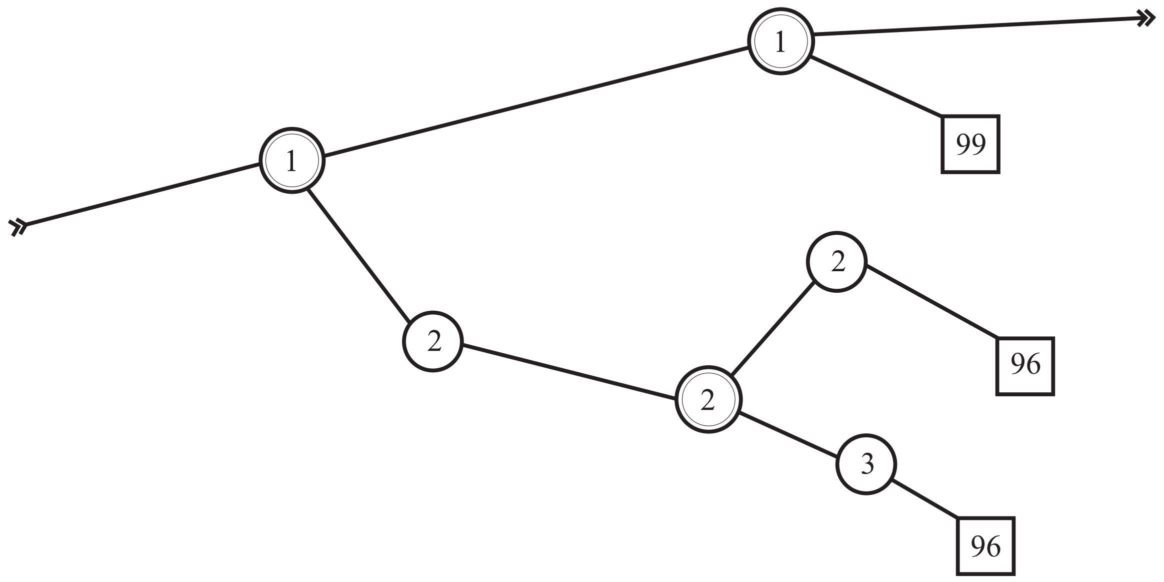

Figure 2-2: Visual depiction of the events within Table 2-2. Error! Reference source not found.

Here is a description of the track segment's event history:

1. In the first event, the primary particle—with a generation number equal to one—has an incoming energy of 23.2 keV and ionizes the medium. It loses 61.1 eV of its energy in the process. The energy actually deposited to the medium is 16 eV, with the difference going into the kinetic energy of the freed electron. The distance traveled by the primary particle between its last event and this event is

222Å. In addition, the location of the interaction is at (x,y,z) = (892 Å, 2470 Å, 15300 Å) relative to the beam entry point.

2. In the second event, the electron freed from event 1, of generation 2, excites the medium.
3. The second generation electron then ionizes the medium to produce a third generation electron.
4. The third generation electron excites the medium.
5. The third generation electron dies.
6. The second generation electron is analyzed once again. It excites the medium.
7. The second generation electron dies.
8. The first generation is analyzed once again. It frees another electron through ionization.
9. The freed electron dies immediately with no excitation or ionization of the medium.

Relating Table 2-2 with Figure 2-2 shows that the `track_info` array is a very simple data structure known as a stack. Stacks are linear repositories of data that can only be accessed from their top element. In this case the top element is the end of the array. All events for a charged particle are pushed onto `track_info` until it creates a child through ionization. The events of the child are then followed until it too creates a child. This process continues until a thermal death occurs, wherein the parent of the dying electron is followed. Viewing Figure 2-2 as a tree, each item is placed on the stack in a depth first manner.

2.2 The Scoring Routine

As previously mentioned, track scoring is optional with PITS. If it is requested in inpits.dat, then an additional module must be supplied and linked with the PITS project. PITS interfaces with this module through a set of subroutine calls with specific names and parameters. PITS drives the scoring process by calling each of these subroutines at the appropriate time in the simulation. The first scoring subroutine to be called, before any tracks have been generated, is `inlinit()`. This is the place to initialize the scoring structures, read in the scoring parameters from the inpits.dat data file (or another file), and carry out any tasks that need to be accomplished prior to the calculation of the tracks. As track generation ensues in the main loop of PITS, the `score()` function is called to process each track. Scoring usually involves looping through the `track_info` array event by event, analyzing a quantity of interest, such as the energy deposited within a site, and then building up a histogram of that quantity. Analyzing the energy deposited for a single spherical site, the algorithm would look like this:

```
energy_deposited = 0 eV
for each event
    (event_x,event_y,event_z) = x,y,z coordinates of event
    (site_x,site_y,site_z) = x,y,z coordinates of site

    r_squared = (site_x - event_x)^2 + (site_y - event_y)^2 +
        (site_z - event_z)^2
    R_squared = (radius of sphere)^2

    if r_squared < R_squared
        // This event is within the target sphere

        energy_deposited = energy_deposited +
            energy_deposited_this_event
    endif
endfor

// Bin energy_deposited in the histogram for energy deposited
// Histograms will be covered in Section 2.3.2.1
```

Algorithm 1: Analyzing energy deposited for a single spherical site.

Finally, after the PITS main loop executes, the subroutines `add_freq()` and `outpfin()` are called. In `add_freq()`, the scoring data structures from each of the processes are aggregated to a single process. Subroutine `outpfin()` performs cleanup duties such as freeing allocated memory and outputting information to the scoring file scor_out.dat.

2.2.1 Running PITS in Parallel

Because particle tracks in a PITS simulation do not interact with each other, the tracks can easily be calculated and scored in parallel on separate processors. The scoring results can then be combined and analyzed on a single processor. This track independence makes the PITS code *embarrassingly parallel*, meaning that there is little cross-talk between processes. The *speedup* in this case is almost perfect: running PITS on *p* processors makes the parallel code almost *p* times faster than the serial code. For example, if it takes 20 minutes to run the serial version of PITS, then the parallel version should probably run in about 4 minutes on 5 processors. In this case the speedup is about 5.

MPI, or Message Passing Interface, is the parallel computing library used by this study. It is currently the de-facto standard library used on machines with processors and memories that are independent from one another. On such machines, every processor is separate from every other processor, and each processor has its own independent memory bank. To run an MPI program, the source code is compiled using a special MPI compiler, a copy of the executable is loaded onto each processor, and the executables are run simultaneously. The executables run independent of one another but share the same original source code. Programming for MPI can therefore be a bit tricky. Communication is achieved between the processes via messages. The simplest messages in MPI are the `send` and `receive` commands, where a process can send data to a target processor or receive data from a target process, respectively. More complex message passing schemes include the communication of a message to all processes at once, known as a one-to-all-broadcast, as well as the summation of a variable from all the processes, known as a reduction.

Some minor changes to the code had to be made in order to run PITS in parallel. The modules affected were the main driver, pits.f, as well as the scoring module, scor.f. In the new code, the header file associated with the MPI library, mpif.h, is #included at the top of the modules. MPI is then initialized by calling a set initialization functions from its library. These functions return the number of processors being used on the system as well as the id of the current process. The tracks are then divided amongst the processors in the source code as illustrated in Algorithm 2:

```
// Divide the tracks equally among the processors
num_tracks_this_processor = num_tracks/num_processors

// If there is a remainder then give the slack to processor 0
extra_tracks = modulus(num_tracks, num_processors)
if (extra_tracks != 0)
      if (id == 0)
            num_tracks_this_processor = num_tracks_this_processor + 1
      endif
endif

// Process each individual track
for track = 1 to number_of_tracks_this_processor
      // Produce the history of one ion track
      // Score the track
endfor
```

Algorithm 2: PITS main driver loop using MPI

The tracks are distributed as evenly as possible amongst the processors by dividing the total number of tracks by the number of processors. The extra tracks, created by taking the modulus of `num_tracks` and `num_processors`, are absorbed by processor zero. For example, suppose that `num_tracks` is 35 and `num_processors` is 4. Then each processor gets 35/4 = 8 tracks and processor zero gets an additional set of tracks equal to modulus(35,4) = 3. Finally, after processing the tracks in the scoring module, the results are aggregated to processor zero by calling the MPI reduction function, and then the MPI shutdown function is called to complete the program.

2.2.2 Random Numbers Using SPRNG

New issues are encountered with random number generation when you parallelize a Monte Carlo simulation code. It is also imperative that a *reliable* random number generation algorithm be used for the Monte Carlo simulations, which typically require the generation of *millions* of random numbers. In general, a parallel random number generation system should

1. provide "high quality" pseudo-random numbers in a computationally inexpensive and scalable manner;
2. provide totally reproducible streams of parallel pseudo-random numbers, independent of the number of processors used in the computation and of the loading produced by sharing of the parallel computer;
3. allow for the creation of unique pseudo-random number streams on a parallel machine with minimal inter-processor communication; and
4. be portable between serial and parallel platforms and must be available on the most commonly used workstations and supercomputers.

A random number package, called SPRNG, incorporates these features. It provides interfaces written in C and FORTRAN to facilitate uncorrelated pseudo-random number streams on different processors, with minimal inter-processor communication. Version 0.1 of SPRNG has been implemented in the version of PITS used for this study.

2.3 Multiple Independent Electron Track Statistics in Cartesian Space

In this section the statistical tools necessary to characterize the distribution of radiation in the simulation space are developed. The section begins by describing the scoring geometry that will be used and outlining its advantages over other geometries. A mathematical review is then made of important topics needed for reference in the body of the paper, namely histograms, probability distributions, probability concepts related to PITS, the estimation of parameters, and goodness of fit testing procedures. Following the review, a discussion is made of the spatial nature of electron scattering wherein the concept of p90 is introduced. The section finally closes with a discussion of the reasoning behind using 1 micron diameter spheres as the targets in the representations.

2.3.1 Cylinder Scoring using Packed Spheres

As introduced in Section 1.4, in order to characterize how microdosimetric stochastic quantities get distributed in space, the target volume absorbing the quantity must first be identified. The simulation space is then packed with identical versions of this volume to get a spatial sense of how the radiation quantity is deposited within *that particular target* when a simulation is run. The resulting data is only pertinent for targets identical with those used to create the data, both in geometry and location. Representing the data with a hypersurface, obtained through either interpolation or a best fit approximation, opens the door to predicting the quantity's deposition for targets *between* the original target sites. This information, however, is still only valid for targets of the same dimensions as the original. If the original targets were one micron diameter spheres, for example, then all future questions about radiation deposition only pertain to one micron diameter spheres.

Which target geometry and packing orientation, then, would best characterize the distribution of radiation in terms of generality, accuracy, flexibility, and applicability to real-world biological problems? Packing cubes in the simulation space would fill the entire space and capture every single event from the tracks, yielding the most statistically accurate results for the same number of tracks. But cells and cell parts are not cubic in nature, and this arrangement must use all three spatial dimensions. Spheres, on the other hand, are easier to work with. They are the best general-purpose target for building the hypersurfaces. How the spheres are packed, however, can make a huge difference. Packing them like a pile of oranges at a grocery store or in a cubic array does not exploit the symmetry inherent in the PITS simulation space.

For a very large number of tracks, the beam of electrons naturally has azimuthal symmetry around the beam entry point. A given track is equally likely to move in any angle around its entry point and the statistical quantities only depend on radius and penetration. It therefore makes sense to exploit this symmetry by running the simulation in a three-dimensional (x,y,z) space, and then summarizing the statistics to a two-dimensional (r, h) space. The two-dimensional space is constructed by packing the spheres in concentric rings as illustrated in Figure 2-3. Then a sphere's location is identified by its distance above the beam entry point, h, and its distance, r, from the beam entry point on the bottom (x,y) plane. Statistically, all spheres within a given ring are assumed to be identical. Using six rings of packed spheres and four layers of rings would result in the two-dimensional layout illustrated in Figure 2-4.

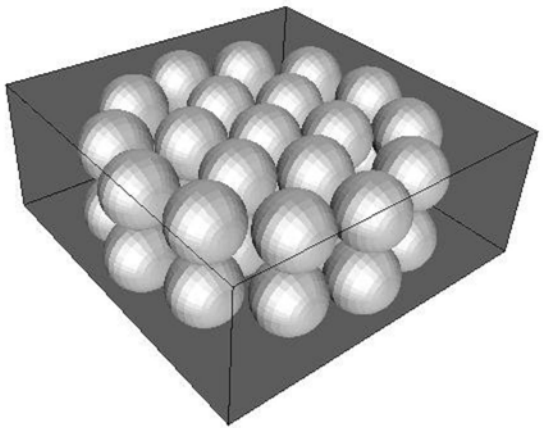

Figure 2-3: Physical arrangement of packed spheres in the simulation space necessary to take advantage of the natural azimuthal symmetry of the electron microbeam.

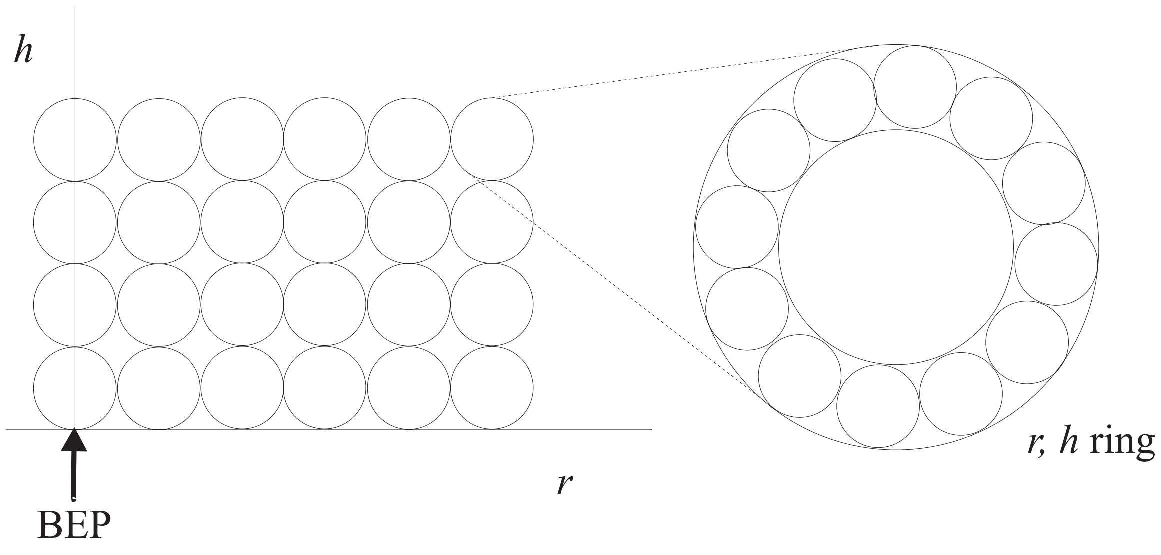

Figure 2-4: Pictorial of spherical target sites in radial distance, *r*, and penetration, *h*. Each sphere in the diagram on the left has a ring of equivalent spheres in the simulation space. Data is presented only for the (*r*,*h*) space, but simulation statistics are gathered in the (*x*,*y*,*z*) space.

The question for all simulations using cylinder scoring is the following: “If an *x* micron radius spherical target is placed a given distance in radius (*r*), penetration (*h*), and angle (θ) from the beam entry point, then, given that a track reaches the sphere, what is the probability distribution of a particular statistic?” The problem, therefore, has two parts: finding the probability that a particle will actually make it to a given sphere in three space, and finding how a given statistic is distributed once the sphere is hit. The mathematics for answering such questions is covered next.

2.3.2 Mathematical Background

2.3.2.1 Histograms and Probability Density Distributions

Think of a histogram as a group of bins that represent values. As an "experiment" proceeds, these bins are then filled with items depending upon the value of the item. For example, suppose the first bin of a histogram represents the value '1'. Then if '1' occurs in the experiment, an item must be placed in the first bin. The total number of items in the first bin at the end of the experiment represents the total number of '1's that occurred in the experiment.

When a statistic is "scored" for a track at a site, the values of that statistic for all events taking place within the site are added together and the histogram bin associated with that statistic is incremented. As additional tracks are run, the histogram keeps track of the counts of the statistic's values. For example, after one hundred tracks, bin three, corresponding with an energy deposited between 200 eV and 300 eV, may contain the value of ten, which means ten out of one hundred tracks deposited energy in that range.

A decision must be made on the binning system to be used. A histogram typically has bins of equal size so that the height of the resulting probability distribution is independent of the bin width used. Many quantities in a given PITS simulation, however, occur primarily at low values, and occur less frequently as their value increases. For example, the energy that a single 25 keV electron microbeam track deposits within a one micron diameter sphere is most likely around 1 keV, yet depositions of 10 keV are possible. It is for this reason that a *logarithmic* binning system was developed for most PITS quantities. Every logarithmic bin has four linear bins within it. The first logarithmic bin is centered at 1.5 and has a total width of 1.0. It is split into four sub-bins of width 0.25 each. The second logarithmic bin has a width of 2.0, meaning the n^{th} logarithmic bin has a width of 2^{n-1}. In the case of energy deposited, the bin centers and widths would be in units of eV. Any unit pertaining to the quantity at hand may be used for the bins.

A histogram must be adjusted, however, if unequal bin widths are used. An adjustment is necessary because the number of tallies in a bin interval is represented in a histogram by the area under the histogram over that interval, calculated by taking the count in the bin times the width. Wider bins would have larger areas for the same number of tally counts. This effect must be compensated for by dividing the height of each bin by its width. Then the area of all bins in the histogram would be equal to their respective tally counts.

The *probability density distribution* for a quantity results from dividing the height of the histogram at each bin by the total number of trials. This distribution is useful because the area under it for a contiguous series of bins gives the *probability* of the quantity lying within those bins. If the height of a bin is four and its width one, then the probability density at that bin after ten trials is 4/(1x10) = 0.4 and the probability of the quantity in question lying within the bin is 1x0.4 = 0.4. The example in Table 2-3 shows these principles in more detail. It calculates the probability density distribution for a histogram with 6 unequal bin widths and a total of 21 trials. The probability of the histogram's quantity lying within the first *i* bins is given in the last column. Notice that the probability of the quantity being in any of the bins is one, or 100%. A visual representation of the original histogram and its transformation into a probability density distribution is given in Figure 2-5.

Bin=i	Center=C_i	Width=W_i	Height=H_i	H_i/W_i	$P_i = H_i/W_i$/num_trials	Area up to right edge of bin
1	0.5	1	1	1	0.048	0.048
2	1.5	1	3	3	0.143	0.191
3	3	2	8	4	0.190	0.571
4	5	2	6	3	0.143	0.857
5	6.5	1	2	2	0.095	0.952
6	7.5	1	1	1	0.048	1

Table 2-3: Example histogram table for a size bin unequal width histogram.

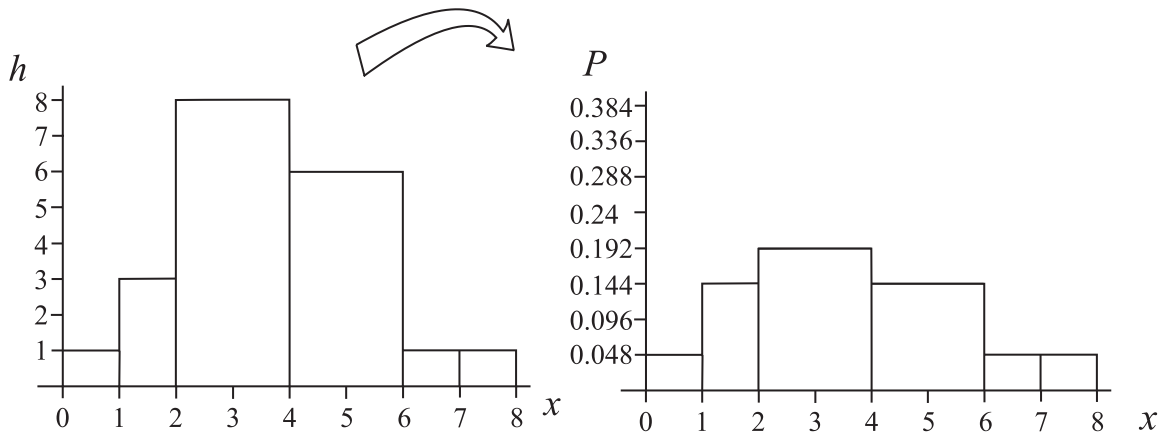

Figure 2-5: Visual representation of Table 2-3. The diagram on the left is a histogram that has not been normalized with respect to its bin widths. The diagram on the right is a proper probability density distribution for the histogram.

2.3.2.2 PITS Probability Density Distributions

Since the behavior of tracks reaching any sphere in a ring at radius *r* and penetration *h* is statistically the same, a single histogram can be built for all targets in the ring. The probability density distribution resulting from this histogram then applies to any sphere located in the ring, independent of the number of spheres in the ring. A histogram's height is incremented at bin *i* when a track crosses one of the spheres in the ring and deposits the a value of the stochastic associated with that bin. Letting

x_i = center of scoring bin *i*,

Δx_i = width of scoring bin *i*,

$n(r,h)_i$ = bin height for a scoring site ring with radius r and penetration h,

the probability density function within an equivalent sphere at ring (r,h) for the stochastic *X* and bin *i* becomes

$$f_X(x_i,r,h) = \frac{n(r,h)_i}{\Delta x_i \sum_{i=1}^{N} n(r,h)_i}, \tag{1}$$

where *N* is the total number of bins in the histogram. Once a track reaches a particular site at $r = r_s$ and $h = h_s$, the probability of the track depositing the stochastic *X* within the interval $[x_a, x_b]$ from bin *a* to bin *b* is given by the equation

$$P(x_a \le X < x_b) = \sum_{i=a}^{b} f_X(x_i,r_s,h_s)\Delta x_i\,. \tag{2}$$

The probability of the track depositing *X* over *all* values of *x* is 1. In the limit as the number of bins approaches infinity and their width approaches zero, the probability density function, $f_X(x_i,r_s,h_s)$, becomes a continuous curve, and probabilities are directly related to *areas* under the curve. In mathematical terms, the sum becomes an integral, and the expression for the probability becomes

$$P(x_a \le X < x_b) = \int_{x_a}^{x_b} f(x,r_s,h_s)dx\,. \tag{3}$$

2.3.2.3 Computing the Moments of a PITS Distribution

The *first moment* of a stochastic's distribution about the origin is the weighted average or "mean" of the stochastic. Given an arbitrary *r* and *h*, for the discrete PITS probability density function in (1) it is given by the equation

$$\mu_X^{(1)}(r,h) = E(x) = \sum_{i=1}^{N} x_i f(x_i, r, h)\Delta x_i\,, \tag{4}$$

with *N* being the total number of bins in the distribution and $E(X)$ being read as "the expected value of *x*". The area over the i^{th} bin, $f_X(x_i, r, h)\Delta x_i$, can be thought of as the "weight" of the stochastic at that bin, and the summation of the stochastic over all of these weights gives its central tendency. Taking this concept further, the *second moment* of the stochastic about the origin is the sum of the stochastic over all of the bins raised to the second power, and the *n moment* about the origin is the sum of the stochastic over all of the bins raised to the n^{th} power, or

$$\mu_X^{(n)}(r,h) = E(x^n) = \sum_{i=1}^{N} x_i^n f(x_i, r, h)\Delta x_i\,. \tag{5}$$

When the distribution is continuous, the sum in Equation (5) becomes an integral that is evaluated over the entire domain of *x*. If *x*'s domain is the interval $(-\infty, +\infty)$, for example, Equation (5) becomes

$$\mu_X^{(n)}(r,h) = E(x^n) = \int_{-\infty}^{+\infty} x^n f(x, r, h)dx\,. \tag{6}$$

The preceding are moments about the origin. Moments can also be calculated about any other value of *x*, with moments about the mean being particularly useful. The second moment about the mean, for example, gives a measure of the dispersion of the distribution about the mean, and is known as the *variance*. For a continuous PITS distribution, the *n moment* about the mean μ_X is given by

$$\mu_X^{(n)'}(r,h) = E\left[(x-\mu_X)^n\right] = \int_{-\infty}^{+\infty}(x-\mu_X)^n f(x,r,h)dx \,. \tag{7}$$

2.3.2.4 Computing the Probability of Reaching a Spherical Target in the Cylindrical Scoring Model

In the cylindrical scoring model, the probability that a particle will reach a given spherical target is the product of two separate probabilities. It is the probability of the particle reaching the ring of spheres containing the target multiplied by the probability of the particle reaching a particular sphere within the ring once the particle has reached the ring. Letting

$n(r,h)_i$	= bin height for a scoring site ring with radius *r* and penetration *h*,
t	= number of tracks in the simulation,
$s(r)$	= number of spheres in a ring at radius *r*, and

the probability that a particle will reach a given spherical target in the cylindrical scoring model is

$$\Phi(r,h) = \frac{\sum_{i=1}^{N} n(r,h)_i}{t}\frac{1}{s(r)}, \tag{8}$$

for *N* total bins in the histogram. The overall probability of a track depositing the stochastic *X* within the interval $[x_a, x_b]$ at a given site centered at $r = r_s$ and $h = h_s$, for example, is

$$P(x_a \le X < x_b) = \Phi(r_s,h_s) \times \sum_{i=a}^{b} f(x_i,r_s,h_s)_i \,. \tag{9}$$

2.3.2.5 Normal and Lognormal Probability Distributions

A stochastic *X* is said to be *normally* distributed with parameters μ and σ if its probability density function is given by

$$h_X(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma} \right)^2 \right), \quad -\infty < x < \infty \tag{10}$$

Referring to Figure 2-6 (a) with parameters $\mu = 0$ and $\sigma = 1$, this function is the recognizable "bell" curve that describes a wide range of phenomenon, such as the distribution of foot sizes amongst people or the distribution of their intelligence quotients. The integral of $h_X(x)$ cannot be analytically evaluated, but using the fact that $\int_{-\infty}^{\infty} h_X(x)dx = 1$, along with integration by parts, the calculation of $\mu_X^{(1)}$ for $h_X(x)$ yields μ exactly, while the calculation of its variance yields σ^2. The median, or point *m* such that $P(X < m) = \frac{1}{2}$, as well as the mode, or peak of the distribution, are identical with the mean for the normal distribution because of the distribution's symmetry about the mean.

For a positive stochastic *X*, if the stochastic $Y = \ln(X)$ is normally distributed with mean μ and variance σ^2, then *X* is *lognormally* distributed with parameters μ and σ^2. The lognormal distribution is often used in simulations of variables such as personal incomes, age at first marriage, or tolerance to poison in animals. Applying the transformation $Y = \ln(X)$ to Equation (10) yields a distribution for *X* of the form

$$g_X(x) = \frac{1}{x\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{\ln(x)-\mu}{\sigma} \right)^2 \right), \quad 0 < x < \infty\,. \tag{11}$$

The graph of a lognormal distribution function with parameters $\mu = 0$ and $\sigma = 1$ is given in Figure 2-6 (b). Notice how the function rises quickly up to a peak and then exponentially decays after that. Also notice that the mean, median, and mode each have three different values. For a general set of parameters, the mean, median, and mode are $e^{\mu+\frac{1}{2}\sigma^2}$, e^{μ}, and $e^{\mu-\sigma^2}$, respectively. If the mean is denoted by α then the variance is $\beta^2 = \alpha^2\eta^2$, where $\eta^2 = e^{\sigma^2} - 1$ [Aitchison, 1957].

