

**SIMULATING PARTICLE  
TRANSPORT WITHIN A  
MICROCALORIMETER TO  
UNFOLD THE DETECTOR  
RESPONSE**

By

Adam E. Brown

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

Bachelor of Science

Houghton University

May 2023

Signature of Author.....

Department of Physics

.....

Dr. Katrina Koehler  
Assistant Professor of Physics  
Research Supervisor

.....

Dr. Brandon Hoffman  
Professor of Physics

# **SIMULATING PARTICLE TRANSPORT WITHIN A METALLIC MAGNETIC CALORIMETER TO UNFOLD THE DETECTOR RESPONSE**

By

Adam E. Brown

Submitted to the Department of Physics  
on 10 May 2023 in partial fulfillment of the  
requirement for the degree of  
Bachelor of Science

## **Abstract**

The measurement technique of Decay Energy Spectroscopy (DES) utilizes high-energy resolution ( $7.5 \pm 0.2$  eV FWHM at 6539 eV) [1] low temperature microcalorimeters to measure the total energy of a decay from an embedded radioactive source. DES spectra are histograms of the total decay energy thermalized in the absorber. Some of this energy is lost, largely due to decay products escaping the absorber or energy stored in metastable states (the latter depends on source preparation and is not considered in this work). This results in a measurement of energy that is lower than the decay energy. The escape probability is not constant as a function of initial decay energy but is dependent on the absorber material and the source's energy, type, location, and distribution—all of which form what we call the detector response. In this work, the response matrix for a microcalorimeter is built using EGSnrc—a Monte Carlo particle transport software—to simulate the energy deposition of a point source of monoenergetic beta particles ranging from 10 keV to 2 MeV. This response matrix may be used to deconvolve the detector response from a DES measurement so systematic uncertainty can be reduced. This will result in a more precisely known beta decay shape, important for fields such as nuclear medicine and testing theoretical descriptions of beta decay at low energies.

Thesis Supervisor: Dr. Katrina Koehler

Title: Assistant Professor of Physics

## TABLE OF CONTENTS

<b>Chapter 1 .....</b>	<b>6</b>
<b>1.1. What Are Beta Decay and Electron Capture?.....</b>	<b>6</b>
1.1.1. Original Experiments and Discovery.....	6
1.1.2. Theory of Beta Decay and Electron Capture.....	10
<b>1.2. Decay Energy Spectra .....</b>	<b>11</b>
1.2.1. Measuring Decay Energy Spectra .....	11
1.2.2. Physical Measurements and Interpretations .....	13
<b>Chapter 2 .....</b>	<b>16</b>
<b>2.1. Low Temperature Detectors .....</b>	<b>16</b>
2.1.1. Principles of Microcalorimeters .....	17
2.1.2. Cryogenics.....	18
2.1.3. Transition-Edge Sensors .....	20
2.1.4. Metallic Magnetic Calorimeters.....	22
<b>Chapter 3 .....</b>	<b>24</b>
<b>3.1. EGSnrc .....</b>	<b>24</b>
<b>3.2. The Physics of Particle Transport Simulation .....</b>	<b>25</b>
3.2.1. Photon Interactions.....	25
3.2.2. Electron Interactions .....	28
<b>3.3. Response Matrix .....</b>	<b>30</b>
3.3.1. Theory Behind Response Matrix.....	32
<b>3.4. Assumptions.....</b>	<b>33</b>
<b>Chapter 4 .....</b>	<b>34</b>
<b>4.1. Simulation Parameters .....</b>	<b>34</b>
4.1.1. Deconstruction of an Input File .....	34
4.1.2. Monoenergetic Simulations .....	39
<b>4.2. Response Matrix .....</b>	<b>40</b>
<b>Chapter 5 .....</b>	<b>41</b>
<b>5.1. Construction of Response Matrix .....</b>	<b>41</b>
5.1.1. Simulation Analysis.....	41
5.1.2. Inverse of the Response Matrix .....	43
<b>Chapter 6 .....</b>	<b>44</b>
<b>6.1. Conclusions and Future Plans.....</b>	<b>44</b>
<b>Appendix A.....</b>	<b>45</b>
<b>Appendix B.....</b>	<b>56</b>
<b>Appendix C .....</b>	<b>63</b>

## TABLE OF FIGURES

Figure 1. Continuous distribution of beta particle energies emitted from Radium E ( $^{210}\text{Bi}$ ) decay.....	7
Figure 2. Comparison of different values of $\mu$ (neutrino rest mass) on the shape of a beta spectrum near the endpoint energy $E_0$ . ....	9
Figure 3. Depiction of beta decay.....	10
Figure 4. Schematic of the internal measurement mode for an ideal absorber. ....	12
Figure 5. Comparison of different methods of spectroscopy for the alpha decay of $^{239}\text{Pu}$ and $^{240}\text{Pu}$ .....	13
Figure 6. Beta spectrum of $^{14}\text{C}$ , measured with an MMC (blue), and calculated spectrum using the code BetaShape (red). ....	14
Figure 7. Comparison of resolving capabilities of a MMC (green) and a HPGe (red) gamma measurement.....	17
Figure 8. An ideal calorimeter.....	18
Figure 9. (Left) Schematic of an adiabatic demagnetization refrigerator (ADR).....	19
Figure 10. (Left) Schematic of a $^3\text{He}/^4\text{He}$ dilution refrigerator []. (Right) Phase diagram of liquid $^3\text{He}/^4\text{He}$ mixture.....	20
Figure 11. The superconducting to normal phase transition of a Mo/Cu film.....	21
Figure 12. Electrical schematic of a TES coupled to a SQUID. ....	22
Figure 13. Linear attenuation coefficient of as a function of photon energy for NaI.....	26
Figure 14. Diagram depicting the process of electron-positron pair production.....	26
Figure 15. (Left) Depiction of Compton Scattering.....	27
Figure 16. (Left) Depiction of the photoelectric absorption. ....	28
Figure 17. Depiction of electron energy loss via Bremsstrahlung.....	29
Figure 18. Plot of X-ray mass attenuation coefficients as a function of energy for Au. ...	31
Figure 19. Particle tracks from an ENSnrc simulation of monoenergetic electrons travelling through an Au microcalorimeter absorber. ....	39
Figure 20. Histogram of energy escape of electrons from an Au absorber. ....	40
Figure 21. Four energy deposition histograms created from simulating various monoenergetic electron sources inside a $0.6 \times 0.6 \times 0.6 \text{ mm}^3$ Au absorber. ....	42
Figure 22. Comparison of $^{36}\text{Cl}$ beta spectra with the simulated measured histogram (left) and the deconvolved histogram (right). ....	43

## Chapter 1

### INTRODUCTION

#### **1.1. *What Are Beta Decay and Electron Capture?***

##### 1.1.1. Original Experiments and Discovery

The beginning of the 20<sup>th</sup> century marked a notable expansion of the scientific community's understanding of the atomic nucleus, especially regarding nuclear decay. After the initial discovery of radiation in 1896 by Henri Becquerel [2] using uranium—and subsequent experimentation carried out by Marie Curie exploring the basic properties of radioactive materials—the process of classifying different types of radiation began. Ernest Rutherford noticed changes in electrical current when covering different uranium salts (uranium sulfate and uranium oxide) with varying thicknesses of metal [3]. Rutherford noticed the current was significantly reduced with only  $2 \times 10^{-3}$  cm of aluminum, but the current did not reduce appreciably until the thickness was  $6 \times 10^{-3}$  cm. This led him to conclude there are two types of radioactive emissions from uranium, which he called alpha rays and beta rays, differentiated by their penetrating power. Alpha rays have very low penetrating power and can be blocked by a thin sheet of paper, and beta rays have more penetrating power but are still stopped by a thin sheet of aluminum. It was later shown by Becquerel in 1900 [4] that the beta rays were actually electrons by comparing their mass-to-charge ratios, using the same method that J. J. Thomson used to originally identify the electron: deflecting a cathode ray with an electric field. Because the beta rays deflected in the same way as the electrons deflected, Becquerel concluded that beta rays were electrons.

However, further study of beta decay led to an interesting discrepancy that would become fundamental to our understanding of nuclear structure and fundamental particles. In alpha and gamma decay, the alpha particle or gamma ray emitted is monoenergetic: dependent on the mass difference of the parent and daughter nucleus or the nuclear deexcitation between nuclear energy levels, respectively. Thus, it seems a reasonable conclusion that beta particles emitted from beta decays should exhibit the same property. However, James Chadwick showed in his 1914 experiment that the beta particle was emitted with a continuous

spectrum of energies [5] Charles D. Ellis and William A. Wooster further provided proof of the continuous beta decay spectrum with their investigation of Radium E (Bismuth-210) [6], as shown in Figure 1.

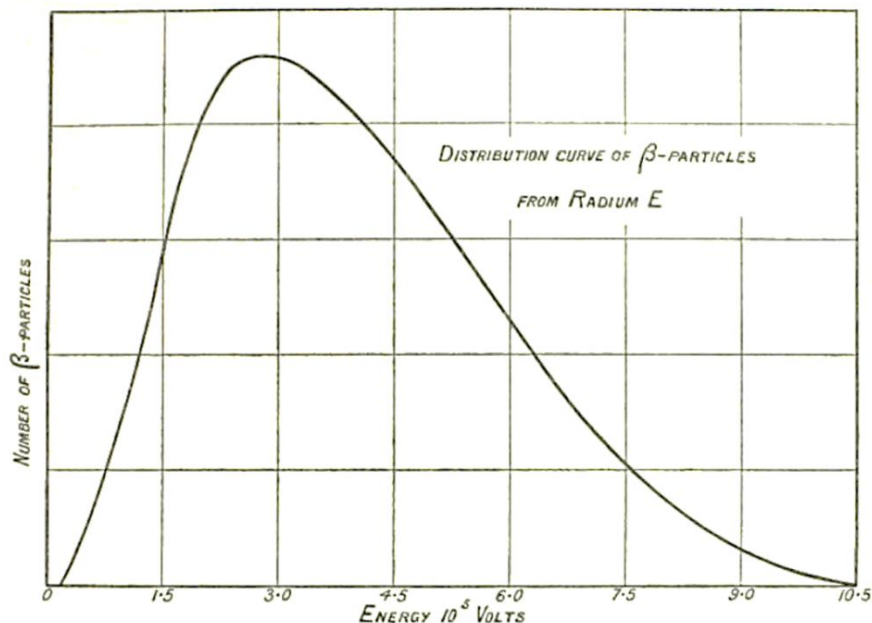


Figure 1. Continuous distribution of beta particle energies emitted from Radium E ( $^{210}\text{Bi}$ ) decay. Instead of beta decays emitting monoenergetic beta particles, a continuous spectrum of energies is allowed. This seemed to violate the conservation of energy and momentum, providing motivation for scientist to reconsider their understanding of nuclear decay. Taken from Ref. [6].

From this discovery arose a supposed contradiction: if beta decay was simply electron emission, then the energy should have a single defined value due to the conservation of momentum and energy. If the only decay products were the daughter nucleus and beta particle, then they would have equal and opposite momenta after the decay. This would result in a monoenergetic beta spectrum. This measured continuous spectrum of beta emission energies seemed to violate the conservation of energy.

It was not until 1930 that progress on this beta spectrum issue was made when Wolfgang Pauli proposed the existence of another particle emitted during beta decay that had thus far been undetected [7]. This particle would account for the missing energy seemingly lost, since momentum conservation with three particles does not have a single solution. He

characterized this particle to be neutral, very low mass,  $\text{spin} = \frac{1}{2}$ , obey the exclusion principle, and have extremely high penetrating power. Many of these assumptions were informed by the fact that this particle had never before been detected. The mass had to be much smaller than that of a proton, with a first limit on the particle's mass of less than 1% the mass of the proton [7]. This is because the Q value, or mass difference between the parent and daughter nucleus, is known for a beta decay. If the particle were more massive, the difference between the Q value and the measured energies would be much more pronounced, as a noticeable amount of the decay energy would be taken away by the unseen particle in the form of mass. The same goes for its neutral charge: if it were charged, its electromagnetic interactions with the environment would be noticed, and an additional charged product would violate the conservation of charge of the nuclear reaction.

Pauli's proposed particle would factor into Enrico Fermi's landmark theory of beta decay [8,9] published in 1934, where he posited the existence of the neutrino. He was able to demonstrate that the mass of the neutrino must be either zero or very small in comparison to the mass of an electron. He did so by determining the theoretical shape of a beta spectrum and examining the effect of different neutrino rest masses on the shape of the curve near the endpoint energy of the reaction and discovered that the greatest agreement with empirically found curves was when the mass of the neutrino was zero (see Figure 2).

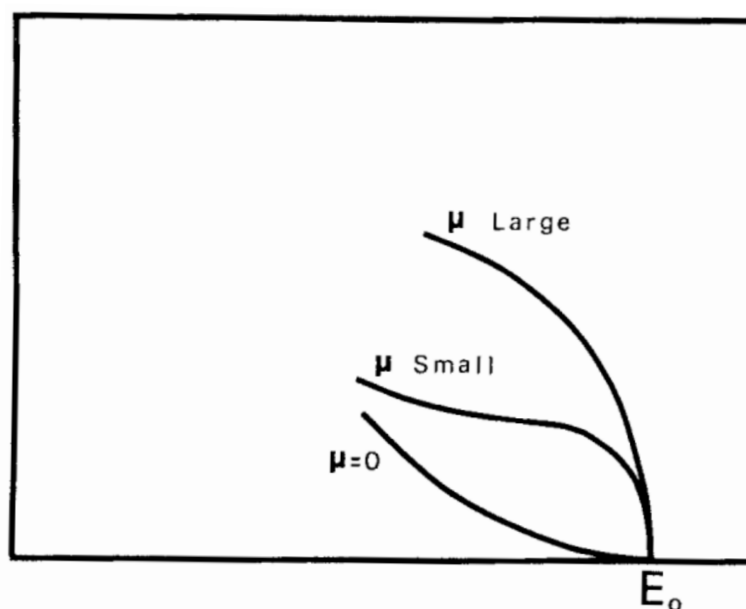
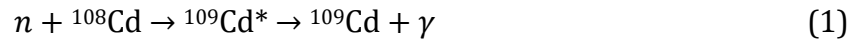




Figure 2. Comparison of different values of  $\mu$  (neutrino rest mass) on the shape of a beta spectrum near the endpoint energy  $E_0$ . The greatest agreement with physical measurements was when  $\mu = 0$ , suggesting the rest mass of the neutrino is either zero or extremely low. Understanding the precise shape of measured beta spectra, especially near the endpoint energy, allows for experimental determination of the neutrino rest mass. Taken from Ref. [9].

One problem with this theory is that experimental validation of the existence of the neutrino seemed to be nearly impossible, due to it having such weak interactions with matter. In 1956, neutrinos were finally detected in the Cowan-Reines experiment [10], which was based off the unique signatures that occur when a neutrino does interact with a proton. Despite the reaction probability being extremely low, Fermi's theory predicts that an electron antineutrino could interact with a proton to create a neutron and a positron. Positron annihilation with an electron creates two coincident gamma rays, and neutron capture by certain nuclei will result in an excited nuclear state, which quickly de-excites via the emission of a gamma ray. Detecting the coincidence of these two events provides a method to validate the existence of the neutrino particle. To achieve this, a nuclear reactor was used to provide a large neutrino flux, which would theoretically come from the large number of beta decays of daughter nuclei following the fission of  $^{235}\text{U}$ . The neutrino flux was incident upon two tanks each containing 200 L of water, which acted as sources of huge numbers of protons ( $1.3 \times 10^{28}$ ), increasing the chances of an interaction occurring. To absorb the product neutron, 40 kg of cadmium chloride ( $\text{CdCl}_2$ ) was dissolved in the water. Cadmium is an effective neutron absorber, and after neutron capture occurs, the product nucleus is in an excited nuclear state that emits a gamma ray to de-excite, shown by Equation (1).



Liquid scintillators between the water tanks were used for the detection of the gamma rays from both positron annihilation and neutron capture. After months of data collection, the experimental data showed evidence of neutrino existence, with a rate of three interactions occurring per hour.

### 1.1.2. Theory of Beta Decay and Electron Capture

Our current understanding of beta decay and electron capture stems from these historical discoveries. Beta decay is a type of radioactive decay where a beta particle ( $\beta^-$  or  $\beta^+$ ) is emitted from the nucleus. In  $\beta^-$  decay, a neutron ( $n$ ) within a nucleus is converted into a proton ( $p$ ) and an electron ( $e^-$ ) and electron antineutrino ( $\bar{\nu}_e$ ) are emitted, as shown in Equation (2),

$$n \rightarrow p + e^- + \bar{\nu}_e. \quad (2)$$

Figure 3 depicts a  $\beta^-$  event, where an element with atomic number  $X$  and mass number  $N$  decays to an element with the same mass number, but an atomic number increased by one to  $Y$ .

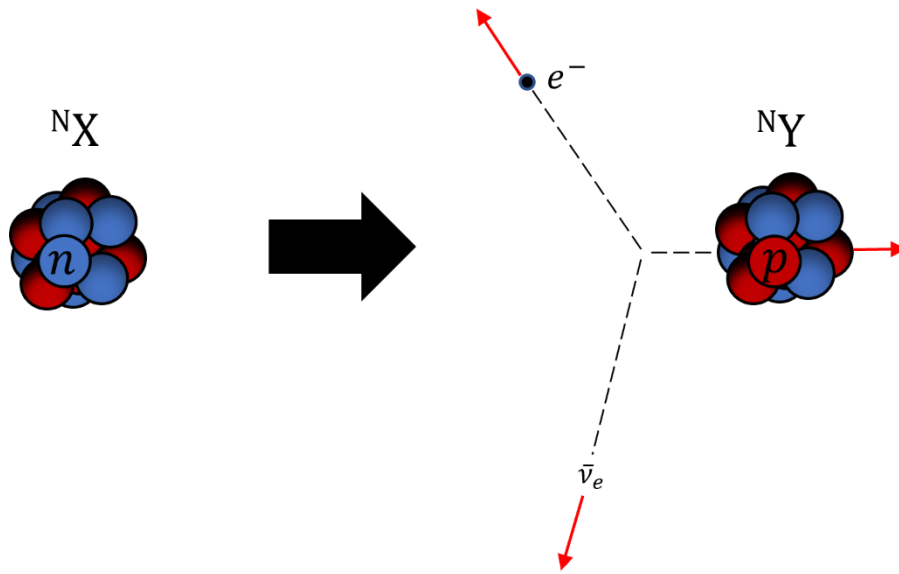


Figure 3. Depiction of beta decay. (Left) The parent nucleus has  $N$  nucleons. Through  $\beta^-$  decay, the highlighted neutron decays into a proton, maintaining the mass number but increasing the atomic number of the nuclei by one, and in the process releasing an  $e^-$  and  $\bar{\nu}_e$ . (Right) The red arrows indicate that the decay products have momentum. The kinetic energy of the products can be measured to form a  $\beta^-$  spectrum.

