Validation of a Simulation Model of Intrinsic ^{176}Lu Activity in LSO-Based Preclinical PET Systems

by

Bryan McIntosh

A Thesis submitted to the Faculty of Graduate Studies of

The University of Manitoba

in partial fulfilment of the requirements of the degree of

MASTER OF SCIENCE

Department of Physics and Astronomy

University of Manitoba

Winnipeg, Manitoba, Canada

Abstract

The LSO scintillator crystal commonly used in PET scanners contains a low level of intrinsic radioactivity due to a small amount of ^{176}Lu. This is not usually a concern in routine scanning but can become an issue in small animal imaging, especially when imaging low tracer activity levels. Previously there had been no systematic validation of simulations of this activity; this thesis discusses the validation of a GATE model of intrinsic ^{176}Lu against results from a bench-top pair of detectors and a Siemens Inveon preclinical PET system. The simulation results matched those from the bench-top system very well, but did not agree as well with results from the complete Inveon system due to a drop-off in system sensitivity at low energies that was not modelled. With this validation the model can now be used with confidence to predict the effects of ^{176}Lu activity in future PET systems.

Acknowledgements

This project would simply not have been possible without the help of the following talented and downright wonderful people:

- Dr. Andrew Goertzen, who has been a wonderful mentor throughout my degree.
- The Manitoba Health Research Council, who funded this work under a Graduate Studentship Award.
- Bob Miller, Chad Harris, and the rest of the staff in the Medical Devices department at CancerCare Manitoba for help designing and building the detector holders and for being great floor hockey opponents.
- Vic Goertzen and the CancerCare Manitoba Nuclear Electronics department for help designing the custom readout circuit boards.
- All the graduate students in the Medical Physics program. The road to a degree is long and difficult, but having good friends makes it much more bearable.
- Dr. Melanie Martin, whose work sparked my interest in Medical Physics and gave me my first chance to do research.
- My family for supporting me unconditionally through all the years as I worked to get my degree.

Contents

Acknowledgements **i**

List Of Tables **v**

List Of Figures **vii**

List Of Copyrighted Material **viii**

1 Introduction **1**

1.1 Background 1

1.2 PET Scanner Design 6

1.2.1 Overview 6

1.2.2 Scintillator Crystals 7

1.2.3 Photomultiplier Tubes 13

1.2.4 PET Detectors 14

1.3 PET Data Processing 17

1.4 Intrinsic Activity in Lu-based scintillators 22

1.5 Monte Carlo Simulations 26

1.6 Modelling of ^{176}Lu Intrinsic Activity in GATE 28

2 Bench-Top Data Acquisition **30**

2.1 Data Acquisition Hardware 30
2.1.1 Inveon Block Detectors 31
2.1.2 NIM Signal Processing 32
2.1.3 DAQ Card 38
2.2 Data Acquisition Software 39
2.2.1 Acquisition Details 41
2.3 Bench-Top Data Processing 41
2.3.1 Calibration 42
2.3.2 Energy Spectrum Generation 43

3 Experimental Details **46**
3.1 Bench-top Measurements 46
3.2 Bench-Top Simulations 48
3.3 Results 51
3.3.1 Sensitivity 51
3.3.2 Intrinsic Activity Measurements 52

4 Inveon System Comparison **56**
4.1 Full System Measurements 56
4.2 Full System Simulations 57
4.3 Results 59

5 Conclusions **63**

A Appendix A: Data Acquisition Code **73**

B Appendix B: Analysis Code **89**
B.1 MATLAB Scripts 89

B.1.1 manualpeaks 89
B.1.2 Flood_MATLAB.m 90
B.1.3 crystalenergy.m 92
B.1.4 calibrate511.m 93
B.1.5 crystalenergy_duo 93
B.1.6 simcoincidence.m 94
B.1.7 readonly.m 95
B.1.8 loadmes_10keV.m 95
B.1.9 sinomash.m 96
B.1.10 masher_direct.m 99
B.1.11 simspectrum_10keV.m 100
B.1.12 sum20.m 100

C Appendix C: GATE Simulations **102**

C.1 Bench-Top Script 102
C.2 Full System Simulations 106
C.2.1 Intrinsic Activity Only 106
C.2.2 Water Tube Simulation 109

List of Tables

1.1 Properties of some common PET scintillators. Data taken from [23] and [24] 12

3.1 Bench-Top Intrinsic Count Rates (counts/second) 54

List of Figures

1.1 Positron production and annihilation schema 2

1.2 Coincidence detection in PET 3

1.3 Clinical image from a ^{18}F-FDG PET scan. 4

1.4 Schematic of a simple PET system. 7

1.5 Kinematics of the photoelectric effect. 9

1.6 Kinematics of the Compton effect. 10

1.7 Inorganic scintillator function. 11

1.8 Attenuation coefficients for various scintillator materials. 13

1.9 Photomultiplier tube cut-away view. 14

1.10 Position sensitive photomultiplier tube functionality. 16

1.11 Segmented flood histogram example. 16

1.12 Examples of true, scattered, and random PET coincidences. 19

1.13 Sample sinogram from a PET scanner. 20

1.14 Explanation of ring difference in PET. 21

1.15 ^{176}Lu decay scheme. 23

1.16 Coincidence caused by intrinsic LSO decay. 24

1.17 Effects of intrinsic LSO activity in weak source imaging. 24

1.18 Mock-up of proposed ultra-portable PET scanner. 29

2.1 Sample assembled Inveon block detector. 31

2.2 Inveon detector module. 32
2.3 Custom readout board diagram. 33
2.4 NIM detector electronics used for this experiment. 34
2.5 Example of detector output. 35
2.6 Block diagram of bench-top NIM electronics. 36
2.7 Constant fraction pulse timing. 37
2.8 Example of leading edge timing leading to timing walk. 37
2.9 Data acquisition software for the bench-top detector system. 40
2.10 Segmented flood histogram from a calibration measurement. 42
2.11 Individual crystal energy spectra. 44
2.12 Sample bench-top coincidence energy spectrum. 45