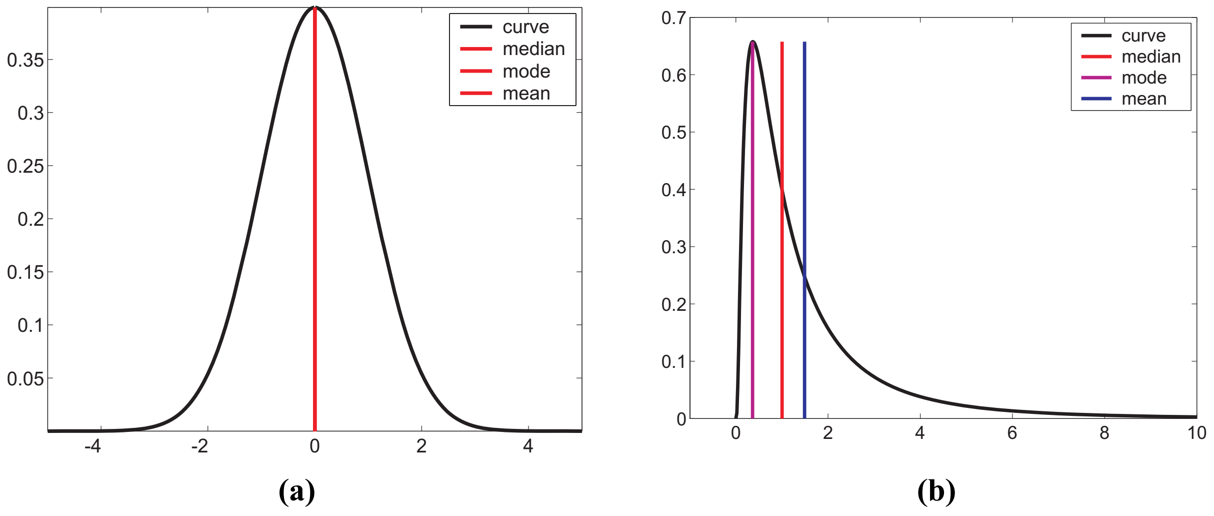

Figure 2-6: Normal distribution and lognormal distribution with parameters (μ , σ^2) = (0, 1), respectively.

2.3.2.6 Estimation of Distribution Parameters

Often a data set is known to be governed by a particular distribution, such as a normal distribution, but the parameters of the distribution are unknown. There are several methods that can be employed to find these parameters, each with varying levels of success given the characteristics of the data set. Four different methods of finding these point estimates are given below:

1. Method of nonlinear least squares

In the method of nonlinear least squares, the unknown parameters in the distribution curve are determined such that the sum of the squared deviations between the distribution and data points is minimized. This method is more or less “brute force”, ignoring the statistical properties of the underlying distribution, and many points spanning the entire distribution must be supplied for a successful fit. The technique of fitting with nonlinear least squares will be highlighted in Section 2.5.5.

2. Probability plotting

Probability plotting involves linearizing the distribution and plotting the data on a special type of paper such that the parameters can be determined from the best fit line through the data. This process can be automated using linear least squares on the transformed data set. Probability plotting is cumbersome and requires special tables and plotting axes for each distribution used. Therefore, it will not be considered in this paper.

3. Method of maximum likelihood

In the method of maximum likelihood, each data point is assumed to come from the distribution being considered, and as a consequence has a certain probability of occurring. Finding the joint probability density function for all of the data points and then maximizing this *likelihood function* (or its natural logarithm) with respect to the unknown parameters gives parameters that are most compatible with the data. For a single parameter θ, if $X_1, X_2, \ldots, X_n$ is the randomly sampled data set from the distribution $f_X(x;\theta)$, then the likelihood function is

$$\begin{aligned} L(\theta) &= f_{X_1,X_2,\ldots,X_N}(x_1,x_2,\ldots,x_N;\theta) \\ &= f_X(x_1;\theta)\cdots f_X(x_N;\theta) \\ &= \prod_{i=1}^{N} f_X(x_i;\theta) \end{aligned} \tag{12}$$

For the normal distribution, after creating $L(\mu,\sigma^2)$ and then maximizing its natural logarithm in terms of μ and σ^2, the maximum likelihood estimators $\hat{\mu}$ and $\hat{\sigma}^2$ of μ and σ^2 become

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i \text{ and } \hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{\mu})^2 , \tag{13}$$

which are just the mean and variance of the data set, respectively. The maximum likelihood estimators for the same parameters in the lognormal distribution are

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} \ln x_i \text{ and } \hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N}(\ln x_i - \hat{\mu})^2 . \tag{14}$$

Maximum likelihood estimates have nice properties [Larsen, 1986], but they can be difficult to compute for certain distributions.

4. Method of moments

The method of moments finds the estimates by setting the distribution moments equal to their corresponding data moments and then solving the resulting equations for the estimates. This method is often easier to execute than the method of maximum likelihood and produces "reasonable" estimators, but the estimators are not guaranteed to have the nice properties associated with maximum likelihood estimators.

Let the jth data moment be denoted by $m_X^{(j)} = \frac{1}{N}\sum_{i=1}^{N} x_i^j$. For the normal distribution there are two unknown parameters, so the first two data and distribution moments are needed to find the parameters. Plugging the estimators into the first two distribution moments and then setting them equal to the first two data moments gives

$$E(X) = \hat{\mu} = m_X^{(1)} \text{ and } E(X^2) = \hat{\sigma}^2 + \hat{\mu}^2 = m_X^{(2)} ,$$

so

$$\hat{\mu} = m_X^{(1)} \text{ and } \hat{\sigma}^2 = m_X^{(2)} - \hat{\mu}^2 . \tag{15}$$

For the lognormal distribution, the same process gives

$$E(X) = \exp\left(\hat{\mu} + \tfrac{1}{2}\hat{\sigma}^2\right) = m_X^{(1)} \text{ and } E(X^2) = \exp\left(2\hat{\mu} + 2\hat{\sigma}^2\right) = m_X^{(2)} ,$$

meaning

$$\hat{\mu} = 2\ln\left(m_X^{(1)}\right) - \tfrac{1}{2}\ln\left(m_X^{(2)}\right) \text{ and } \hat{\sigma}^2 = \ln\left(m_X^{(2)}\right) - 2\ln\left(m_X^{(1)}\right). \tag{16}$$

Notice how the maximum likelihood method and method of moments each give identical estimators for the normal distribution parameters but different estimators for the lognormal distribution parameters.

2.3.2.7 Goodness of Fit Testing

Often an experimenter wants to know if a given data set is governed by a particular probability density distribution. In the discipline of statistics there exists a standard way of determining this. Known as goodness of fit testing, it involves arranging the data in non-overlapping intervals, estimating the parameters if needed, calculating a statistic known as a chi-square for each interval, and then rejecting the hypothesis that the data fits the distribution if the sum of the chi-square statistics is greater than a certain value for a given level of confidence. The chi-square statistic here essentially measures the relative error between the expected frequency from the distribution within the interval and the observed frequency. A very large chi-square means the data does not fit the distribution well at all. Details of the goodness of fit testing process will not be introduced here. Rather, it will be introduced in Section 3.4.3 where it will be applied to data for the stochastic quantity Λ at a given location and beam energy.

2.3.3 Statistical Nature of Electron Scattering

Two primary factors are at work in determining the degree of scattering for a charged particle as it travels through matter. The first factor is the particle's mass. Heavy particles, such as ions, tend not to scatter much because their mass is comparable to the mass of the atoms in the medium. Electrons, which are much lighter than the atoms in the interaction medium, tend to scatter a great deal. The kinetic energy of the particle is also a consideration in its degree of scattering. Slower particles have a greater impact on their colliding neighbors, so they can scatter with a greater angle than faster moving particles. These factors explain the track of a typical 25 keV electron from a PITS simulation as shown in Figure 1-2. The electron initially has enough kinetic energy to retain its forward direction, but then tends to curve to a greater extent as its energy decreases.

The nature of the mono-energetic electron beam is also dependent on the average lifetime of the electrons and their average number of events. These in turn are dependent on the initial energy of the electrons. Higher energy electrons produce longer tracks with more events. The number of inelastic events, for example, varies *linearly* with the beam energy, which is evident in Figure 2-7.

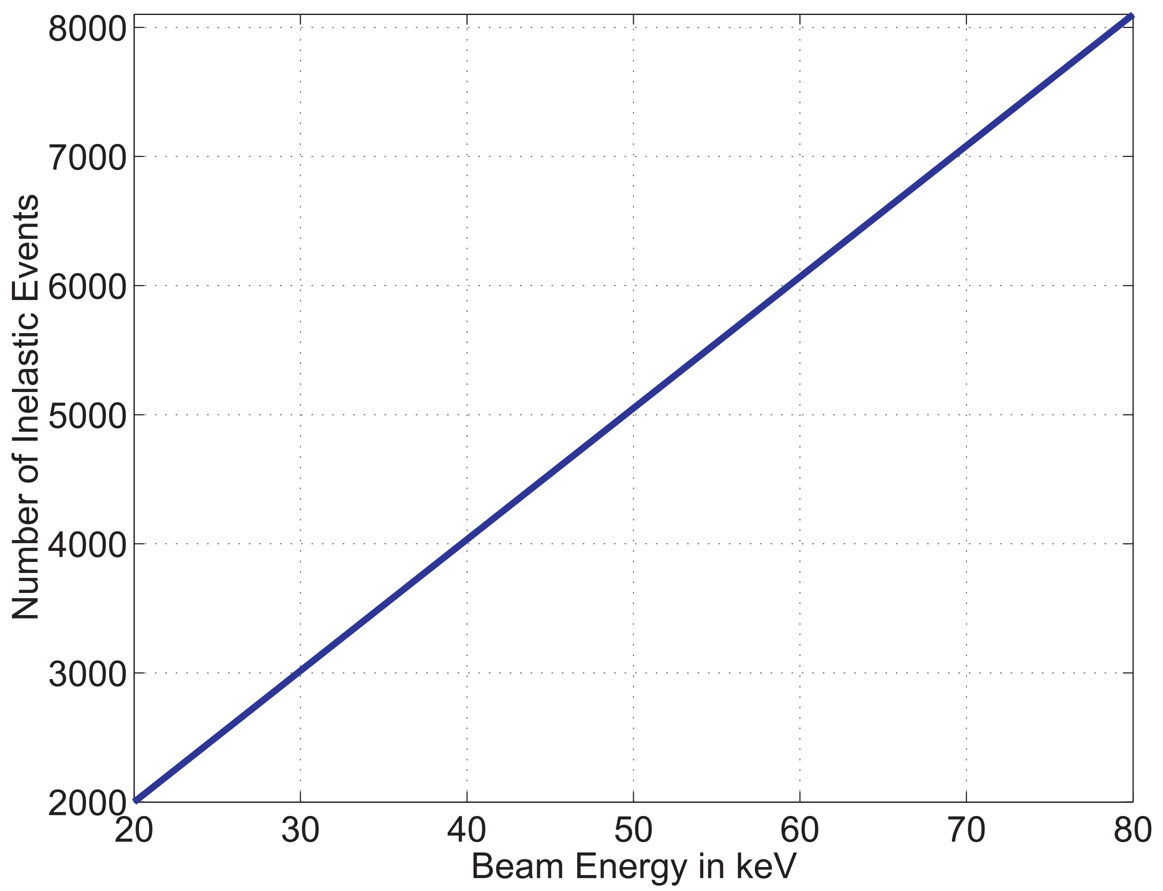

Figure 2-7: Graph showing the number of inelastic events versus the beam energy computed from a series of PITS simulations.

The quantity Φ, defined in Equation (8), gives a picture of the collective nature of the tracks in a mono-energetic electron microbeam. In the present study it represents the probability that an electron track will reach a one micron spherical site at various locations in forward and radial penetration. Figure 2-8 shows $\log_{10}(\Phi)$ versus radial penetration (r) on the x-axis and forward penetration (h) on the y-axis for 25, 50, and 80 keV electron microbeams. The beam entry point is on the lower left of the sub-figures. Notice from the red peaks that the beams have a high propensity to propagate forward near their entry points and then diffuse outward as the electrons lose energy. These high probability areas near the entry points are actually about the same size for all energies, extending about 10 microns along the beam axis. The very low probability outer boundaries extend to 14 microns for the 25 keV case, 40 microns for the 50 keV case, and 85 microns for the 80 keV case. Notice how rapidly Φ drops off in each subfigure, remembering that these are log plots. In the 25 keV case, Φ is $10^0 = 1$ for a 1 micron diameter spherical target lying right above the beam entry point. For a sphere centered at h = 9.5 microns and r = 9.0 microns, Φ is $10^{-5.63}$ = 2.339E-06, a very low probability.

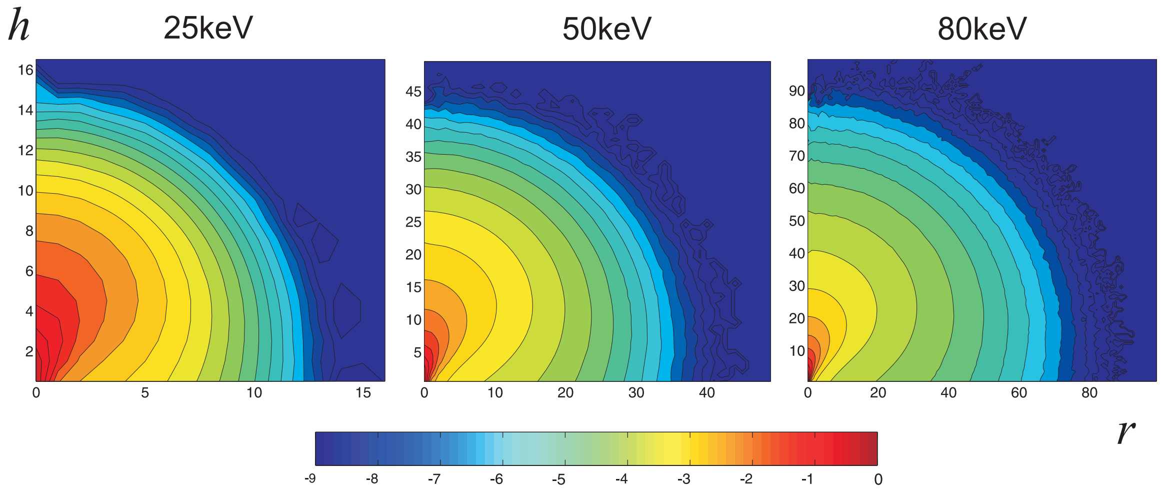

Figure 2-8: $\log_{10}(\Phi(r,h))$ **contour plots for 25keV, 50keV, and 80 keV electron microbeams, respectively.**

The distance from the beam entry point at which 90% of a beam's energy has been absorbed is called the "p90" of that beam. Every mono-energetic electron microbeam of a given energy has a unique p90. In the case of the 25, 50, and 80 keV beams using the liquid-based PITS model, it is 8.7, 26.8, and 56.2 microns, respectively. The p90 is measured by placing concentric spherical shells centered at the beam entry point in the simulation space, simulating many particle tracks, and then measuring the track energy depositions within each shell. The shell that envelops 90% of the deposited energy represents the p90. A graph of the p90 for beam energies between 20 keV and 80 keV is shown in Figure 2-9. Notice that the p90 is approximately quadratic over this range. The p90 values will become valuable later in the thesis when the (r, h) spatial axes for each beam energy will be scaled by its corresponding p90 to develop a common spatial grid.

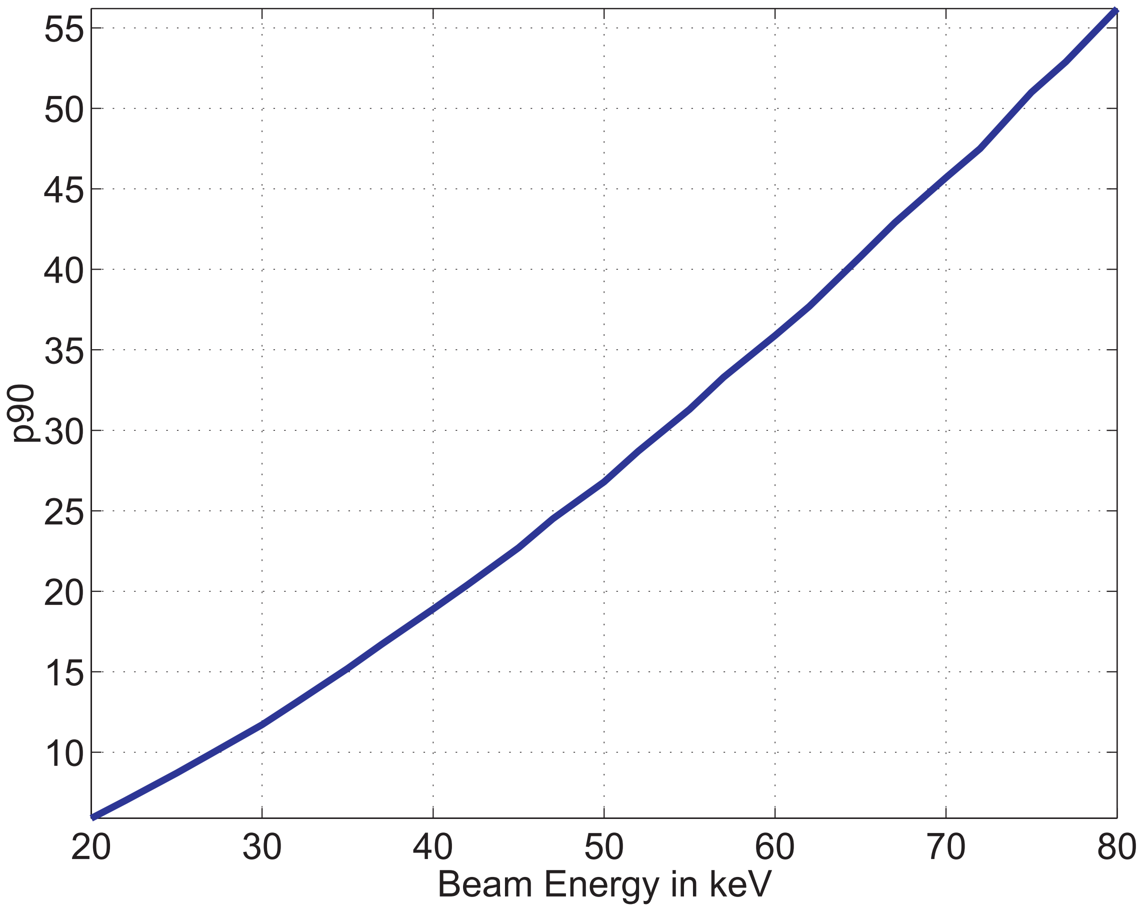

Figure 2-9: Beam energy versus p90 for the liquid-based PITS model.

2.3.4 Choice of Target Size: 1 micron Diameter Spheres

Remember that once a target is chosen in the cylindrical scoring scheme, all future inferences of the scoring data must pertain to targets of that exact size. So why use one micron diameter spheres? What is so special about them? Spheres of any diameter could be used, but one micron diameter spheres were chosen primarily for the following reasons.

1. Targets in microdosimetry have historically been cell nuclei, which have dimensions on the order of a micron (versus a nanometer).
2. They have set a precedent. Experimenters were using them before the modern study of molecular biology.
3. They are convenient to work with. Spheres are easier to define and score within the code than other volume geometries.
4. Packed spheres of this size are small enough to generate data with a sufficient level of detail over the simulation domain, yet large enough to make the data set size manageable.

2.4 Production and Visualization of Stochastic Distributions

This section gives the nuts and bolts of gathering and processing the stochastic distribution data from PITS simulations. It describes the software developed to run the PITS simulations, build histograms for the stochastic distributions, parse the resulting data files, and process the resulting distributions.

2.4.1 Scoring with Module scor_s.f

As described earlier in Section 2.2, the scoring module is the place in the PITS code where generated tracks are analyzed. The module used for scoring in this study is named scor_s.f. In the function `inlinit()` it sets up the binning system and geometry describing the cylindrically packed spheres. To process a track in the function `scor()`, it loops through the events from the track and determines the amount of each stochastic accumulated in the spheres. The amount is then binned in the histogram for that stochastic and ring where the sphere resides. There is one histogram per stochastic per (*r*, *h*) ring. After all of the tracks have been calculated, the function `add_freq()` is called to aggregate the histograms onto one processor and `outpfin()` is called to write an XML data file of the histograms. The actual probability distributions are calculated and analyzed by post-processing scripts. Table 2-4 shows the per-track stochastics scored by module scor_s.f. A detailed pseudocode listing for scor_s.f is given in the appendix.

Stochastic	Description
E	Total energy deposited by all events in the track
E_p	Total energy deposited by the primary particle only
L	Total length of the track, including the primary particle and delta rays
L_p	Total length of the primary particle's track
Λ	E/L_p
Λ'	E/L
Λ''	E_p/L_p
N_p	Total number of primary inelastic events

Table 2-4: Per-track stochastics scored by module scor_s.f.

When scor_s.f was originally written, it allocated a FORTRAN array for the accumulation of each stochastic. The number of cells in each array was equal to the number of cylinders times the number of layers times the maximum number of sites in the cylindrical rings. For very large domains, such as the 80 keV case with tracks reaching as far as 100 microns from the beam entry point, the memory usage of these arrays was prohibitive. Since each track only goes through a very small percentage of the packed target spheres, a *sparse representation* of the arrays was a logical fix for the memory issue. The implementation of a sparse array involves using one array to hold the quantity of interest and another array to hold the indices for that quantity. Only non-zero items are stored in these arrays. The energy deposited for a site at layer 4, cylinder 5, and sphere 6 within the cylinder would have the indices (4, 5, 6) stored in the indices array and the accumulated energy stored in the energy deposited array.

The data generated by a PITS simulation made with scor_s.f is encapsulated within a hierarchy of tags that conform to the XML 1.0 standard. Each of these files contain both a header section and a frequency distribution section. The header section contains data that describes the simulation setup, such as the time and date, the input parameters needed by PITS, the binning geometry, and the packed sphere geometry. The frequency distribution section has a layer→ cylinder→ stochastic tag hierarchy as shown in the XML snippet below:

```
<frequency_distribution>
      <layer id="1">
            <cylinder id="1">
                  <E histogram>
                        count for histogram bin 1
                        count for histogram bin 2
                        ⋮
                  </E histogram>
                  ⋮
            </cylinder>
            ⋮
      </layer >
      ⋮
</frequency_distribution>
```

Figure 2-10: XML code snippet showing the scor_s.f frequency distribution storage scheme.

The individual histogram bins are represented by a single line of integers separated by spaces. XML was used in order to make the data files self-describing and to take advantage of the plethora of XML parsing libraries and tools that are available to developers. Caution should be heeded, however, when choosing to store data within XML tags. The parsing of large XML files can be very slow and memory intensive.

2.4.2 Gathering the Data

A python script, called run_scor_s.py, was written to automate the gathering of data for the production of this study's data set representations. As input, the script uses a set of command line flags to set the initial beam energy, the number of tracks, the sphere radii, the number of cylinders, the number of layers, and the random number seed. It also has flags for controlling the script's execution, such as setting the number of PITS runs that need to be made or whether the output file needs to be zipped. The script's algorithm is outlined below.

```
read in the command line parameters
produce a list of random number seeds, one for each iteration

for each iteration
      create the unique filename for the PITS data file

      generate the inpits.dat file for the PITS run, using the
      iteration's random number seed

      run the parallel version of PITS on all processors

      copy the output to the unique filename and zip if specified
endfor
```

Algorithm 3: Automating the production of PITS data with run_scor_s.py.

This script, however, produces iterations of PITS runs at the same energy and number of tracks. Another script was written to automate the generation of data for many energies and numbers of tracks that calls run_scor_s.py as a subroutine.

2.4.3 Parsing the PITS Data Files

XML can be parsed using a variety of different languages and toolkits. In this study Python was chosen because of its simplicity and familiarity.

Python's XML parsing module, appropriately named "xml", can parse XML using two different methods. The first method, called the Simple API for XML (SAX), searches the document for particular types of events, such as the parser hitting a "start element" tag, and when an appropriate event is found a callback function is invoked to handle that event. The second method, called the Document Object Model (DOM), places the entire XML tree structure into memory, and this tree is traversed using function calls. DOM is much slower and memory intensive than SAX, but it is also more flexible and intuitive for the programmer.

A mixture of the two techniques is used in the PITS post-processing scripts by importing the xml.dom.pulldom module. With pulldom, a SAX-like parsing scheme is used to find each section in the output file. Once found, these sections are then processed using a DOM-like approach.

2.4.4 Aggregating Multiple Runs

Breaking up a PITS simulation into many separate runs and then aggregating the data into a single output file has a couple of advantages. To begin with, it creates data incrementally rather than all at once, thus saving most of the data if a computer or software failure occurs. Multiple runs also allow for the calculation of the data's mean and standard deviation from run to run. These quantities will prove important in the analysis of the data representations produced in Section 3. A major downside to making multiple runs, however, is the additional storage space that is needed to contain the extra data files.

A script was written to aggregate data from multiple runs of a PITS simulation and create a summary data file. The script parses the output file from each run and then combines the histograms and calculates the standard deviations for each bin in the histograms. The resulting aggregate file is similar to the original files, except it includes a few updated tags, aggregated histograms, and the addition of tags for the standard deviations of the histograms.

2.4.5 Visualizing Output Using MATLAB

To visualize the probability density distributions for the stochastics in PITS, a script was written to parse the scor_s.f XML data files, calculate the density distributions from the histograms in those files, write these distributions to a file, and then generate a MATLAB script file to load these distributions and visualize them. A modified version of the script was written to process and visualize the aggregated data files.

Figure 2-11 and Figure 2-12 show what the MATLAB scripts can produce for each distribution at a given initial beam energy. In this case the beam consisted of 3 million electron tracks with an initial energy of 20 keV. Figure 2-11 is the probability density distribution for the energy deposited per unit track length of the primary electron, Λ, for the target sphere directly above the beam entry point at $r = 0$ μm and $h = 0.5$ μm. The area under this curve between λ_1 and λ_2, designated in red, represents the probability that a future track reaching the target sphere and depositing Λ will deposit a value Λ between those values. The horizontal error bars are the same width as the bins, meaning they represent the uncertainty in Λ over their respective domains. Vertical error bars at the bin centers, on the other hand, would represent the uncertainty in $f_{\Lambda}(\lambda_i)$ at bin *i*. They are not displayed because of their very small size relative to the magnitude of $f_{\Lambda}(\lambda)$.

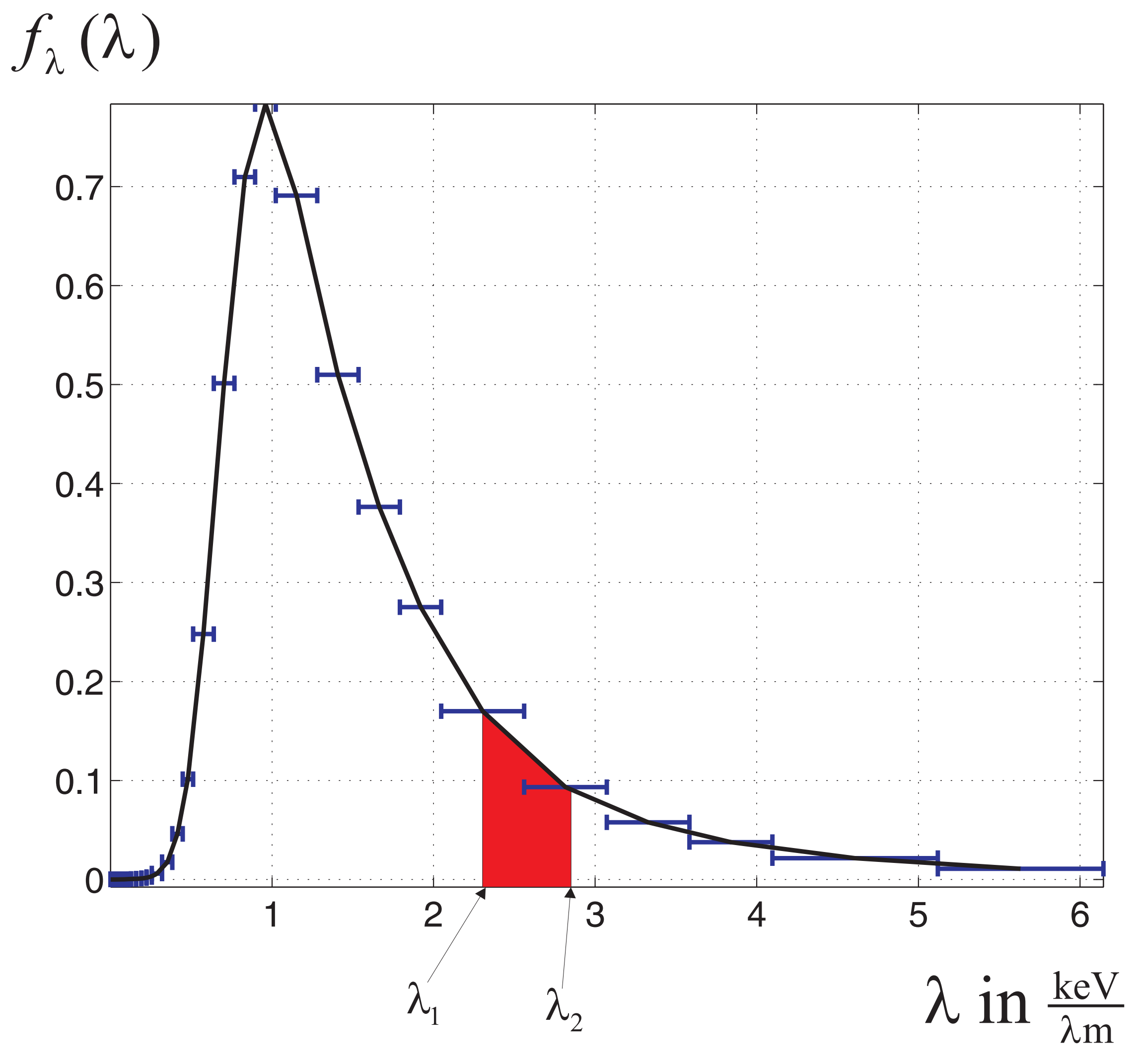

Figure 2-11: Probability density distribution for the stochastic Λ at r = 0 µm and h = 0.5 µm for a 20 keV electron microbeam.