In a decay similar to  $\beta^-$  decay,  $\beta^+$  decay is the process whereby a proton is converted to a neutron, emitting an anti-electron (positron) and an electron, as seen in Equation (3):

$$p \rightarrow n + e^+ + \nu_e, \quad (3)$$

where  $e^+$  is a positron and  $\nu_e$  is the electron neutrino. Both forms of  $\beta$  decay are facilitated by the weak nuclear force.

Electron capture is another example of weak decay in nuclei. During electron capture, a proton within the nucleus absorbs one of the atomic electrons and is converted into a neutron, resulting in the emission of an electron neutrino. Electrons in the s orbital are most likely to be subject to electron capture, while those in orbitals with higher angular momentum, such as p or d, are less likely.

$$p + e^- \rightarrow n + \nu_e. \quad (4)$$

See Table 1 for a more detailed description of the different conservation laws for beta decays and electron capture. In addition, charge conservation, baryon number, and lepton number are also conserved. The baryon numbers of nucleons are 1, while electrons and positrons have baryon numbers of 0. The lepton number of electrons and neutrinos is 1, while their antiparticles (positrons and antineutrinos) have a lepton number of -1; nucleons have a lepton number of 0.

Table 1: Conservation Laws in Different Types of Weak Decays

	$\beta^-$ Decay: $n \rightarrow p + e^- + \bar{\nu}_e$	$\beta^+$ Decay: $p \rightarrow n + e^+ + \nu_e$	Electron Capture: $p + e^- \rightarrow n + \nu_e$
Electric Charge	$0 = 1 - 1 + 0$	$1 = 0 + 1 + 0$	$1 - 1 = 0 + 0$
Baryon Number	$1 = 1 + 0 + 0$	$1 = 1 + 0 + 0$	$1 + 0 = 1 + 0$
Lepton Number	$0 = 0 + 1 - 1$	$0 = 0 - 1 + 1$	$0 + 1 = 0 + 1$

## 1.2. Decay Energy Spectra

### 1.2.1. Measuring Decay Energy Spectra

Decay Energy Spectroscopy (DES) is a measurement technique where a radioactive source is embedded inside an absorber, which is thermally coupled to a very sensitive thermometer. In DES, the total energy of decay products (i.e., X-rays, gamma rays, and kinetic energy of the daughter nucleus and electrons) are thermalized within the absorber. The thermalization of

the decay energy of a source internal to an absorber is depicted in Figure 4. The energy from all the decay products is thermalized within the time period of thermal diffusion in the detector, creating a single change in temperature for the entire decay energy, rather than for the energy of individual decay products. However, this excludes energy of radiation that escapes the absorber, such as neutrinos and high energy photons. In contrast, if the source were external to the absorber, only the energy of a single decay product (i.e., X-ray,  $\alpha$  particle, electron) would be thermalized.

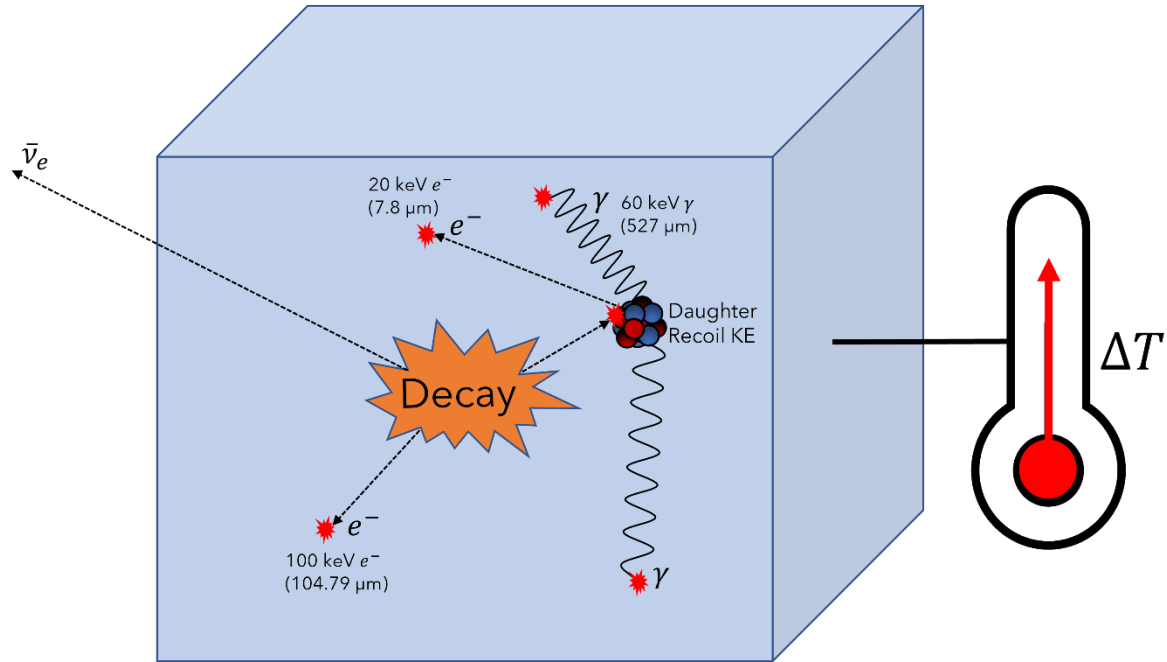


Figure 4. Schematic of the internal measurement mode for an ideal absorber. When the source decays, all energies of all decay products are thermalized by the absorber and converted to heat, causing the temperature to increase.

This change in temperature is measured with a microcalorimeter: a type of Low Temperature Detector (LTD). There are different types of microcalorimeters, such as transition-edge sensors (TES) or metallic magnetic calorimeters (MMC), but all are designed to measure extremely small changes in temperature by exploiting the electrical or magnetic properties of materials transitioning between superconducting and normal phases. When operated near the critical temperature of the phase transition, any small change to a material's temperature results in a measurable change in some other property, such as resistance for TESs or magnetization for MMCs. When this change is read out electronically,

it results in a pulse whose amplitude is proportional to the energy deposited from a single decay. A decay energy spectrum is a histogram of many of these pulses.

DES offers not only excellent energy resolution, but also has the advantage of measuring total decay energy independent of decay path. This results in a simplified spectrum, making analysis of total activity by isotope less prone to systematic bias, important for determining radionuclide composition. This effect is shown in Figure 5.

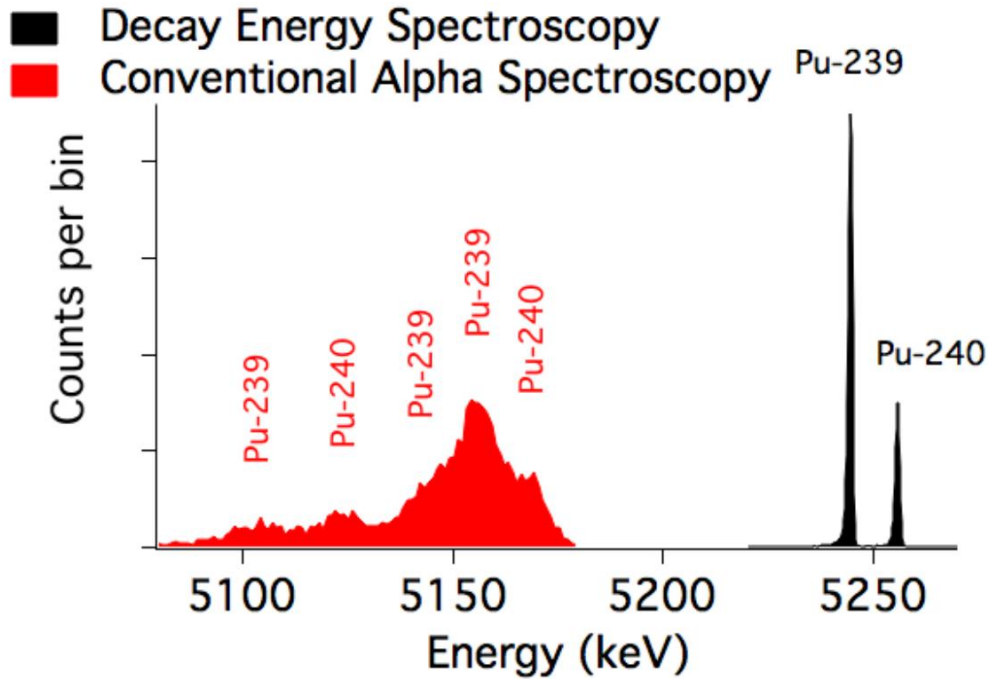


Figure 5. Comparison of different methods of spectroscopy for the alpha decay of  $^{239}\text{Pu}$  and  $^{240}\text{Pu}$ . The red region shows conventional alpha-particle spectroscopy, where only the energy of an emitted alpha particle is measured. The black region shows a DES measurement of each isotope, peaking at total decay energy for the alpha decay. Taken from Ref. [11].

### 1.2.2. Physical Measurements and Interpretations

Several DES projects measuring weak decays with MMCs have been developed. These include MetroMMC for exploring electron capture schemes [12] and MetroBeta for beta spectrum shape [13, 14]. MetroBeta studied four beta decaying radionuclides:  $^{151}\text{Sm}$  (1<sup>st</sup> forbidden non-unique;  $Q = 76.4$  keV),  $^{14}\text{C}$  (allowed;  $Q = 156.476$  keV),  $^{99}\text{Tc}$  (2<sup>nd</sup> forbidden non-unique;  $Q = 293.8$  keV), and  $^{36}\text{Cl}$  (2<sup>nd</sup> forbidden non-unique;  $Q = 709.53$  keV). Of these, just the beta spectrum of  $^{14}\text{C}$  has been measured (See Figure 6).

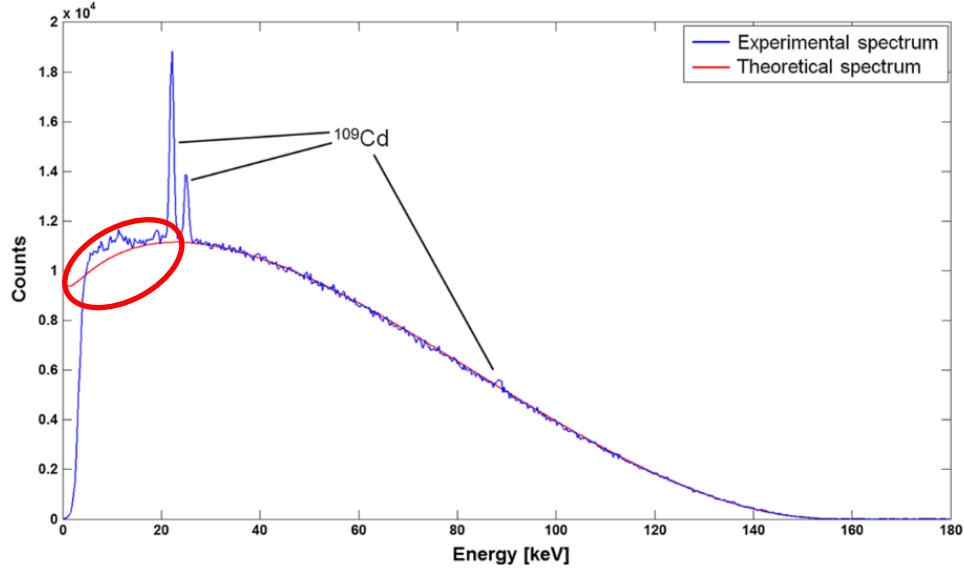


Figure 6. Beta spectrum of  $^{14}\text{C}$ , measured with an MMC (blue), and calculated spectrum using the code BetaShape (red). Three discrete photon lines from a  $^{109}\text{Cd}$  source external to the detector were used for energy calibration. The discrepancy between experimental and theoretical spectra in the low-energy region highlighted motivates further exploration into interpretation of physical microcalorimeter measurements. Figure taken from Ref. [13].

The energy resolution achieved (1 keV FWHM at 22 keV) was severely degraded by pile-up. Results showed good agreement with the theoretical beta spectrum, except in the low-energy range where most of the histogram bins were overpopulated compared to prediction. This discrepancy is possibly a result of the energy of certain events not being completely thermalized, resulting in a lower energy deposited in the detector. Other possibilities include imperfect heat transfer between the absorber and the microcalorimeter, resulting in nonlinear energy loss, or a lack of fidelity in the theoretical prediction of the beta spectrum for low energies. Determining the effect of energy escape on microcalorimeter measurements of  $\beta^-$  decay will enable more precise determinations of the  $\beta^-$  spectrum for the MetroBeta project and other beta spectroscopy experiments.

Analytically determining the probability of energy escape from an absorber is nearly impossible. This is because the escape probability is not constant as a function of initial decay energy but is dependent on the absorber's material and size, as well as the source's energy, type, location, and distribution. This work will simulate the energy deposition of beta

particles within an absorber using EGSnrc—a Monte Carlo simulation software—in order to quantify the escape probability and the goal of removing this effect from microcalorimeter measurements.

By iterating through monoenergetic electrons ranging from 10 keV to 2 MeV, the response matrix for an MMC detector can be constructed. This response matrix may be used to deconvolve the detector response from a DES measurement, which will reduce the systematic uncertainty in MMC measurements of beta spectra, particularly for low energies.

## Chapter 2

### DETECTORS

#### **2.1. Low Temperature Detectors**

The field of low temperature calorimetry and cryogenic detectors first emerged in 1984 out of a need for enhanced sensitivity from detectors, specifically to investigate fundamental issues in nuclear physics, such as the mass of the electron neutrino. When compared with semiconductor detectors, like High Purity Germanium(HPGe) or Silicon, LTDs have vastly superior energy resolutions (see Figure 7), making them a desirable choice for many experiments. The energy resolution ( $\Delta E$ ) of a detector is proportional to its temperature ( $T$ ) times the square root of its heat capacity ( $C$ ), as shown in Equation (5).

$$\Delta E \propto T\sqrt{C}, \quad (5)$$

This holds throughout the energy range a detector is applied to. While semiconductor detector energy resolution approaches a physical limit of 8-10 keV at full-width-half-maximum (FWHM) for 5 MeV alpha particles, which is a resolving power of around 500 [15], low temperature  $\alpha$  detectors yield resolutions less than 1 keV FWHM for 5.3 MeV alphas (resolving power of 5300) [16]—an order of magnitude better energy resolution. X-rays energy resolution of  $\Delta E_{FWHM} = 3.4$  eV at 6.5 keV has been demonstrated [17, 18], as has gamma energy resolution of 22 eV FWHM at 97.43 keV [19]. The high energy resolution of microcalorimeters can be exploited to study both electron capture and  $\beta^-$  decay [20,21,22], allowing for extremely precise measurements of the spectral shape.



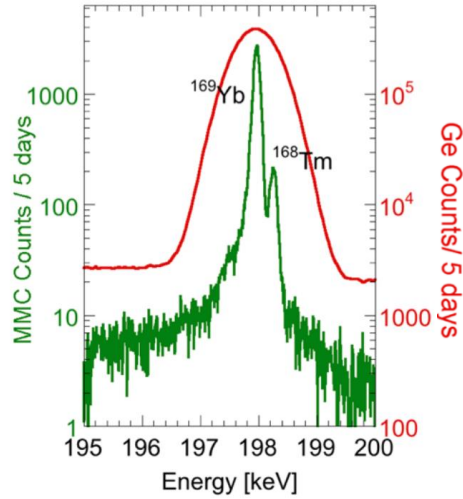


Figure 7. Comparison of resolving capabilities of a MMC (green) and a HPGe (red) gamma measurement. The microcalorimeter can resolve the gamma peaks from <sup>169</sup>Yb and <sup>168</sup>Tm, while the HPGe detector cannot. Taken from Ref. [23].

#### 2.1.1. Principles of Microcalorimeters

The detector is composed of three parts: an absorber, a calorimeter, and a heat sink [24]; This ideal system is shown in Figure 8. For any measurement to be made, a particle must deposit its energy in the absorber in the form of heat. The particle may either be incident upon the absorber, or the particle may originate from a decay event within the absorber. As the absorber thermalizes all of the particle's energy, the absorber's temperature increases. A thermal link to an extremely sensitive calorimeter allows for the heat of a single decay to be measured via the intrinsic temperature change in the absorber. This entire system is then weakly linked to a heat sink, allowing the absorber and calorimeter to slowly return to a baseline temperature. A good absorber has high stopping power to ensure that no energy from events escapes, and it has a low heat capacity so that there is a greater change in the absorber's temperature from the heat of a single event.