Figure 2-12 is a montage showing $f_{\Lambda}(\lambda)$ for target sites located in the domain of spheres lying in packed rings corresponding with the first four layers and cylinders in the simulation space. The beam entry point is in the lower left corner of the figure, and each "window" represents the equivalent site for a particular layer and cylinder located a distance of (r, h) from the beam entry point. Notice how the distributions look approximately lognormal. This issue will be explored later in Section 3.4.3.

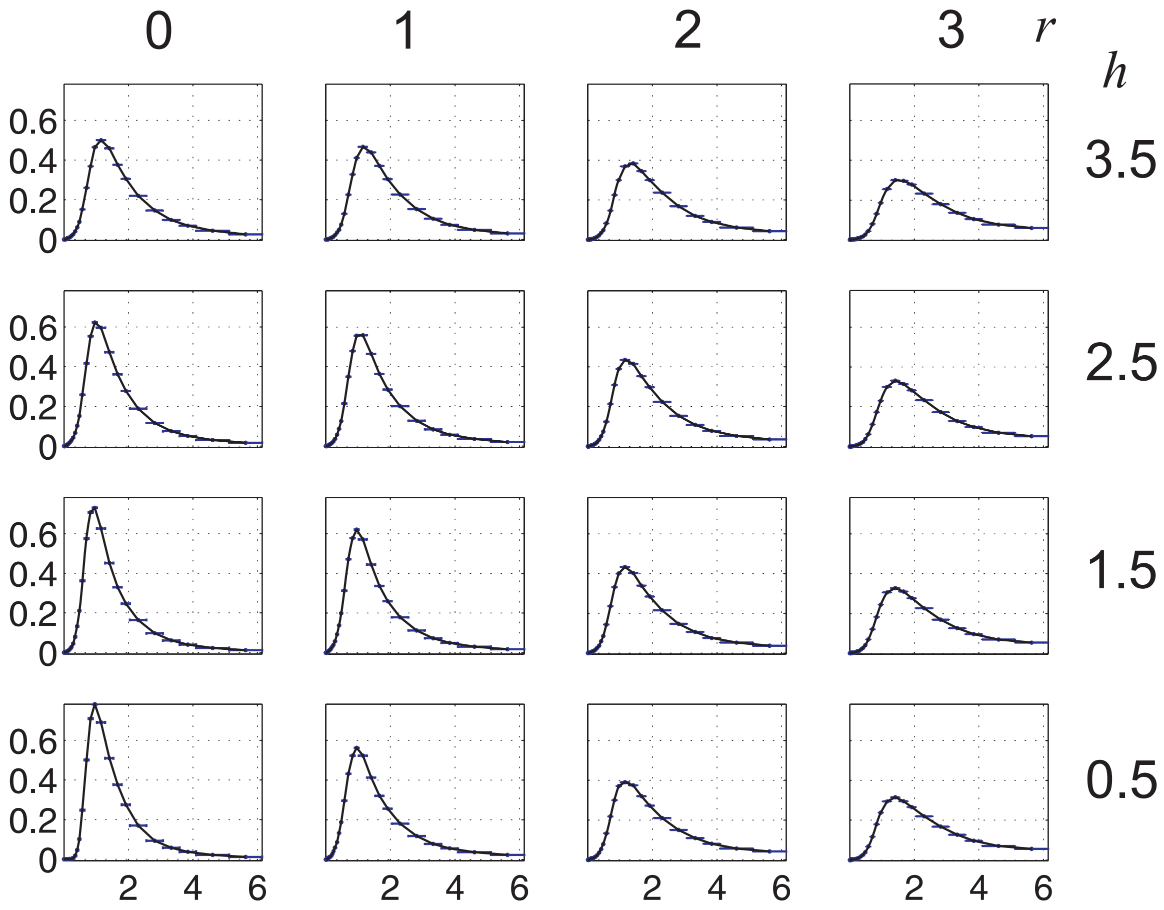

Figure 2-12: Probability density distribution for the stochastic Λ for sites residing in rings located at the first four values of *r* and *h* for a 20 keV microbeam. Each window is the distribution for the respective value of *r* and *h* shown on the top and right of the figure, respectively.

2.5 Mathematical Tools for the Modeling of Data

As stated in the thesis introduction, the project's theme is on the mathematical approximation of data. In this section several differing methods of approximation are introduced and compared with one another based on such criteria as the speed of creation and retrieval, the ease of implementation, accuracy of representation, and storage space requirements. From this information, appropriate methods can be chosen to represent the various PITS quantities and their respective data sets.

2.5.1 Process of Building a Mathematical Representation

Data from a simulation is represented as a discrete set of tuples. Each tuple in the set is called a *point*, and every item in a tuple represents a particular quantity of interest, such as the primary beam energy in a PITS simulation. For example, a data set describing the energy of an electron at three particular locations in Cartesian space could be

$$\mathrm{A} = \{(0,0,0,100),(0,1,2,90),(7,3,4,80)\}, \tag{17}$$

assuming that the first three items in each tuple are the *x*, *y*, and *z* coordinates of the locations, respectively, and the fourth is the electron's energy. Representing the variables themselves as tuples, their values would be $\vec{x} = (0,0,7)$, $\vec{y} = (0,1,3)$, $\vec{z} = (0,2,4)$, and $\vec{e} = (100,90,80)$ for this data set, and their *intervals* would be $I_{\vec{x}} = [0,7]$, $I_{\vec{y}} = [0,3]$, $I_{\vec{z}} = [0,4]$, and $I_{\vec{e}} = [80,100]$. The values for those variables lie within their intervals. For each interval, a bracket denotes equality at its respective edge, while a parenthesis is used in continuous intervals where the edge is not a part of the interval.

The first job in making a representation for a data set is to identify the input and output quantities to the data set. For example, from the data set in Equation (17), you would take snapshots of the position of the electron as it travels away from the beam entry point and then determine from the experiment what the energy was. Or you could view the data in the opposite light, by asking the experiment to give the position for a list of energies that you provide it. Each quantity, whether input or output, is represented by a set. The input quantities are known as the *independent* quantities and their set of tuples constitute the *domain* of the data set. The output quantities are known as the *dependent* quantities and their set of tuples constitute the *range* of the data. The *dimension* of the domain is considered to be the number of quantities in the domain, while the dimension of the range is the number of quantities in the range. The domain for the sample data set in Equation (17), for example, is composed of the set $\{(0,0,0),(0,1,2),(7,3,4)\}$, while the range is the set $\{100, 90, 80\}$. This makes the dimension of the domain 3 and the dimension of the range 1. Data sets, in general, can have many dimensions in the domain and range, but the dimension of the range is usually 1.

Identifying the domain and range of the data set leads directly to the domain and range of the approximating representation. Data from numerical simulations is inherently discrete, but representations are usually continuous in both their domain and range. For continuous representations, each simulation quantity must have a continuous interval associated with it. This interval would naturally be a subset of the variable's corresponding data interval. The representation's domain and range would be the *Cartesian product* of the continuous intervals for variables in the domain and range, respectively. This means that in the example data set, the representation's continuous domain would contain all real numbers (x', y', z') such that $0 \le x' \le 7$, $0 \le y' \le 3$, and $0 \le z' \le 4$.

A typical set of input parameters to a fitting process is comprised of 1) the domain and range of the representation, 2) the domain and range of the data set, 3) the entire group of data set points, and 4) an error tolerance. Sometimes an initial "guess" for the fit must be provided as well. The chosen method of approximation is then run until the error tolerance is reached between the data set points and the fit. Or in place of the tolerance, a specific number of refinement levels could be specified. The evaluation of a fit using the representation is performed via a sub-routine where, given a tuple in the representation's domain, a tuple lying in the representation's range is returned. Assuming the domain and range of the representation are continuous, any tuple lying in the domain can be evaluated. The performance of the approximation method is gauged by the storage needed for the representation, accuracy of the fit, running time needed to create the representation, running time needed to evaluate the fit, and the complexity of implementing the method.

2.5.2 Assessing a Representation's Accuracy

There are a variety of ways to test a representation against its original data set. The crudest, of course, is to evaluate the representation at the data points and compare the two sets of points on the same set of axes over the fit's domain. Another method is to calculate the error between the dependent variables of the fit and some test points. In the case of one dimension in the domain, if the set of *N* test points is written as $\{(x_1, f(x_1)), (x_2, f(x_2)), \ldots, (x_N, f(x_N))\}$ and the fit evaluated at $f(x_i)$ is $g(x_i)$, then the error at x_i for $1 \le i \le N$ is $\mathrm{E}(x_i) = g(x_i) - f(x_i)$. Calculating the error over the test points, however, does not give a global sense of the fit's accuracy. What is needed in this case is a summary statistic of the error function $\mathrm{E}(x)$. The mean of the error itself is not a useful quantity, since $\mathrm{E}(x)$ can be both positive and negative over the fit domain. A statistic on the error called the "root mean square", or RMS for short, is more useful. It gives a single number that assesses the average of the fit's absolute *deviation* from the test data. The RMS for the case of one dimension in the domain is defined as

$$\sigma(\mathrm{E}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\mathrm{E}(x_i))^2}, \tag{18}$$

and it has the same unit as the range of the data set. A measure of the *relative* error of the fit against the range of a test point is given by a quantity called the percent difference. For x_i it is defined as

$$\mathrm{P}(x_i) = \frac{|\mathrm{E}(x_i)|}{x_i} \times 100\%. \tag{19}$$

2.5.3 Original Data as the Representation

2.5.3.1 Description

It may seem ludicrous at first, but the original data can be used as a representation for itself. In this case, the data is stored in a file, and an application that needs that data would use that file as a look-up table. Within the file, every tuple in the data set domain has a one-to-one correspondence with a tuple in its range.

2.5.3.2 Advantages of the Representation

- **Fast to execute –** The primary advantage of the table look-up representation is its speed. Looking up values from the table does not require any processing.

- **Simple –** The table look-up representation is also very simple to implement. The data is placed in a text file and simple access functions are implemented to set and get values from the table.
- **Accurate –** When an independent value is actually in the table then the table will return a 100% accurate result with respect to the data.

2.5.3.3 Disadvantages of the Representation

- **Discrete in nature –** A table of values is by nature discrete, meaning that it only contains a finite number of items. The representation can therefore only be used for tuples in its discrete domain.
- **Storage intensive** – Every data point must be stored in the table. The size of the table grows along with the number of data points collected.
- **No distillation of relationships** – This representation reveals nothing about the underlying characteristics of the data.

2.5.4 Linear Interpolation

2.5.4.1 Description of the Representation

A very simple and accurate representation can be constructed by taking the original data set and connecting the points using linear functions. The representation is the "skin" that is formed around the data. Unlike the look-up table approach, its domain and range are continuous. The representation is stored on disk as a file containing the interval and coefficients for each linear function. Given a tuple in the representation's domain, the corresponding tuple in its range is calculated by determining the data points enclosing it, looking up the appropriate linear function coefficients, and evaluating the linear functions at the domain tuple. Below is an example that illustrates the linear interpolation of a two dimensional data set with a domain of $\{1,2,3,4,5\}$ and a range of $\{1,4,9,16,25\}$.

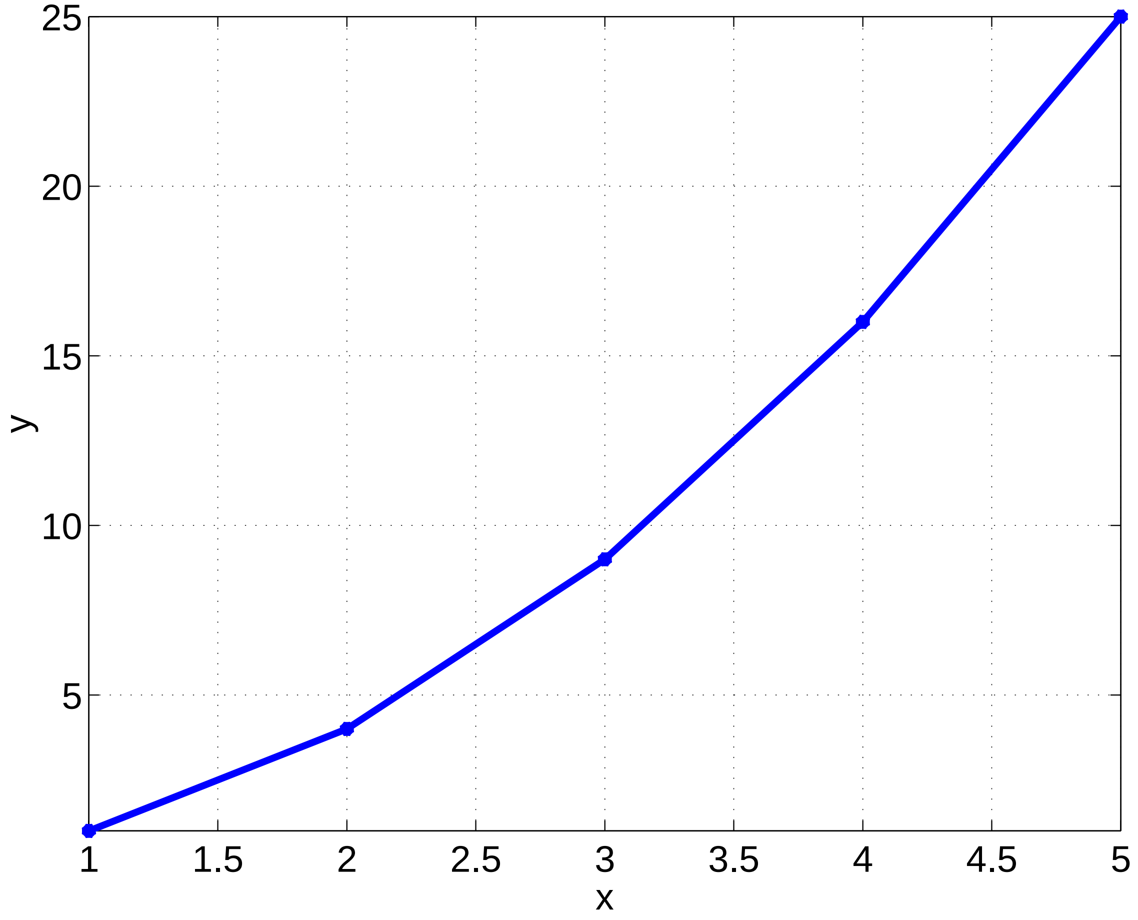

Figure 2-13: Example linear interpolation of a two dimensional data set.

The interval and coefficients for each linear function would be

Interval	C_1	C_2
$[1,2)$	3	-2
$[2,3)$	5	-6
$[3,4)$	7	-12
$[4,5]$	9	-20

To determine y for $x = 2.5$ using the representation,

1. the interval $[2,3)$ is identified,
2. its coefficients are determined as $C_1 = 5$ and $C_2 = -6$, and
3. $y = C_1 x + C_2 = 5(2.5) - 6 = 6.5$.

2.5.4.2 Advantages of the Representation

- **Fast to create and execute –** Creating the table of coefficients and the interval for each linear function is very fast owing to the simple nature of the linear functions. As well, evaluating the linear functions for a particular independent value requires just one calculation of a linear function, though the appropriate interval must be found.
- **Accurate –** The linear interpolation representation is 100% accurate at each original data point, but possibly deviates from new data that is collected between these points.

2.5.4.3 Disadvantages of the Representation

- **Storage intensive –** If there are *n* values in a domain tuple, then *n* + 1 coefficients are necessary for each linear function. Considering there is about one linear function for every data point, and some PITS data sets contain over 50 million points, the amount of storage necessary for such a representation can become daunting.

- **Representation not smooth –** The first derivative of the representation at each original data point is discontinuous. As a consequence, there is no smooth transition between any two adjacent linear functions. Use of higher order polynomials can take care of this problem, but then there are more coefficients to store.
- **Little distillation of relationships –** The linear interpolation method merely takes the data set and "connects the dots". This process does not illuminate any underlying relationships in the data.

Quadratic or cubic polynomials can be used for a closer fit, but the cost of this additional accuracy is the need to store $2n + 1$ coefficients for the quadratic case and $3n + 1$ coefficients for the cubic case. In general, a kth degree spline interpolation requires the storage of $k(n + 1)$ coefficients.

2.5.5 Least Squares Fitting Using Analytic Functions

2.5.5.1 Description of the Representation

If a data set $\{(x_1, f(x_1)), (x_2, f(x_2)), \ldots, (x_N, f(x_N))\}$ looks like it may emanate from an analytic function, such as a straight line, then the process of least squares fitting can be carried out to find the function's parameters such that the function "best fits" the data set. Least squares techniques optimize the fit by minimizing the sum of the square of the deviations between the data and the fitting function. The result is a best estimate for the unknown coefficients in the function in terms of the data points. Symbolically, if

$g(x;a_1,\ldots,a_M)$ is the fit function with coefficients $a_1,\ldots,a_M$, and σ_i is the standard deviation of the i^{th} data point, then this amounts to minimizing the chi-square statistic

$$\chi^2(a_1,\ldots,a_M)=\sum_{i=1}^{N}\left(\frac{g(x_i;a_1,\ldots,a_M)-f(x_i)}{\sigma_i}\right)^2 \tag{20}$$

over all *M* coefficients. If σ_i is not known then it defaults to 1 for all data points.

The process of *linear* least squares can be used if the guessed functional form has its coefficients multiplying each term of the equation, such as the function $f(x)=ae^x+b/x+c$. This holds great advantages because linear least squares fitting is very fast, stable, and guaranteed to give a result. If the coefficients are embedded within a term, such as in the function $f(x)=\sin(e^{ax})$, then the process of *nonlinear* least squares must be applied, and the chance of getting a result that makes sense lowers considerably. Nonlinear least squares techniques ultimately reduce to finding the roots for systems of nonlinear equations, which is a notoriously fickle problem in applied mathematics. An initial guess at the solution must be provided as input to such routines, and convergence to a solution is not guaranteed.

As an example of fitting using linear least squares, suppose you have a data set with the domain {1,2,3,4,5,6,7,8,9,10} and range {1.52, 2.93, 3.71, 4.23, 5.45, 6.17, 7.97, 8.36, 9.05, 10.76}. Fitting a line of the form $y=C_1x+C_2$ to the data gives coefficients $C_1=0.9762$ and $C_2=0.6458$, yielding the best-fit line in Figure 2-14.

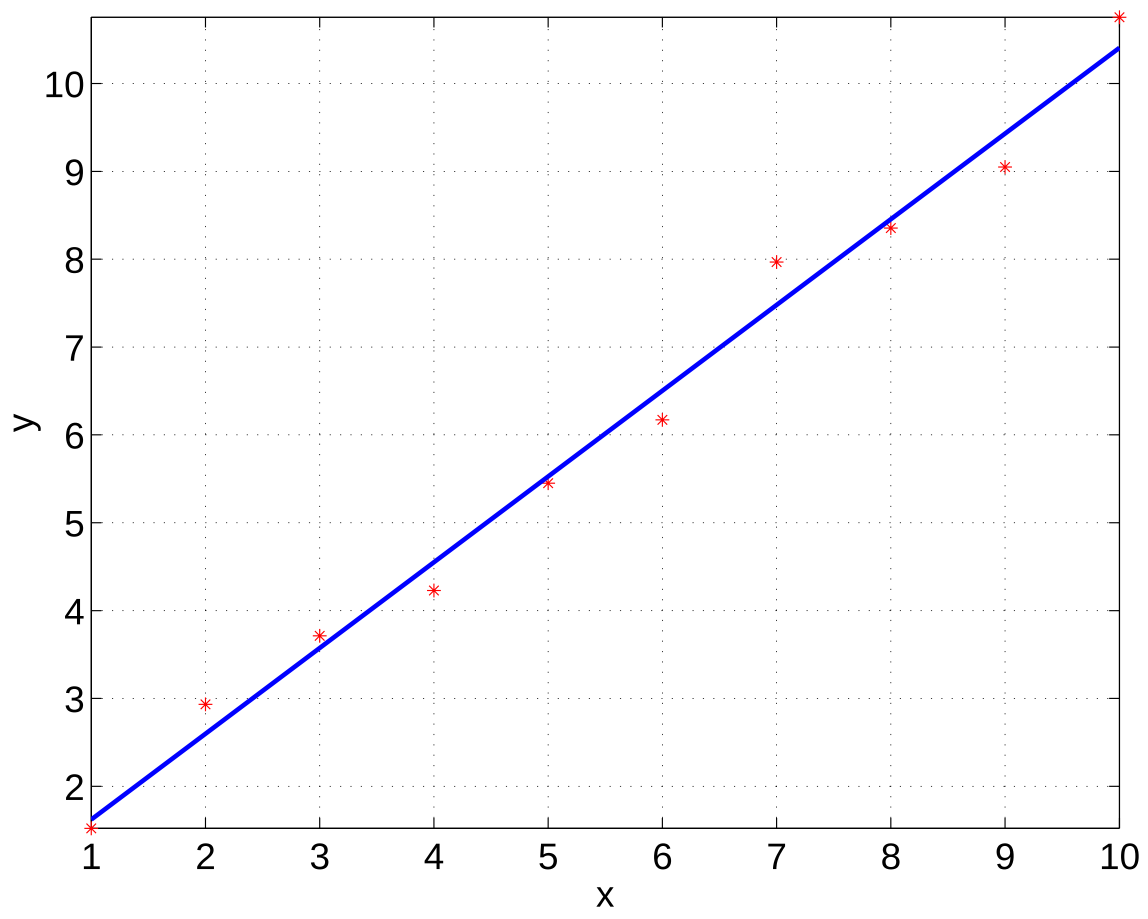

Figure 2-14: An example of fitting a 2-D data set using linear least squares. The data set is represented by the red dots and the best-fit line is the blue line.

The least squares method can be extended to work with multiple dimensions as well. Fitting a data set with two dimensions in its domain and one in its range, for example, would require a function in the form of a surface. Guessing an appropriate functional form in multiple dimensions, however, can be a challenge, since visualizing data beyond two dimensions in the domain is difficult. Multiple dimensions would also add complexity to the task of solving the underlying system of nonlinear equations. Good insight into the nature of the data, in addition with plenty of patience, is needed to arrive at a sensible solution.

2.5.5.2 Advantages of the Representation

- **Minimal storage required –** The representation only has to store the coefficients of the analytic functions, making it extremely efficient with storage.
- **Suggests relationships in the data –** An analytic function is more than just a calculating device. It suggests a trend in the data and can lead to physical insights that other methods of data fitting cannot give.
- **Fast to execute –** Once the analytic function coefficients have been calculated, the evaluation of the function is very fast, especially if it is a polynomial.

2.5.5.3 Disadvantages of the Representation

- **Difficult to find an appropriate fitting function** – Finding a function that fits the data well can be a challenge. The trend in the data must first be recognizable. Then a trial and error process must be undertaken to try out candidate fit functions against the data. Once a candidate is chosen, it is run through a least squares optimization process to get the best fit coefficients. This may need repeated as the fit function is refined. For a large majority of data sets a fit function is all but impossible to find, especially if the data is multidimensional.
- **Nonlinear fits may never converge** – If the fit function is nonlinear then the least squares optimization process many never converge to the best fit. This problem is inherent in nonlinear optimization methods.

2.5.6 B-spline Fitting Using Least Squares

2.5.6.1 Motivation

As illustrated in Section 2.5.5, finding an analytic function to fit a set of data using least squares can be difficult. But what if, instead of using a function over the entire domain, the domain were broken up into pieces and polynomial functions were used to fit the data over each interval in the least squares sense? Then the guesswork involved with using an analytic function for fitting would be overcome, additional control over the fitting process would be gained, and the computation's efficiency would be increased through the use of polynomials. Such techniques exist, and their principle tool is the piecewise polynomial aptly called a *spline function*. Spline functions derive their name from the days of draftsman and the wooden splines they used to make curves.

In Section 2.5.4 splines were used without even knowing it. The lines from the interpolation combined to make up a *spline function of degree 1*, and the points on the domain that made up the interval edges were *knots*. In that case the function itself was continuous at the knots, but its first derivative (slope) was not. In general, a spline function of degree M is a piecewise polynomial of degree M whose M-1 derivatives are all continuous at each knot. The higher degree a spline function is the better it can blend in at the knots because of its higher level of continuity, and the better it can bend between the knots. This increase in fitting flexibility, however, is offset by an increased level of complexity and computational cost. Third degree spline functions seem to provide a nice compromise in practice. They carry enough fitting flexibility for most applications yet are efficient to compute.

2.5.6.2 Introduction to B-spline Curves

A *spline curve* is a device that uses a set of control points with piecewise polynomial weights to generate a curve over the set of knots. If an appropriate set of polynomial weights can be found, then the shape of the final curve can be manipulated by merely changing the location of the control points. The curve can even be forced to move through a given data set and hence interpolate it, or it can approximate the data set through an optimization process such as least squares. Let any tuple in the domain of the curve be associated with a single parameter, *u*. Then, the point on the spline curve at the parameter *u*, represented by $\vec{C}(u)$, can be found by weighting a set of control points, $\vec{P}_i$, by a set of blending functions, $\vec{R}_i(u)$, by the relation

$$\vec{C}(u) = \sum_{i=0}^{h} R_i(u)\vec{P}_i \,. \tag{21}$$

The question now is how to produce the set of blending functions to define the curve for any set of *h*+1 control points. This is where the knots come in. Given the knots defined as a vector in terms of the parameter *u*, there exist several families of blending functions that can be used to generate every possible spline that can be defined over those knots. One such family, the B-spline family of basis functions, has particularly nice properties and is used by the computational spline fitting routines in this project.

The *i*th B-spline basis function using piecewise polynomials of degree *p* over the parameter *u* is written as $N_{i,p}(u)$. Equation (21) then becomes

$$\vec{C}(u) = \sum_{i=0}^{h} N_{i,p}(u)\vec{P}_i \,. \tag{22}$$

Given the knot vector $\vec{u} = (u_0, u_1, \ldots, u_m)$ where $m = h + p + 1$, the basis functions can be recursively created via the formula

$$N_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \le u < u_{i+1} \\ 0 & \text{otherwise} \end{cases}$$
$$N_{i,p}(u) = \left(\frac{u - u_i}{u_{i+p-1} - t_i} \right) N_{i,p-1}(u) + \left(\frac{u_{i+p} - u}{u_{i+p} - u_{i+1}} \right) N_{i+1,p-1}(u) \quad . \qquad (23)$$

But what should the knot vector be to create the set of basis functions? The only requirement is that $m = h + p + 1$ knots be defined within the domain of the parameter *u*. But certain knot spacings are more optimal than others, and if the curve must be clamped to the first and last control points then the first *p*+1 knots must be identical and the last *p*+1 knots must be identical. For example, a knot vector for a B-spline curve with 10 control points (*h* = 9) using cubic polynomials (*p* = 3) over a parameter domain of [0,1] would require $m = h + p + 1$ = 13 knots and a knot vector of the form $\{0,0,0,0,\_,\_,\_,\_,\_,1,1,1,1\}$. Filling in the blanks with equally spaced knots is suitable in most situations.