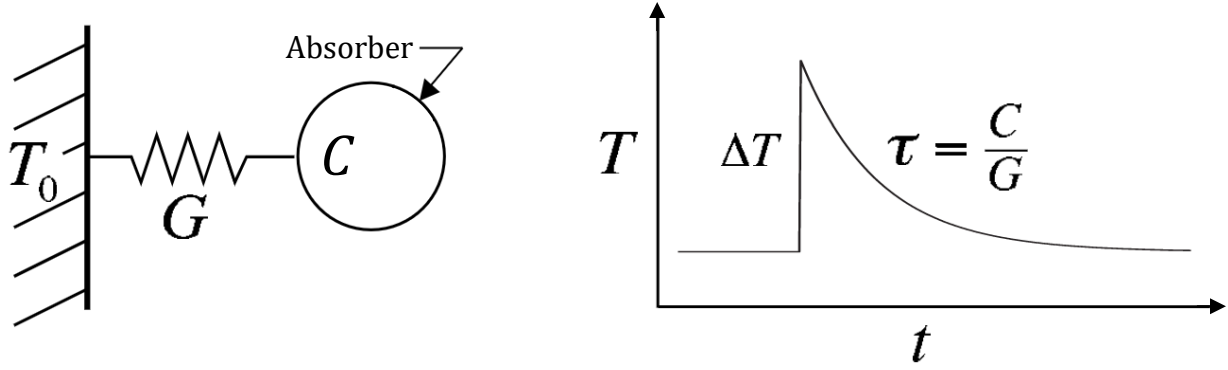


Figure 8. An ideal calorimeter. On the left, an absorber with heat capacity  $C$  is connected to a heat sink of temperature  $T_0$  via a thermal link with thermal conductivity  $G$ . On the right, an instantaneous input of energy  $E_0$  will raise the calorimeter's temperature by  $\Delta T = E_0/C$ , and it will then decay back to its initial temperature with a time constant  $\tau = C/G$ . Figure modified from Ref. [24].

### 2.1.2. Cryogenics

Both transition-edge sensors and metallic magnetic calorimeters require cryogenics for operation. The first reason is to reduce thermal noise in measurements, which increases with increasing temperature. The second is to ensure that the sensors remain in the superconducting phase. Any LTD design must make considerations for cryogenics to keep the system in an acceptable temperature range, usually less than 400 mK for a TES or less than 50 mK for a MMC. To ensure sufficient cooling, a variety of different cryostat systems have been used for LTDs, including  $^3\text{He}/^4\text{He}$  dilution refrigerators and adiabatic demagnetization refrigerators (ADR).

**Adiabatic Demagnetization Refrigerators:** An ADR cools by using the thermodynamic properties of paramagnetic materials in magnetic fields. Figure 9 depicts a simplified ADR design. When a paramagnetic solid, referred to as a “salt pill”, is placed in a strong magnetic field, the magnetic moments of the molecules in the pill align with the field, decreasing the entropy of the system. When the strength of the field decreases, the spins in the salt pill become more disordered and absorb heat to increase the entropy, resulting in the cooling of the pill.

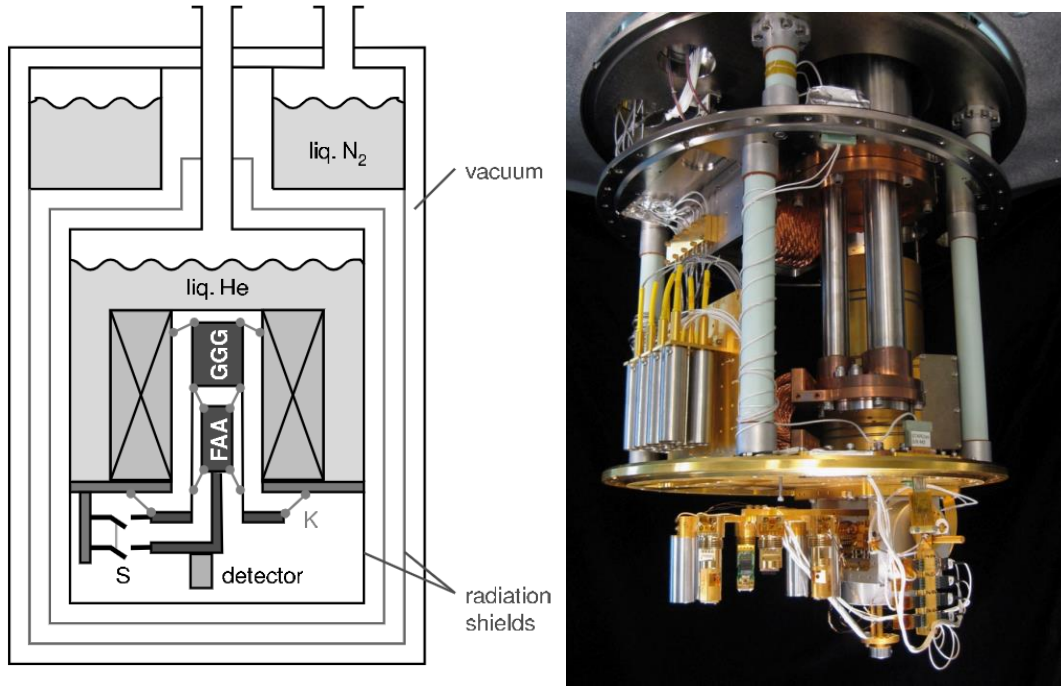


Figure 9. (Left) Schematic of an adiabatic demagnetization refrigerator (ADR). This commercial design uses both a liquid nitrogen and liquid helium bath to keep the detector at a temperature of 4.2 K. Two paramagnetic salt pills, GGG (Gadolinium-Gallium-Garnet) and FAA (Ferric-Ammonium-Alum), are in contact with the helium bath via heat switches (S). A detector is suspended from the FAA salt pill. Figure taken from Ref. [29]. (Right) Picture of ADR used for DES. The ADR is used to ensure that the LTDs operate in the correct temperature range.

An ADR works cyclically to cool the environment to temperatures less than 1 mK. The whole system is thermally shielded by a liquid N<sub>2</sub> bath and radiation shields throughout the cooling process. In the first part of the cycle, the paramagnetic salt pills are thermally isolated from the liquid H<sub>2</sub> heat sink and a magnetic field is applied to the solid. As the spins align in the salt and the entropy increases, the salt heats up. Then, the salt pills are thermally connected via switches to the heat sink and cooled down back close to their starting temperature. Finally, the thermal link to the bath is broken and the magnetic field is decreased. To increase in entropy, the salt pills absorb thermal energy from the experimental platform and detector, resulting in cooling of both.

**Dilution Refrigerator:** A <sup>3</sup>He/<sup>4</sup>He dilution refrigerator uses a mixture of <sup>3</sup>He and <sup>4</sup>He as the cooling agent. When a mixture of these isotopes is cooled beneath 870 mK, it undergoes a

phase separation, creating a concentrated phase of nearly pure  $^3\text{He}$  and a dilute phase of about 6.6%  $^3\text{He}$  and 93.4%  $^4\text{He}$  (see Figure 10).

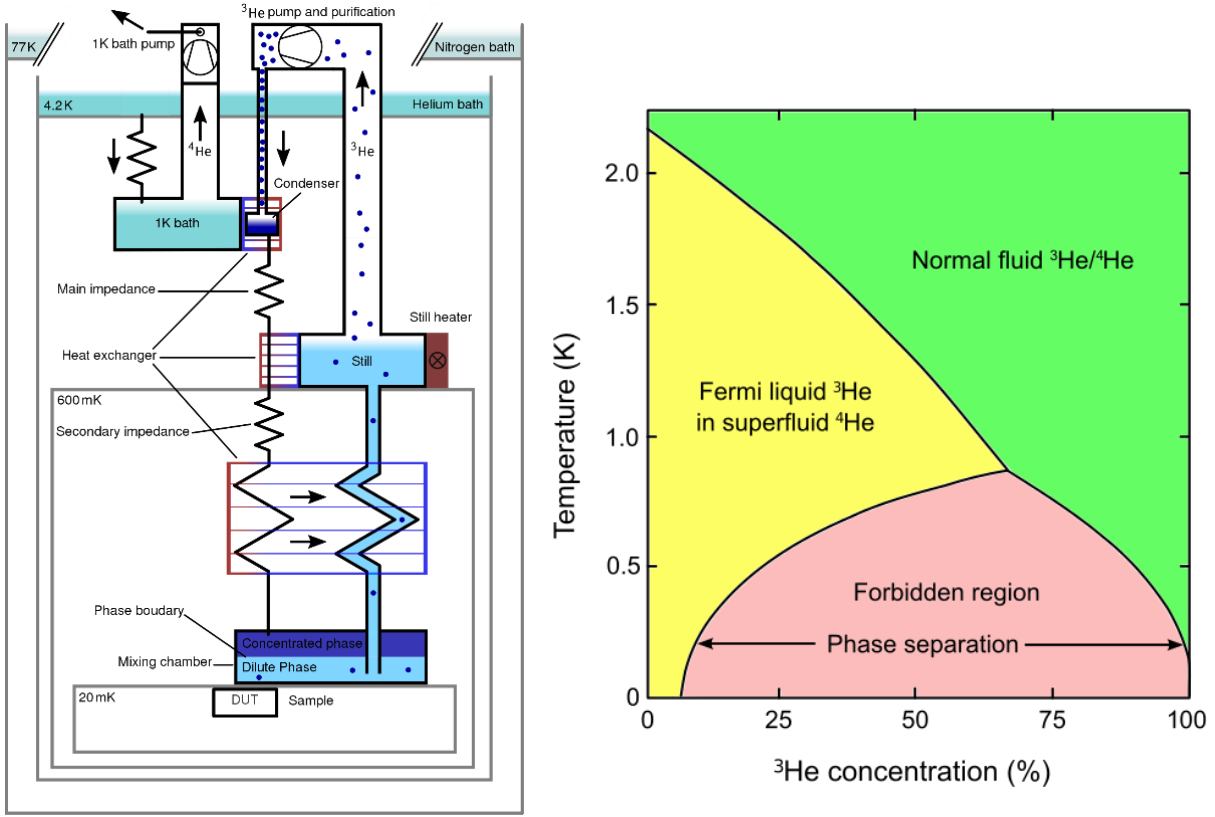


Figure 10. (Left) Schematic of a  $^3\text{He}/^4\text{He}$  dilution refrigerator [25]. (Right) Phase diagram of liquid  $^3\text{He}/^4\text{He}$  mixture. The phase separation begins at approximately 870 mK, leading to a concentrated phase of  $^3\text{He}$  (green region) and a dilute phase of 6.6%  $^3\text{He}$  and 93.4%  $^4\text{He}$  (yellow region). Fermi liquids and superfluids are the states of matter describing  $^3\text{He}$  and  $^4\text{He}$  at extremely low temperatures, respectively. This is another cryostat option to achieve the low temperatures necessary for DES measurements using LTDs. Taken from Ref. [26].

In the mixing chamber of the cryostat, these two phases are in equilibrium, creating a phase boundary. When  $^3\text{He}$  enters this chamber, it must cross the phase boundary and become diluted, an endothermic process that removes heat from chamber environment and serves as the primary method of the extreme cooling.

### 2.1.3. Transition-Edge Sensors

The first demonstration of using the superconducting phase transition for microcalorimetry was in 1941, when D. H. Andrews applied a current to a tantalum wire that had been cooled

down to its superconducting phase and was able to measure the resistance change in the material caused by infrared radiation [27]. The same researchers then measured the voltage pulses from bombarding a superconducting niobium nitride strip with alpha particles.

When certain materials are cooled below a critical temperature  $T_c$  that varies by material, they transition into a superconducting state with zero electrical resistance. This phase transition into superconductivity can be extremely sharp, creating a sort of edge, as seen in Figure 11.

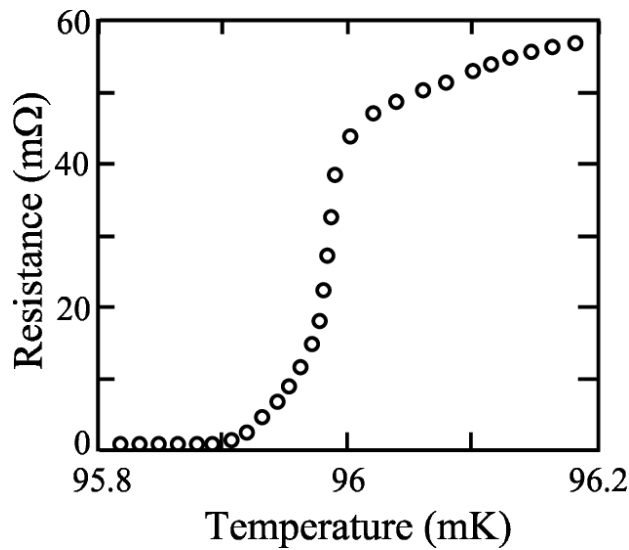


Figure 11. The superconducting to normal phase transition of a Mo/Cu film. Near 96 mK, a small change in temperature will cause a measurable change in the film's resistance, due to the fact that the transition is continuous. This is used in TESs to translate energy deposition to electronic pulses. Figure taken from Ref. [28].

The temperature control is done with a combination of temperature controls in the cryostat and either voltage or current biasing, which sets constant electronic operating conditions in the device.

One of the initial challenges in developing TES detectors was obtaining an accurate signal readout, especially in a device with such low impedance. Whenever a TES was connected to a current bias, joule heating would drive the detector out of its superconducting state, a process called positive electrothermal feedback. However, this issue was resolved by voltage biasing the TES. With a constant voltage, when the temperature of the absorber increases

from an energy deposition, the resistance goes up, lowering the current, causing the joule heating to go down. This returns the device to its equilibrium temperature, a process called negative electrothermal feedback. The change in current from an energy deposition is inductively coupled to a superconducting quantum interference device (SQUID) current amplifier. The current pulse through the TES coil changes its magnetic field and thus the input magnetic flux to the SQUID, whose output is easily amplified and measured (see Figure 12).

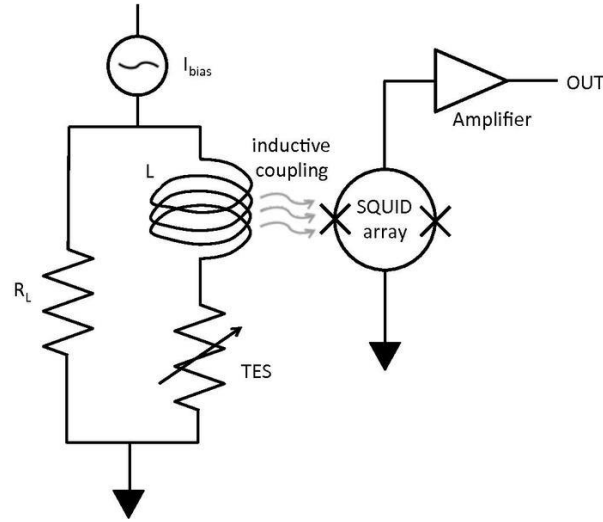


Figure 12. Electrical schematic of a TES coupled to a SQUID. The design is such that the TES is voltage biased by the current source,  $I_{bias}$ , and the load resistor  $R_L$ . Whenever an event is absorbed by the TES, the extra heat causes the TES resistance to increase and current to drop. This change in current is inductively coupled to a SQUID, whose output is then amplified even further.

The breakthrough of TES-SQUID coupling has made TES detectors a viable option for current experimentation. The biggest advantages in their use are the spectral detail and energy resolution. TES detectors are now being developed for measurements of various radiation, including alpha particles, beta particles, and photons ranging from eVs to MeVs of energy [16, 17, 28]. TESs used for DES measurements of  $\alpha$  decay have demonstrated excellent energy resolution, with  $\Delta E_{FWHM} = 7.5 \pm 0.2$  eV for 6539 eV [1].

#### 2.1.4. Metallic Magnetic Calorimeters

Another common type of low temperature detector is a metallic magnetic calorimeter (MMC). Rather than dealing with the electrical properties of superconducting materials,

MMCs use their magnetic properties. These properties are very strongly dependent on temperature within many materials, and thus form a good candidate for precise thermometry. The idea of low-temperature magnetic calorimetry was first proposed in the thesis of W. Seidel, from the Technical University in Munich, in 1986 [29, 30]. Various experiments demonstrated the usage of magnetization in a dielectric material to measure small energy doses to an absorber. In 1993, it was proposed to use metallic host materials instead, because the stronger interactions between magnetic moments and conduction electrons in the metal meant the detector response would be faster. A typical MMC has a paramagnet contained within a small magnetic field. This is placed in strong thermal contact with an absorber. When energy is deposited in the absorber, it increases in temperature, which changes the magnetization of the paramagnet. This can be read by a SQUID magnetometer. The relationship between a change in magnetization  $\delta M$  caused by an energy absorption of amount  $\delta E$  is

$$\delta M = \frac{\partial M}{\partial T} \delta T = \frac{\partial M}{\partial T} \frac{\delta E}{C_{\text{TOT}}}, \quad (6)$$

where  $C_{\text{TOT}}$  is the total heat capacity of the absorber and the paramagnetic thermometer. MMCs have demonstrated excellent energy resolution, with  $\Delta E_{FWHM} = 3.4$  eV for X-ray energies up to 6.5 keV [18] and  $\Delta E_{FWHM} = 340$  eV for 122 keV gamma rays emitted from a  $^{57}\text{Co}$  source [29]—on par with other types of low temperature detectors such as the TES.

## Chapter 3

### SIMULATIONS FOR A DETECTOR RESPONSE MATRIX

#### **3.1. *EGSnrc***

The Electron Gamma Shower (EGSnrc) system is a software application designed for the Monte Carlo simulation of transport of different types of ionizing radiation through a user-defined geometry. The Monte Carlo technique is a widely used scientific tool for problems that are often too difficult to solve analytically. The Monte Carlo method can require high computational power, and the accuracy of results is dependent on the input parameters, but in certain cases where physical experimentation is too time-consuming or costly, Monte Carlo simulation provides a reasonable alternative. In this technique, particles are initialized with a certain energy and physical distribution. These particles are propagated probabilistically through the material, using libraries of cross sections for various interactions. For each interaction, the collision particles leave with an energy and direction chosen from distributions. This process is continued until all particles are either absorbed or leave the geometry of interest [31].

EGSnrc is capable of simulating electrons, photons, and positrons with kinetic energies from 1 keV to several hundreds of GeV [32]. Radiation transport may be simulated within any element, compound, or mixture. There are two types of particle transport: charged and uncharged. Uncharged particle transport refers to the interactions taken by uncharged particles through a medium, such as electromagnetic radiation, while charged refers to the transport of charged particles, such as electrons and positrons. Because electrons and positrons differ only by having opposite charges, both are referred to as “electrons” for simplicity when discussing charged particle transport. EGSnrc has very high accuracy for electron and photon transport simulations based on cross-section data.

EGSnrc uses the physics of Compton scattering, electron-positron pair production, Rayleigh scattering, and photoelectric effect for photon transport, and it uses inelastic collisions and radiative energy loss for electron transport. One limitation of the software is the ability to



track electron transport event-by-event. In cases where electrons have high kinetic energy, each simulated particle undergoes hundreds of thousands of interactions with the surrounding atoms during the process of slowing down. Modern computing power is insufficient to completely track every event that occurs for one electron, much less the thousands of initial histories that are simulated. The solution employed by EGSnrc is called the “condensed history” technique [33], where the cumulative effect of large numbers of these electron transport and collision processes are condensed into a single “step”. The state of subsequent steps is determined by calculating the particle’s total change in energy, velocity, and position. This technique makes Monte Carlo simulation for a charged particle’s transport possible [34].

Other software packages exist that also employ a Monte Carlo method to simulate particle transport, such as MCNP, GEANT4, and PENELOPE. EGSnrc and PENELOPE have comparable accuracy for particle transport [35], while GEANT4 can achieve similar agreement under certain parameters, but needs further improvement to its modelling.

### **3.2. *The Physics of Particle Transport Simulation***

#### **3.2.1. Photon Interactions**

Photons interact with matter via four processes: pair production, Compton scattering, photo-electric absorption, and Rayleigh scattering. All these are used for the simulation of uncharged particle transport.

**Pair Production:** Pair production is the process whereby an electron-positron pair is created when a photon passes through the electromagnetic field created by atomic nuclei and surrounding electrons in the medium, as seen in Figure 14. The energy of the photon ( $E$ ) can be converted into particle mass ( $m$ ) described by Einstein’s equation

$$E = mc^2, \tag{7}$$

using the speed of light ( $c$ ). From this mass-energy relation, it is clear that for pair production to occur, the photon must have energy higher than the sum of the rest mass energies of the electron-positron pair. Since both the electron and positron have a rest mass of 0.511 MeV,

the photon must have an energy of above 1.022 MeV. Once above this energy threshold, pair production dominates over other photon interactions, as shown in Figure 13

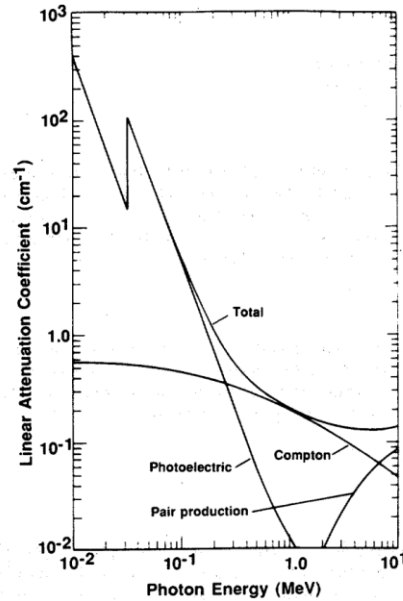


Figure 13. Linear attenuation coefficient of as a function of photon energy for NaI. The total attenuation coefficient has contributions from pair production, Compton scattering, and photoelectric absorption. Notice how pair production dominates at higher energies, while photoelectric absorption dominates at low energies. Taken from Ref. [36].

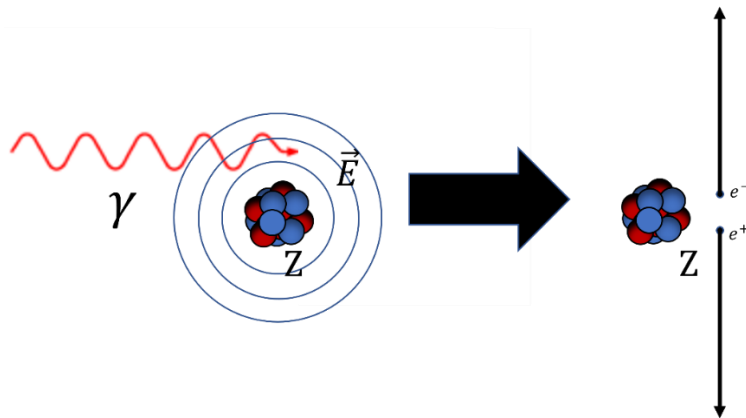


Figure 14. Diagram depicting the process of electron-positron pair production. As a photon with sufficient energy passes through the electromagnetic field of an atom, it can be converted into an electron-positron pair with nearly collinear velocities. The energy of the products is deposited in an absorber during a DES measurement.

**Compton Scattering:** Compton scattering is a scattering process of photons with charged particles, usually electrons. In this interaction, the photon transfers some of its energy, resulting in a lower scattered photon energy and a recoil electron. Because the energy of a photon is inversely proportional to its wavelength, the change in energy can be described mathematically as a function of the scattering angle  $\theta$ , determined by:

$$\lambda' - \lambda = \frac{h}{m_e c} (1 - \cos \theta), \quad (8)$$

where  $\lambda$  is the photon's initial wavelength,  $\lambda'$  is the scattered wavelength,  $h$  is Planck's constant,  $m_e$  is the rest mass of an electron, and  $c$  is the speed of light. The process is shown in Figure 15.

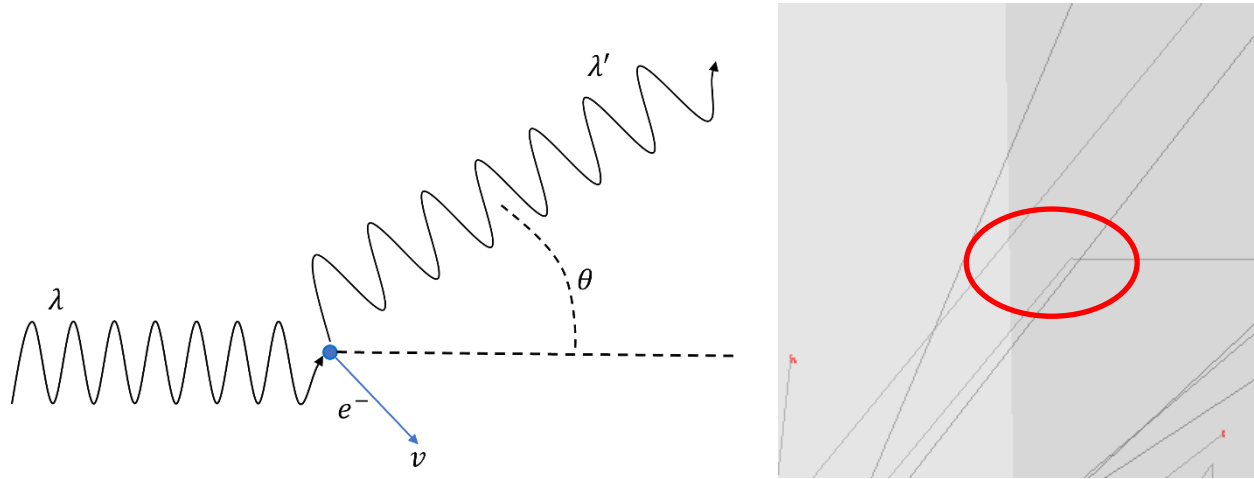


Figure 15. (Left) Depiction of Compton Scattering. An incident photon with wavelength  $\lambda$  interacts with an electron, scattering at an angle  $\theta$  with some larger wavelength  $\lambda'$  while the electron conserves momentum by recoiling at some velocity  $v$ . (Right) An EGSnrc simulation shows a photon undergoing Compton scattering and depositing energy within an Au absorber in the highlighted section.

**Photoelectric Absorption:** In the photoelectric effect, photons transfer all their energy to atomic electrons. When an incident photon to a material has more energy than the binding energy of an electron to that material, then the electron is likely to be ejected. Any additional energy the photon has above the binding energy of the electron becomes the electron's kinetic energy (see Figure 16).

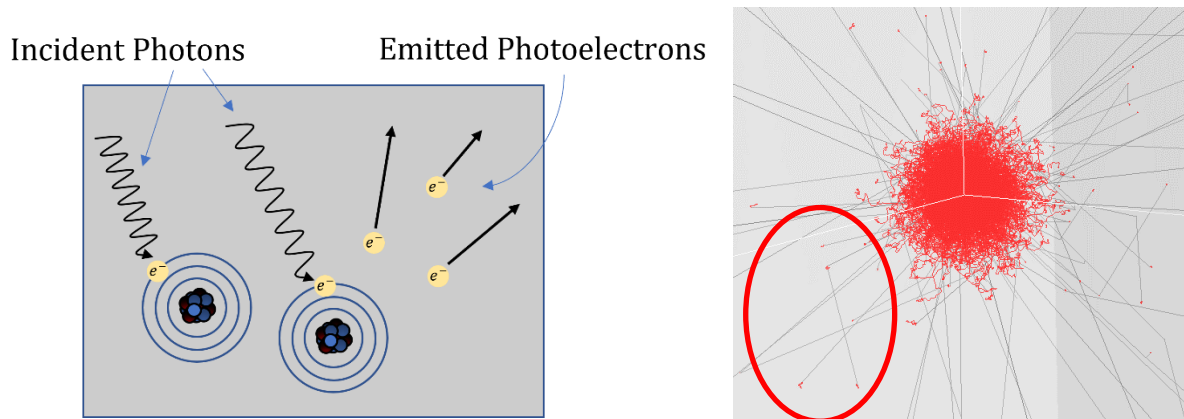


Figure 16. (Left) Depiction of the photoelectric absorption. Photons incident upon atomic electrons are absorbed if their energy is greater than the binding energy of the electron. The electrons are then freed from the atom and are emitted. Within an absorber, the electrons then deposit their energy. (Right) An EGSnrc simulation shows a photon undergoing photoelectric absorption within an Au absorber in the highlighted section. Notice how the black photon track ends in a red electron track, and the electron deposits its energy in the absorber.

This phenomenon is primarily dominant for low-energy photons.

**Rayleigh Scattering:** Rayleigh scattering is a form of elastic scattering of light by particles much smaller than the incident wavelength. Because the collision is elastic, the overall effect is to simply change the direction of the incident light while its energy remains approximately the same. The amount of scattering is inversely proportional to the fourth power of the wavelength of the photon. This is the only of the four mentioned photon interactions that does not directly transfer photon energy to electrons.

### 3.2.2. Electron Interactions

Electrons lose energy as they traverse a medium via two processes: radiative energy loss and inelastic collisions with atomic electrons. The radiative energy loss primarily occurs by bremsstrahlung and positron annihilation, while inelastic electron collisions with atomic electrons lead to atomic excitation and ionization, which can then result in the emission of x-rays and electrons during atomic de-excitation.

**Radiative Energy Loss:** As electrons traverse through a material, radiative energy loss occurs primarily via two processes: bremsstrahlung and positron annihilation.

Bremsstrahlung is electromagnetic radiation that is produced by the deceleration of an electron by interacting with the electric fields of other charged particles, such as other electrons or an atomic nucleus. Deceleration causes a photon to be emitted with energy equal to the loss in kinetic energy of the initial electron after interacting with a charged particle. Bremsstrahlung is the dominant mechanism of electron energy loss at high energies, and transfers this energy from electrons back to photons. Figure 17 represents the Bremsstrahlung process.

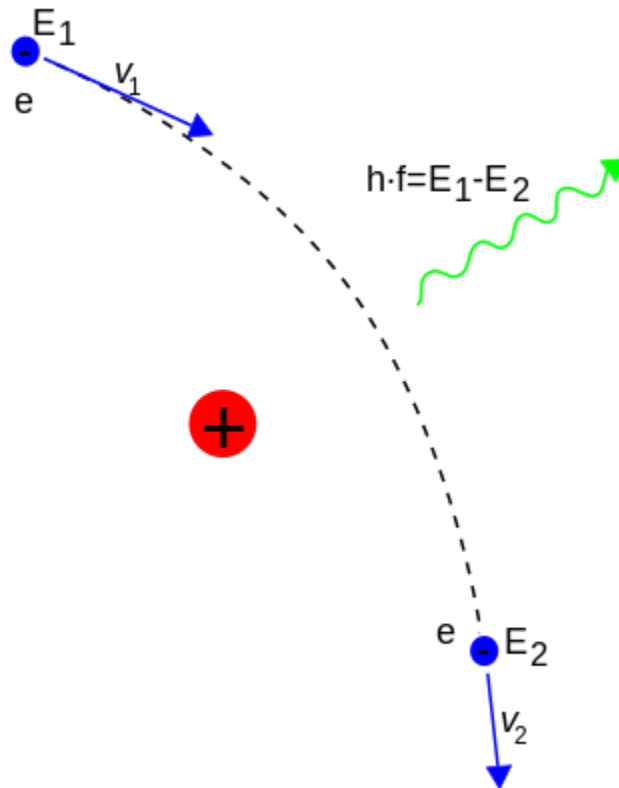


Figure 17. Depiction of electron energy loss via Bremsstrahlung. The initial electron with energy  $E_1$  passes by and decelerates in the electric field of the charged nuclei. The electron loses kinetic energy and continues with energy  $E_2$ , while a photon is emitted with energy  $E = hf = E_1 - E_2$  to conserve energy.

Electron-positron annihilation can be viewed as the reverse process of pair-production. In the annihilation process, a collision between an electron and its anti-particle (the positron) results in their destruction and the emission of energetic photons. Due to the laws of energy and momentum conservation, the creation of a single photon is forbidden. In the most

common case, two photons are created, each with energy equal to the rest mass of an electron (0.511 MeV) and emitted in opposite directions.

**Inelastic collisions:** This type of interaction leads to various excitations and ionizations of atoms along the path of the electron. With any inelastic collision, the electron transfers some of its energy to the collided particle; this can take the form of exciting inner-shell atomic electrons to higher energy levels or ionizing atomic nuclei. Ionized electrons traverse the material and interact via the processes described above, while highly excited atoms deexcite via the emission of photons and electrons with characteristic energies.

### **3.3. *Response Matrix***

Measuring beta spectra using low-temperature detectors yields very high-resolution results, but there are still systematic uncertainties that can be accounted for. Energy escape is an unavoidable reality of DES, and fractions of the initial energies of beta particles emitted inside the absorber will not be thermalized. Microcalorimeter absorbers are designed to be large enough to ensure that all beta particles are stopped, but the mechanisms of electron transport (i.e. bremsstrahlung, electron-positron annihilation) within the absorber often lead to the creation of photons, which are much more likely to escape. Electron attenuation within any absorber is theoretically 100%, but photons, especially high energy photons, have a lower attenuation, as shown in Figure 18, and withhold decay energy from being thermalized.

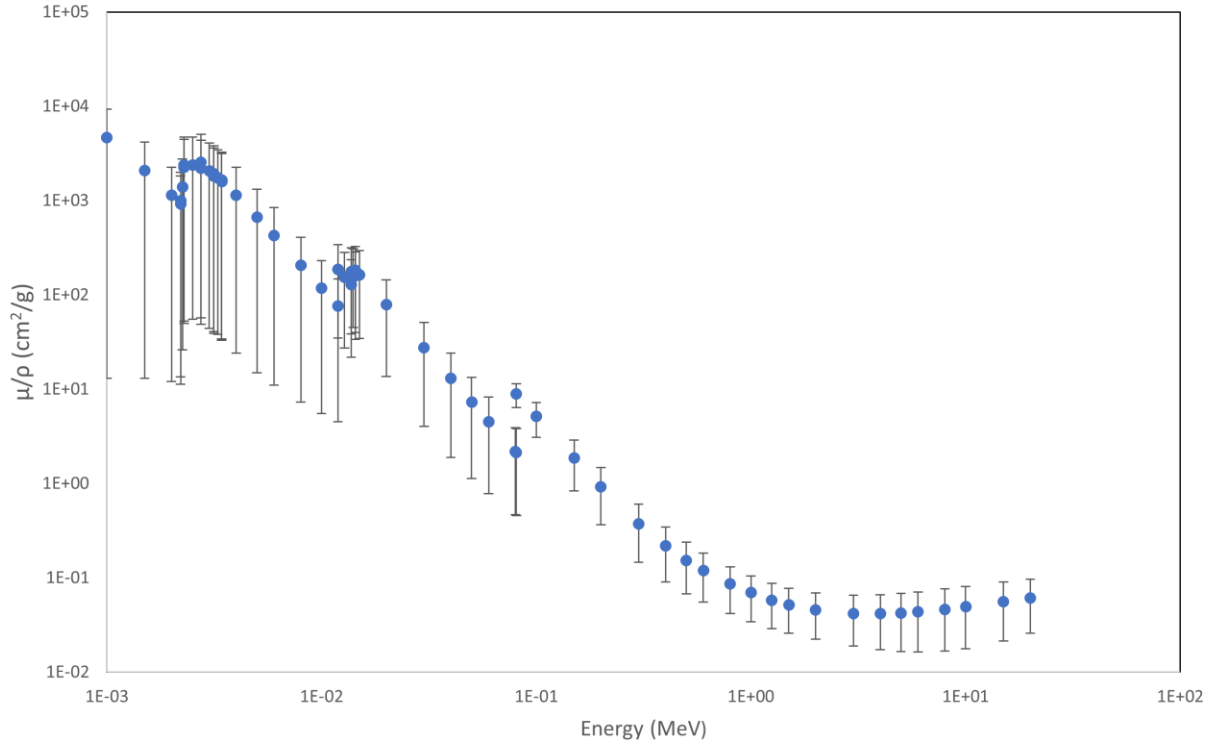


Figure 18. Plot of X-ray mass attenuation coefficients as a function of energy for Au. Higher energy photons have a much lower attenuation coefficient (3–4 orders of magnitude) than ones with low energy. Data from Ref. [37].

The overall effect is beta particles with defined initial energies from a radioactive source are detected with reduced energies. This leads to a systematic skew in a measurement of a beta spectrum histogram, where lower energy bins are overpopulated, and higher energy bins are underpopulated. This systematic energy loss can be accounted for by using EGSnrc. The power behind using this MC software is the ability to accurately simulate the energy deposition of beta particles within a defined absorber's geometry, thereby quantifying the escape probability. The information garnered from the EGSnrc simulations is used to construct the detector's response, which would be used to account for the systematic loss in energy stemming from interactions within the detector itself and thus give a more accurate measurement of the beta spectrum.