2.5.6.3 Using B-splines to Interpolate and Approximate Data

Spline functions interpolate data by associating every data point with a knot and then creating piecewise polynomials between the knots by plugging the data into the polynomials and solving for the coefficients. With B-spline curves it is the control points that become the unknowns, since they dictate the shape of the curve given an appropriate knot vector. For *h*+1 data points, finding the *h*+1 required control points amounts to doing the following [Shene, unit 9].

1. Settling on a choice for the degree of the polynomials. This is usually *p* = 3.
2. Finding an appropriate parameter u_k for each data point $\vec{D}_k$ with $0 \le k \le h$.

   There are many ways to do this, but a very simple method is to uniformly space the parameters over [0,1]. There is one item in $\vec{D}_k$ (and $\vec{P}_i$) for each dimension in the data set.

3. Creating a knot vector for the B-spline coefficients with $m = h + p + 1$ knots and a multiplicity of p+1 on both ends of the vector. The vector can be equally spaced or it can be a function of the parameters.
4. Plugging the data points $\vec{D}_k$ into $\vec{C}(u_k)$.
5. Calculating the B-spline coefficients $N_{i,p}(u)$ using the recurrence relation (23).
6. Solving for the control points $\vec{P}$ in terms of $\vec{D}$ and the resulting coefficient matrix $\vec{N}$.

With B-spline interpolation, the curve was forced to contain every data point, and for each data point p coefficients and a corresponding control point were computed. For large data sets this process can be prohibitive both in term of computational cost and storage. The B-spline curve may also display excessive wiggle in order to contain all of the data points. An approximation by a B-spline curve with a limited number of control points that follows the general shape of the data points can prevent these wiggles and reduce the overall storage costs. The only restriction is that the approximation fit the data at the endpoints. Accordingly, let there be n+1 data points, h+1 control points, and B-spline polynomials of degree p, where $1 \le p \le h < n$. Then, given a set of n+1 parameters over [0,1] and an associated knot vector with multiplicity of p+1 at its ends, the control points for the B-spline best fit approximation curve are found by [Shene, unit 9]

1. setting the first data point equal to the first control point and the last data point equal to the last control point, so that $\vec{D}_0 = \vec{C}(0) = \vec{P}_0$ and $\vec{D}_n = \vec{C}(1) = \vec{P}_h$ and

$$\vec{C}(u) = N_{0,p}(u)\vec{D}_0 + \left(\sum_{i=1}^{h-1} N_{i,p}(u)\vec{P}_i\right) + N_{h,p}(u)\vec{D}_n\,; \tag{24}$$

2. creating a function of the unknown control points that is the sum of the square of the errors between the data points and the curve at each control point, viz.

$$f(P_1,\ldots,P_{h-1}) = \sum_{k=1}^{n-1}\left(\vec{D}_k - C(u_k)\right)^2\,; \tag{25}$$

3. and then minimizing this error function with respect to the unknown control points.

The ideas in the previous sections can easily be expanded to multiple dimensions by introducing additional sets of knots, control points, parameters, and basis functions, for a total of one additional set for each additional dimension. For example, in the two dimensional case, given the following:

1. a set of m+1 by n+1 control points $\vec{P}_{i,j}$, where $0 \le i \le m$ and $0 \le j \le n$;
2. a knot vector of h+1 knots in the u-direction;
3. a knot vector of k+1 knots in the v-direction;
4. the degree p in the u-direction;
5. the degree q in the v-direction;

a B-spline surface is defined by the equation,

$$\vec{C}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}(u)N_{j,q}(v)\vec{P}_{i,j}\,. \tag{26}$$

2.5.6.4 Advantages of the Representation

Depending on the size of the data set, number of dimensions in the data set, and number of control points used in each dimension, B-spline approximations carry the following advantages.

- **Have a high degree of Accuracy** – The fit can be "molded" within reason to conform to the local data points or the global nature of the data. Improvements in the fit can be made by having several levels of refinement. Increasing the accuracy, however, has the side effect of increasing the computational and storage cost.
- **Be efficient to create and evaluate** – *Creating* the control points for a *d*-dimensional B-spline has a computational cost in both time and storage that is approximately $O\left(3^D m_0 m_1 \cdots m_{D-1}\right)$ for *D* dimensions and m_i control points in dimension *i* [Weiss, 2000]. Therefore, reducing the number of dimensions in the data can have a significant impact on the overall cost of the control point creation algorithm. For a low number of dimensions (a typical scenario being $D = 2$ or 3), the control point mesh density is the prominent influence in the computational cost. This is not the case for the *evaluation* of the B-splines at a single point, however, since this operation depends solely on the number of dimensions. Keeping the dimensionality low, then, results in a very quick computation for the evaluation.
- **Be extended easily to multiple dimensions** – Because of the flexibility inherent in B-splines, extending the method to multiple dimensions is trivial.
- **Be efficient with coefficient storage** – The density of the nodal point mesh can be easily adjusted as required. Reducing the density, however, has the repercussion of reducing the accuracy of the solution.

- **Be easy to manipulate** – The greatest benefit of using B-splines in the data representations is their level of flexibility. There exists a plethora of "knobs" that can be turned to manipulate the B-spline approximations against the data.

2.5.6.5 Disadvantages of the Representation

B-splines, however, suffer from the following problems.

- **Can distort errant data points –** B-splines are only as good as the data they are fitting. A sparse collection of data points or errant data points can result in a fit that peaks at the data points, even though a flatter surface is envisioned (in 2-D).
- **Do not suggest relationships in the data –** Splines are more or less computational approximation tools, and that is it. They fit the data without any suggestion of a functional form, outside of the fit actually looking like a certain function.

2.6 Tools for Least Squares Analytic Fitting

There are a variety of packages that perform both linear and nonlinear least squares analytic function fitting. For the nonlinear case, nonlinear optimization and nonlinear equation solvers must be present as well. MATLAB's optimization toolkit provides all of these in addition with an easy to use interface, scripting language, and help system. The GNU scientific library is another choice, but is more difficult to use. It was constructed in the C programming language and currently has both C and FORTRAN interfaces. On the Python front, at least two packages are available that perform nonlinear least squares fitting, one being SciPy and the other being Scientific Python. For reasons of convenience, Scientific Python was the package chosen for the nonlinear least squares fitting in this study.

The fitting routine in Scientific Python, called `leastSquaresFit()`, requires a fit function, an initial guess for each of the parameters of the fit function in the form of a tuple, and a list of tuples that contain the data and their associated standard deviations to fit against. The routine then uses the Levenberg-Marquardt algorithm with the automatic calculation of the fitting function's derivatives to find the optimal parameter values. The result is a list containing the optimal parameter values and the chi-square statistic describing the quality of the fit. For example, suppose the fit function is $f(t) = c_1 \exp(-c_2/t)$, the initial guess is (1e13, 4700), and the data set is in the list of tuples called "data", then the Python code to determine the fit would be:

```
from Numeric import exp

from ScientificPython.Scientific.Functions.LeastSquares import \
      leastSquaresFit

def f(param,t):
      return param[0]*exp(-param[1]/t)

data = [(t1,f1),...,(tn,fn)]
print leastSquaresFit(f, (1e13,4700), data)
```

Algorithm 4: Example use of Scientific Python's module Scientific.Functions.LeastSquares.

2.7 Tools for B-spline Fitting

The backbone of all B-spline fitting in this thesis was performed by a multidimensionl B-spline approximation package called BSPLND. BSPLND has a simple interface, computes fits efficiently in multiple dimensions, and is based on a set of state-of-the-art algorithms. It was developed here at WSU Tri-Cities in 2000 by Michael Weiss [Weiss, 2000].

2.7.1 Introduction to BSPLND

BSPLND, which stands for B-spline N-dimensional, is a library of functions coded in the C programming language that implements the multilevel B-spline approximation (MBA) method presented by [Lee et. al., 2000], and extends it to an arbitrary number of dimensions in the domain and range. Techniques used in the MBA method are related to those introduced in Section 2.5.6. The key functions and data structures in the BSPLND library include:

1. **BSplineNDimensional** fit structure

   Description:

   An instance of this structure is the actual fit object that is returned by the fitting routine. It contains all of the data that is needed to describe the fit and can be stored on disk for future evaluations.

   C interface:

   ```
   // Assume DIMS is the maximum number of dimensions needed
   typedef struct
   {
   int D, R;               // Dimensions of the domain and range
   int stride[DIMS];        // A spline property
   int n[DIMS];             // Control lattice mesh density
   double xMin[DIMS];       // Lower bound of the domain
   double xMax[DIMS];       // Upper bound of the domain
   double slope[DIMS];      // A spline property
   double intercept[DIMS];// A spline property
   double *phi;             // Spline coefficients
   } BSplineNDimensional
   ```

2. **bsplnd_fit**

   Description:

This routine takes the data and input parameters and returns a pointer to a `BSplineNDimensional` structure. It also returns the rms error of the fit against the data points. The definition of rms is given in Section 2.5.2.

C interface:

```
BSplineNDimensional *bsplnd_fit
(
int ptCnt,          // Number of scattered data points
int D,              // Dimensionality of the domain
double x[],         // Data from the domain
int R,              // Dimensionality of the range
double y[],         // Data from the range
int n[],            // Initial control lattice mesh density
double xMin[],      // Lower bound of the domain
double xMax[],      // Upper bound of the domain
int h,              // Number of levels of fit refinement
double *rmsError, // Root mean square of the fit errors
int *errCode        // Error code
)
```

3. **bsplnd_eval**

Description:

This routine evaluates the fit at specified points on the fit's domain and returns an error code if any problems occur. The evaluation can be anywhere within this domain.

C interface:

```
int BSPLND_eval
(
BSplineNDimensional *BSPLND,// Pointer to the fit structure
double x[],                 // Values from the domain
double result[]             // Evaluated values in the range
```

```
        )
```

The BSPLND parameters `n` and `h` control the characteristics of the fit, including accuracy, smoothness, and size of the final fit structure. The parameter `n` is the initial density of the control point mesh in each dimension of the independent variable. It must be at least 2 in each dimension. The parameter `h`, on the other hand, is an integer and refers to the number of refinement levels you want the fit routine to use. At every level of refinement the lattice density doubles in each dimension, creating a final control point mesh with $2^h(n_i-1)+3$ control points in each dimension, i, of the independent variable. With an increase in n and no increase in refinement, the accuracy of the fit at the data points is improved at the expense of smoothness in the vicinity of the data points. An increase in h increases accuracy near the data points, but retains some level of smoothness in their vicinity. This mesh information is created dynamically as the `bsplnd_fit` routine runs and is referenced from the fit object via the pointer `phi`.

Using BSPLND to fit a data set involves writing a driver routine in C that reads the data from disk, defines the fitting parameters, allocates memory for the fit structure, calls `bsplnd_fit` to fill the fit structure with the pertinent fit information, and stores the fit structure and control mesh nodes to disk as a binary file. The binary file, then, represents the fit for the data. This fit can later be evaluated by writing a C routine that defines the domain of the evaluation points, loads the fit structure and its control mesh nodes from disk, and calls `bsplnd_eval` for each evaluation point.

Assuming

p is the number of data points defined in the space of refinement,

D is the number of dimensions in the domain,

R is the number of dimensions in the range,

h is the number of refinement levels, and

$m_i = \mathrm{n}_i - 1$ for dimension i,

the performance of the MBA `bsplnd_fit` algorithm used by the BSPLND package is

$$O\left(R\left(hp4^D D + 2^h m_1 m_2 \cdots m_D 3^D\right)\right) \tag{27}$$

in time and

$$O\left(p(D+R) + R2^h m_1 m_2 \cdots m_D\right) \tag{28}$$

in space as the algorithm runs [Weiss, pg 19]. For the PITS data set these benchmarks are not nearly as important as the *storage requirements* of the binary fit file. Its growth depends entirely on the growth of the final control mesh, which is

$$O\left(R2^h m_1 m_2 \cdots m_D\right). \tag{29}$$

To evaluate the fit at a single data point in the domain using `bsplnd_eval`, the complexities in time and space are $O\left(D4^D R\right)$ and $O(D)$, respectively.

2.7.2 MATLAB Gateway Routines for BSPLND

MATLAB was chosen as the tool for visualizing and analyzing the simulation data and fits from the PITS simulations. The BSPLND library, however, was written in the C programming language. Using C functions in MATLAB involves writing a gateway routine called a MEX file, which is a special C file that MATLAB can recognize. You create a MEX file for each C function you want to call from MATLAB. Within it you define the input and output syntax of the associated MATLAB function call and tie this in with the function call to the C routine.

The following MEX file gateway routines were written to access the BSPLND library functions:

1. **bsplnd_fit.c** to interface with bsplnd_fit

   MATLAB call syntax:

   ```
   [fit_struct, rms, err] = bsplnd_fit(x, y, minx, maxx, n, h)
   ```

Parameter	**Description**
In	
`x`	Matrix of data points in the independent variables with each row being a dimension of data and each column a point (D by ptCnt)
`y`	Matrix of data points in the dependent variables with each row being a dimension of data (R by ptCnt)
`minx`	Vector of minimums in each dimension of x
`maxx`	Vector of maximums in each dimension of x
`n`	Vector of the initial mesh density in each dimension of x
`h`	Number of refinements in the bsplnd_fit algorithm
Out	
`fit_struct`	MATLAB structure that corresponds with the BSPLND struct produced from the fit
`rms`	Root mean square of the error of the fit
`err`	Error code

2. **bsplnd_eval.c** to interface with bsplnd_eval using one point

   MATLAB call syntax:

   ```
   [result, err] = bsplnd_eval(fit_struct, x)
   ```

Parameter	Description
In	
`fit_struct`	MATLAB structure that corresponds with the BSPLND struct produced from the fit
`x`	Vector of length D containing a single point in the independent variable to be evaluated.
Out	
`result`	Vector of length R containing the dependent variable
`err`	Error code

3. **bsplnd_eval_points.c** to interface with bsplnd_eval using multiple points

   MATLAB call syntax:

```
[result, err] = bsplnd_eval_points(fit_struct, x)
```

Parameter	Description
In	
`fit_struct`	MATLAB structure that corresponds with the BSPLND struct produced from the fit
`x`	Matrix of P points in the independent variable you want evaluated, with D columns and P rows
Out	
`result`	Matrix of results in the dependent variable, with R columns and P rows
`err`	Error code

Several additional MEX files were written to speed up some of the computations in the MATLAB test scripts. One set of MEX files (each working with a different dimension of data) reads in a list of points and converts the data to a MATLAB n-dimensional grid array. Another set performs the opposite function, converting the MATLAB n-dimensional grid arrays into a list of points that BSPLND can process. Any small routines that were referenced frequently and deeply nested became prime targets for the conversion to C.

3 Results

3.1 Introduction

In this section the spatial characteristics of three previously defined quantities, Φ, μ_E, and $\Lambda = E/L_p$ are approximated. Beam energies in the data set range from 20 keV to 80 keV, and the targets are one micron diameter spheres packed in the simulation space consistent with the guidelines outlined in Section 2.3.1.

As discussed in Section 1.4, each of these quantities was chosen because of their biological significance, as well as their abilities to illustrate the methods. In particular, Φ was chosen because it gives the probability of a future track reaching the target wherever the target is placed in the domain. Knowing this is important in biological experiments using radiation because irradiated target cells have neighbors, and these neighbors too are exposed by that radiation. Φ gives the *chance* of that exposure. The summary statistic μ_E, on the other hand, gives the *degree* of that exposure if a track actually deposits energy in the target. It is the mean of the energy deposited stochastic, summarizing the track's ability to deliver energy to the target, independent of the number of tracks in the experiment. Finally, Λ was chosen because it gives a better indication of radiation quality than the density distribution of energy imparted. It describes the ability of the track to deposit energy over the primary particle's path, thus giving more information on the track's ability to do damage to the target if that target was biological in nature. And, unlike other stochastic quantities, its individual density distributions appear to follow a standard continuous distribution known as the lognormal distribution. Goodness-of-fit testing can confirm if the data is statistically lognormal after an estimation is made of the proposed lognormal distribution fit parameters.

3.2 Data Preparation

All representations developed for this thesis use the same PITS data set. It was obtained by running ten sets of 300,000 track simulations for each initial beam energy of 20 keV up to 80 keV in 5 keV increments. As described in Section 2.4.1, scoring was performed by the module scor_s.f, wherein the simulation domain was cylindrically packed with one-micron diameter spheres and a total of eight different stochastic quantities were analyzed. The output of each simulation was in the form of an XML file containing the ingredients necessary to build the probability density distributions for any of those eight quantities, including the histogram for each distribution, the number of tracks, and the number of spheres in each cylinder. The structure of this XML file is described in Section 2.4.1. The XML files for all ten runs at a given energy were then processed to create a single XML file, wherein the bin values from all of the realizations were coalesced for each bin and their standard deviation was calculated. Together, these aggregated XML files make up the *complete data set* used as the basis for all of the approximations in this thesis. Additional data sets at beam energies of 22, 32, 42, 52, 62, and 72 keV, using the same number of tracks and realizations, were computed for testing purposes.

There are a total of five dimensions in the complete data set for a particular stochastic. Four of these are in the domain and one is in the range. The set of values in each dimension of the domain corresponds with a particular variable in the simulation. As outlined earlier in the paper, these variables include

- *e*, the initial beam energy in units of keV;
- *h*, the penetration of the target's center point above the beam entry point in units of μm;
- *r*, the radial distance from the beam entry point of the target's center point in units of μm; and
- *b*, the histogram bin center, in units of the current stochastic.

Normally the complete data set would be defined by the Cartesian product of the set of values for each of these variables. But the number of target sites required to encompass all of the tracks is not identical at each initial beam energy. This is because higher energy particles tend to penetrate further into the simulation domain than lower energy particles. This is evident from Figure 2-9 in Section 2.3.3, where the p90 values were shown to increase quadratically with an increase in the initial beam energy. The unevenness of the data with energy makes the Cartesian product impossible to compute.

There are several solutions to forming a complete data set. One is to force the target sites to be identical by padding the lower energy target domains with additional sites. All energy levels would then contain the same targets as the highest energy level. Another solution involves first scaling the target site coordinates at each beam energy by the beam's p90 value and then forcing the new p90 sites to be identical by interpolating their grids. This second solution is superior to the first since the target sites at each energy level extend spatially at similar percentages of the p90. Dividing by the p90 places the site data at each energy level on common ground. Recovering the original spatial target coordinates for an energy level is accomplished through multiplying by the p90 for that energy.

To rigorously pin down the storage requirements for the PITS data set, first define the following quantities.

- $\vec{e} = [e_1, e_2, \ldots, e_i, \ldots, e_n]$ = vector of initial beam energies with $e_1 \le e_i \le e_n$, and $1 \le i \le n$. e_i is known as the ith energy level.
- $\vec{r}_i$ = vector of target site radial distances from the beam entry point for the ith energy level.
- $\vec{h}_i$ = vector of target site penetration distances from the beam entry point for the ith energy level.
- n_b = number of bins in each histogram, which is a constant for all histograms.

Using this nomenclature, the set of all coordinates for each target site for the ith energy level is defined by the Cartesian product $\vec{r}_i \times \vec{h}_i$, and the number of target sites is defined by the size of this set, or $\left\|\vec{r}_i \times \vec{h}_i\right\| = \left\|\vec{r}_i\right\| \cdot \left\|\vec{h}_i\right\|$. Therefore, for all n energy levels, the storage requirement growth factor is

$$\begin{aligned} \Psi &= O(\left\|\vec{r}_1\right\| \cdot \left\|\vec{h}_1\right\| + \left\|\vec{r}_2\right\| \cdot \left\|\vec{h}_2\right\| + \cdots + \left\|\vec{r}_n\right\| \cdot \left\|\vec{h}_n\right\|) \\ &= O(n \cdot \left\|\vec{r}_n\right\| \cdot \left\|\vec{h}_n\right\|). \end{aligned} \tag{30}$$

This means that the rate of growth of the storage requirements depends on the total number of energy levels times the number of target sites. The number of bins in each histogram is not a part of Ψ since it is constant for all energy levels. Ψ is important because, even though the data is stored on disk and disk space is relatively cheap, the running time performance and memory usage in the data processing scripts is directly affected by the final size of the data set.

Finally, a note should be made about the simulation parameter values. An energy range of 20 keV to 80 keV was chosen because the department's current grant focuses on electron energies lying within this range. Further, 3,000,000 tracks were generated for each energy level in order to make the data sets for the levels as accurate as possible given the computational resources available. Running a large number of tracks like this does not have an impact on the storage requirements of the project data files. As covered earlier in this section, the storage requirements only depend on the initial beam energy of the simulation and the number of energy levels run.

3.3 Approximating the PITS Data Set Using B-splines

Introduced in Section 2.5.6, B-splines are a very flexible tool for approximating arbitrary data sets. They are easy to manipulate, work equally well over various data set dimensionalities, are efficient to compute, and have a fitting accuracy that is easily adjustable, depending on the accepted nodal mesh density. If implemented in a software package such as BSPLND, they can also be very convenient to work with in the fitting process.

Accordingly, BSPLND is used to create approximations of various accuracies for all of the simulation quantities outlined in this study. In each case, the initial nodal mesh density is held to a minimum, and the final density is adjusted through the refinement parameter in BSPLND. This strategy helps to maintain the fit's smoothness in the vicinity of the data points. The accuracy of the fit against the original and test data sets is then assessed for each level of refinement in terms of the techniques outlined in Section 2.5.2 and compared with the storage cost required for the nodes. *The objective is to find a refinement level that yields an acceptably accurate fit with the smallest possible storage requirements for each of the data sets*.

3.3.1 Approximating the Probability of Reaching a Target Sphere and the Conditional Mean of Energy Deposited for a 25keV Microbeam

3.3.1.1 Background

The first data sets to be approximated are those representing the probability of reaching a target sphere, Φ, and the conditional mean of energy imparted, μ_E, from a 25 keV electron microbeam. These sets were calculated directly from the 25 keV energy level histograms of the parent data set using Equations (8) and (4), respectively. The objective here is to begin the B-spline approximation process using relatively simple data sets in order to develop the concepts needed for fitting and testing the higher dimensioned sets. It is relatively easy to visualize data in three dimensions, since it can be represented as a *surface*, but adding dimensionality to the domain, and range for that matter, makes visualization and analysis increasingly difficult.

One such concept developed in this section is the separation of the data domain into multiple regions in order to obtain a more accurate fit. If the range of the data slowly varies over the entire domain then this technique is not especially useful. The data set for μ_E, for example, follows this trend. If the range rapidly changes over a section of the domain, on the other hand, then accuracy can greatly increase over that region if it is cordoned off and fitted separately, perhaps using a higher level of control mesh refinement. Φ 's data set, for example, shows such characteristics. There is a very high probability that an event will occur in a site very close to the beam entry point, but this probability drops rapidly as distance from the beam entry point increases. At all levels of BSPLND refinement, fits for Φ and μ_E are made using both one and two regions in order to assess of the effectiveness of the technique.

3.3.1.2 Approximating the Probability of Reaching a Target Sphere at 25 keV

The domain of Φ 's data set for a 25 keV beam energy extends out 18 microns in radial distance, *r*, and 18.5 microns in penetration, *h*, yielding a total of 324 (*r*,*h*) tuples. This corresponds with 19 cylinders and 19 layers used in the simulation. Through observation of the data, the high probability region was determined to lie within $r = 4$ microns and $h = 8.5$ microns.

Approximating the original data set proved difficult because of Φ 's very sharp drop near the beam entry point. For this reason, the logarithm base ten of the data set's range was approximated instead. Accuracy of the fit was then assessed by 1) evaluating the fit on the data set's domain tuples, 2) raising the range of this fit data by the power of 10 to convert it to the same base as the original data, and 3) comparing the original data with the converted fit data.

Figure 3-1 visually shows $\log_{10}$ of the original data and its corresponding fit on the same domain, with a BSPLND refinement level of 4. The converted data is represented by black dots in the figure and the fit is represented by the shaded surface. The two fitting regions are displayed as well, separated by a bold line. Notice how closely the converted data matches the evaluated fit points, giving a percent difference of

0.0153% at (0,0.5) and a percent difference of 0.0121% at (4,4.5). The cost of this fit in terms of storage is 2 * 19^2 = 722 nodes.

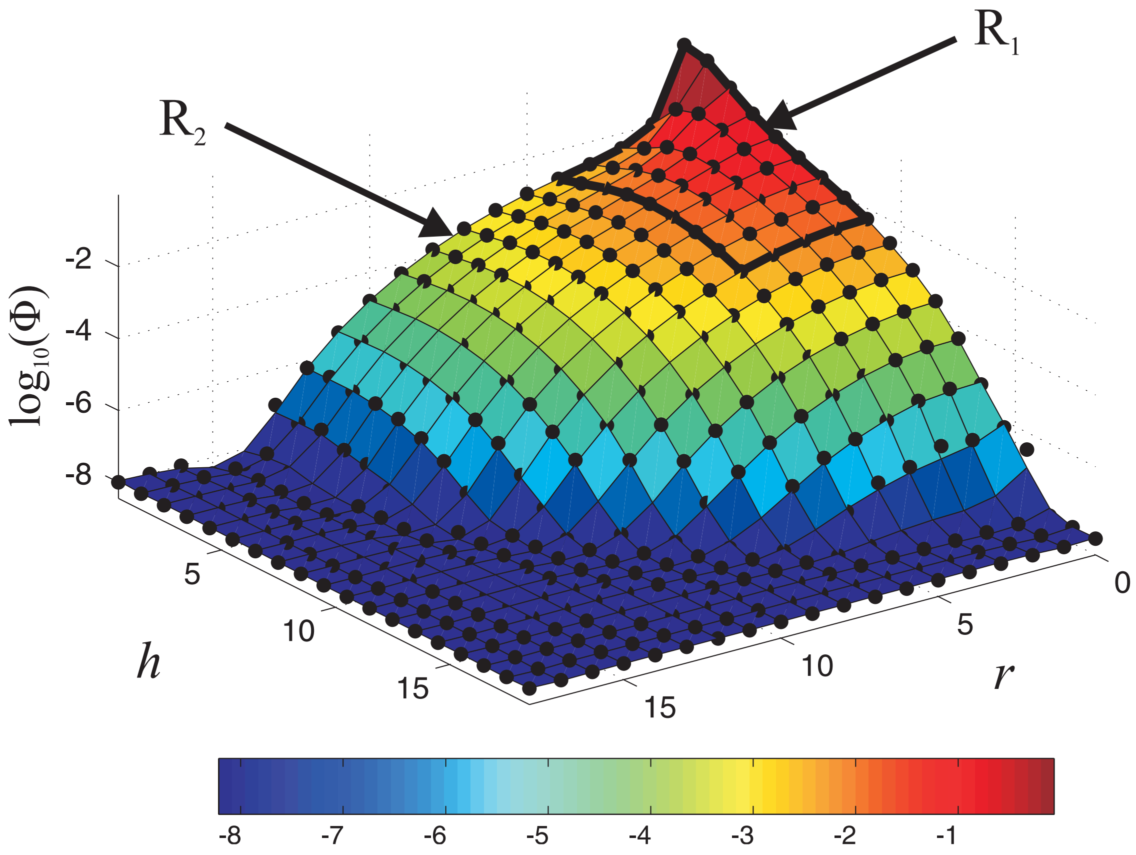

Figure 3-1: Graph comparing $\log_{10}(\Phi_{data})$, represented as the black dots, and $\log_{10}(\Phi_{fit})$, represented as the composite surface composed of regions R_1 and R_2. The BSPLND refinement level is 4.

The domain of Φ at 25 keV extends over twice as far in *r* and *h* as the beam's p90 value. Since the data within the p90 is most valuable, for benchmarking purposes it is useful to restrict the domain to a value just past the p90 in the analysis of the data. *In the forthcoming analysis both r and h are restricted to 1.2 times their p90, or 9.0 and 9.5 microns, respectively.*

Table 3-1 summarizes the fitting of Φ's data for BSPLND refinement levels of 1 through 6, with and without the data split into two regions. The first column indicates the refinement level, K, used in the call to the function `bsplnd_fit`. An 's' is appended if the data is split. The second column contains the RMS for the fit error over the restricted domain, indicated by the symbol σ_{fit}. The data itself has error in it as well, so the third column compares σ_{fit} with σ_{data}, the RMS for the data's error, by taking the *ratio* of the two, $\Sigma = \sigma_{fit} / \sigma_{data}$. If $\Sigma \leq 1$ then, on average, the fit error is within the data error over the relevant domain, and the fit is considered "reasonable"; i.e. it is within the "noise" of the data. The Σ statistic will be the dominate tool in deciding on a sufficient refinement level for all forthcoming PITS data sets. **The cut-off, K_c, will be a K chosen such that C is a minimum for a $\leq$ / , or within half of a "reasonable" fit**. Column four contains the cost, C, of the fit in terms of the number of B-spline nodes required in the fit. Splitting a data set doubles C, owing to the fact that each region requires its own set of nodes. Finally, columns five through nine contain the percent difference at five different tuples in the restricted domain, one at the beam entry point, three near the p90 at the edge of the restricted domain, and one in the center of the domain at (4, 4.5).

K	σ_{fit}	$\Sigma = \sigma_{fit} / \sigma_{data}$	C	$\mathrm{P}(r,h)$				
				(0, 0.5)	(0, 9.5)	(9.5, 0)	(9.5, 9.5)	(4, 4.5)
1	0.1119	718.62	5^2	95.19	81.27	12.92	59.37	81.13
1 s	0.0837	537.70	$2 * 5^2$	73.08	96.39	41.24	67.72	16.57
2	0.0991	636.41	7^2	86.87	13.94	12.10	50.70	17.15
2 s	0.0427	274.25	$2 * 7^2$	38.56	55.08	6.51	42.63	3.90
3	0.0921	591.52	11^2	81.90	22.13	15.74	26.36	0.6940
3 s	0.0019	12.36	$2 * 11^2$	0.0823	14.09	13.35	28.33	2.37
4	0.0370	237.85	19^2	38.82	2.470	1.637	1.831	0.9152
4 s	1.563e-04	1.0042	$2 * 19^2$	0.0153	5.534	1.132	2.580	0.0121
5	0.0087	56.14	35^2	9.496	0.5027	0.0412	2.336	0.0841
5 s	1.663e-06	0.0107	$2 * 35^2$	1.11e-14	0.0079	0.0704	2.252	0
6	1.941e-10	1.247e-06	67^2	1.55e-13	2.11e-13	2.04e-13	0	5.72e-07
6 s	2.16e-12	1.39e-08	$2 * 67^2$	1.11e-14	9.57e-14	2.04e-13	1.99e-13	1.01e-13

Table 3-1: Results of fitting Φ's data for BSPLND refinement levels of 1 through 6, both with the data split into two regions and without.

From the table, notice how significantly splitting the data into regions affects the accuracy of the fit near the beam entry point in terms of the percent difference. Notice as well how this improvement is amplified as the refinement level is increased. With a general refinement level of 4, for example, multiple regions give over a 99.96% improvement in $\mathrm{P}(0,0.5)$ and a 99.58% improvement in Σ, yet with a refinement of 2 the improvement is only 55.61% and 56.91%, respectively. As it improves, Σ first falls below $1/2$ with a minimal cost of $2{*}35^2$ nodes at K = 5s. Therefore, K_c is 5s for this data set.

The improvement of the fit is illustrated graphically in Figure 3-2. It is a montage showing how the fit error, $\mathrm{E}_{fit} = \Phi_{data} - \Phi_{fit}$, relates with the standard deviation in the data, E_{data}, along the domain for various levels of split refinement. The independent variable in each window is the forward penetration, *h*, and the dependent variable is the ratio of the fit error to the data error, $\mathrm{E}_{fit} / \mathrm{E}_{data}$. This is assuming the radial penetration, *r*, and the refinement level, K, are constant for that window, each corresponding with the window's column and row, respectively. If $\mathrm{E}_{fit} / \mathrm{E}_{data} < \pm 1$ for a window, i.e. lies within the bars, then E_{fit} is within the data's error range. From the figure, as K is increased for all values of *r*, $\mathrm{E}_{fit} / \mathrm{E}_{data}$ approaches the bars, lying almost wholly within them at K = 4s and completely within them at K = 5s.

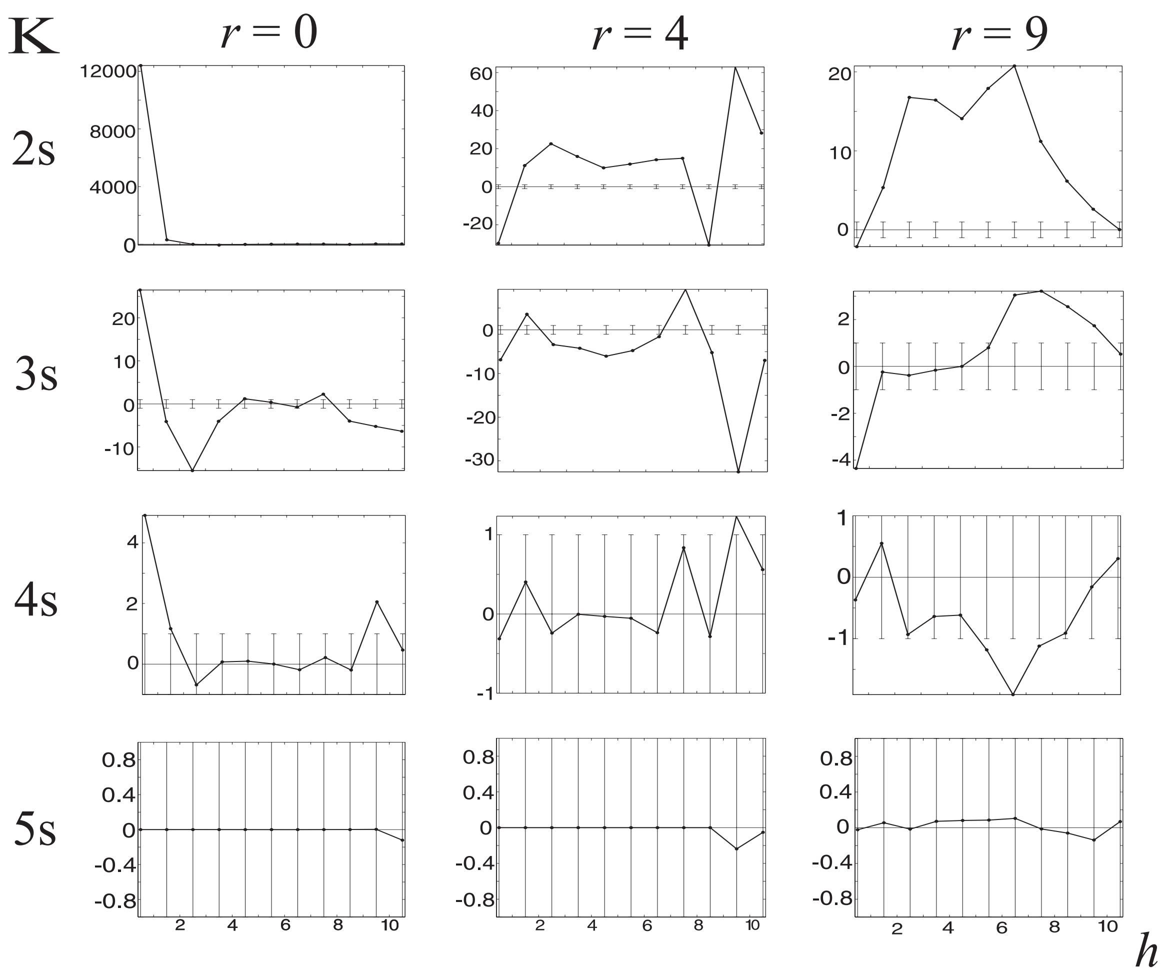

Figure 3-2: A montage showing the ratio of the fit error to the data error along the domain for various levels of split refinement.

Summary:			
K_c	$\Sigma = \sigma_{fit} / \sigma_{data}$	C_{data}	C_{fit}
5s	$\Sigma = 0.0107$	$19^2 = 361$	$2 \times 35^2 = 2450$

3.3.1.3 Approximating the Conditional Mean of Energy Deposited

Unlike the data set representing Φ, which has a very sharp peak at the beam entry point and then rapidly descends with increasing *r* and *h*, the data set representing μ_E has a relatively even slope over most of its domain. This behavior can be seen in Figure 3-3, where the black dots represent the actual data set and the surface represents the B-spline approximation of that data set for a BSPLND refinement level of 4. μ_E is around 1400 eV near the beam entry point and then slopes upward at an approximate rate of 160 eV/micron in both *r* and *h*. It reaches a maximum value of approximately 7000 eV at 12 microns in both *r* and *h* and then quickly descends to zero after that. The upward slope is a consequence of the primary electrons losing their kinetic energy as they travel over the domain. The electromagnetic force exerted by a charged particle is more influential the slower the charged particle moves in relation to its surrounding medium. The descent in μ_E, on the other hand, is a consequence of the beam's very low probability of reaching a target at such distances from its entry point. Not enough electrons made it to those distances to calculate a statistically significant first moment of the energy deposited stochastic. It is for these reasons that μ_E's data set too will be restricted to 1.2 times the p90 when analyzed.

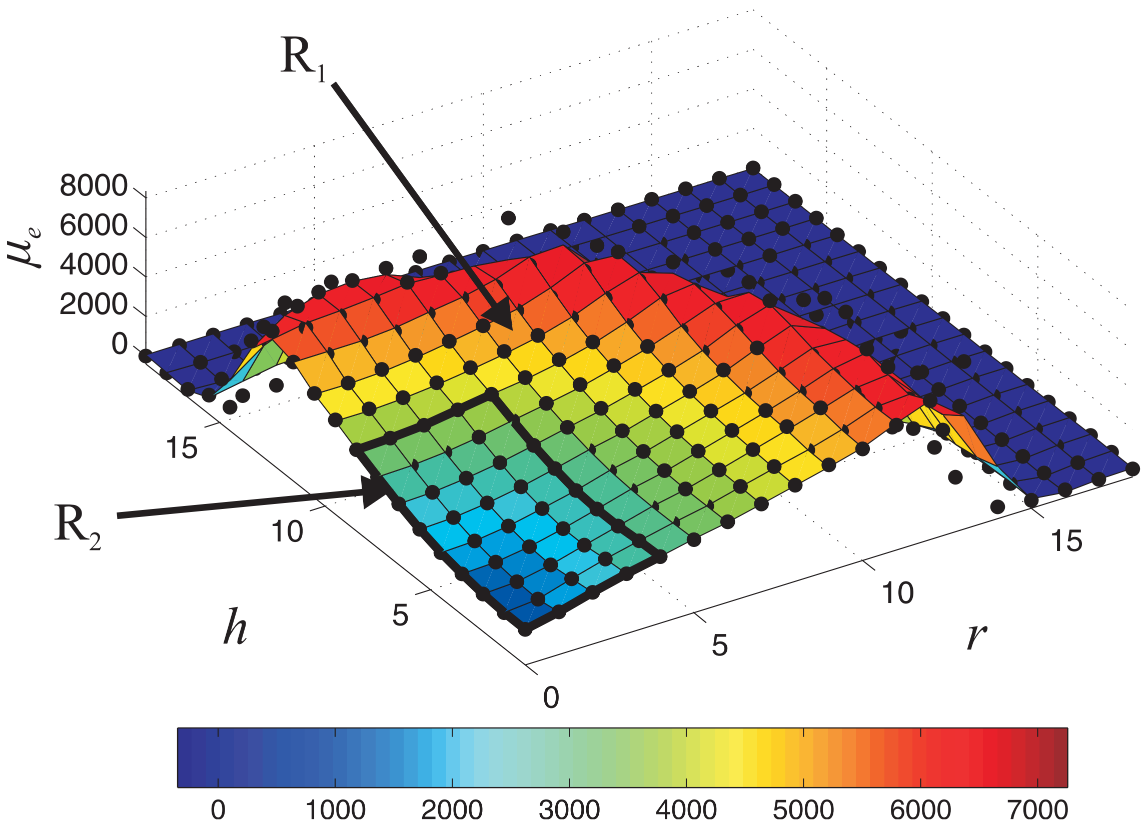

Figure 3-3: Data set and fit for μ_E. The surface represents the fit at a refinement level of K = 4 and the dots represent the data set.

The results from fitting μ_E's data set over the restricted domain are shown in Table 3-2. The regions, refinement levels, and columns are identical with those used for Φ in Table 3-1. Notice that splitting the domain into two regions still provides a significant improvement in the percent difference near the beam entry point, improving it at (0,0.5) by 81.9% when K = 2 and 98.6% when K = 4, yet the split *does not improve* Σ *enough over the restricted domain to be worthwhile*. Σ actually increases by 5.36% when the domain is split at K = 2, and only decreases by 0.0177% at K = 4. Another trend to notice from the table is the much smaller value for Σ in this restricted data set versus that for Φ's set at all levels of refinement. With no domain split, Σ is 99.7% smaller at K = 2 and 99.9% smaller at K = 4 versus Φ's fit. The source of this discrepancy is the relatively large error in μ_E near the outer fringes of this restricted set, which inflates σ_{data} and reduces the overall value of the Σ statistic as a refinement criterion. Regardless, Σ dips below $1/2$ with a minimal cost of 19^2 nodes at K = 4, without refinement, making $K_c = 4$ for the μ_e data set.

K	σ_{fit}	$\Sigma = \sigma_{fit}/\sigma_{data}$	C	$P(r,h)$				
				(0, 0.5)	(0, 9.5)	(9.5, 0)	(9.5, 9.5)	(4, 4.5)
1	1.588e+03	3.719	5^2	114.08	17.65	25.60	52.94	36.64
1 s	1.288e+03	3.017	$2 * 5^2$	11.88	2.832	17.91	50.06	1.515
2	773.27	1.811	7^2	14.30	1.451	3.76	29.15	23.88
2 s	814.32	1.908	$2 * 7^2$	2.805	18.68	3.127	29.24	1.358
3	387.73	0.9083	11^2	0.2620	1.531	3.879	7.072	3.648
3 s	380.39	0.8911	$2 * 11^2$	0.4331	9.905	3.033	7.492	2.517
4	120.26	0.2817	19^2	2.430	1.723	2.704	6.153	0.549
4 s	120.28	0.2818	$2 * 19^2$	0.03380	1.077	2.348	6.302	0.01682
5	24.646	0.0577	35^2	0.1096	0.1151	0.3000	1.382	0.0884
5 s	24.608	0.0576	$2 * 35^2$	0	0.1449	0.3378	1.363	0
6	0	0	67^2	0	0	0	0	0
6 s	0	0	$2 * 67^2$	0	0	0	0	0

Table 3-2: Results from fitting μ_e's data set over the restricted domain.

Figure 3-4 is a montage illustrating the nature of μ_E's error for a non-split domain, similar to the montage for Φ's error in Figure 3-3. Again, each window associates the level of refinement with its row and radial penetration with its column, and the individual windows plot the ratio of E_{fit}/E_{data} versus the forward penetration. E_{fit} lies for the most part within E_{data} over the restricted domain at the cut-off value of K_c =

4, but look how the window at K = 4 and $r = 0$ shows E_{fit} at near 15 times the value of E_{data} at $h = 0.5$. This results from the domain not being split into two regions and highlights the risk with using Σ as a criterion for choosing $\mathrm{K_c}$.

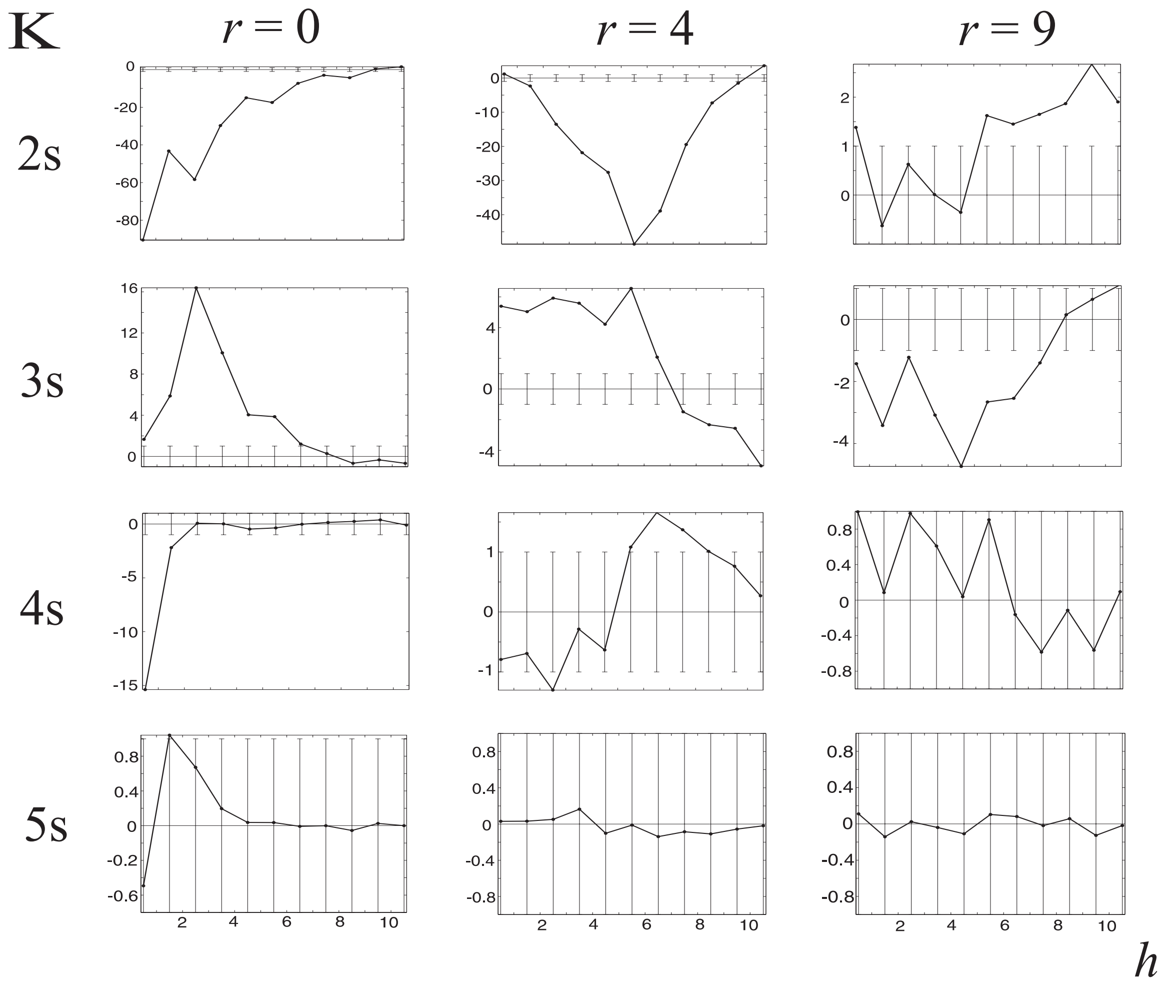

Figure 3-4: A montage showing the ratio of the fit error to the data error along the domain for various levels of split refinement.

Summary:			
$\mathrm{K_c}$	$\Sigma = \sigma_{fit} / \sigma_{data}$	$\mathrm{C_{data}}$	$\mathrm{C_{fit}}$
4	$\Sigma = 0.2817$	$19^2 = 361$	$19^2 = 361$

3.3.2 Approximating the Probability of Reaching a Target Sphere and the Conditional Mean of Energy Deposited for Energies 20keV to 80keV

3.3.2.1 Background

In Section 3.3.1 fits were created for Φ and μ_E over the spatial dimensions of *r* and *h* at a *single* primary beam energy. Doing this helped to develop the concepts of fitting using the BSPLND package. In this section fits are created over the *entire* domain of Φ and μ_E.

As introduced in Section 3.2, the number of sites over *r* and *h* is different for each energy level, e_i, in the data set. The domain for the data set cannot be properly defined until there is a common domain for the spatial dimensions at all energy levels. Section 3.2 suggested scaling the spatial coordinates of the data by the beam's p90 value. A common domain at each energy level could then be achieved by 1) identifying the domain along each scaled spatial dimension that is common to all data points and then 2) deleting the points that don't lie within that domain. Letting $\vec{r}_i^{\,\prime} = \vec{r}_i / p_{90_i}$ and $\vec{h}_i^{\,\prime} = \vec{h}_i / p_{90_i}$ be the scaled target site center vectors at the i^{th} energy level, this involves computing $[\min(\bigcap_{i=1}^{n} \vec{r}_i^{\,\prime}), \max(\bigcap_{i=1}^{n} \vec{r}_i^{\,\prime})]$ and $[\min(\bigcap_{i=1}^{n} \vec{h}_i^{\,\prime}), \max(\bigcap_{i=1}^{n} \vec{h}_i^{\,\prime})]$ over all *n* energy levels. For the Φ and μ_E data sets these domain intervals were found to be [0,2.034] and [0.0847,2.119], resulting in a final fit domain of $D_3 = [0,2.034] \times [0.0847,2.119] \times [20,80]$. A consequence of scaling each (*r*, *h*) grid, however, is that the data point densities will be different amongst the grids.

There are several possible directions to follow in creating the fit over the common domain D_3. Each deals differently with the problem of uneven data point densities. The most logical method is to approximate the data directly by BSPLND. The result would be a fit more influenced by the higher energy levels and difficult to analyze in MATLAB. Ameliorating these issues involves either *removing* data points from each energy level to match the grid at the lowest level or *adding* data points to the energy levels to match the grid at the highest level. Adding information is certainly better than deleting it, so each grid was forced to match the 80 keV grid through the process of interpolation.

Approximations of Φ and μ_E were then carried out on both the complete grid, D_3, as well as a two region version of D_3. Recalling Section 3.3.1, splitting the domain and fitting the data separately on each region can increase the accuracy of the fits if the "high feature" regions are properly chosen. The expense to this, however, is an increase in nodal storage proportional with the number of regions to be approximated over. The data set representing Φ has peaks at all energy levels close to the beam entry point, so two regions were created from D_3 for Φ (and μ_E as well): $R_1 = [0,0.4]\times[0.0847,0.8]\times[20,80]$ and $R_2 = D_3 - R_1$. Fits were then performed for Φ and μ_E over each region separately, requiring the storage of a B-spline node set for each region.

Finally, the performance of each fit was monitored using a third-party data set that was not used in the creation of the fits. This set includes energy levels of 22, 32, 42, 52, 62, and 72 keV.

3.3.2.2 Approximating the Probability of Reaching a Target Sphere

As in Section 3.3.1.2, the data representing Φ itself was not approximated over D_3. Rather, $\log_{10}(\Phi)$ was approximated. The performance of the fitting process was assessed against the test data set by 1) evaluating the fit at the domain tuples of the test data set, 2) exponentiating the resulting values by the power of 10, and 3) comparing the new values directly against the range of the test data set. Figure 3-5 shows slices of the $\log_{10}(\Phi)$ data set over a subset of the domain D_3, fenced off at the p90 value for all energies. The range of the data points in the figure is encoded using color, with the higher probabilities shown in red and the lower probability areas shown in dark blue. Observe how the high probability peak dominates the region within the p90 at 20 keV, reducing in area as the energy level increases. This is mainly a product of the scaling process, but it demonstrates that higher energy beams deposit a greater percentage of their energy via low energy diffusing electrons. The two fitting regions can easily be identified in Figure 3-6, which displays the absolute value of the difference between $\log_{10}(\Phi)$ and its fit at each of the grid points in D_3 on the same subset as Figure 3-5. The BSPLND refinement level in this case was 4. A bump resides in the figure at the interface of the two regions because each region was fitted independent of the other and share no points on their respective domains.

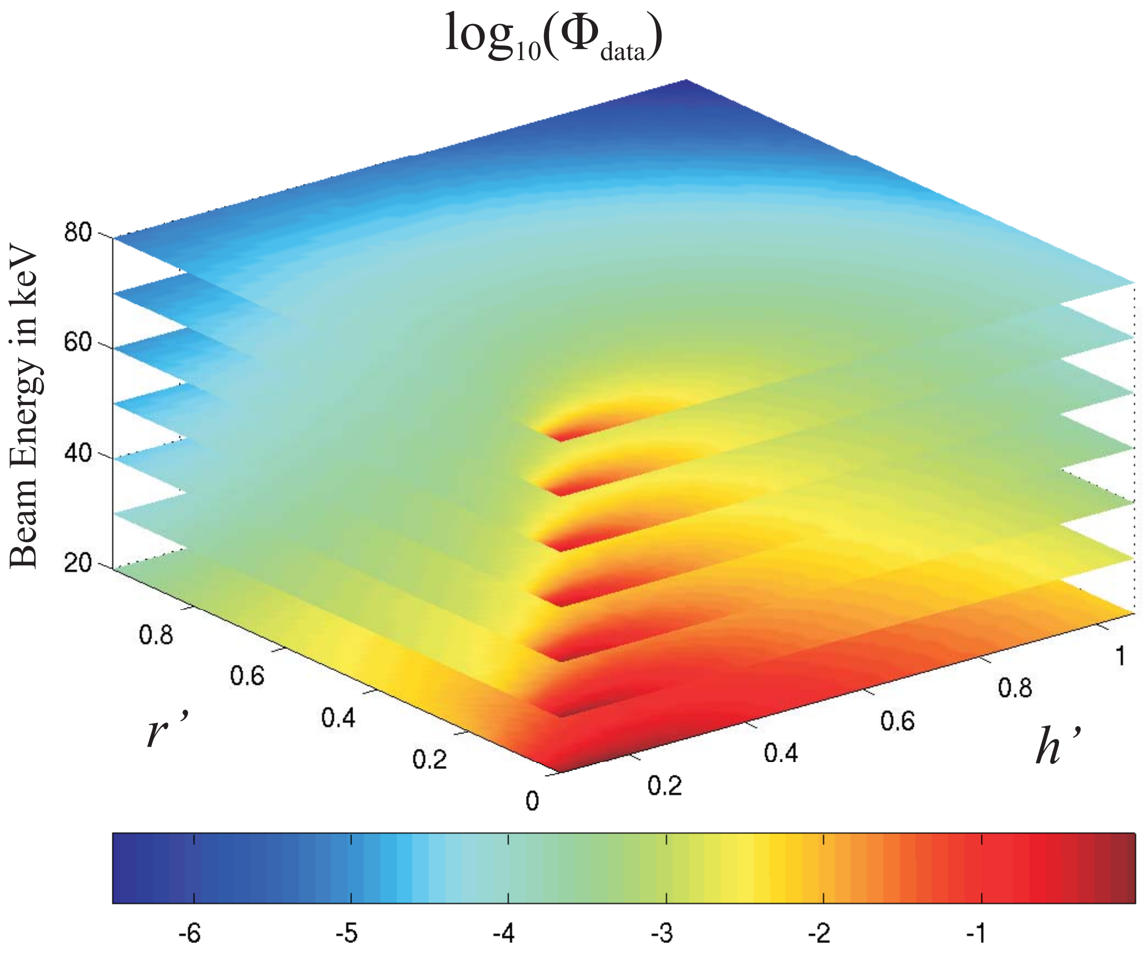

Figure 3-5: Four dimensional smooth graph of $\log_{10}(\Phi_{data})$. The beam energy is sampled from the data at every 10 keV.