### 3.3.1. Theory Behind Response Matrix

One way to describe the spectrum of a beta source is with a column vector  $\mathbf{D}_N = \begin{pmatrix} D_1 \\ D_2 \\ \vdots \\ D_N \end{pmatrix}$ ,

where  $N$  is the number of energy bins over a defined energy range, and  $D_j$  is the counts of the  $j^{\text{th}}$  bin of the energy histogram. The vector  $\mathbf{D}_N$  is the true beta spectrum, representing the total decay energy of a given source without any energy escape. Define  $\mathbf{M}_N$  as the measured beta spectrum with the same length as  $\mathbf{D}_N$ . Each  $M_i$  is from experimental measurements with real microcalorimeters. The response matrix for the detector is given by the  $N \times N$

matrix  $\mathbf{R}_N = \begin{pmatrix} R_{11} & \cdots & R_{1N} \\ \vdots & \ddots & \vdots \\ R_{N1} & \cdots & R_{NN} \end{pmatrix}$ , where any value  $R_{ij}$  is proportional to the probability of the

events in the  $j^{\text{th}}$  bin of  $\mathbf{D}_N$  being measured in the  $i^{\text{th}}$  energy bin of  $\mathbf{M}_N$ , such that

$$\mathbf{R}_N \mathbf{D}_N = \mathbf{M}_N. \quad (9)$$

To find the matrix elements of  $\mathbf{R}$ , monoenergetic electrons are simulated inside an absorber, and the energy deposition inside the absorber is recorded as a histogram. In doing so, the true beta spectrum  $\mathbf{D}_N$  is known, since all electrons begin with the same energy.  $\mathbf{D}_N$  will take

the form of  $\mathbf{D}_N = \begin{pmatrix} 0 \\ \vdots \\ n \\ \vdots \\ 0 \end{pmatrix}$ , where  $n$  is the number of electrons simulated and the value of  $j$

depends on the initial electron energy. The histogram of the electron energy deposition inside the absorber is calculated with EGSnrc, and it is proportional to measured beta spectrum  $\mathbf{M}_N$ . The resulting histogram has heights corresponding the percentage of electrons simulated in a specific bin, rather than total counts, so each height must be multiplied by  $n$  to yield  $\mathbf{M}_N$ . With both  $\mathbf{D}_N$  and  $\mathbf{M}_N$  known, each element of the response matrix  $\mathbf{R}$  is defined to be

$$R_{ij} = \frac{M_i}{D_j}. \quad (10)$$

Once the response matrix  $\mathbf{R}_N$  is determined, assuming it is invertible,



$$\mathbf{R}_N^{-1} \mathbf{R}_N \mathbf{D}_N = \mathbf{D}_N = \mathbf{R}_N^{-1} \mathbf{M}_N, \quad (11)$$

This means that, with the inverse of the detector response matrix, the true beta spectrum can be obtained by multiplying the inverse matrix onto the measured spectrum. This will remove the detector response from a spectrum and account for one of the main systematic uncertainties involved in this type of measurement at low energies [38].

### **3.4. Assumptions**

The proposed methodology is first based on the following assumptions:

1. The physics within EGSnrc is an accurate representation of particle transport within an MMC absorber.
2. The time period of energy deposition for any particle's specific decay path is short compared to the measurement time.
3. Any measured decay is independent from all other events. This mean that no memory or pile-up issues are factored into the simulation parameters.

While assumptions 2 and 3 pose real issues for any DES measurement, the largest source of systematic uncertainty comes from the incomplete absorption of all decay radiation energy, and is why the Monte Carlo simulation approach lends itself as a viable process to obtaining the detector's response.

## Chapter 4

### THE ANATOMY OF A SIMULATED SPECTRUM

To account for the many factors contributing to energy escape from an absorber, EGSnrc is used to calculate the escape probabilities via the Monte Carlo method. Monoenergetic beta particles are simulated within an absorber, and the energy deposition within the various regions of the simulation environment is recorded. Creating 1000s of monoenergetic simulations across an entire range of energies yields the escape probability of an entire beta spectrum. This information can be used to unfold energy loss that comes from escape.

#### **4.1. *Simulation Parameters***

##### 4.1.1. Deconstruction of an Input File

EGSnrc is used to simulate the source, absorber, and transport of particles within an absorber. The user controls the geometry of the source, its location, and the type of radiation emitted, as well as the geometry, composition, and location of the absorber. The parameters for particle transport are largely determined by distributions internal to the software, but limits can be placed on maximum and minimum particle energies. To replicate real beta spectroscopy measurements, beta-decay point sources are simulated inside an absorber and all energy deposited within the geometry of the absorber is recorded. The weights of this information is its scoring.

EGSnrc input files are broken into a series of input blocks, each of which is responsible for defining parameters relating to the simulation.

**Run Control:** This input block determines how many histories to initialize in the simulation, which is the number of radioactive source particles used. For example, to simulate one million histories, the following block is used:

```
:start run control:
  ncase = 1e6
:stop run control:
```

The computational time of the simulation increases as the number of histories is increased and as the initial energy of each history increases. The full simulation time for 0.25 MeV electrons simulated in an Au absorber was 5 minutes, but increased to 35 minutes when the energy was raised to 1.00 MeV.

**Geometry Definition:** The geometry input block determines the size and shape of any objects within the simulation. Two types of geometry must be defined: source geometry and phantom geometry. The source geometry defines the shape of the radioactive source, while the phantom geometry defines the volume within which energy deposition will be scored. The following blocks were used to define the phantom (Au absorber) and source geometries:

```
:start geometry definition:
  # Define phantom geometry (Au absorber)
  :start geometry:
    name = absorber
    library = egs_ndgeometry
    type = EGS_XYZGeometry
    x-slabs = -0.03, 0.06, 1
    y-slabs = -0.03, 0.06, 1
    z-slabs = -0.03, 0.06, 1
    :start media input:
      media = Au
    :stop media input:
  :stop geometry:

  # Define the source geometry (point source)
  :start geometry:
    name = seed
    library = egs_spheres
    midpoint = 0 0 0
    type = EGS_cSpheres
    radii = 0.0001
    :start media input:
      media = AIR_TG43_LD
      set medium = 0 0
    :stop media input:
  :stop geometry:

  phantom geometries = absorber
  source geometries = seed

:stop geometry definition:
```

**Media Definition:** This input block defines the materials for each volume. Multiple media can be defined in this input block. Media are defined by calling material data files that contain default mass densities and density correction information. For example, many different media are all defined in the material.dat file, as shown in the following input block:

```
:start media definition:
  material data file = /egs_home/egs_brachy/lib/media/material.dat
:stop media definition:
```

Once the media have been defined in an input file, individual geometry regions are assigned a medium within the geometry definition input blocks. For example, within the definition of the phantom geometry, the geometry was defined to be gold:

```
:start geometry:
  name = absorber
  library = egs_ndgeometry
  type = EGS_XYZGeometry
  x-slabs = -0.03, 0.06, 1
  y-slabs = -0.03, 0.06, 1
  z-slabs = -0.03, 0.06, 1
  :start media input:
    media = Au
  :stop media input:
:stop geometry:
```

**Source Definition:** The source definition input block determines the characteristics of the radioactive source in the simulation. The user can create five types of sources: collimated, Fano, isotropic, parallel beam, and point. A collimated source only irradiates in a certain defined area or solid angle. Fano sources deliver particles proportional to the mass in the source region. An isotropic source emits particles uniformly distributed in a  $4\pi$  angle (and a point source is a special subset of isotropic sources that emit from a single point in space). Parallel beam sources emit particles uniformly in a single direction. Electron, positron, and photon sources can be created with different energy distributions using a spectrum input block nested within the source definition input block. For example, to define a 1 MeV monoenergetic electron point source:

```
:start source:
  library = egs_point_source
```

```

name = PointSource
charge = -1
position = 0 0 0
:start spectrum:
    type = monoenergetic
    energy = 1
:stop spectrum:
:stop source:

```

**Transport Parameters:** Global parameters for the simulation, including maximum and minimum energy cutoffs for photons and electrons are defined in this block. The parameters used are shown in the following input block:

```

:start MC transport parameter:
    Global ECUT                = 0.512
    Global PCUT                = 0.001
    Source ECUT                = 0.512
    Source PCUT                = 0.001
    Fluorescent Photon Cutoff  = 0.001
    Brems Cross Sections      = NRC
    Rayleigh Scattering        = On
    Electron Impact Ionization = On
:stop MC transport parameter:

```

**Scoring Options:** The user must define the mass attenuation coefficients as a function of energy (muen) for each material used in the simulation, which is done by calling a library of attenuation data for many different materials. This block is also used to score energy deposition in different regions of the simulation. This information is returned in the form of a histogram. For example, scoring energy deposition in a gold volume:

```

:start scoring options:
    # The path to a file containing mass-energy absorption data for
    # the relevant media in the simulation
    muen file = egs_home/egs_brachy/lib/muen/
    brachy_gold_1.5MeV.muendat

    muen for media = Au

    #Energy deposition histogram
    pulse height regions = 1 #Index of geometry region to score in
    pulse height bins = 100 #Number of energy bins

:stop scoring options:

```

Additional options for scoring different types of spectra can be chosen by the user, including the absolute counts of particles escaping the external surface of the source, an energy weighted spectrum of particles on the surface of the source, and photon fluence in a geometry region (See Appendix B).

**Ausgab Objects:** Ausgab definitions provide more options for additional outputs from a simulation. Different ausgab objects pulled from the EGS\_AusgabObject library provide different information for scoring. Two very useful ausgab objects are EGS\_TrackScoring [39] and EGS\_DoseScoring [40], which record the directional paths taken by every particle over the course of the simulation and record energy deposition in each geometry, respectively. EGS\_TrackScoring outputs particle track information to a separate file that is used to visualize both the geometries and the particles' transport through the material, as seen in Figure 19. To define these two ausgab objects:

```
:start ausgab object definition:
  ### Particle tracks
  :start ausgab object:
    name = tracks
    library = egs_track_scoring
  :stop ausgab object:
  ### Dose scoring
  :start ausgab object:
    library = egs_dose_scoring
    name = my_dose_scoring
    region dose = yes
    volume = 0.000216
    dose regions = 1
  :stop ausgab object:
:stop ausgab object definition:
```

The exact input files of the simulations used to create the response matrix can be found in Appendix A. In each simulation, a million monoenergetic beta particles (electrons) are emitted isotropically from a point source in the center of a  $0.6 \times 0.6 \times 0.6 \text{ cm}^3$  Au box, which represents the absorber of a microcalorimeter and is like actual absorber dimensions and composition used for real  $\beta$ -spectrum measurements [41,38]. This was simulated in the center of a  $30 \times 30 \times 30 \text{ cm}^3$  box of air. Energy deposition from all particle interactions was scored in all regions across the simulation environment i.e., within the point source, absorber, and surrounding air box. The total energy deposited in each region is returned in

the simulation output file. The EGSnrc codes `egs_track_scoring` and `egs_dose_scoring` were used to create the particle tracks and record energy deposition, respectively. The particle tracks are shown in Figure 19.

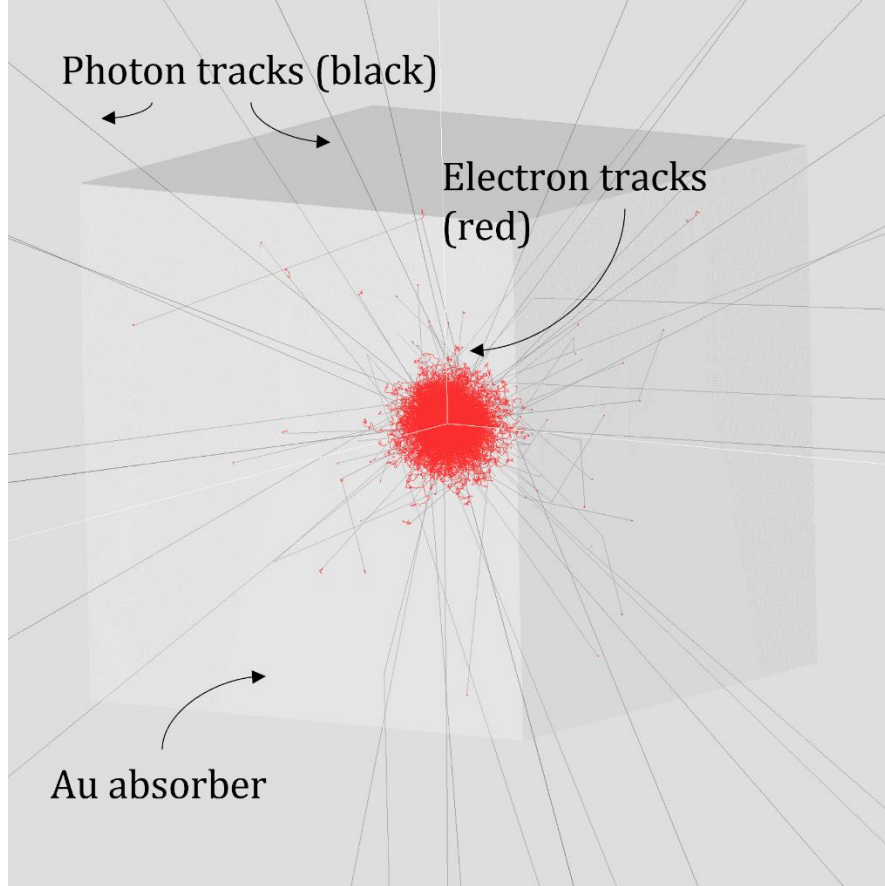


Figure 19. Particle tracks from an ENSnrc simulation of monoenergetic electrons travelling through an Au microcalorimeter absorber. The cube is the Au absorber, with dimensions of  $0.6 \times 0.6 \times 0.6 \text{ cm}^3$ . Within the absorber, the particle tracks of photons and electrons have been displayed. Black lines are photons, while red are electrons.

#### 4.1.2. Monoenergetic Simulations

The response matrix is constructed with a series of monoenergetic simulations. In each simulation, monoenergetic electrons are simulated within the absorber, and the total energy deposited by each of the simulated decays is recorded, resulting in a histogram corresponding to a measured energy spectrum corresponding to the decays with a single energy (see Figure 20).

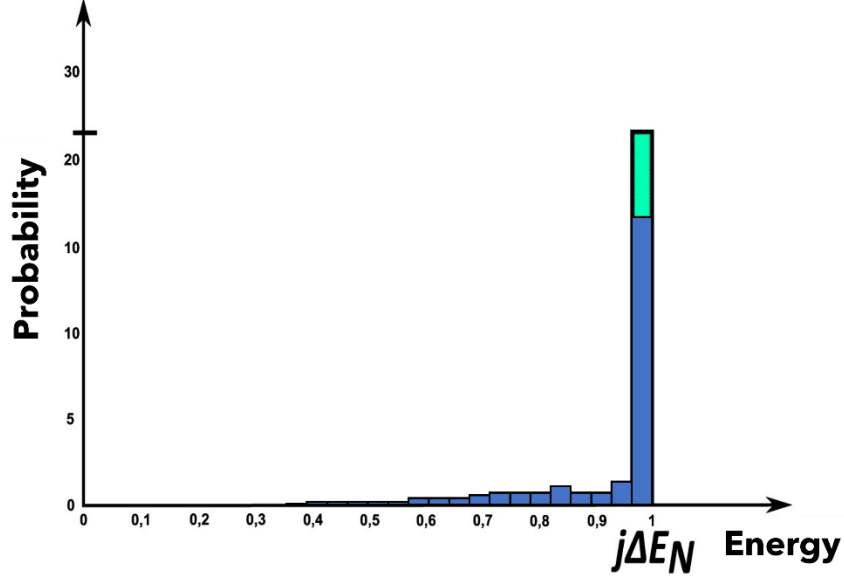


Figure 20. Histogram of energy escape of electrons from an Au absorber. The histogram of the initial energies of electrons in the simulation is shown in green; all the particles are within the bin  $j\Delta E$ , where  $j$  is the energy and  $\Delta E$  is the bin width. The histogram of energies output from the simulation is shown in blue. Energy escape means that not all the energy of each electron was deposited in the absorber, resulting in a spectrum shift. Figure modified from Ref. 38.

#### 4.2. Response Matrix

To construct the response matrix, monoenergetic simulations will be run across the energy range of 10 keV to 2 MeV, incrementally increasing by 10 keV. Finer incrementing between monoenergetic simulations creates a more continuous response matrix. Once created, the inverse response matrix will be numerically solved in Python. Having the inverse response matrix then allows for the detector response to be deconvolved from a measured DES spectrum, resulting in a new histogram that is closer to the true beta spectrum.



## Chapter 5

### BEGINNINGS OF A RESPONSE MATRIX

#### **5.1. Construction of Response Matrix**

When using DES to measure the total energy of a decay from an embedded radioactive source, some of the total decay energy is lost due to incomplete thermalization, largely from decay products escaping the absorber. This results in a DES spectrum with measured energies lower than the decay energy. The escape probability is not constant as a function of initial energy and depends on a variety of factors, including absorber material and source energy, type, location, and distribution. By working with EGSnrc to simulate varying monoenergetic beta sources within a microcalorimeter, this detector response can be quantified in the form of a matrix  $R$ . The value  $R_{ij}$  of a particular element of the response matrix is proportional to the probability that a decay will deposit that much energy in the absorber.

The tutor7pp code from EGSnrc is used to create a histogram of the energy deposition, which is written to a text file (.egslog). The initial energy of the electrons corresponds to the  $j$  index, the x-axis binning from the histogram corresponds to the  $i$  index, and the height of each bin corresponds to the value  $a_{ij}$  of the response matrix.

To explore the escape probabilities as a function of source energy, monoenergetic electron sources of 0.25 MeV, 0.50 MeV, 0.75 MeV, and 1.00 MeV were simulated using the tutor7pp program. For each, the number of bins in the histogram was adjusted so that the width of each bin would be the same across all four input energies. Figure 21 shows the resulting energy histograms created from these simulations.

##### 5.1.1. Simulation Analysis

To simplify the process of data analysis of the simulations, a python script (see Appendix C) was written to read the .egslog files for each simulation and save the energy deposition histogram.