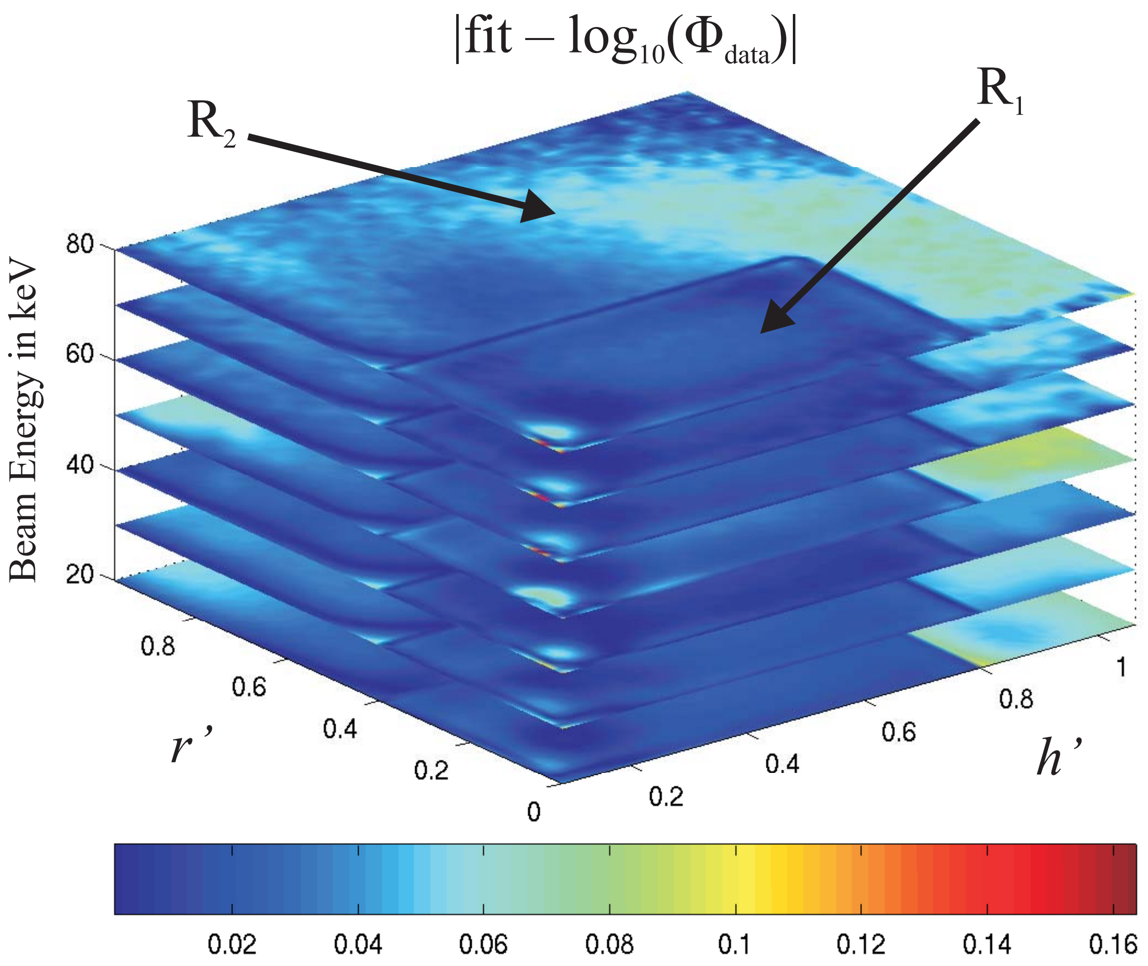

Figure 3-6: Four dimensional smooth graph showing the absolute value of the error between the fit of $\log_{10}\left(\Phi_{data}\right)$ and $\log_{10}\left(\Phi_{data}\right)$ itself. The BSPLND refinement level is 4 and the beam energy is sampled from the data at every 10 keV.

Assessment of the fits for refinement levels 1 through 6 can be seen in Table 3-3. Again, as in Section 3.3.1.2, the refinement level is appended by an "s" if the domain is split, and the cost of the fit in terms of the required nodal storage is given by the parameter C. As well, the RMS of the fit error against the original data set is denoted by σ_{fit}, and the RMS of the data's standard deviation is denoted by σ_{data}, yielding Σ_{fit} as the ratio of the two. In order to assess the fits against the test data set, two additional parameters have been added to the mix. These include σ_{test}, the RMS of the fit error against the test data set, and Σ_{test}, the ratio of σ_{test} to σ_{data} over the test data points. Just as before, σ_{fit}, σ_{test}, and σ_{data} were calculated at up to 1.2 times the p90 along *r'* and *h'*, giving the restricted domain of $[0,1.2]\times[0.0847,1.2]\times[20,80]$ in their analysis.

K	σ_{fit}	σ_{test}	$\Sigma_{fit} = \sigma_{fit}/\sigma_{data}$	$\Sigma_{test} = \sigma_{test}/\sigma_{data}$	C
1	0.0151	0.0166	397.55	409.29	5^3
1 s	0.0126	0.0141	331.93	347.72	$2 * 5^3$
2	0.0152	0.0167	399.32	412.65	7^3
2 s	0.0103	0.0118	269.84	289.64	$2 * 7^3$
3	0.0144	0.0159	378.28	392.55	11^3
3 s	0.0050	0.0060	131.85	149.01	$2 * 11^3$
4	0.0105	0.0118	277.04	290.06	19^3
4 s	0.0024	0.0031	62.90	75.98	$2 * 19^3$
5	0.0064	0.0094	168.73	232.75	35^3
5 s	0.0013	0.0024	34.67	58.28	$2 * 35^3$
6	0.0022	0.0094	56.60	230.62	67^3
6 s	0.0011	0.0024	28.57	58.10	$2 * 67^3$

Table 3-3: Result of fitting $\log_{10}(\Phi_{data})$ for refinement levels 1 through 6s.

From the table, notice how Σ_{test} is greater than Σ_{fit} at every level of refinement. For example, at a refinement level of K = 5s, Σ_{fit} = 34.67, while Σ_{test} is 68% greater at 58.28. This occurs because BSPLND created the fit on only thirteen energy levels, and the fit is biased slightly toward the data points at each energy level. Also notice from the table that the cut-off criterion K_c, introduced in Section 3.3.1.2, cannot be used here because the fit no longer improves appreciably when K>5. This suggests using a revised criterion in this case, namely choosing K such that the improvement in Σ_{test} is less than 1% when increasing from a given level to the next This occurs at a refinement level of 5s. The RMS statistics are larger for this data set versus the single energy level set used in Section 3.3.1.2 because the beam energy was added to the domain with only thirteen energy levels.

Figure 3-7 demonstrates the perils of using the Σ statistic as a measure of fit accuracy. The sub-figures (a), (b), and (c) show the percent difference, P(*r'*,*h'*,*e*), evaluated at 22 keV, 42 keV, and 62 keV, respectively, for five selected values of *r'* and *h'* and refinement levels ranging from K = 2 to K = 6s. The percent difference is encoded by the shade of gray for a given (K, *r'*,*h'*) block, with lighter shades representing higher percent differences. In the 22 keV case, the percent difference drops most dramatically at all selected values of *r'* and *h'* , with the exception of (0.25, 0.25), when K moves from 3s to 4. But Σ_{test} moves from 149.01 to 290.06 in this case. A similar story results for the other energies as well. The reasoning is simple: a relatively high spike in Φ near the beam entry point creates a large error in that region if the domain is not split, swamping the Σ_{test} statistic. The statistic is not scaled in relation with the magnitude of its data points. The region of most interest at each energy level, however, *is* the high probability spike, so the Σ_{test} statistic is a reasonable method of assessing the fits.

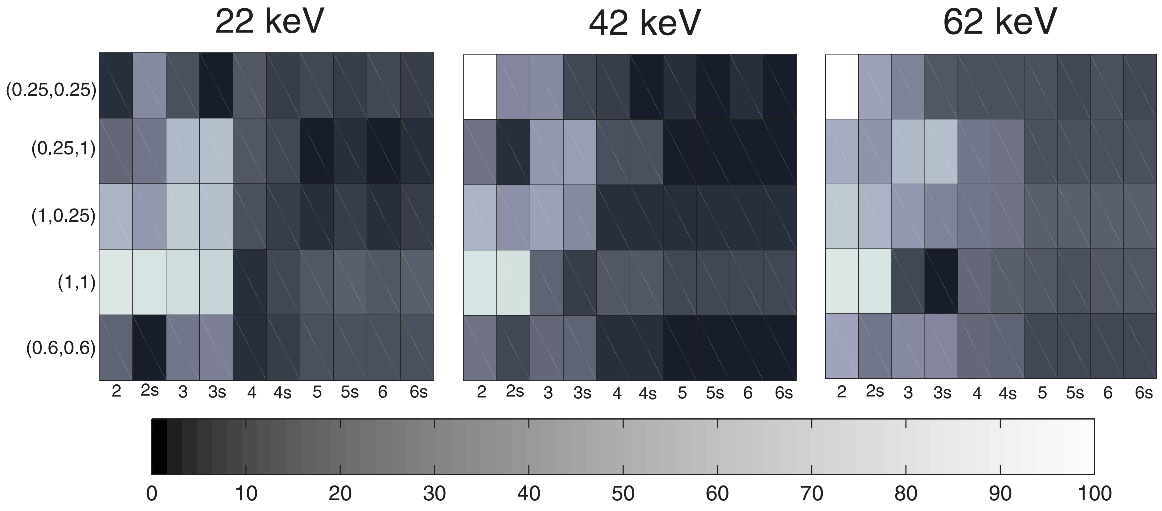

Figure 3-7: The percent difference, P(*r'*,*h'*,*e*), evaluated at 22 keV, 42 keV, and 62 keV, respectively, for five selected values of *r'* and *h'* and refinement levels ranging from K = 2 to K = 6s.

Several actions could be undertaken in order to improve the fit further, albeit with strings attached. One technique that has proven useful is the restriction of the domain, which improves the fit by reducing the level of detail that BSPLND must deal with. In the present case, it would be appropriate to reduce the domain to cover the region encompassed by the p90 at all energy levels. Of course, this reduction has the price of limiting the scope of the representation. Another technique is to adjust the density of the data grid to better capture the high feature regions in the fit, such as areas near the beam entry point in the Φ data set. The BSPLND package allows for non-uniform scattered data sets such as these. Data density adjustment could also be used over entire energy levels to give them equal footing in the fit after scaling. Currently the lowest energy level, 20 keV, has 13^2 original data points and the highest, 80 keV, has 144^2, yielding an increasing density with energy level after scaling. Additional overlapping sites could be added to match the densities. Adding sites, however, means rerunning simulations in order to build up the necessary histograms for the stochastics. This is a major obstacle to adjusting the data density and implementing these last two techniques.

Summary:			
K_c	$\Sigma_{test} = \sigma_{test} / \sigma_{data}$	C_{data}	C_{fit}
5 s	$\Sigma = 58.28$	$13^2 + 19^2 + \cdots + 144^2 = 85{,}059$	$2 \times 35^3 = 85{,}750$

3.3.2.3 Approximating the Conditional Mean of Energy Deposited

Figure 3-8 shows a portion of the data set representing μ_E after interpolation and scaling. Slices are shown for the 20, 40, 60, and 80 keV energy levels and at *r*'=1.2 and *h*' = 1.2. The range is color coded to show higher values in red and lower values in blue. The smooth gradually increasing characteristic of μ_E at each energy level, reminiscent of Figure 3-3 for the 25 keV case, can be seen on the slices. At 20 keV μ_E rises relatively rapidly to a high of about 7000 eV near the 20 keV p90. Higher beam energies show μ_E rising much more gradually in comparison. Notice how jumpy the data is near the outer fringes of each energy level.

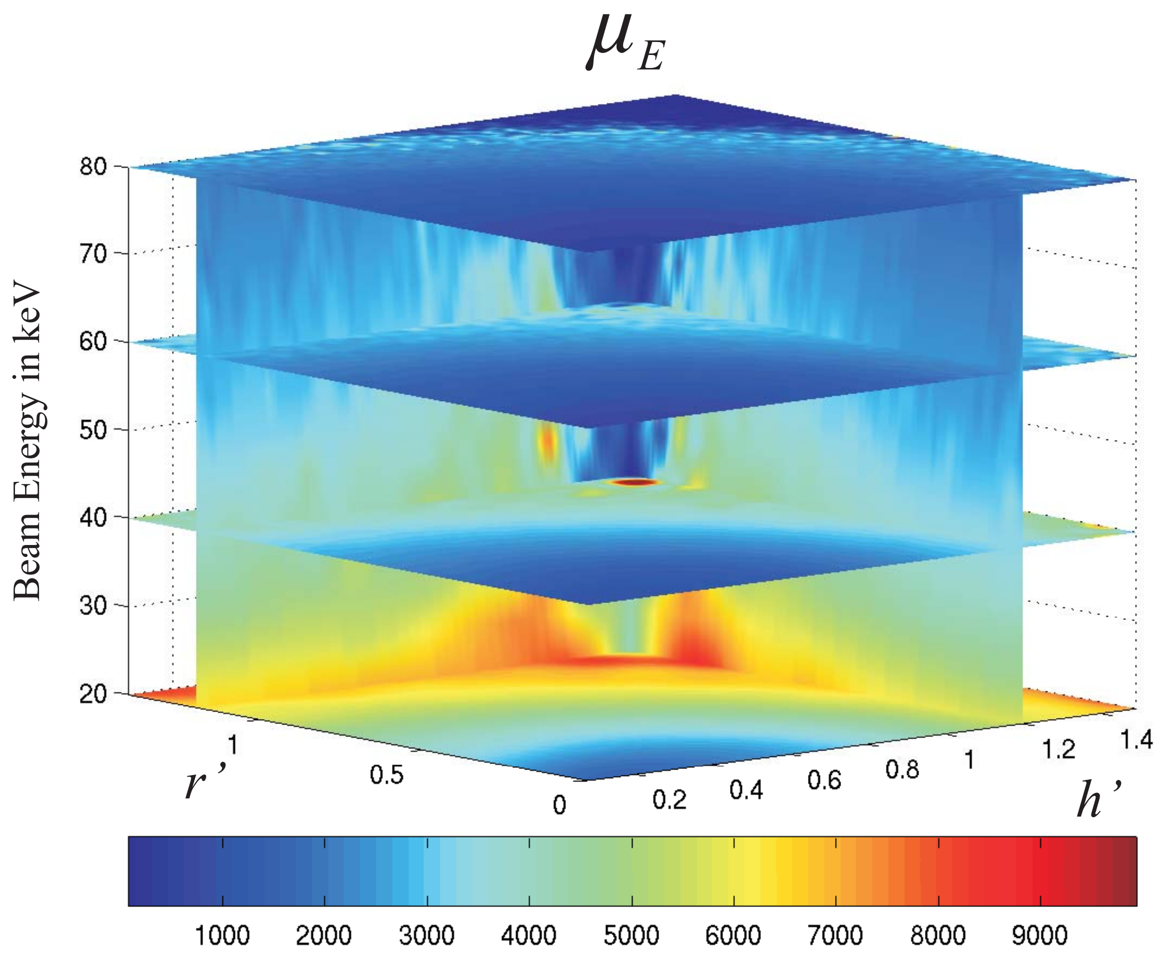

Figure 3-8: Four dimensional smooth graph showing the data set for μ_E for energy levels 20, 40, 60, and 80 keV.

The process of preparing the data, fitting the data, and assessing the fits for μ_E was identical with Φ, aside from fitting the logarithm of the data. The split regions were identical, as was the restricted region of error assessment on the domain $[0,1.2]\times[0.0847,1.2]\times[20,80]$. Fitting was performed over the entire domain of the common scaled, interpolated data set at six levels of refinement both with split regions and without. Table 3-4 gives the results.

K	σ_{fit}	σ_{test}	$\Sigma_{fit} = \sigma_{fit} / \sigma_{data}$	$\Sigma_{test} = \sigma_{test} / \sigma_{data}$	C
1	876.76	906.42	1.378	1.547	5^3
1 s	806.46	826.61	1.268	1.4111	$2 * 5^3$
2	473.09	479.19	0.7436	0.8180	7^3
2 s	506.16	516.40	0.7957	0.8816	$2 * 7^3$
3	309.63	298.54	0.4867	0.5097	11^3
3 s	306.30	294.43	0.4815	0.5026	$2 * 11^3$
4	224.03	204.63	0.3522	0.3493	19^3
4 s	223.30	203.82	0.3510	0.3479	$2 * 19^3$
5	191.47	190.84	0.3010	0.3258	35^3
5 s	191.36	190.74	0.3008	0.3256	$2 * 35^3$
6	151.07	190.85	0.2375	0.3258	67^3
6 s	151.00	190.74	0.2374	0.3256	$2 * 67^3$

Table 3-4: Result of fitting μ_E for refinement levels 1 through 6s.

Notice how low Σ_{test} is for the μ_E data set compared with Σ_{test} for the Φ data set in Section 3.3.2.2. This was expected here after it occurred for the 25 keV case in Section 3.3.1.3. At K = 4, for instance, Σ_{test} is 829% larger for Φ than for μ_E over the restricted assessment range. As noted in Section 3.3.1.3 this discrepancy is due to larger data errors past the p90 relative to the data set's magnitude. Observe as well from the table how little improvement is made to the fit by splitting the domain. Splitting the domain at K = 4 only gives a 0.396% improvement in Σ_{test}, while increasing K to 5 gives a much larger 6.74% improvement. As mentioned earlier this lack of improvement in the

Σ_{test} statistic is due in large part to the swamping effect of the high data error near and past the p90. The error near the beam entry point is being drowned out.

The error improvement is actually quite modest within R_1, as can be seen in the top row of shaded blocks in each plot of Figure 3-9. As in Figure 3-7, a block's shade is indicative of the percent difference, P, at the associated point (row) and refinement level (column) for that energy level (plot). The top row corresponds with (0.25,0.25), which is the only point from each plot residing in R_1. P improves at (0.25,0.25) for every region split, with the exception of 3 to 3s at 22 keV and 2 to 2s at 42 keV.

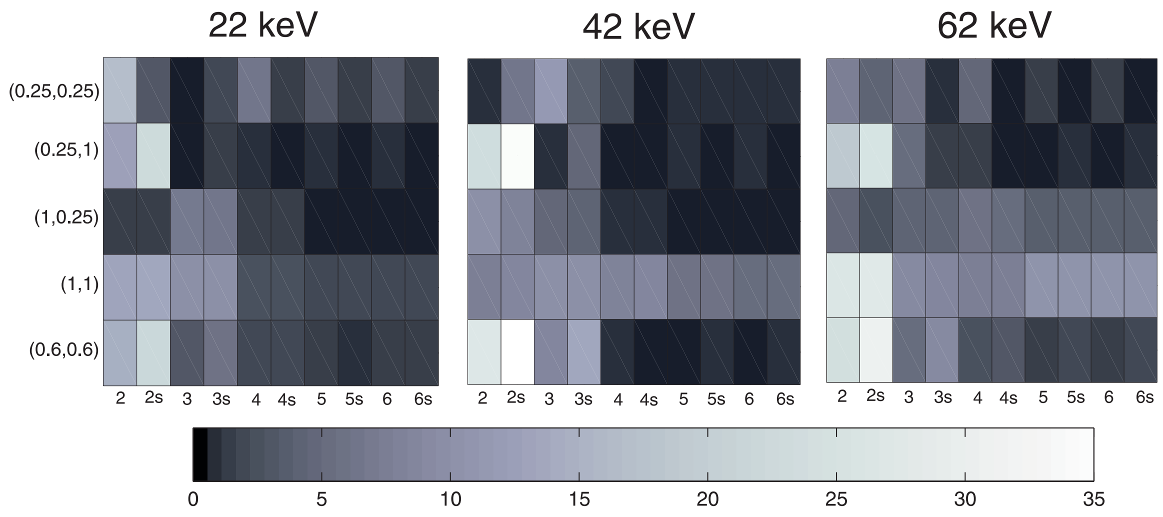

Figure 3-9: The percent difference, P(*r'*,*h'*,*e*), evaluated at 22 keV, 42 keV, and 62 keV, respectively, for five select values of *r'* and *h'* and refinement levels ranging from K = 2 to K = 6s.

Additionally, notice that *max*(P) is only 35% in these plots, versus 100% in Φ's plots. This shows the difficulty with fitting the high feature region of Φ near the beam entry point. Also notice the relatively large decrease in P at all energy levels and most points in the figure when K moves from 3s to 4. This evidence, in addition with the fact that Σ_{test} first dipped below 0.5 at a refinement level of 4 with no domain split, supports giving K_c the value of 4.

Summary:			
K_c	$\Sigma_{test} = \sigma_{test} / \sigma_{data}$	C_{data}	C_{fit}
4	$\Sigma = 58.28$	$13^2 + 19^2 + \cdots + 144^2 = 85{,}059$	$19^3 = 6{,}859$

3.3.3 Approximating Distributions of the Energy Deposited Per Unit Track Length of the Primary Particle for Energies 20keV to 80keV

Recall that the distribution of the stochastic $\Lambda = E / L_p$ at the site centers is denoted by f_Λ. The domain of the data set representing f_Λ is identical with the domain representing Φ and μ_e, except an additional variable has been added, λ. This variable represents the bin centers of the probability density distribution $f_\Lambda(\lambda | e_s, r_s', h_s')$ at any given beam energy, e_s, and site center (r_s', h_s'). For convenience, this one-dimensional distribution function will be written as $f_\Lambda(\lambda)$ for a given site and beam energy. Introduced in Section 2.3.2, the area under $f_\Lambda(\lambda)$ over the region $[\lambda_a, \lambda_b]$ yields the probability that a track reaching the site will deposit an energy per unit track length of the primary electron between λ_a and λ_b. The nice part about adding λ to the data set domain is that no additional scaling and interpolation is required to make the set complete. Adding the beam energy to the domain in Section 3.3.2.1 made that work necessary, since the domain of the site centers varied with the beam energy. Adding λ to the domain does, however, add to the running time and memory requirements of the algorithm according to Equations (27) and (28), respectively, and to the space requirements of the control point nodes according to Equation (29).

In the simulations, Λ was scored using the logarithmic binning system described in Section 2.3.2.1, with the width of each bin multiplied by the scaling factor 0.001. This factor was necessary to adjust the bins to the magnitude of the stochastic being scored. For Λ, the distributions typically begin at around 0.03 keV/micron (bin 20) near the beam entry point, reach a peak around 1 keV/micron (bin 40), and then tail off to a maximum value of about 100 keV/micron (bin 67). The original binning structure introduced in Section 2.3.2.1, however, was devised for the energy deposited stochastic, *E*. It has a distribution that typically begins around 9 eV (original bin 13) near the beam entry point, reaches a maximum value at around 1152 eV (original bin 41), and then ends near 18,432 eV (original bin 57). The factor of 0.001 for λ takes care of this transformation from ε (the independent axis for the stochastic *E*). With a total of 88 bins, the binning structure used for Λ's histogram in the simulations begins with 0.001125 keV/micron at bin 1 and then ends with 3932 keV/micron at bin 88. This yields a fit domain of $D_4 = [0, 2.034]\times[0.0847, 2.119]\times[20, 80]\times[0.001125, 3932]$ if all bins are used.

Restrictions on the data set representing f_Λ, however, must be made in order to produce an acceptable fit. Low event frequencies at large r' and h' combined with low histogram counts at high λ produce malformed distributions at these ranges. Restricting the domain yields a cleaner data set and correspondingly better fits. This has the consequence, however, of reducing the overall size and applicability of the fit. Limiting both r' and h' to 1.4 times the p90 as well as λ to 5.632 keV/micron (bin 50) produces a nice compromise between these two factors, giving a final restricted fit domain of $D_{4r} = [0, 1.4]\times[0.0847, 1.4]\times[20, 80]\times[0.001125, 5.6320]$.

Figure 3-10 and Figure 3-11 each illustrate a slice of the f_Λ data set at constant penetration and beam energy. In Figure 3-10 the target's penetration is set to 0.2143 microns with a beam energy of 22 keV. In Figure 3-11 the target is at a penetration of 0.0928 microns and the beam energy is 62 keV. The white dots in the figures represent the test data set and the surfaces represent the slices taken from the representations developed for $f_\Lambda(\lambda, r')$ evaluated at a refinement level of 5s. The axes are identical in each plot to highlight the differing behavior of $f_\Lambda(\lambda, r')$ with beam energy. A few trends emerge from the figures. Notice how the peaked region in each plot flattens as r' increases. Also notice how much more peaked the 62 keV distributions are than the 22 keV distributions for each value of r'. In the figures, for a fixed value of r', areas under the curve $f_\Lambda(\lambda)$ over an interval of λ represent the probability of depositing Λ within that interval. The higher peaks of $f_\Lambda(\lambda, r')$ lying over lower values of λ and r', therefore, indicate a greater chance of depositing the stochastic Λ over these regions once the electron track reaches the target.

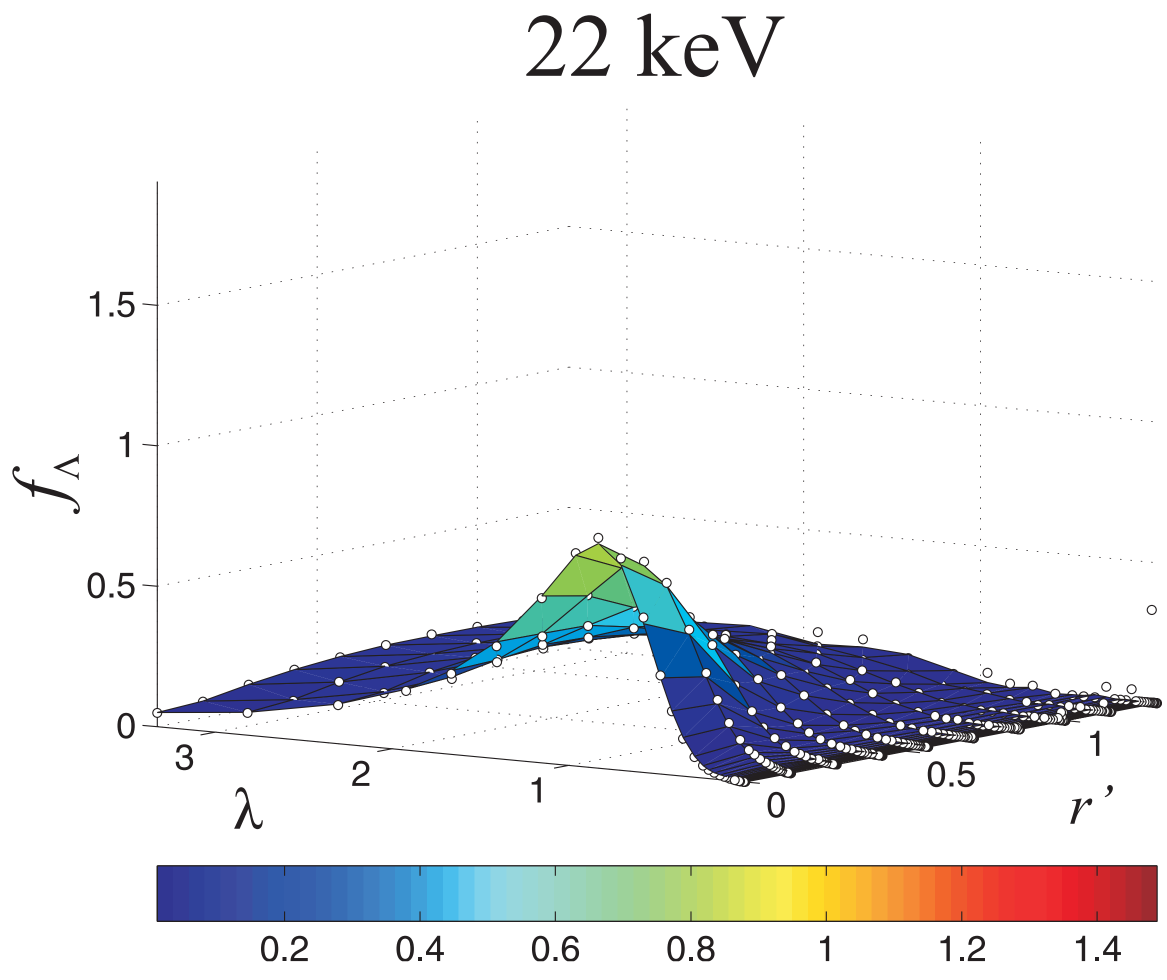

Figure 3-10: Data set (white dots) and fit (surface) representing f_Λ for a beam energy of 22 keV with h' = 0.2143.

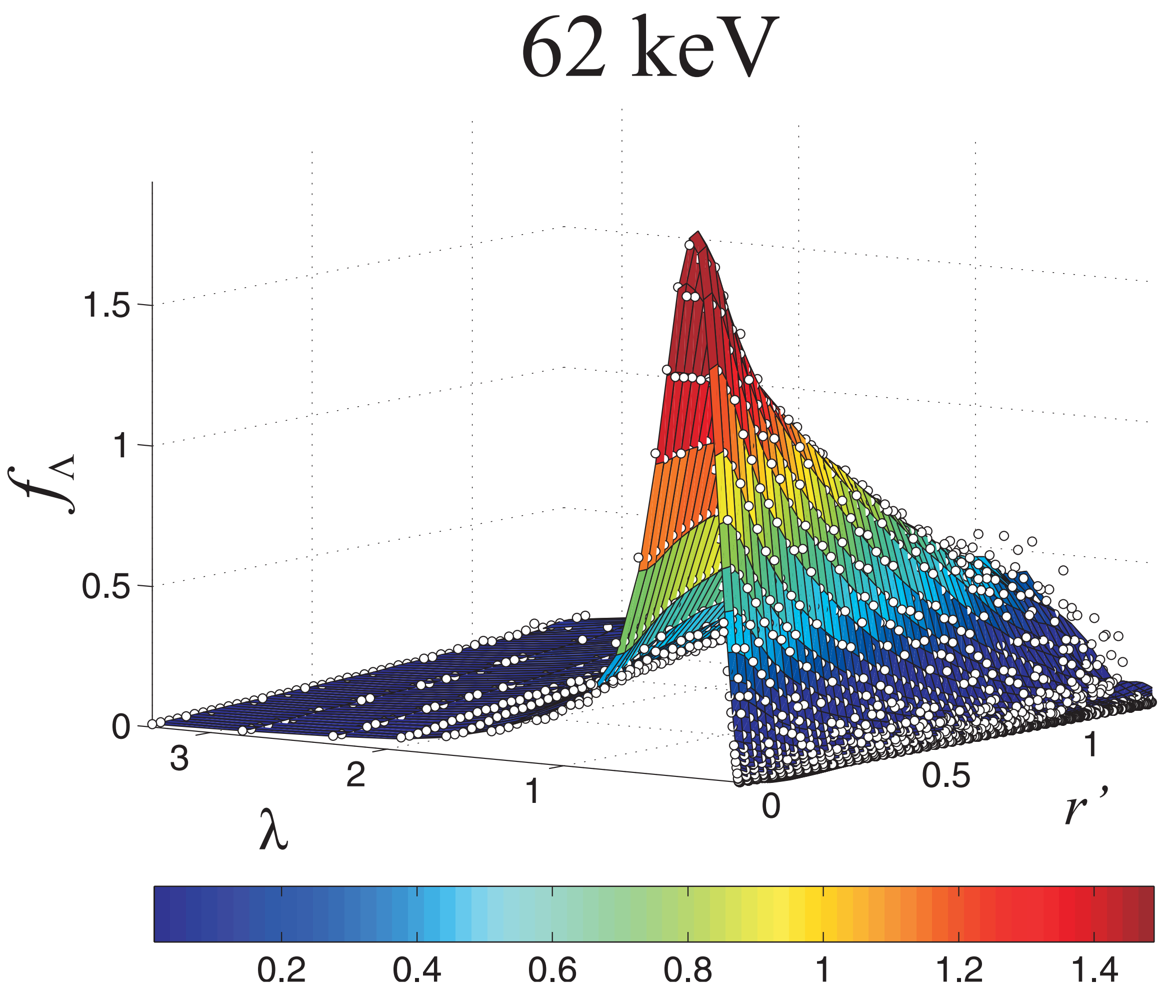

Figure 3-11 Data set (white dots) and fit (surface) representing f_Λ for a beam energy of 62 keV with *h'* = 0.0928. For an easy comparison, the axis is identical with the axis in Figure 3-10.

The consistent nature of the peaks along λ suggests an optimization. As with Φ and μ_E, splitting the domain to capture these peaks within their own region increases the accuracy of the overall fit. This time, however, r' and h' is left alone in the creation of the two regions R_1 and R_2, and only λ is split. Referring to Figure 3-10, an effective choice for this split at both energies displayed occurs approximately at $\lambda = 2$. This split yields the two intervals $0 < \lambda_{R1} \le 2$ and $2 < \lambda_{R2} \le \max(\lambda)$ over λ and creates the two regions $R_1 = [0, 1.4] \times [0.0847, 1.4] \times [20, 80] \times [0.001125, 2]$ and $R_2 = D_{4r} - R_1$.

Table 3-5 gives the results of fitting the interpolated common data grid for all levels of refinement from 1 to 6s. The fit statistics in the table are identical with those in Table 3-3 and Table 3-4. Notice how much smaller in magnitude the Σ_{fit} and Σ_{test} statistics are in this table than those for Φ in Table 3-3. They are consistently 10% of their corresponding values in Table 3-3. This behavior was not expected, since the current case has an additional dimension and relatively eccentric peaks in the distributions at the highest beam energies. Another oddity is how much larger Σ_{test} is than Σ_{fit} for all levels of refinement in this case versus the previous fits. Σ_{test} *should* be larger since the test data set was not used to make the approximation. But 37% larger at K = 5s, for example, is not consistent with BSPLND's previous performances. Regardless, improvements in the RMS statistics *is* consistent with increasing refinement level and a split domain. From the table, a logical stopping point is at K = 5s, when the Σ_{test} statistic actually reaches a minimum. Figure 3-10 and Figure 3-11 each visually attest to the accuracy of the fit at this level of refinement. The resulting cost in terms of the required nodal storage is now to the fourth power.

K	σ_{fit}	σ_{test}	$\Sigma_{fit} = \sigma_{fit} / \sigma_{data}$	$\Sigma_{test} = \sigma_{test} / \sigma_{data}$	C
1	0.2867	0.2532	10.690	30.506	5^4
1 s	0.2507	0.2130	9.3471	25.658	$2 * 5^4$
2	0.2670	0.2318	9.9546	27.922	7^4
2 s	0.1869	0.1434	6.9699	17.281	$2 * 7^4$
3	0.2186	0.1776	8.1509	21.403	11^4
3 s	0.1227	0.0667	4.5746	8.0313	$2 * 11^4$
4	0.1496	0.1069	5.5776	12.877	19^4
4 s	0.1046	0.0437	3.8989	5.2689	$2 * 19^4$
5	0.1085	0.0713	4.0454	8.5916	35^4
5 s	0.1012	0.0429	3.7723	5.1716	$2 * 35^4$
6	0.0974	0.0709	3.6299	8.5381	67^4
6 s	0.0911	0.0430	3.3973	5.1819	$2 * 67^4$

Table 3-5: Results of fitting the interpolated common data grid for all levels of refinement from 1 to 6s.

Figure 3-12 makes it easy to compare the data and fit of individual distributions over the entire domain of λ for the single refinement level of K = 5s. Here the focus of each subplot is the distribution itself, with the beam energy varying over the subplot columns and the sample locations varying over the rows. In each subplot the solid line represents the data, the dashed line represents the fit, and the horizontal error bars represent the bins and their corresponding uncertainty. All subplots share the same axes in λ and f_Λ in order to help illustrate the behavior of f_Λ over e, r', and h'. As with distribution slices taken from Figure 3-10 and Figure 3-11 at constant values of r', notice how $f_\Lambda(\lambda)$ peaks higher at the larger beam energy. Also notice how $f_\Lambda(\lambda)$ flattens as the distance from the beam entry point increases. Both of these trends are indicative of the increasing randomness of electrons as they lose energy. Another thing to notice is the choppiness of the data in the subplot associated with (r', h') = (1,1) and e = 62 keV. This indicates how the statistical quality of the data diminishes with increasing distance from the beam entry point. Finally, it should be noted that the area under $f_\Lambda(\lambda)$ over $0 < \lambda < \infty$ for each of these subplots should in theory be the same. As was covered in Section 2.3.2.1, all true probability distribution functions enclose an area of 1. Therefore, the flatter curves in the figure keep larger values past the cutoff value of λ used in the domain of the fits.

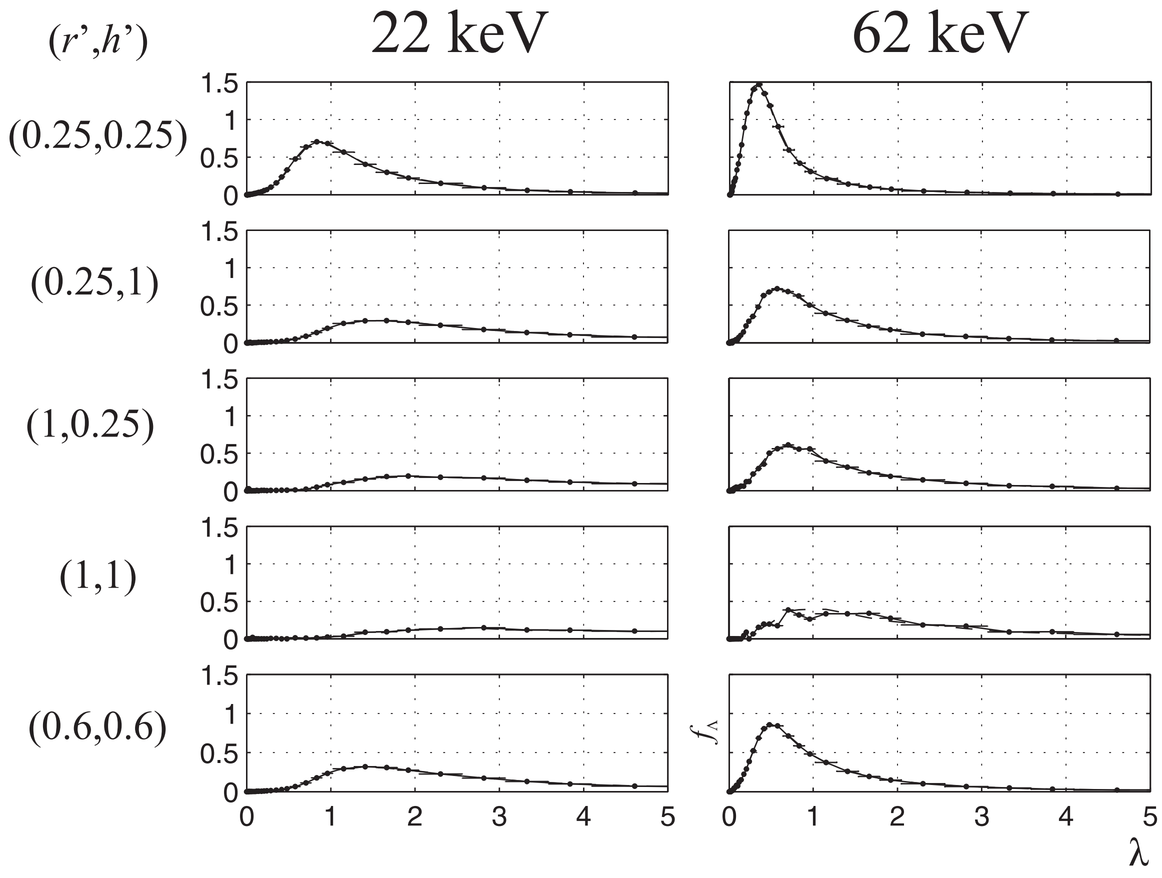

Figure 3-12: Montage of f_Λ vs. λ over *e*, r', and h'. The position of each axis in the montage represents its energy (by column) and its location (by row).

Summary:			
K_c	$\Sigma_{test} = \sigma_{test} / \sigma_{data}$	C_{data}	C_{fit}
5s	$\Sigma = 5.172$	$50 \times (1.4\vec{p}_{90})^2 = 50 \times (8.26^2 + \cdots + 78.68^2) = 1.357e6$	$2 \times 35^4 = 3.001e6$

3.4 Approximating Distributions of the Energy Deposited Per Unit Track Length of the Primary Particle using Lognormal Functions

As data for f_Λ was collected, it appeared that the distributions were lognormal in nature. Figure 2-11 from Section 2.4.5 shows this behavior for a 25 keV electron microbeam. The distributions in the data rise up quickly to a peak and then exponentially taper off after that. It is uncertain from physical arguments, however, why f_Λ would be lognormal in nature. Several other distribution functions follow a similar trend, such as the Wiebull and fatigue life distributions, and they too could have been used to possibly model f_Λ. Nevertheless, the *process* of fitting f_Λ is emphasized in this section rather than the accuracy of the model used, and the lognormal distribution seemed to fit the data particularly well early in the course of research. Unfortunately, applying the goodness of fit testing process introduced in Section 2.3.2.7 to the resulting lognormal distribution fits gives disappointing results. This testing process will be discussed in Section 3.4.3. First, a comparison is made of the various point estimate techniques in Section 3.4.1, and then an analysis of the trends in μ and σ for the lognormal parameters is made in Section 3.4.2.

3.4.1 SEP Lognormal Parameter Estimation

Section 2.3.2.6 introduced the notion of estimating the parameters for a probability distribution given a set of data that is assumed to be governed by that distribution. A total of four methods were outlined: the method of nonlinear least squares (MNLS), the method of maximum likelihood (MML), the method of moments (MM), and probability plotting. In this section the first three of these methods will be applied to finding the parameters of a lognormal function at each of the target sites for a 20 keV electron microbeam.

Figure 3-13 shows the lognormal curves at the nine targets within [0.25, 0.6, 1.0]x[0.25, 0.6, 1.0] resulting from parameter estimation by the various methods. The curves associated with MM, MML, and MNLS are in magenta, blue, and red, respectively, while the data points are represented by the black dots and error bars. The axes are scaled differently to fully accommodate the respective curves. Notice how the curve using MML falls short of the peak at the data point closest to the beam entry point and then catches up with the curve using MNLS as r' and h' increases. The MML curve peak is consistently below the MNLS curve peak. In addition, the MM curve is skewed to the left compared with the other curves. Overall, the estimated distributions are close to the data points, but they fall well short in accuracy at this beam energy when compared with the corresponding curves extracted from the B-spline approximation developed in Section 3.3.3. Figure 3-14 shows this discrepancy at the site (0.25, 0.25).

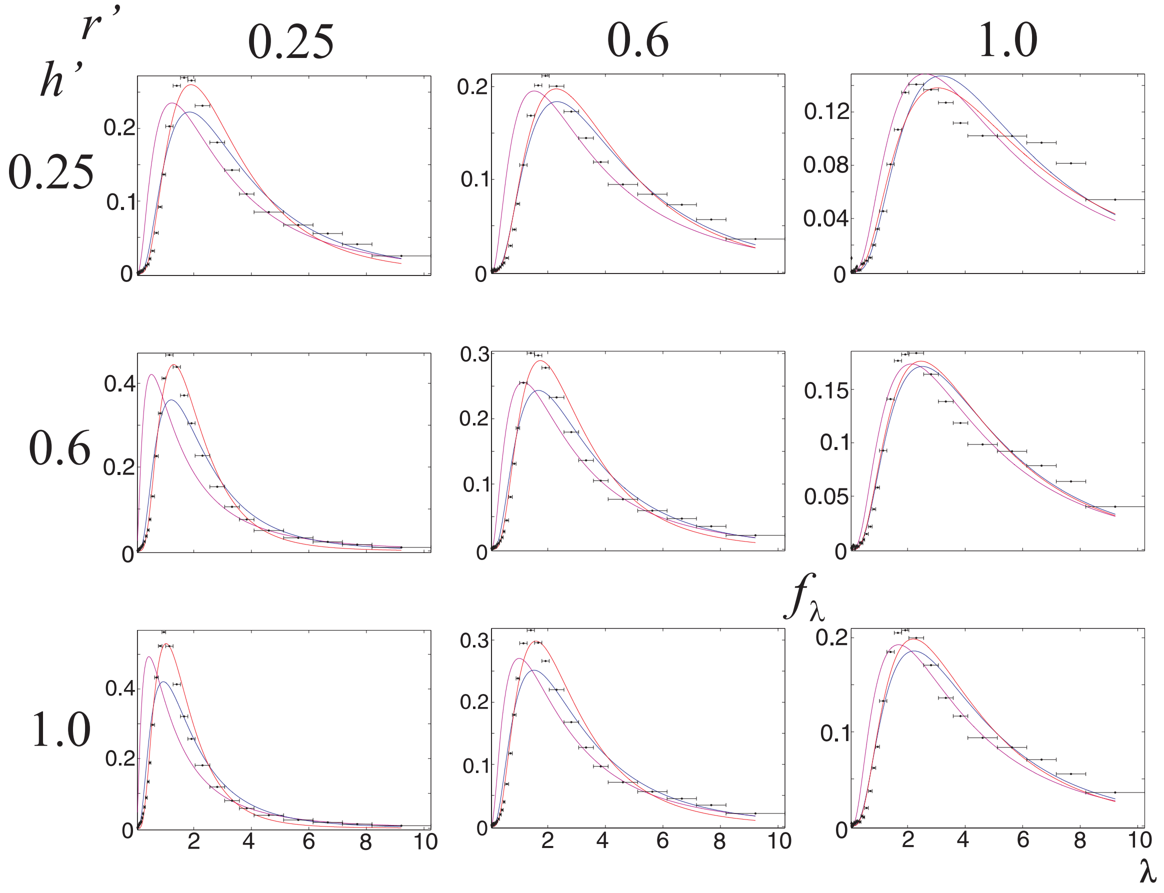

Figure 3-13: The lognormal curves at the nine targets within [0.25, 0.6, 1.0]x[0.25, 0.6, 1.0] resulting from parameter estimation by MM (magenta), MML (blue), and MNLS (red).

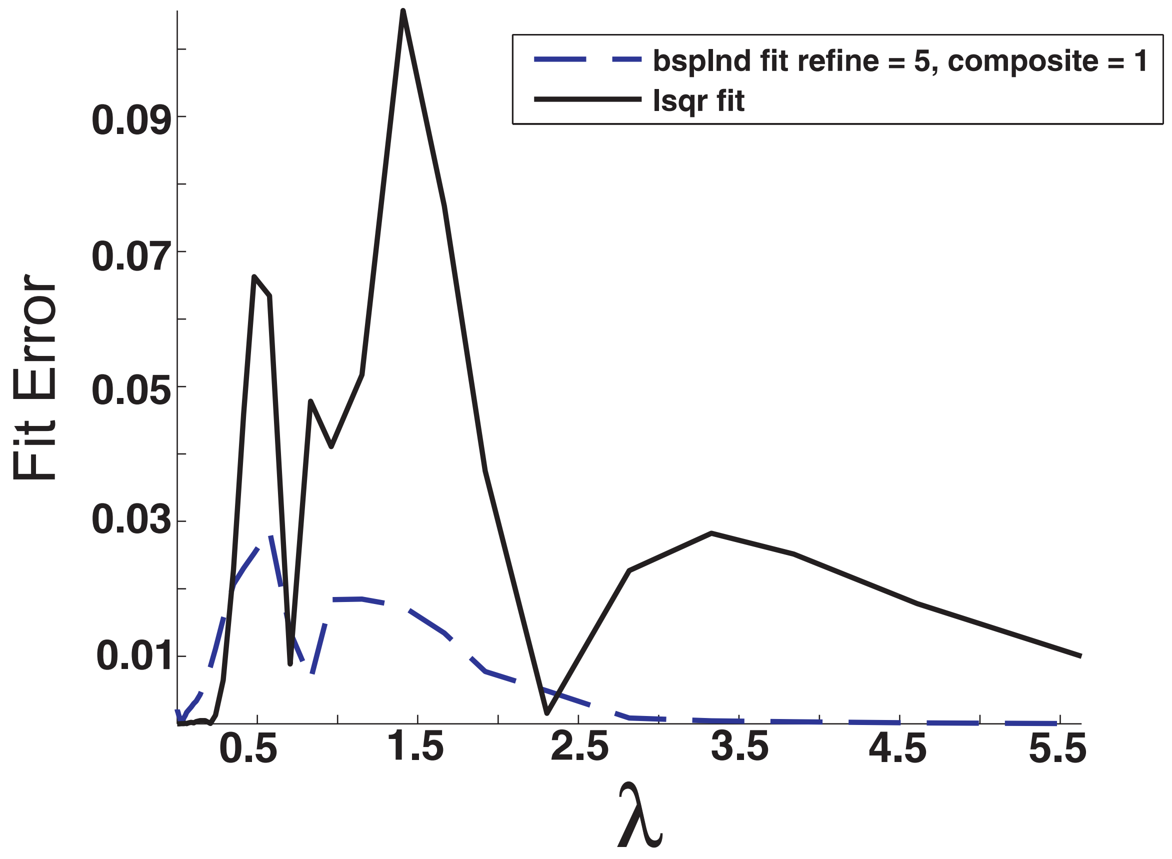

Figure 3-14: Error between the data for $f_{\Lambda}(\lambda)$ and its fit using the f_{Λ} BSPLND representation (dashed blue curve) and the best parameter estimation method (black curve) at the site (0.25,0.25) for a 20 keV microbeam.

3.4.2 Trends in the Lognormal Parameters over *r'*, *h'*, and *e*

Figure 3-15 shows the trend in the fit parameters μ and σ of the lognormal distributions over r', h', and *e*. The first column of plots in the figure represents μ, while the second column represents σ. The rows indicate *e*, and the axes in the individual plots run over the domain of $\vec{r}_i^{\,\prime} \times \vec{h}_i^{\,\prime}$ (see Section 3.3.2.1) up to 1.2 of the p90 value for the energy level *i*. The parameter μ has units of ln(keV/micron) while σ is unitless.

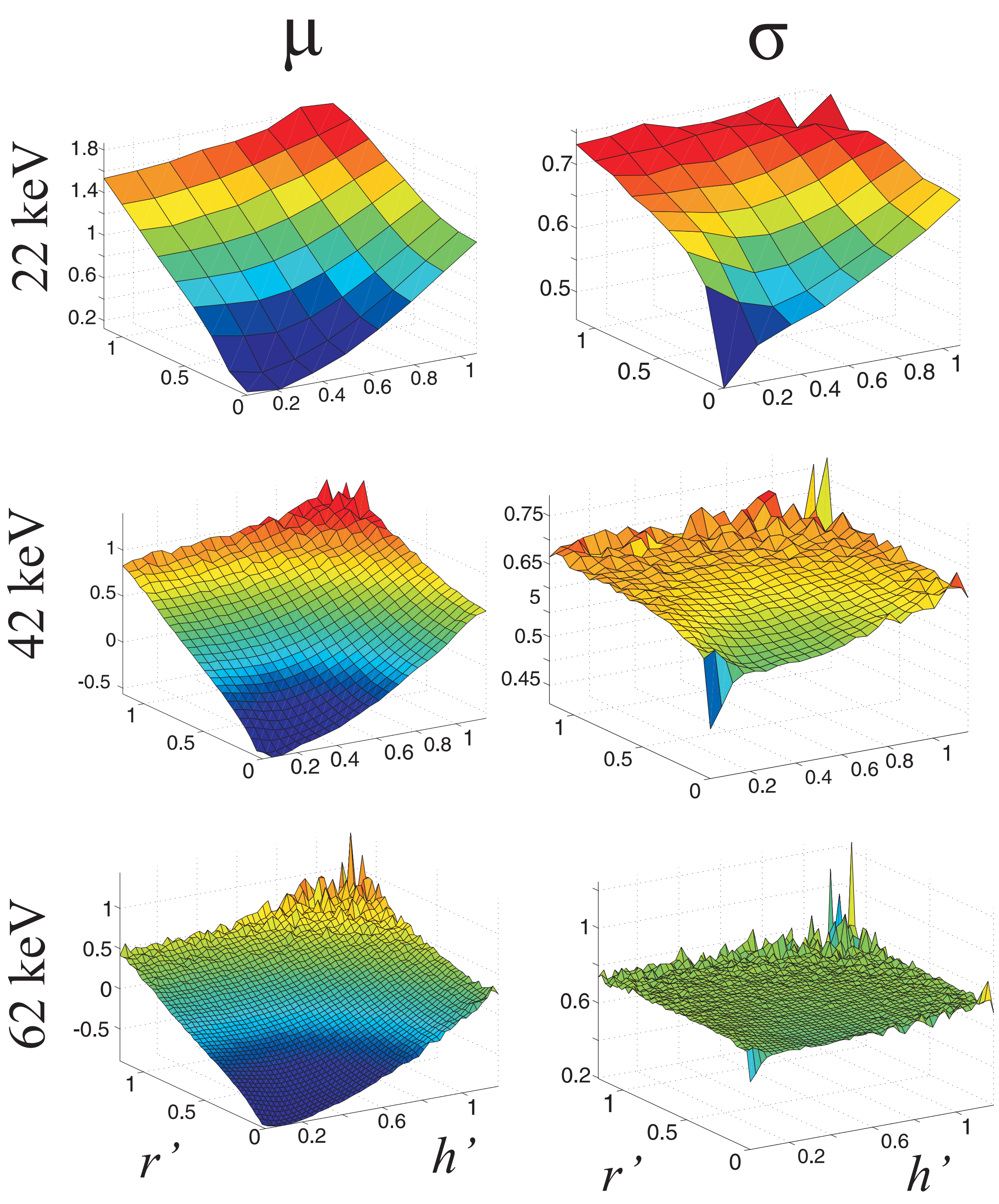

Figure 3-15: Trend in the fit parameters μ and σ of the lognormal distributions over r', h', and *e*.

Notice how μ for each energy resembles the conditional mean of energy deposited surface in Figure 3-3. It is near a minimum at the beam entry point and then bowls up to an asymptotic value past about 1.5 times the p90. It also has about the same distance between its minimum and maximum on each plot and decreases over the entire spatial domain with increasing beam energy. The surfaces for σ at each beam energy reach asymptotic values as well, but they do not hold their shape with increasing energy.

With some visual effort and mathematical insight the surfaces for μ and σ can be represented by the functions

$$\mu(r',h') = a\left(1 - \frac{1}{\exp(bh'^2 + cr'^2)}\right) \tag{31}$$

and

$$\sigma(r',h') = d\left(1 - \frac{1}{eh'^2 + fr'^2 + 1}\right) + g\,, \tag{32}$$

with parameters *a*, *b*, *c*, *d*, *e*, *f*, and *g*. If equations (31) and (32) are plugged into the original equation for the lognormal distribution, then a fit function in terms of these seven new parameters results, namely

$$f_\Lambda(\lambda) = \frac{1}{\lambda\sigma(r',h')\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{\ln(\lambda) - \mu(r',h')}{\sigma(r',h')}\right)^2\right), \quad 0 < \lambda < \infty\,. \tag{33}$$

The new fit function can then be pushed through the process of nonlinear least squares fitting yet again over λ, r', and h' at each fixed beam energy. For *n* energy levels, the representation is then reduced to a set of *n* equations, each a function of λ, r', and h'.

3.4.3 Goodness of Fit Testing of Lognormal Fits

Goodness of fit testing was introduced in Section 2.3.2.7 as a standard method of determining if a data set is governed by a particular distribution representation. This method will now be applied to a sample $f_\Lambda(\lambda)$ data set at a fixed site and a fixed energy level to assess its lognormality. Sites very close to the beam entry point at low energy levels produce $f_\Lambda(\lambda)$ data sets that look the most lognormal, so that is a good place to begin. Over the domain D_{4r} from Section 3.3.3, the best choice for $f_\Lambda(\lambda)$ is at the site $(r', h') = (0, 0.0847)$ with a beam energy of 20 keV. Data sets representing $f_\Lambda(\lambda)$ at other sites and energy levels could be assessed if this sample data set passes the goodness of fit testing process.

The first step in a chi-square goodness of fit test is to estimate the distribution parameters using one of the methods introduced in Section 2.3.2.6. For this discussion the method of maximum likelihood is used, since it was shown to give a slightly better result than the method of nonlinear least squares. Using the method of maximum likelihood, the parameters are estimated to be $\mu = 0.3535$ and $\sigma = 0.5458$, yielding a lognormal representation of

$$g_\Lambda(\lambda) = \frac{1}{\lambda(0.5458)\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{\ln(\lambda) - (0.3535)}{0.5458}\right)^2\right), \quad 0 < \lambda < \infty . \tag{34}$$

All histograms in the PITS data set are split into 88 non-overlapping intervals. In each interval there is an error between the actual value of the data point and its expected value given the representation in Equation (34). Overall, the chi-square statistic is a measure of the relative error between these two quantities. If *n* is the number of data points in the *k* included intervals, then the expected value of the distribution in the *i*th interval is $ng_\Lambda(\lambda_i)\Delta\lambda_i$ and the chi-square statistic is

$$C = \sum_{i=1}^{k} \frac{\left(\lambda_i - ng_\Lambda(\lambda_i)\Delta\lambda_i\right)^2}{ng_\Lambda(\lambda_i)\Delta\lambda_i} . \tag{35}$$

From the data set, only intervals 23 to 60 are used in order to keep the denominator greater than zero. In addition, there are approximately 3 million data points within these intervals, so *C* becomes

$$C = \sum_{i=23}^{60} \frac{\left(\lambda_i - (3e6) g_{\Lambda}(\lambda_i)\right)^2}{(3e6) g_{\Lambda}(\lambda_i)}. \tag{36}$$

The statistic *C* has a χ^2 distribution with (60-23)-1-2 = 34 degrees of freedom. The hypothesis that the data follows a lognormal distribution with parameters $\mu = 0.3535$ and $\sigma = 0.5458$ is rejected at the α level of significance if $C \geq \chi^2_{1-\alpha,34}$. At $\alpha = 0.01$, for example the hypothesis should be rejected if $C \geq 56.06$. Since *C* is an astounding 2.314e5, the conclusion is that the distribution is not lognormal at this site.