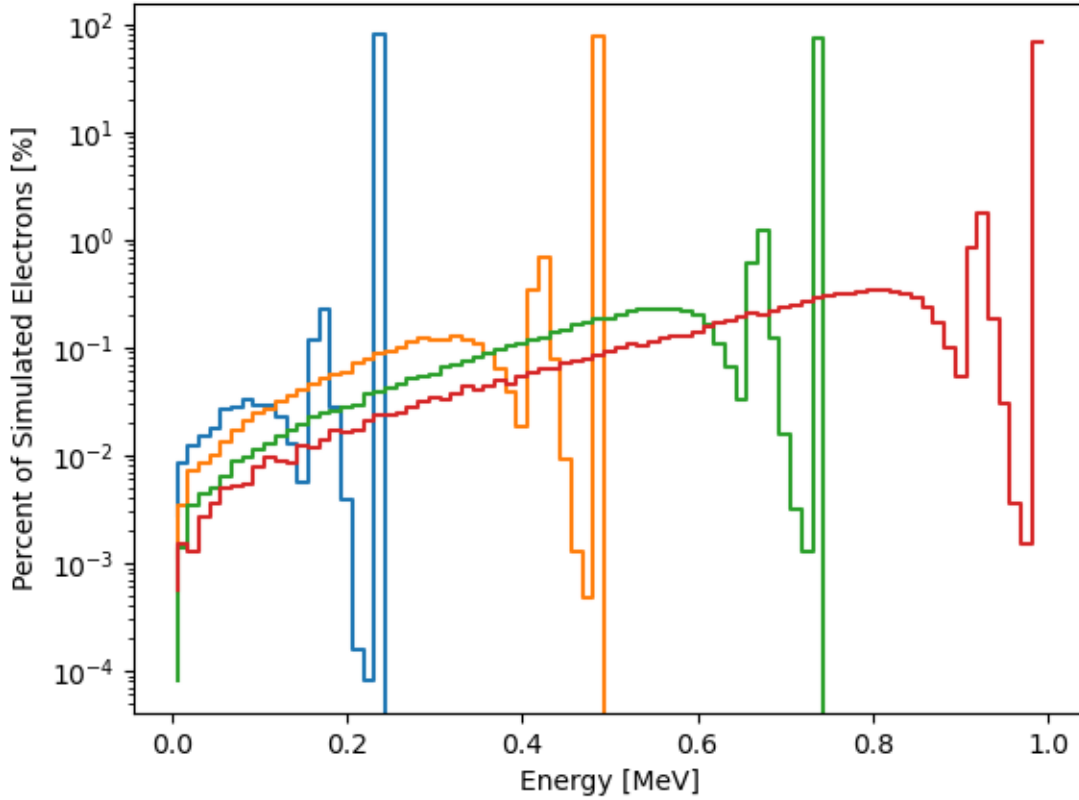


Figure 21. Four energy deposition histograms created from simulating various monoenergetic electron sources inside a  $0.6 \times 0.6 \times 0.6 \text{ mm}^3$  Au absorber. Blue is a 0.25 MeV source, orange is 0.50 MeV, green is 0.75 MeV, and red is 1.00 MeV. All histograms have the same binning width of 12.5 keV and have similar escape peaks.

Finer binning within an energy deposition histogram reveals structure within the energy escape probability. For example, the peak at 935 keV corresponds to electrons with 1 MeV of energy that have deposited only 935 keV. This occurs about 2% of the time, and it corresponds to gold fluorescent X-rays escaping. Gold  $K\alpha_1$  and  $K\alpha_2$  X-rays are 66.8037 keV and 66.9895 keV, respectively [42].

This process can be used to create a finely binned response matrix, by simulating the entire energy range of 10 keV to 2 MeV with 10 keV steps. This can be done by running a batch mode in EGSnrc.

### 5.1.2. Inverse of the Response Matrix

As was discussed in Chapter 2, the actual quantity of interest is not  $R$ , but  $R^{-1}$ , or the inverse of the response matrix. We can show that this is actually the desired matrix by showing:

$$R^{-1}R = I$$

It is this inverse matrix that can be used to deconvolve detector response from a measured beta spectrum. When the matrix operates on measured DES histogram, the detector response is accounted for, and the result is the true beta spectrum. The overall effect is a slight shift in the DES histogram away from lower energy bins that were overpopulated due to energy loss (see Figure 22).

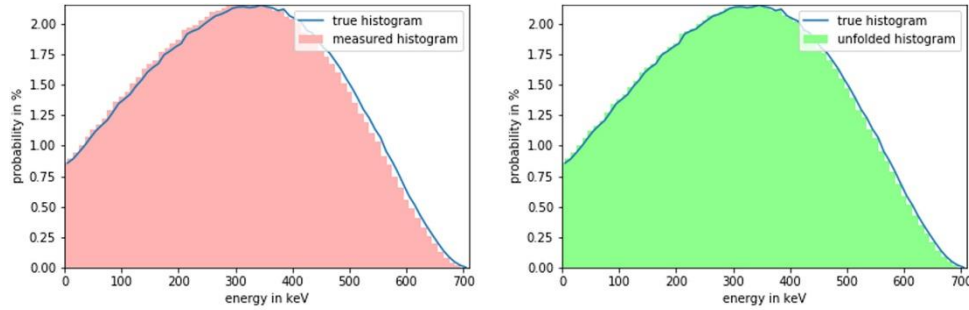


Figure 22. Comparison of  $^{36}\text{Cl}$  beta spectra with the simulated measured histogram (left) and the deconvolved histogram (right). Taken from Ref.38.

## Chapter 6

### DISCUSSION

#### **6.1. *Conclusions and Future Plans***

A method for determining the response matrix for DES absorber was proposed and initial results were presented for simulations of monoenergetic electrons for four energies (0.25, 0.50, 0.75, and 1.00 MeV) within a  $0.6 \times 0.6 \times 0.6 \text{ mm}^3$  Au absorber. Provided sufficiently fine binning and a large number of histories, a response matrix can be created to unfold the detector response from beta spectroscopy measurements, offering superior measurements and lower uncertainties.

Then, the response matrix could be applied to real DES beta spectra to see how the theoretical calculations compare to experimental measurements.

This tool can be used to study the effect of absorber materials and geometries on escape probabilities. Future studies should include uncertainty calculations of this approach and benchmarking the results with experimental data from a monoenergetic electron source.

## Appendix A

The following is the source code for the tutor7pp software used to run the simulations.

```
/*
#####
###
#
#   EGSnrc egs++ tutor7pp application
#   Copyright (C) 2015 National Research Council Canada
#
#   This file is part of EGSnrc.
#
#   EGSnrc is free software: you can redistribute it and/or modify it under
#   the terms of the GNU Affero General Public License as published by the
#   Free Software Foundation, either version 3 of the License, or (at your
#   option) any later version.
#
#   EGSnrc is distributed in the hope that it will be useful, but WITHOUT ANY
#   WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS
#   FOR A PARTICULAR PURPOSE. See the GNU Affero General Public License for
#   more details.
#
#   You should have received a copy of the GNU Affero General Public License
#   along with EGSnrc. If not, see <http://www.gnu.org/licenses/>.
#
#####
###
#
#   Author:          Iwan Kawrakow, 2005
#
#   Contributors:    Frederic Tessier
#
#####
###
#
#   A relatively simple EGSnrc application using the C++ interface. It
#   implements the functionality of the original tutor7 tutorial code written
#   in mortran except that now, due to the use of the general geometry and
#   source packages included in egspp, any geometry or any source can be used
#   in the simulation.
#
#   In addition, tutor7pp derives from the EGS_AdvancedApplication class and
#   therefore automatically inherits the ability to do restarted and parallel
#   simulations, to combine the results of parallel runs or to re-analyze the
#   results of single/parallel runs. It also inherits the ability to run for
#   a
#   user specified maximum amount of cpu time or to terminate the simulation
#   when a user specified uncertainty has been reached.
#
#
#   TERMINOLOGY
#   -----
#
```

```

# Simulations are split into 'chunks'. For simple simulations (no parallel
# runs, etc.) there is a single simulation chunk with the number of
# histories specified in the input file. For parallel runs the number of
# chunks and number of histories per chunk are determined by a 'run control
# object' (see below).
#
# Each simulation chunk is split into 'batches'. The batches are not required
# for statistical analysis (by using the provided scoring classes it is easy
# to have a history-by-history uncertainty estimation). Instead, simulation
# chunks are split into batches so that the progress of the simulation can
be
# reported after the completion of a batch and the current results can be
# stored into a data file. By default there are 10 batches per simulation
chunk
# but this can be changed in the input file.
#
# The simulation is controlled via a 'run control object' (RCO) The purpose
# of the run control object is to give to the shower loop the number of
# histories per simulation chunk, number of batches per chunk and to possibly
# terminate the simulation prematurely if certain conditions are met (e.g.
# maximum cpu time allowed is exceeded or the required uncertainty has been
# reached).
#
# egs++ provides 2 run control objects:
#
# 1) simple: the simple RCO always uses a single simulation chunk.
#
# 2) JCF: a JCF object is used by default for parallel runs
# JCF stands for Job Control File as this type of object
# uses a file placed in the user code directory to record
# the number of histories remaining, the number of jobs
# running, etc., in parallel runs. This is explained in
# more details in PIRS-877. A JCF object uses by default
# 10 simulation chunks but this can be changed in the
# input file.
#
# It is possible to use a simple control object for parallel runs by giving
# the -s or --simple command line option. In this case, each parallel job
# will run the number of histories specified in the input file but
# automatically adjust the initial random number seed(s) with the job index.
# This additional possibility has been implemented because several users
have
# reported problems with file locking needed for a JCF run control object.
#
# It is also possible to have other RCO's compiled into shared libraries and
# automatically loaded at run time (e.g., one could implement a RCO that
# communicates via TCP/IP with a remote server to obtain the number of
# histories in the next simulation chunk).
#
#
# USAGE
# -----
#
# - Geometry and particle source are specified in an input file as explained
# in PIRS-899 and PIRS-898.
#
# - Run control is specified in a section of the input file delimited by

```

```

#      :start run control: and :stop run control: labels.
#
# - A simple RCO is used for single job runs.
#
# - A JCF RCO is used by default for parallel runs, unless -s or --simple
#   is specified on the command line.
#
# - A simple RCO understands the following keys:
#   ncase                = number of histories to run
#   nbatch               = number of batches to use
#   statistical accuracy sought = required uncertainty, in %
#   max cpu hours allowed = max. processor time allowed
#   calculation          = first | restart | combine | analyze
#
#   All inputs except for ncase are optional (a missing ncase key will result
#   in a simulation with 0 particles).
#
# - A JCF object understands all the above keys plus
#   nchunk = number of simulation chunks
#
# - The simulation is run using
#
#   tutor7pp -i input_file -p pegs_file [-o output_file] [-s] [-P n -j i]
#
#   where command line arguments between [] are optional. The -P n option
#   specifies the number of parallel jobs n and -j i the index of this job.
#   On Linux/Unix systems it is more convenient to use the 'exb' script for
#   parallel job submission (see PIRS-877)
#
#####
###
*/
#include "egs_advanced_application.h"
#include "egs_scoring.h"
#include "egs_interface2.h"
#include "egs_functions.h"
#include "egs_input.h"
#include "egs_base_source.h"
#include "egs_rndm.h"
#include <cstdlib>
using namespace std;
class APP_EXPORT Tutor7_Application : public EGS_AdvancedApplication {
    EGS_ScoringArray *score;    // scoring array with energies deposited
    EGS_ScoringArray *eflu;    // scoring array for electron fluence at back
of geometry
    EGS_ScoringArray *gflu;    // scoring array for photon fluence at back
of geometry
    EGS_ScoringArray **pheight; // pulse height distributions.
    int nreg;    // number of regions in the geometry
    int nph;    // number of pulse height objects.
    double Etot; // total energy that has entered the geometry
    int rr_flag; // used for RR and radiative splitting
    EGS_Float current_weight; // the weight of the initial particle
that
    // is currently being simulated
    bool deflect_brems;
    EGS_Float *ph_de; // bin widths if the pulse height distributions.

```

```

    int          *ph_regions; // region indices of the ph-distributions
    static string revision;    // the CVS revision number
public:
    Tutor7_Application(int argc, char **argv) :
        EGS_AdvancedApplication(argc,argv),  score(0),    eflu(0),    gflu(0),
    pheight(0),
        nreg(0),    npg(0),    Etot(0),    rr_flag(0),    current_weight(1),
    deflect_brems(false) { };
    ~Tutor7_Application() {
        if (score) {
            delete score;
        }
        if (eflu) {
            delete eflu;
        }
        if (gflu) {
            delete gflu;
        }
        if (nph > 0) {
            for (int j=0; j<nph; j++) {
                delete pheight[j];
            }
            delete [] pheight;
            delete [] ph_regions;
            delete [] ph_de;
        }
    };
    void describeUserCode() const;
    int initScoring();
    int ausgab(int iarg);
    int outputData();
    int readData();
    void resetCounter();
    int addState(istream &data);
    void outputResults();
    void getCurrentResult(double &sum, double &sum2, double &norm,
                          double &count);
protected:
    int startNewShower();
};

string Tutor7_Application::revision = " ";
extern "C" void F77_OBJ_(egs_scale_xcc,EGS_SCALE_XCC) (const int *,const
EGS_Float *);
extern "C" void F77_OBJ_(egs_scale_bc,EGS_SCALE_BC) (const int *,const
EGS_Float *);
void Tutor7_Application::describeUserCode() const {
    egsInformation(
        "\n          *****"
        "\n          *                                     *"
        "\n          *                                     *"
        "\n          *                                     *"
        "\n          *                                     *"
        "\n          *****"
        "\n\n");
    egsInformation("This is Tutor7_Application %s based on\n"
        "          EGS_AdvancedApplication %s\n\n",
        egsSimplifyCVSKey(revision).c_str(),
        egsSimplifyCVSKey(base_revision).c_str());

```



```

}
int Tutor7_Application::initScoring() {
    // Get the number of regions in the geometry.
    nreg = geometry->regions();
    score = new EGS_ScoringArray(nreg+2);
    //i.e. we always score energy fractions
    eflu = new EGS_ScoringArray(200);
    gflu = new EGS_ScoringArray(200);
    // Initialize with no russian roulette
    the_egsvr->i_do_rr = 1;
    EGS_Input *options = input->takeInputItem("scoring options");
    if (options) {
        EGS_Input *scale;
        while ((scale = options->takeInputItem("scale xcc"))) {
            vector<string> tmp;
            int err = scale->getInput("scale xcc",tmp);
            //egsInformation("Found          'scale          xcc',          err=%d
tmp.size()=%d\n",err,tmp.size());
            if (!err && tmp.size() == 2) {
                int imed = EGS_BaseGeometry::getMediumIndex(tmp[0]) + 1;
                if (imed > 0) {
                    EGS_Float fac = atof(tmp[1].c_str());
                    egsInformation("\n ***** Scaling xcc of medium %d with
%g\n",imed,fac);
                    F77_OBJ_(egs_scale_xcc,EGS_SCALE_XCC) (&imed,&fac);
                }
            }
            delete scale;
        }
        while ((scale = options->takeInputItem("scale bc"))) {
            vector<string> tmp;
            int err = scale->getInput("scale bc",tmp);
            //egsInformation("Found          'scale          xcc',          err=%d
tmp.size()=%d\n",err,tmp.size());
            if (!err && tmp.size() == 2) {
                int imed = EGS_BaseGeometry::getMediumIndex(tmp[0]) + 1;
                if (imed > 0) {
                    EGS_Float fac = atof(tmp[1].c_str());
                    egsInformation("\n ***** Scaling bc of medium %d with
%g\n",imed,fac);
                    F77_OBJ_(egs_scale_bc,EGS_SCALE_BC) (&imed,&fac);
                }
            }
            delete scale;
        }
        vector<string> choices;
        choices.push_back("no");
        choices.push_back("yes");
        deflect_brems = options->getInput("deflect electron after
brems",choices,0);
        if (deflect_brems) {
            egsInformation("\n *** Using electron deflection in brems
events\n\n");
            setAusgabCall(AfterBrems,true);
        }
        int n_rr;
        if (!options->getInput("Russian Roulette",n_rr) && n_rr > 1) {

```

```

        the_egsvr->i_do_rr = n_rr;
        setAusgabCall(BeforeBrems,true);
        setAusgabCall(AfterBrems,true);
        setAusgabCall(BeforeAnnihFlight,true);
        setAusgabCall(AfterAnnihFlight,true);
        setAusgabCall(BeforeAnnihRest,true);
        setAusgabCall(AfterAnnihRest,true);
        //setAusgabCall(FluorescentEvent,true);
        egsInformation("\nUsing Russian Roulette with survival
probability 1/%d\n",n_rr);
    }
    // The user has provided scoring options input.
    // See where she/he wants to score a pulse height distribution
    // and how many bins to use for each pulse height distribution
    vector<int> regions;
    int err = options->getInput("pulse height regions",regions);
    vector<int> nbins;
    int err1 = options->getInput("pulse height bins",nbins);
    if (!err && !err1) {
        if (regions.size() != nbins.size() && nbins.size() != 1)
            egsWarning("initScoring(): you must input the same "
                "number of 'regions' and 'bins' inputs or a single
'bins'"
                    " input\n");
        else {
            EGS_ScoringArray **tmp = new EGS_ScoringArray* [nreg+2];
            for (int i=0; i<nreg+2; i++) {
                tmp[i] = 0;
            }
            for (int j=0; j<regions.size(); j++) {
                int nb = nbins.size() == 1 ? nbins[0] : nbins[j];
                if (nb < 1) {
                    egsWarning("zero bins for region %d?\n",regions[j]);
                }
                if (regions[j] < -1 || regions[j] > nreg) {
                    egsWarning("invalid region index %d\n",regions[j]);
                }
                if (nb > 0 && regions[j] >= 0 && regions[j] < nreg+2) {
                    int ij = regions[j];
                    if (tmp[ij]) egsInformation("There is already a "
                        "PHD object in
region %d => ignoring it\n",ij);
                    else {
                        tmp[ij] = new EGS_ScoringArray(nb);
                        ++nph;
                    }
                }
            }
            if (nph > 0) {
                pheight = new EGS_ScoringArray* [nph];
                ph_regions = new int [nph];
                ph_de = new EGS_Float [nph];
                EGS_Float Emax = source->getEmax();
                int iph = 0;
                for (int j=0; j<nreg+2; j++) {
                    if (tmp[j]) {
                        pheight[iph] = tmp[j];