So why is the chi-square statistic so large? One would think that three million tracks should generate a data set that is very accurate to its underlying standard distribution, if it exists. As more tracks are added, the data *converges* to the distribution. The PITS data set, however, does not seem to converge to the lognormal distribution. Some irregularities do not disappear as the number of tracks is increased. Also, the estimation of parameters would have been much more accurate if the *raw data* were collected in the original simulation in lieu of the binned quantities. A higher density binning system would help as well, but the resulting chi-square statistic would probably be just as large. It should be concluded that the distributions are *approximately* lognormal but not *statistically* lognormal.

4 Application of the BSPLND Representations: Irradiation of a Cell Nucleus

4.1 Background

The representations developed for the quantities within this thesis can only be applied to one micron diameter spherical targets. The use of other target volumes would require a complete rerun of the simulation and fitting process. But what if it were possible to pack *any* volume with one micron diameter spheres and calculate the sought after quantities *directly from the existing fits*? If so, then what about automating the process so that the user could graphically specify their target in three dimensions, set the appropriate simulation parameters, press a button initiating the sphere packing, and then press another button to evaluate the quantity over the domain specified? At the beginning of this study these suppositions seemed plausible, and they drove the development of the fits and their associated software. Unfortunately, for the most part, calculating any quantities, whether stochastic or not, for an arbitrary volume using these methods is not possible. It is the purpose of this section to explain why this is so and to expose the methods that *can* give a limited amount of information concerning larger volumes.

It would be most useful if the stochastic distributions themselves could be aggregated from the one micron diameter spheres into a stochastic distribution for the larger packed volume. Letting Λ_i be the i^{th} stochastic for $1 \le i \le N$ with distribution density $g_{\Lambda_i}(\lambda_i)$, then the aggregate distribution would represent the new stochastic

$$\Lambda_{ag} = f(\Lambda_1, \Lambda_2, \ldots, \Lambda_N) \tag{37}$$

and have a distribution density function of $g_{\Lambda_{ag}}(\lambda_{ag})$. The big hurdle to making the aggregation successful, however, is the *dependency* rooted within each particle track. This track "history" makes the deposition of a stochastic in one target dependent on the track's previous depositions in other targets. Unless this dependency can be accounted for, there is no way to find the distribution for all of the combined targets. If the distributions were independent then they could undergo a process called *convolution* to produce the desired aggregate distribution.

4.1.1 Simulation Setup

It was known from the outset that the aggregation of distributions was futile. Others, such as Kellerer [1969] have previously explored this issue and come to the same conclusion as myself. The aggregation of the distribution summary statistics, however, was assumed to be possible, and equations were developed to perform the aggregation using the developed fits. The entire process was to be showcased and tested in a sample problem from Biology, in which a 10 micron diameter spherical cell nucleus is irradiated by a 50 keV electron microbeam positioned 6 microns directly below the cell's center point. Comparisons were to be made for the relevant quantities between their simulated values and their calculation using the fits and aggregation. A diagram of the simulation setup is shown in Figure 4-1.

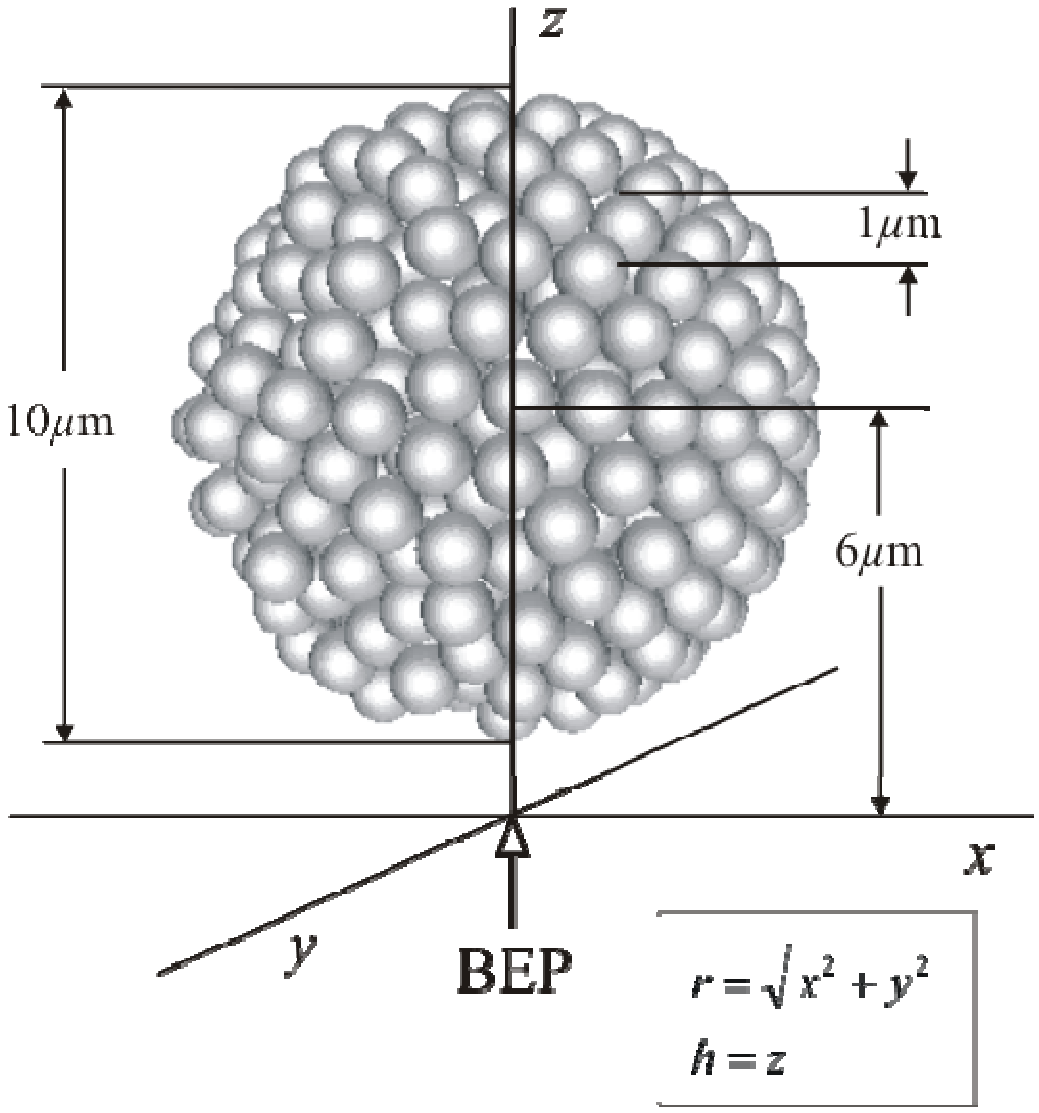

Figure 4-1: The simulation setup for the cell aggregation problem. Only the packed micron diameter spheres are shown in the diagram.

The exact steps in carrying out the comparisons are given below.

1. Pack the nucleus with one micron diameter spheres using a 3rd party sphere packing routine. A Python utility called packer.py is capable of this. It can be obtained via the world wide web from http://packinon.sourceforge.net. In

packer.py you can specify any two of the following: small sphere diameter, large sphere diameter, and number of spheres. It saves the packed sphere centers in a text file.

2. Construct a scoring file where
    a. the target sphere geometries are loaded in from the text file produced in list item 1;
    b. a simulation is carried out with those spheres as targets, where the desired quantities are scored; and
    c. an output XML file is produced containing the relevant scoring quantities.
3. Run the simulation for millions of tracks at a beam energy that is represented in the fits (e.g. within 20 to 80 keV).
4. Construct a MATLAB file that
    a. loads the sphere geometry;
    b. loads the PITS simulation results;
    c. loads the previously developed fit structures for Φ, μ_E, and Λ;
    d. uses a yet to be determined algorithm to calculate the aggregate of μ_E and the aggregate of μ_Λ from the micron diameters spheres, taking into account that for Λ only lineal energies up to 5.632 keV/micron are represented;
    e. and compares each aggregated quantity with the respective quantity in the large sphere. The space between the packed spheres would be compensated for in this comparison.

These steps were executed for the 50 keV electron beam simulation mentioned at the beginning of the section. Unfortunately, the simulation and aggregate calculations for the most part did not agree. Insights into the source of these discrepancies will be discussed in the remaining sections of this chapter.

4.1.2 Issues with the Aggregation Problem

Issue 1: The probability of a track reaching either of two volumes is not the probability of the track reaching the first volume plus the probability of the track reaching the second volume.

Specifically, if Φ_1 is the probability of a track reaching sphere 1 and Φ_2 is the probability of a track reaching sphere 2, then the probability of a track reaching the volume enclosed by both spheres, $\Phi_{(1\text{ or }2)}$, is not $\Phi_1 + \Phi_2$. This is intuitive if you consider spheres 1 and 2 lying very near the beam entry point. The value of Φ for each of these spheres would be near 1, so their aggregate, $\Phi_{(1\text{ or }2)}$, could not possibly be 2. Probabilities can never be greater than 1. The issue here lies with tracks passing through *both* spheres at the same time, whose probability of occurring is written $\Phi_{(1\text{ and }2)}$. The quantity $\Phi_{(1\text{ or }2)}$ must be compensated for by subtracting $\Phi_{(1\text{ and }2)}$ from the sum of Φ_1 and Φ_2, yielding

$$\Phi_{(1\text{ or }2)} = \Phi_1 + \Phi_2 - \Phi_{1\text{ and }2}\,. \tag{38}$$

The overlap of every possible combination of spheres, however, is necessary in order to calculate $\Phi_{(1\,\text{and}\,2)}$. For n spheres there are $2^n - n - 1$ possible combinations, meaning that this problem has a $O(2^n)$ growth rate!

Issue 2: PITS stochastic quantities can be divided into two groups: additive and non-additive. Only additive stochastics have summary statistics that can be aggregated.

Depositions of additive stochastics in non-overlapping volumes sum to give the deposition in the combined volume. Depositions of non-additive stochastics, on the other hand, do not have this property. A stochastic becomes non-additive when it is the ratio of two additive stochastics. The total energy deposited in two spheres, for example, is composed of the energies from each individual sphere. Hence, the total energy deposited stochastic is additive. The stochastic Λ, however, is not additive, since it is the ratio of two additive stochastics. To see why, consider two adjacent targets labeled 1 and 2. Then, defining λ_1 and λ_2 as the quantities of the stochastic Λ deposited in each target and λ_{ag} as the quantity deposited in both targets gives

$$\lambda_1 + \lambda_2 = \frac{e_1}{l_1} + \frac{e_2}{l_2} = \frac{l_2 e_1 + l_1 e_2}{l_1 l_2} \tag{39}$$

and

$$\lambda_{ag} = \frac{e_1 + e_2}{l_1 + l_2} \neq \frac{l_2 e_1 + l_1 e_2}{l_1 l_2}. \tag{40}$$

Therefore, $\lambda_{ag} \neq \lambda_1 + \lambda_2$.

If a stochastic is additive, then its summary statistics can be aggregated from the smaller enclosing volumes. To prove this, first suppose e_i is the energy deposited in the i^{th} target and $e_{1,2,\ldots,n}$ is the total energy deposited in all targets if n is the number of targets. Then, since the energy deposited stochastic is additive,

$$e_{1,2,\ldots,n} = \sum_{i=1}^{n} e_i \,. \tag{41}$$

For many track realizations, the energy deposited in a particular target is approximately equal to the probability of reaching that target, Φ, times the expected deposition of energy within the target once it is reached, μ_E. Therefore, for the i^{th} target,

$$e_i \approx \Phi_i \mu_{Ei} \,, \tag{42}$$

and for all individual targets taken as one large target,

$$e_{1,2,\ldots,n} = \Phi_{(1 \text{ or } 2 \text{ or} \cdots \text{or } n)} \mu_{E1,2,\ldots,n} \,, \tag{43}$$

where $\mu_{E1,2,\ldots,n}$ is the as yet undetermined expectation of energy deposited in all of the individual targets. Combining Equations (41), (42), and (43) gives

$$\sum_{i=1}^{n} \Phi_i \mu_{Ei} \approx \Phi_{(1 \text{ or } 2 \text{ or} \cdots \text{or } n)} \mu_{E1,2,\ldots,n} \,, \tag{44}$$

so that

$$\mu_{E1,2,\ldots,n} \approx \frac{\sum_{i=1}^{n} \Phi_i \mu_{Ei}}{\Phi_{(1 \text{ or } 2 \text{ or} \cdots \text{or } n)}} \,. \tag{45}$$

This process unfortunately does not work for non-additive stochastics because Equation (41) is violated. The best that can be done in finding the aggregated summary statistic is to average the summary statistics from all of the individual targets.

Issue 3: The probability of a track reaching the larger enclosing volume, Φ_{big}, is required in order to relate any additive summary statistic in that volume—such as μ_E—to the same statistic of its packing spheres. Determining Φ_{big}, however, requires rerunning simulations for the larger volume.

Relating an additive summary statistic for the larger volume to the same statistic for its enclosing volumes requires the introduction of the *packing density*. The packing density, denoted by ρ, is defined as the ratio of the total volume of the small packing objects to the volume of the larger enclosing object. If *n* is the number of *r* radius spheres packed in a larger sphere of radius *R*, then

$$\rho = n\left(\frac{r}{R}\right)^3. \tag{46}$$

Packing the sample "nucleus" with micron diameter spheres using the packer.py utility creates a total of 537 spheres. Since the ratio of a micron diameter sphere's volume and a 10 micron diameter sphere's volume is 1 to 1000, the packing density in this situation is 0.537. This means that just under 50% of the large sphere's volume is empty space. A packing density of 0.537 may seem low at first sight, but the maximum achievable packing density for spheres is only $\pi/3\sqrt{2} \approx 0.74048$ when they are stacked like oranges in a grocery store. This arrangement is not possible in the nucleus problem, however, because the small spheres must be contained wholly within the larger sphere. The relatively low packing density in the nucleus problem means that compensating for ρ will be a large factor in achieving accurate aggregation results.

The packing density concept can be applied to additive stochastic depositions by making use of the relation

$$\frac{\sum_{i=1}^{n} e_i}{e_{big}} \approx \frac{\sum_{i=1}^{n} V_i}{V_{big}} = \rho \,. \tag{47}$$

Here e_i is the deposition of the stochastic in the i^{th} packing sphere, V_i is the i^{th} packing sphere's volume, e_{big} is the deposition in the larger packed sphere, and V_{big} is the larger packed sphere's volume. Then, since $e_i \approx \Phi_i \mu_{E_i}$ from Equation (42),

$$\begin{aligned} e_{big} &\approx \frac{1}{\rho}\sum_{i=1}^{n} e_i \\ &\approx \frac{1}{\rho}\sum_{i=1}^{n} \Phi_i \mu_{E_i} \,. \end{aligned} \tag{48}$$

The technique does not make sense for non-additive stochastics because the ratio of $\sum_{i=1}^{n} e_i \Big/ e_{big}$ in Equation (47) is meaningless.

So far the quantity $\mu_{E_{big}}$ has been absent from the discussion. Bridging the gap between μ_{E_i} and $\mu_{E_{big}}$ requires the introduction of Φ_{big}, through the relation $e_{big} \approx \Phi_{big} \mu_{E_{big}}$. Then

$$\begin{aligned} \mu_{E\,big} &\approx \frac{e_{big}}{\Phi_{big}} \\ &\approx \frac{1}{\Phi_{big}\rho}\sum_{i=1}^{n} \Phi_i \mu_{E_i} \end{aligned} \tag{49}$$

A problem arises here, however. The quantity Φ_{big} was never computed in the project's simulations, nor is it computable from Φ_i without knowing $\Phi_{(1 \text{ and } 2 \text{ and } \cdots \text{and } n)}$ (see Equation (38)). It could manually be created for the large sphere by rerunning the simulation, but this defeats the purpose of creating the fits in the first place. Luckily, for the sample problem in Figure 4-1, the large sphere is close enough to the beam entry point to yield $\Phi_{big} = 0.9827 \approx 1$. Considering this, Φ_{big} can be eliminated and Equation (49) becomes

$$\mu_{E\,big} \approx \frac{1}{\rho}\sum_{i=1}^{n}\Phi_i \mu_{E_i}. \tag{50}$$

Despite all of these restrictions, Equation (50) gives a value of 8.0416×10^3 for the sample problem, versus 1.1546×10^4 eV calculated from the simulation, a 30% difference. Placing the large sphere further away from the beam entry point would degrade this result.

5 Conclusions and Future Work

The overall focus of this thesis was on building a system for the efficient production, approximation, and visualization of microdosimetric data derived from Monte Carlo particle track simulations. From these approximations a better understanding of the nature of electron microbeams and their microdosimetric impact on targets of a set geometry emerged. One micron diameter spheres were used as targets primarily to exploit the azimuthal symmetry of the simulated tracks. The methods used could easily translate to targets of other shapes and sizes by making a couple of parameter changes in the scripts and rerunning the track simulations. As well, the microdosimetric quantities μ_E, Φ, and Λ were highlighted throughout the thesis because of their biological significance, but any other stochastic or non-stochastic quantities could have been chosen in their place. The *methods* exposed by the thesis are just as important as the results presented. So too is the educational value of the write-up itself, where the concepts of ionizing radiation, microdosimetry, electron track structures, the mathematical approximation of data, B-spline fitting techniques, statistical distributions, parameter estimation, and goodness-of-fit testing have been introduced.

Even though the representations developed for this thesis are only applicable to one micron diameter spherical targets, they still have merit in helping to visualize the nature of an electron microbeam's radiation field. The representations give a sense of how the microbeam impacts the spherical targets as they are moved within the simulation space and as the beam energy is altered. Other targets would respond much the same to a change in location and beam energy as one micron diameter spheres. Targets very far from the beam entry point, for example, would have a relatively high conditional mean of energy imparted owing to the slowdown of the electrons far from the beam entry point. Likewise, the distribution of stochastics such as Λ within these targets would be less peaked because of the more random nature of the tracks at these distances. Therefore, it would be fruitful to use the methods developed in this thesis to study how targets of different geometries respond to a change in location and beam energy. The easiest and most precise way of doing this is to make sphere diameter an additional variable in the representations, perhaps at a constant beam energy to reduce the dimensionality of the problem.

Additional time should also be spent on studying the sphere packing problem introduced in Section 4. Finding an efficient method for calculating microdosimetric quantities in larger targets using smaller identical targets with known statistics remains a wide open problem. Further study may find such a method for the summary statistics. Or the problem might be proved to be NP-complete, meaning an efficient polynomial running time method does not exist to solve the problem. As it stands, the problem has been shown to be solvable in Section 4.1.2 (see Equation (49)), but the techniques presented depended on calculating $\Phi_{(1 \text{ and } 2 \text{ and} \cdots \text{and } n)}$ for n spheres, which has a growth rate of $O(2^n)$. Finding a method for stochastic distributions would be much more difficult because they depend on individual depositions and not aggregate quantities. How a track behaves in a target is a direct consequence of its history, which is different for every track in a PITS Monte Carlo simulation. Finding this history from the distributions could then lead to a solution for the larger target, but this too is an unsolved problem. According to Kellerer [1969, pg 2], "The question of whether an operational method can be found to reconstruct the spatial patterns of energy deposition from the [stochastic] distributions $f_1(z)$ [$f_\Lambda(\lambda)$ in this text] is one of the interesting open problems in microdosimetry." He goes on to say "As yet there is no technique to extract all information contained in the [stochastic] distributions $f_1(z)$. Consequently it is also not possible to calculate the distributions for a non-spherical region from those determined in spheres…"

One problem that *can* be solved efficiently using the aggregation technique is estimating the overall amount of an additive stochastic quantity that has been deposited in the large packed volume. The mathematics behind the technique was introduced in Section 4.1.2 (see Equation (48)) , but it was never applied to the cell nucleus problem. Work in the immediate future will involve using the technique to assess the total energy deposited in the sample cell nucleus problem and other such problems in Biology. It will also be interesting to remove the packing density from the calculation by comparing the total energy deposited scaled by a non-stochastic quantity such as the mass of the targets. The total value deposited for the non-additive stochastic Λ will be treated as well by averaging the total quantity of Λ deposited in each of the packing spheres and comparing this value with the total deposited in the larger volume. After preliminary calculations have been made using data exclusively from simulations, an interactive graphical application will be built to use the representations developed in this thesis to solve the aggregation problems for custom geometries. In this application the user would modify the target location, set the beam energy within a range of 20 to 80 keV, and then request a total stochastic quantity to be calculated (either E or Λ). They would then press a button to initiate the packing process, evaluate the necessary representations, and make the final calculations. The spheres would be color coded to show the relative quantities of the stochastic delivered. Such an interactive visualization application could easily be written in MATLAB using its graphical user interface and visualization tools.

Appendix

pits.F Main Procedure

```
>Initialize parallel computing environment
>Open and read input parameter file, inpits.dat
>Initialize random number generator
>Read in the probability cross-section tables for electron transport,
new_crossec.dat
>Initialize electron transport module parameters by calling
eltran.init()
>Initialize interaction history output file, ioncoord.dat

// Divide the tracks equally among the processors.  Assume
number_of_tracks is divisible by number_of_processors
>number_of_tracks_this_processor = number_of_tracks/number_of_processors

// Process each individual track
>For track = 1 to number_of_tracks_this_processor
      // Produce the history of one ion track
      >call pits.iontrac()

      // Score the track
      >call pits.outpend()
>End

>Aggregate scoring data from all processors onto one processor by
calling scor.add_freq()
>Finalize the parallel computing environment
```

```
>Finalize scoring analysis by calling scor.outpfin()
```

Algorithm 5: pits.F main procedure.

pits.F iontrac() Subroutine

```
/*
This the main procedure for ion track structure calculations.  Follows
the history of one ion and stacks the ionization interactions for
later transport of the secondaries.
*/

>While true
      // We have a new interaction
      >Detemine the type of ion interaction and its specifications
      >Calculate the location of this interaction
      >Update the path track length
      >Output the ion event info to a file
      >Transport the secondary electrons produced from this
      interaction
>End
```

Algorithm 6: pits.F iontrac() subroutine.

scor_s.F inlinit() Subroutine

```
>Create scoring bin edges, centers, and widths
>Read in parameters from inpits.dat for scoring
>Define site centers
>Initialize histogram arrays to zero
>Write header information to the output file
```

Algorithm 7: scor_s.F inlinit() subroutine

scor_s.F score() Subroutine

```
>Declare sites_traversed_array
>Create and clear sparse accumulation arrays:
      >accume_sparse(MAX_LAYERS, MAX_CYLINDER, MAX_SECTORS) = total
      energy deposited by all events in a site
      >accumep_sparse = accumulates energy deposited by the primary
      electron in a site
      >accumtl_sparse = accumulates track length accumulated by all
      electrons in a site
      >accumtlp_sparse = accumulates track length accumulated by the
      primary electron in a site
      >accumndp_sparse = accumulates number of primary event
      interactions in a site
      >accumntch_sparse = accumulates number of toucher events in a site
      >accumtltchall_sparse = accumulates track length accumulated by
      all toucher events in a site
      >accumetchall_sparse = accumulates energy accumulated by all
      toucher events in a site
      >accumtltchlg_sparse = accumulates the tracklength of the lowest
      generation events for toucher events
      >lowest_gen_sparse = lowest generation event to hit a site
      >sites_traversed_sparse = sites actually traversed by a particle

>Loop through events to determine the spheres that are traversed
      >Calculate the layer and cylinder that event is in
      >Calculate the exact site the event is in within its cylinder
            >Optimize by assuming the event is in the same site as the
            last event
            >Make a note of this site within sites_traversed_sparse
>End loop

>Loop through events to determine the lowest generation that passes
through each sphere in sites_traversed_array
      >Loop through sites traversed sparse to determine which sphere the
      event is in
            >Determine if this event is lowest in generation in the
            sphere
            >If the event is a primary event then add it to
            accumndp_sparse
      >End loop
>End loop

>Loop through the events to determine parameters such as energy
deposited in hit spheres
      >Loop through sites traversed sparse to determine which sphere the
      event is in
            >Add the energy deposited by this event to accume_sparse
            > If the event is produced by a primary electron, add energy
            deposited to accumep_sparse
            >Add the length of this event’s track since the last event
            to accumtl_sparse
            >If the event is produced by a primary electron, add the
            event’s track length to accumtlp_sparse
            >If the event is a toucher event then
                  >Add the energy to accumetchall_sparse
                  >Add the track length to accumtltchall_sparse
                  >Increment accumntch_sparse
                  >If the event is the lowest generation in the sphere
                        >Add the track length to accumtltchlg_sparse
      >End loop
>End loop

>Loop through the hit spheres to perform scoring
      >Score the number of primary event interactions
            >Get accumndp_sparse for this site
            >Determine the histogram bin this fits in
            >Increment the nfreqndp for that bin
      >Score the accumulated energy in this site
      >Score the accumulated tracklength of all events for this site
      >Score the accumulated energy of the primary electron for this
      site
      >Score the accumulated track length of the primary electron for
      this site
      >Score the energy deposited by all toucher events for this site
      >Score the tracklength accumulated by all toucher events for this
      site
      >Score the tracklength of the lowest generation of toucher events
>End loop
```

Algorithm 8: scor_s.F score() subroutine.

Bibliography

Aitchison, J., Brown, A.C. (1957) *The Lognormal Distribution*. Cambridge University Press, London and New York.

ICRU. (1970) *Linear Energy Transfer*. Report 16, International Commission on Radiation units and Measurements, Bethesda, MD.

ICRU. (1983) *Microdosimetry*. Report 36, International Commission on Radiation units and Measurements, Bethesda, MD.

Kellerer, A.M. (1969) "Analysis of Patterns of Energy Deposition: A Survey of Theoretical Relations in Microdosimetry." *Proceedings of the Second Symposium on Microdosimetry*.

Larsen R.J., Marx, M.L. (1986) *An Introduction to Mathematical Statistics and its Applications*. Prentice Hall, New Jersey.

Lee, S., Wolberg, G., Shin S.Y. (1997) "Scattered Data Interpolation with Multilevel B-Splines", *IEEE Transactions on Visualization and Computer Graphics*, Vol. 3, No. 3.

Lynch, D.J., Wilson, W.E., Batdorf, M.T., Sowa Resat, M.B., Kimmel, G.A., and Miller, J.H. (2005) "The Spatial Distribution of Energy Deposition for an Electron Microbeam." *Radiat. Res.* 163, 468-472.

Miller J.H., Resat M.S., Metting N.F., Wei K., Lynch D.J., Wilson, W.E. (2000) "Monte Carlo Simulation of Single-Cell Irradiation by an Electron Microbeam." *Radiat Environ Biophys* 39:173-177.

Miller, J.H., Batdorf, M.T., Lynch, D.J., Lewis, R.R., and Wilson, W.E. (2004) "Microdosimetry of Electron Microbeams." *Radiat. Res.* 162, 474-479.

Shene, C.K., "CS3621 Introduction to Computing with Geometry Notes" at http://www.cs.mtu.edu/~shene/COURSES/cs3621/NOTES/

Weis, M.P. (2000) “BSPLND, A B-Spline N-Dimensional Package for Scattered Data Interpolation.” M.S.C.S. Thesis, Washington State University.

Wilson, W.E., Lynch, D.J., Wei, K., Braby, L.A. (2001) “Microdosimetry of a 25 keV Electron Microbeam.” *Radiat Res*. 155, 89-94.

Wilson, W.E., Miller, J.H., Lynch, D.J., Lewis, R.R., and Batdorf, M.T. (2004) “Analysis of Low Energy Track Structure in Liquid Water.” *Radiat. Res*. 161, 591-596.

Wilson, W.E., Nikjoo, H. (1994) “PITS: A Code Set for Positive Ion Track Structure.” In *Computational Approaches in Molecular Radiation Biology* (Varma, M.N., and Chatterjee, A., Eds.). Plenum Press, New York.