```

```

        ph_regions[iph] = j;
        int nbin = pheight[iph]->bins();
        ph_de[iph++] = Emax/nbin;
    }
}
}
delete [] tmp;
}
}
else egsWarning("initScoring(): you must provide both, 'regions'"
               " and 'bins' input\n");
delete options;
}
return 0;
}
int Tutor7_Application::ausgab(int iarg) {
    if (iarg <= 4) {
        int np = the_stack->np - 1;
        // Note: ir is the region number+1
        int ir = the_stack->ir[np]-1;
        // If the particle is outside the geometry and headed in the positive
        // z-direction, change the region to count it as "transmitted"
        // Note: This is only valid for certain source/geometry conditions!
        // If those conditions are not met, the reflected and transmitted
        // energy fractions will be wrong
        if (ir == 0 && the_stack->w[np] > 0) {
            ir = nreg+1;
        }
        EGS_Float aux = the_epcont->edep*the_stack->wt[np];
        if (aux > 0) {
            score->score(ir,aux);
        }
        //      if(      the_stack->iq[np]      )      score->score(ir,the_epcont-
>edep*the_stack->wt[np]);
        if (ir == nreg+1) {
            EGS_ScoringArray *flu = the_stack->iq[np] ? eflu : gflu;
            EGS_Float r2 = the_stack->x[np]*the_stack->x[np] + the_stack-
>y[np]*the_stack->y[np];
            int bin = (int)(sqrt(r2)*10.);
            if (bin < 200) {
                aux = the_stack->wt[np]/the_stack->w[np];
                if (aux > 0) {
                    flu->score(bin,aux);
                }
            }
        }
        return 0;
    }
    int np = the_stack->np-1;
    if (iarg == BeforeBrems || iarg == BeforeAnnihRest || (iarg ==
BeforeAnnihFlight &&
        the_stack->latch[np] > 0)) {
        the_stack->latch[np] = 0;
        rr_flag = 1;
        the_egsvr->nbr_split = the_egsvr->i_do_rr;
        return 0;
    }
}

```

```

    if (iarg == AfterBrems && deflect_brems) {
        EGS_Vector u(the_stack->u[np-1],the_stack->v[np-1],the_stack->w[np-1]);

        EGS_Float tau = the_stack->E[np-1]/the_useful->rm - 1;
        EGS_Float beta = sqrt(tau*(tau+2))/(tau+1);
        EGS_Float eta = 2*rndm->getUniform()-1;
        EGS_Float cost = (beta + eta)/(1 + beta*eta);
        EGS_Float sint = 1 - cost*cost;
        if (sint > 0) {
            sint = sqrt(sint);
            EGS_Float cphi, sphi;
            rndm->getAzimuth(cphi,sphi);
            u.rotate(cost,sint,cphi,sphi);
            the_stack->u[np-1] = u.x;
            the_stack->v[np-1] = u.y;
            the_stack->w[np-1] = u.z;
        }
    }
    if (iarg == AfterBrems || iarg == AfterAnnihRest || iarg == AfterAnnihFlight) {
        the_egsvr->nbr_split = 1;
        if (iarg == AfterBrems && rr_flag) {
            the_stack->latch[the_stack->npold-1] = 1;
        }
        rr_flag = 0;
        return 0;
    }
    /*
    if( iarg == FluorescentEvent && the_stack->latch[np] > 0 ) {
        the_stack->latch[np] = 0; the_stack->wt[np] /= the_egsvr->i_do_rr;
        if( np+1+the_egsvr->i_do_rr > MXSTACK )
            egsFatal("Stack size exceeded while splitting fluorescent photon!\n");
        for(int j=1; j<the_egsvr->i_do_rr; j++) {
            EGS_Float cost = 2*rndm->getUniform()-1;
            EGS_Float sint = 1 - cost*cost; sint = sint > 0 ? sqrt(sint) : 0;
            EGS_Float cphi, sphi; rndm->getAzimuth(cphi,sphi);
            the_stack->E[np+j] = the_stack->E[np];
            the_stack->wt[np+j] = the_stack->wt[np];
            the_stack->iq[np+j] = the_stack->iq[np];
            the_stack->ir[np+j] = the_stack->ir[np];
            the_stack->dnear[np+j] = the_stack->dnear[np];
            the_stack->latch[np+j] = the_stack->latch[np];
            the_stack->x[np+j] = the_stack->x[np];
            the_stack->y[np+j] = the_stack->y[np];
            the_stack->z[np+j] = the_stack->z[np];
            the_stack->u[np+j] = sint*cphi;
            the_stack->v[np+j] = sint*sphi;
            the_stack->w[np+j] = cost;
        }
    }
    */
    return 0;
}

int Tutor7_Application::outputData() {
    // We first call the outputData() function of our base class.
    // This takes care of saving data related to the source, the random

```

```

// number generator, CPU time used, number of histories, etc.
int err = EGS_AdvancedApplication::outputData();
if (err) {
    return err;
}
// We then write our own data to the data stream. data_out is
// a pointer to a data stream that has been opened for writing
// in the base class.
(*data_out) << " " << Etot << endl;
if (!score->storeState(*data_out)) {
    return 101;
}
for (int j=0; j<nph; j++) {
    if (!pheight[j]->storeState(*data_out)) {
        return 102+j;
    }
}
if (!eflu->storeState(*data_out)) {
    return 301;
}
if (!gflu->storeState(*data_out)) {
    return 302;
}
return 0;
}

int Tutor7_Application::readData() {
    // We first call the readData() function of our base class.
    // This takes care of reading data related to the source, the random
    // number generator, CPU time used, number of histories, etc.
    // (everything that was stored by the base class outputData() method).
    int err = EGS_AdvancedApplication::readData();
    if (err) {
        return err;
    }
    // We then read our own data from the data stream.
    // data_in is a pointer to an input stream that has been opened
    // by the base class.
    (*data_in) >> Etot;
    if (!score->setState(*data_in)) {
        return 101;
    }
    for (int j=0; j<nph; j++) {
        if (!pheight[j]->setState(*data_in)) {
            return 102+j;
        }
    }
    if (!eflu->setState(*data_in)) {
        return 301;
    }
    if (!gflu->setState(*data_in)) {
        return 302;
    }
    return 0;
}

void Tutor7_Application::resetCounter() {
    // Reset everything in the base class
    EGS_AdvancedApplication::resetCounter();
}

```

```

// Reset our own data to zero.
score->reset();
Etot = 0;
for (int j=0; j<nph; j++) {
    pheight[j]->reset();
}
eflu->reset();
gflu->reset();
}
int Tutor7_Application::addState(istream &data) {
// Call first the base class addState() function to read and add
// all data related to source, RNG, CPU time, etc.
int err = EGS_AdvancedApplication::addState(data);
if (err) {
    return err;
}
// Then read our own data to temporary variables and add to
// our results.
double etot_tmp;
data >> etot_tmp;
Etot += etot_tmp;
EGS_ScoringArray tmp(nreg+2);
if (!tmp.setState(data)) {
    return 101;
}
(*score) += tmp;
for (int j=0; j<nph; j++) {
    EGS_ScoringArray tmpj(pheight[j]->bins());
    if (!tmpj.setState(data)) {
        return 102 + j;
    }
    (*pheight[j]) += tmpj;
}
EGS_ScoringArray tmp1(200);
if (!tmp1.setState(data)) {
    return 301;
}
(*eflu) += tmp1;
if (!tmp1.setState(data)) {
    return 302;
}
(*gflu) += tmp1;
return 0;
}
void Tutor7_Application::outputResults() {
    egsInformation("\n\n last case = %d Etot = %g\n",
        (int)current_case, Etot);
    double norm = ((double)current_case)/Etot;

    egsInformation("\n\n=====\\n
");
    egsInformation(" Energy fractions\\n");

    egsInformation("=====\\n");
    egsInformation("The first and last items in the following list of energy
fractions are the reflected and transmitted energy, respectively. These two
values are only meaningful if the source is directed in the positive z-

```

```

direction. The remaining values are the deposited energy fractions in the
regions of the geometry, but notice that the identifying index is the region
number offset by 1 (ir+1).");
    score->reportResults(norm,
                        "ir+1 | Reflected, deposited, or transmitted energy
fraction",false,
                        " %d %12.6e +/- %12.6e %c\n");
    if (nph > 0) {
        if (nph > 1) {
            egsInformation("\n\n=====\\n
");
            egsInformation(" Pulse height distributions\\n"
"=====\\n\\n");
        }
        else {
            egsInformation("\n\n Pulse height distribution in region %d\\n"
"=====\\n\\n",
                        ph_regions[0]);
        }
        for (int j=0; j<nph; j++) {
            if (nph > 1) egsInformation("\\nRegion %d\\n"
"-----"
\\n\\n",ph_regions[j]);
            double f,df;
            for (int i=0; i<pheight[j]->bins(); i++) {
                pheight[j]->currentResult(i,f,df);
                egsInformation("%g %g %g\\n",ph_de[j]*(0.5+i),
                            f/ph_de[j],df/ph_de[j]);
            }
        }
    }
    /*
    EGS_Float Rmax = 20; EGS_Float dr = Rmax/200;
    egsInformation("\\n\\n Electron/Photon fluence at back of geometry as a
function of radial distance\\n"
"=====
=\\n");
    for(int j=0; j<200; ++j) {
        double fe,dfe,fg,dfg;
        eflu->currentResult(j,fe,dfe); gflu->currentResult(j,fg,dfg);
        EGS_Float r1 = dr*j, r2 = r1 + dr;
        EGS_Float A = M_PI*(r2*r2 - r1*r1);
        EGS_Float r = j > 0 ? 0.5*(r1 + r2) : 0;
        egsInformation("%9.3f %15.6e %15.6e %15.6e
%15.6e\\n",r,fe/A,dfe/A,fg/A,dfg/A);
    }
    */
}
void Tutor7_Application::getCurrentResult(double &sum, double &sum2,
double &norm, double &count) {
    count = current_case;
    norm = Etot > 0 ? count/Etot : 0;
    score->currentScore(0,sum,sum2);

```

```

}
int Tutor7_Application::startNewShower() {
    Etot += p.E*p.wt;
    int res = EGS_Application::startNewShower();
    if (res) {
        return res;
    }
    if (current_case != last_case) {
        if (nph > 0) {
            for (int j=0; j<nph; j++) {
                pheight[j]->setHistory(current_case);
                int ireg = ph_regions[j];
                // In ausgab the scoring array is offset by 1 to include
                // the reflected and transmitted as the first and last regions
                EGS_Float edep = score->currentScore(ireg+1);
                if (edep > 0) {
                    int ibin = min((int)(edep/(current_weight*ph_de[j])),
pheight[j]->bins()-1);
                    if (ibin >= 0 && ibin < pheight[j]->bins()) {
                        pheight[j]->score(ibin,1);
                    }
                }
            }
        }
        score->setHistory(current_case);
        eflu->setHistory(current_case);
        gflu->setHistory(current_case);
        last_case = current_case;
    }
    current_weight = p.wt;
    return 0;
}
#ifdef BUILD_APP_LIB
APP_LIB(Tutor7_Application);
#else
APP_MAIN(Tutor7_Application);
#endif

```

## Appendix B

The following is the input file used for 1 MeV monoenergetic electron simulations in EGSnrc.

```

#####
##
#
# MMC Simulation with monoenergetic point source. Designed for use with
tutor7pp
#
#####
##

```



```

#-----
----
:start run control:
    ncase = 1e6
    geometry error limit = 100
:stop run control:

#-----
----
:start run mode:
    # egs_brachy has 3 run modes:
    #     'normal', 'superposition', and 'volume correction only'
    run mode = normal
:stop run mode:

#-----
----
# This input block allows 'pegsless' runs
:start media definition:
    AE = 0.512
    UE = 2.012
    AP = 0.001
    UP = 1.500

    material data file =
C:/EGSnrc_with_egs_brachy/egs_home/egs_brachy/lib/media/material.dat
    #to run in batch, insert local value of $EGS_HOME/egs_brachy/ before lib
:stop media definition:

#-----
----
# A gold MMC
:start geometry definition:

    # An air box that will hold the entire geometry
    :start geometry:
        name = box
        library = egs_glib          #this is a brachy_dose addition to egs++
                                    #which allows files to be included into the
                                    #input file. Very useful for defining
                                    #commonly used geometries.
        include file =
C:/EGSnrc_with_egs_brachy/egs_home/egs_brachy/lib/geometry/phantoms/50cmx50c
mx50cm_box_xyz_air.geom
        #to run in batch, insert local value of $EGS_HOME/egs_brachy/ before
lib
    :stop geometry:

    # The volume in which we want to dose score (the gold MMC)
    :start geometry:
        name = phantom
        library = egs_glib
        include file =
C:/EGSnrc_with_egs_brachy/egs_home/egs_brachy/lib/geometry/phantoms/0.06cmx0
.06cmx0.06cm_0.03mm_goldMMC_1region.geom

    :stop geometry:

```

```

# The source geometry
:start geometry:
    name = seed
    library = egs_glib
    include file =
C:/EGSnrc_with_egs_brachy/egs_home/egs_brachy/lib/geometry/sources/point_source/sphere/sphere.geom
    #to run in batch, insert local value of $EGS_HOME/egs_brachy/ before
lib
:stop geometry:

# Inscribe the seed in the scoring phantom
:start geometry:
    name = phantom_with_seed
    library = egs_genvelope
    base geometry = phantom
    inscribed geometries = seed
:stop geometry:

# Inscribe the phantom+seed in the large air box
:start geometry:
    name = final
    library = egs_genvelope
    base geometry = box
    inscribed geometries = phantom_with_seed
:stop geometry:

# Source, phantom, and simulation geometries all need to be explicitly
# specified to egs_brachy

source geometries = seed

phantom geometries = phantom

simulation geometry = final

:stop geometry definition:

#-----
----
:start volume correction:

    # Scoring voxels which contain source geometries need to have their
    # volumes corrected to accurately score dose
:start source volume correction:
    correction type = correct
    density of random points (cm-3) = 1E8

    # This include file statement defines a shape that encompasses the
    # entire source geometry.
    # Volume correction will only occur within the boundaries of this
shape.
    include file =
C:/EGSnrc_with_egs_brachy/egs_home/egs_brachy/lib/geometry/sources/point_source/sphere/boundary.shape

```

```

        #to run in batch, insert local value of $EGS_HOME/egs_brachy/ before
lib
        :stop source volume correction:

:stop volume correction:

#-----
----
:start source definition:

    # Typical egs++ source input block
    :start source:
        library = egs_isotropic_source
        name = PointSource
        charge = -1

        include file =
C:/EGSnrc_with_egs_brachy/egs_home/egs_brachy/lib/geometry/sources/point_sou
rce/sphere/sphere.shape
        #to run in batch, insert local value of $EGS_HOME/egs_brachy/ before
lib

        :start spectrum:
            type = monoenergetic
            energy = 0.1
        :stop spectrum:
    :stop source:

    # The position of the source
    :start transformations:
        include file =
C:/EGSnrc_with_egs_brachy/egs_home/egs_brachy/lib/geometry/transformations/s
ingle_seed_at_origin
        #to run in batch, insert local value of $EGS_HOME/egs_brachy/ before
lib
        :stop transformations:

    # The source needs to be explicitly specified to egs_brachy.
    # This should be the same as the 'name' defined in source input block
above.
        simulation source = PointSource

:stop source definition:

#-----
----
:start scoring options:

    # Many scoring options are available in egs_brachy.
    # Please see the documentation for a full list.

    score tracklength dose = yes
    score energy deposition = yes

    # The path to a file containing mass-energy absorption data for the
    # relevant media in the simulation

```

```

muen file =
C:/EGSnrc_with_egs_brachy/egs_home/egs_brachy/lib/muen/brachy_gold_1.5MeV.mu
endat
#to run in batch, insert local value of $EGS_HOME/egs_brachy/ before lib
# Specify which media dose is scored in
muen for media = Au

#pulse height distribution (hopefully)
pulse height regions = 1
pulse height bins = 100

#
# :start spectrum scoring:
# type = surface count
# particle type = photon
# minimum energy = 0.000
# maximum energy = 0.5
# number of bins = 50
# output format = xmgr
# :stop spectrum scoring:

:stop scoring options:

#-----
----
# Transport parameters
include file =
C:/EGSnrc_with_egs_brachy/egs_home/egs_brachy/lib/transport/low_energy_defau
lt
#to run in batch, insert local value of $EGS_HOME/egs_brachy/ before lib

#####
### AUSGAB OBJECTS
#####
:start ausgab object definition: # Only 1 ausgab definition block allowed

### Particle tracks
:start ausgab object:
    name = tracks
    library = egs_track_scoring
:stop ausgab object:

### Dose scoring
:start ausgab object:
    library = egs_dose_scoring
    name = my_dose_scoring
    region dose = yes
    volume = 0.000216
    dose regions = 1
:stop ausgab object:

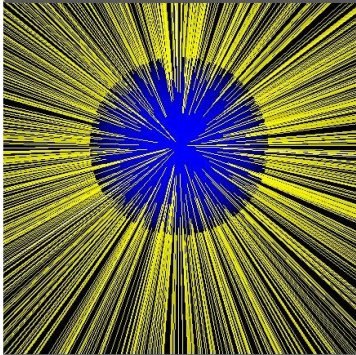
:stop ausgab object definition:

```

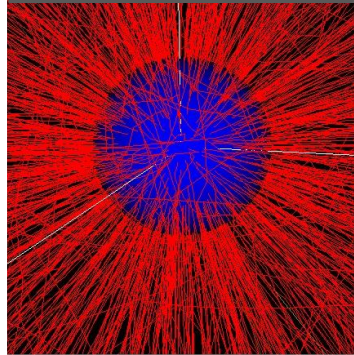
Analysis of different spectrum scoring options available:

# Absolute Counts

1.5 MeV photons emitted from point source



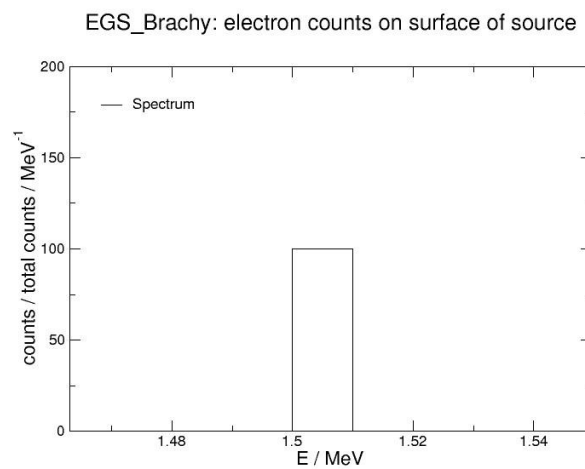
1.5 MeV electrons emitted from point source



- Records how many particles escape the surface of the source
- Applications: useful for determining how much radiation is entering rest of simulation

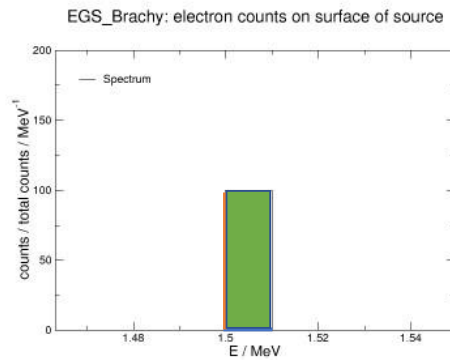
## Histograms/Results

- Once simulation is run, histogram plotting information is saved under a `agr` file and can be plotted using QtGrace
- X-axis is energy, yaxis is proportional to number of counts
- Bin width =  $(E_{\max} - E_{\min}) / \# \text{ of bins}$



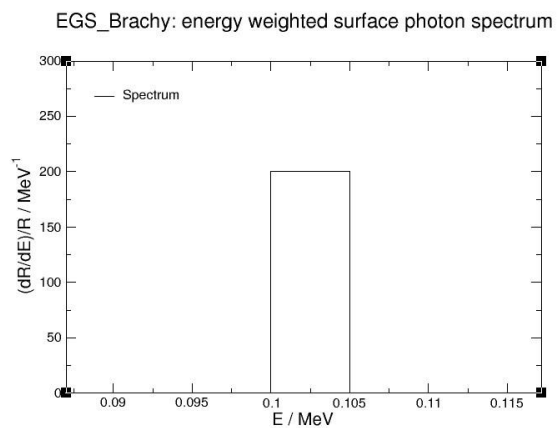
# Interpretation

- Area of each rectangle is the percentage of total counts simulated with that energy
- For this example:  
 $(0.01 \text{ MeV}) * (100 \text{ MeV}^{-1}) = 1$
- This means 100% of the simulated electrons had an energy between 1.5 MeV and 1.51 MeV



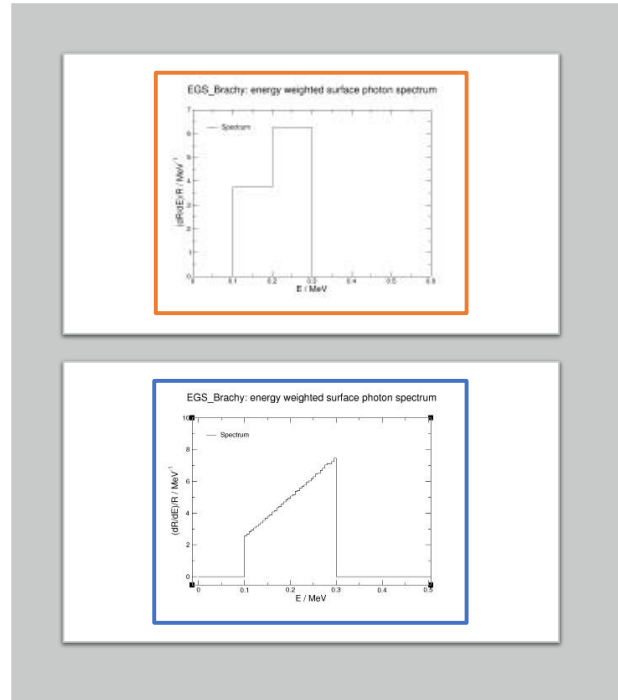
## Energy Weighted Spectra

- Records particles on surface of source weighted by the amount of energy they have
- Histories with lower energies contribute less to the total amount of energy within the simulations
- When all histories have the same energy, only one bin is formed (see right)



# Histogram/Results

- Notes on y-axis:
  - $dR$  is the total amount of energy all histories within the bin have
  - $dE$  is the width of the bin (energy of the histories)
  - $R$  is the total energy of all histories initialized in the simulation
- Consider the orange graph
  - $R = 200400 \text{ MeV}$
  - The first bin has width  $0.1 \text{ MeV}$  and height  $3.747 \text{ MeV}^{-1}$
  - Thus,  $dR/R = dE(3.747) = (0.1 \text{ MeV}) * (3.747 \text{ MeV}^{-1}) = 0.3747$ 
    - This tells us what percentage of the total energy in the simulation came from particles with this amount of energy (37.47%)
  - Additionally,  $dR = 0.3747 * (200400 \text{ MeV}) = 75089.88 \text{ MeV}$ 
    - This is the total amount of energy particles with this specific initial energy had when combined
- Consider the blue graph
  - Now, instead of two bins, there are 40 ( $0.005 \text{ MeV}$  width)
  - We could go through each bin and find the proportions, but I won't
  - Just notice that if you consider the area of the whole thing, we see:
 
$$A_{\text{trapezoid}} = \frac{a+b}{2}h = \frac{2.563 \text{ MeV}^{-1} + 7.466 \text{ MeV}^{-1}}{2} (0.3 \text{ MeV} - 0.1 \text{ MeV})$$
  - $A_{\text{trapezoid}} = 1.00$
  - So, all the energy is accounted for within the graph!



## Appendix C

The following python script was used to read through output files from each of the four simulations and pull the histogram values out for constructing the coarsely binned response matrix.

```
# Importing the required modules
import numpy as np
import matplotlib.pyplot as plt

import linecache
# Generating data for the heat map
response_matrix = np.zeros(( 4 , 80 ))

#Reading in energy deposition from 0.25MeV Monoenergetic Source
with open("C:/Users/aeb11/OneDrive/Documents/ProjectLab
(Koehler)/Dumb_Man's_Response_Matrix_0.25MeV.egslog", 'r') as file:
    fileText = file.readlines()

for i in range (408, 428, 1):
    bin_data = linecache.getline("C:/Users/aeb11/OneDrive/Documents/Project
Lab(Koehler)/Dumb_Man's_Response_Matrix_0.25MeV.egslog", i)
    #print(bin_data)
    energy = bin_data.split(' ') [0]
    N = bin_data.split(' ') [1]
```

```

response_matrix[0][i-408] = N

#print(energy)
#print(N)
#print(empty)

#Reading in energy deposition from 0.50MeV Monoenergetic Source
with open("C:/Users/aeb11/OneDrive/Documents/Project Lab
(Koehler)/Dumb_Man's_Response_Matrix_0.50MeV.egslog", 'r') as file:
    fileText = file.readlines()

for i in range (408, 448, 1):
    bin_data = linecache.getline("C:/Users/aeb11/OneDrive/Documents/Project
Lab (Koehler)/Dumb_Man's_Response_Matrix_0.50MeV.egslog", i)
    #print(bin_data)
    energy = bin_data.split(' ') [0]
    N = bin_data.split(' ') [1]
    response_matrix[1][i-408] = N

#Reading in energy deposition from 0.75MeV Monoenergetic Source
with open("C:/Users/aeb11/OneDrive/Documents/Project Lab
(Koehler)/Dumb_Man's_Response_Matrix_0.75MeV.egslog", 'r') as file:
    fileText = file.readlines()

for i in range (408, 468, 1):
    bin_data = linecache.getline("C:/Users/aeb11/OneDrive/Documents/Project
Lab (Koehler)/Dumb_Man's_Response_Matrix_0.75MeV.egslog", i)
    #print(bin_data)
    energy = bin_data.split(' ') [0]
    N = bin_data.split(' ') [1]
    response_matrix[2][i-408] = N

#Reading in energy deposition from 1.00MeV Monoenergetic Source
with open("C:/Users/aeb11/OneDrive/Documents/Project Lab
(Koehler)/Dumb_Man's_Response_Matrix_1.00MeV.egslog", 'r') as file:
    fileText = file.readlines()

for i in range (408, 468, 1):
    bin_data = linecache.getline("C:/Users/aeb11/OneDrive/Documents/Project
Lab (Koehler)/Dumb_Man's_Response_Matrix_1.00MeV.egslog", i)
    #print(bin_data)
    energy = bin_data.split(' ') [0]
    N = bin_data.split(' ') [1]
    response_matrix[3][i-408] = N

print(response_matrix)

```



## References

- 1 M.P. Croce, et al., "Development of Holmium-163 Electron-Capture Spectroscopy with Transition-Edge Sensors," J. Low. Temp. Phys. 184, 958-968 (2016).
- 2 H. Becquerel, "Sur les radiations émises par phosphorescence," Séanc. Acad. Sci. Paris, 122, 420-421 (1896).
- 3 E. Rutherford, "Uranium Radiation and the Electrical Conduction Produced by It," Philos. Mag. 47, 109-163 (1899).
- 4 H. Becquerel, in *Nobel Lectures, Physics 1901-1921* (Elsevier Publishing Company, Amsterdam, 1967).
- 5 J. Chadwick, "Intensitätsverteilung im magnetischen Spektren der  $\gamma$ -Strahlen von Radium B + C," Verhandl. Dtsc. Phys. Ges. 16, 383-391 (1914).
- 6 C. D. Ellis, W. A. Wooster, "The Average Energy of Disintegration of Radium E," Proc. Roy. Soc. A, 177(776), 109-123 (1927).
- 7 L. M. Brown, "The Idea of the Neutrino," Phys. Tod., 31(9), 23-28, (1978).
- 8 E. Fermi, "Tentativo de una Teoria dei Raggi  $\beta$ ," (1934).
- 9 F. L. Wilson, "Fermi's Theory of Beta Decay," Am. J. Phys., 36(12), 1150-1160 (1968).
- 10 C. L. Cowan, Jr., F. Reines, "Detection of the Free Neutrino," Phys. Rev., 92(3), 830-831, (1953).
- 11 K. Koehler, "Microcalorimeters: A Bright, Bold Future," (2021).
- 12 P. C-O. Ranitzsch, et al., "MetroMMC: Electron-Capture Spectrometry with Cryogenic Calorimeters for Science and Technology," J. Low. Temp. Phys., 199, 441-450, (2020).
- 13 M. Loidl, J. Beyer, L. Bockhorn, C. Enss, D. Györi, S. Kempf, K. Kossert, R. Mariam, O. Nähle, M. Paulsen, M. Rodrigues, M. Schmidt, "MetroBeta: Beta Spectrometry with Metallic Magnetic Calorimeters in the Framework of the European Program of Ionizing Radiation Metrology," J. Low. Temp. Phys., 193, 1251-1256, (2018).
- 14 M. Loidl, J. Beyer, L. Bockhorn, C. Enss, D. Györi, S. Kempf, K. Kossert, R. Mariam, O. Nähle, M. Paulsen, M. Rodrigues, M. Schmidt, "Beta spectrometry with metallic magnetic calorimeters in the framework of the European EMPIR project MetroBeta," Appl. Radiat. Isot., 153, (2019).
- 15 E. Steinbauer, P. Bauer, M. Geretschläger, G. Bortels, J.P. Biersack, P. Burger, "Energy resolution of silicon detectors: approaching the physical limit," Nucl. Instrum. Methods Phys. Res., 85, 642-649, (1994).
- 16 K. E. Koehler, D. A. Bennett, E. M. Bond, M. P. Croce, D. E. Dry, R. D. Horansky, et. al, "Q Spectroscopy with Superconducting Sensor Microcalorimeters," IEEE. Trans. Nucl. Sci., 60(2), 624-629, (2014).
- 17 J. N. Ullom, D. A. Bennett, "Review of Superconducting Transition-Edge Sensors for X-ray and Gamma-ray Spectroscopy," Supercond. Sci. Technol. 28 (2015).
- 18 A. Fleischmann, T. Daniyarov, H. Rotzinger, M. Linck, and C. Enss, "Magnetic Calorimeters for High Resolution X-ray Spectroscopy," Rev. Sci. Instrum. 74, 3947-3954 (2003).
- 19 M. K. Bacrania *et al.*, "Large-Area Microcalorimeter Detectors for Ultra-High-Resolution X-Ray and Gamma-Ray Spectroscopy," IEEE. Trans. Nucl. Sci. 56(4), 2299-2302 (2009).
- 20 E. Cosulich, G. Gallinaro, F. Gatti, S. Vitale, "Detection of  $^{187}\text{Re}$  beta decay with a cryogenic microcalorimeter. Preliminary results," Phys. Lett. B, 295, 143-147, (1992).
- 21 L. Gastaldo, K. Blaum, K. Chrysalidis, et al., "The electron capture in  $^{163}\text{Ho}$  experiment—ECHO," Eur. Phys. J. Spec. Top., 226, 1623-1694, (2017).
- 22 D. A. Bennett et al., "A high resolution gamma-ray spectrometer based on superconducting microcalorimeters," Rev. Sci. Instrum. 83, (2012).
- 23 S. Friedrich, G. B. Kim, D. Lee, J. Ad Hall, R. Cantor, A. Voyles, R. Hummatov, S. P. T. Boyd, "Ultra-high Resolution Magnetic Microcalorimeter Gamma-Ray Detectors for Non-Destructive Assay of Uranium and Plutonium," presented at the IAEA Workshop on Nondestructive Assay of Uranium and Plutonium, Vienna, Austria (2021).
- 24 D. McCammon, in *Cryogenic Particle Detection*, Ed. by C. Enss (Springer-Verlag, Berlin, 2005), p. 1-34.

- 
- 25 S. Schlör, *Sketch of helium dilution refrigerator* [vector graphic], Wikipedia, February 14 2019 ([https://en.wikipedia.org/wiki/Dilution\\_refrigerator](https://en.wikipedia.org/wiki/Dilution_refrigerator)).
- 26 F. Pobell, *Matter and Methods at Low Temperatures* (Springer Berlin, Heidelberg, 2007).
- 27 D. H. Andrews et al. "Attenuated Superconductors I. for Measuring Infra-Red Radiation," *Rev. Sci. Instrum.* 13, 281 (1941).
- 28 K. D. Irwin and G. C. Hilton, in *Cryogenic Particle Detection*, Ed. by C. Enss (Springer-Verlag, Berlin, 2005), p. 63-149.
- 29 A. Fleischmann, C. Enss, and G.M. Seidel, in *Cryogenic Particle Detection*, Ed. by C. Enss (Springer-Verlag, Berlin, 2005), p. 151-216.
- 30 W. Seidel, Ph. D. Thesis, Technische Universität München, 1986.
- 31 R. L. Harrison, "Introduction to Monte Carlo Simulation," *AIP. Conf. Proc.*, 1204, 17-21, (2010).
- 32 I. Kawrakow, D.W.O Rogers, E. Mainegra-Hing, F. Tessier, R.W. Townson, B.R.B Walters. "EGSnrc toolkit for Monte Carlo simulation of ionizing radiation transport", doi: 10.4224/40001303 [release v2021] (2000).
- 33 M. J. Berger, "Monte Carlo Calculation of the penetration and diffusion of fast charged particles," *Methods Comput. Phys.*, 1, 135 – 215 (1963).
- 34 I. Kawrakow, D.W.O Rogers, E. Mainegra-Hing, F. Tessier, B.R.B Walters, "The EGSnrc Code System: Monte Carlo Simulation of Electron and Photon Transport," NRC Canada, (2018).
- 35 B. A. Faddegon, I. Kawrakow, Y. Kubyshin, J. Perl, J. Sempau, L. Urban, "The accuracy of EGSnrc, Geant4 and PENELOPE Monte Carlo systems for the simulation of electron scatter in external beam radiotherapy," *Phys. Med. Biol.*, 54(20), 6151-6163, (2009).
- 36 G. Nelson and D. Reilly, in *Passive Nondestructive Assay of Nuclear Materials*, Ed. by D. Reilly, N. Ensslin, H. Smith, Jr., and S. Kreiner, (U.S. Government Printing Office, Washington, DC, 1991).
- 37 J. H. Hubbell and S. M. Seltzer (1996). "X-Ray Mass Attenuation Coefficients," NIST, V.1.4, Dataset, <https://physics.nist.gov/PhysRefData/XrayMassCoef/ElemTab/z79.html>.
- 38 M. Paulsen, K. Kossert, J. Beyer, "An Unfolding Algorithm for High Resolution Microcalorimetric Beta Spectrometry," *Nucl. Instrum. Methods Phys. Res.*, 953, (2020).
- 39 Iwan Kawrakow, computer code `egs_track_scoring.cpp` (National Research Council, Canada, 2015), [https://nrc-cnrc.github.io/EGSnrc/doc/pirs898/classEGS\\_TrackScoring.html](https://nrc-cnrc.github.io/EGSnrc/doc/pirs898/classEGS_TrackScoring.html)
- 40 Ernesto Mainegra-Hing, computer code `egs_dose_scoring.cpp` (National Research Council, Canada, 2015), [https://nrc-cnrc.github.io/EGSnrc/doc/pirs898/classEGS\\_DoseScoring.html](https://nrc-cnrc.github.io/EGSnrc/doc/pirs898/classEGS_DoseScoring.html)
- 41 H. Rotzinger, et al., "Beta Spectroscopy with Magnetic Calorimeters," *J Low Temp Phys*, 151, 1087-1093, (2008).
- 42 J. A. Bearden, "X-Ray Wavelengths," *Rev. Mod. Phys.*, 39(1), 78-124, (1967).