3.1 Bench-top setup of two opposing Siemens Inveon detectors. 47
3.2 GATE simulation of the bench-top detector setup. 50
3.3 Bench-top sensitivity measurement energy spectra. 52
3.4 Bench-top intrinsic activity energy spectra. 53
3.5 Plot of non-proportional LSO light output. 54
3.6 Illustration of photopeak misalignment. 55

4.1 Simulated Inveon scanner with a water-filled centrifuge tube placed in the centre of the scanner's field of view. 58
4.2 Energy spectra from a complete Inveon system and its simulated counterpart. 60
4.3 Sinogram line profiles from the central 20 sinogram slices with a centrifuge tube in the FOV and intrinsic activity only. 62

List Of Copyrighted Material

- Compton-scattering.svg, image from Wikimedia Commons, originally uploaded by Wikipedia user Jabberwok, released under Creative Commons Attribution ShareAlike License 3.0, found on Page 10 (Figure 1.6)
- Figure from [25], found on Page 14 (Figure 1.8), reprinted by permission of the Society of Nuclear Medicine
- Figure from [29], found on Page 16 (Figure 1.10B), Copyright 1997 with permission from Elsevier.
- Figure from [52], found on Page 31 (Figure 2.1), ©2007 IEEE.
- Figure from [55], found on Page 54 (Figure 3.5), ©2004 IEEE.

Chapter 1

Introduction

1.1 Background

Positron Emission Tomography (PET) is a nuclear medicine imaging technique that uses positron-emitting radioisotopes chemically bound to a tracer molecule [1] to image the molecular interactions of biological processes. Positrons (or β^+ particles) are emitted from a proton-rich nucleus X via the decay process

$$ {}^{A}_{Z}X \rightarrow {}^{A}_{Z-1}Y + \beta^+ + \nu \tag{1.1} $$

where A is the atomic mass of the atom, Z is its atomic number, Y is the daughter nucleus, and ν is a neutrino. The positron emission process is shown in Figure 1.1; an excess proton is converted to a positron with a variable kinetic energy E as well as a neutrino. The much more massive nucleus moves relatively slowly to conserve momentum but with essentially zero kinetic energy; the neutrino produced in the decay thus effectively shares the released energy with the positron, causing a spectrum of positron energies to be seen [2]. The positron interacts with electrons in the

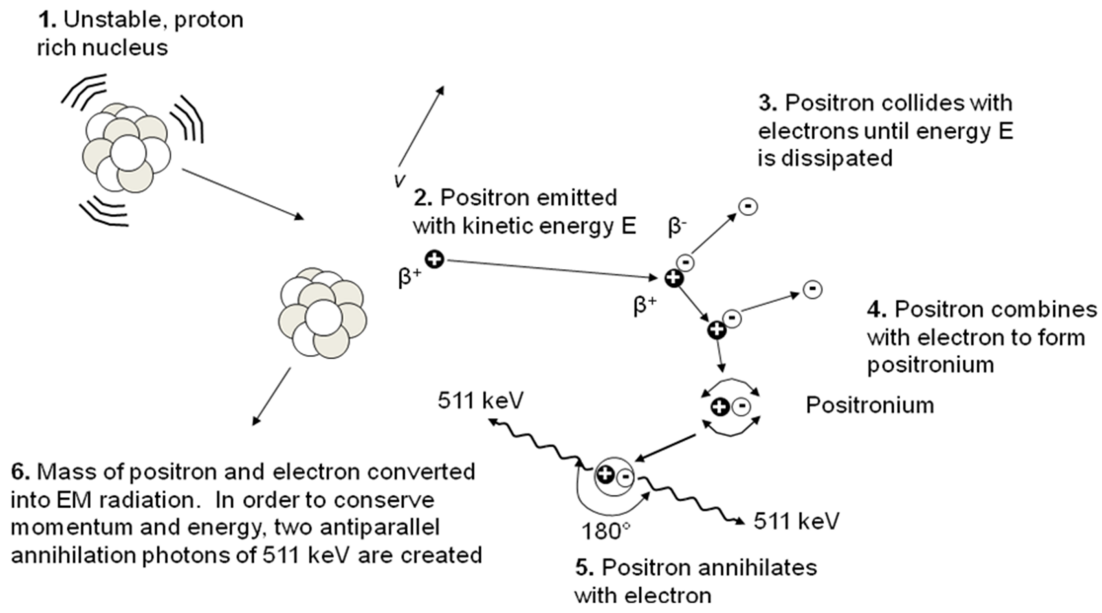

Figure 1.1: Positron production and annihilation schema. Positrons produced in a material ultimately annihilate with electrons to produce pairs of 511 keV annihilation photons which are used to create PET images. Image courtesy Dr. Andrew L. Goertzen.

surrounding material via electromagnetic interactions until its energy E is dissipated, at which point it either annihilates directly with an electron or combines with an electron briefly to form positronium (an electron orbiting a positron) for slightly more than 0.1 ns [3] before annihilating. The anti-matter annihilation causes the rest mass energy of both particles (511 keV each) to be converted to electromagnetic radiation, creating a pair of antiparallel annihilation photons to conserve momentum.

A ring of photon detectors placed around the patient is used to detect coincident pairs of photons; by drawing a line of response (LOR) between the two detectors, the angle and distance of the original annihilation from the centre of the scanner's field of view (FOV) can be determined (Figure 1.2). By acquiring a large number of coincidence events a map of tracer distribution is made, showing more activity in areas where more tracer was taken up.

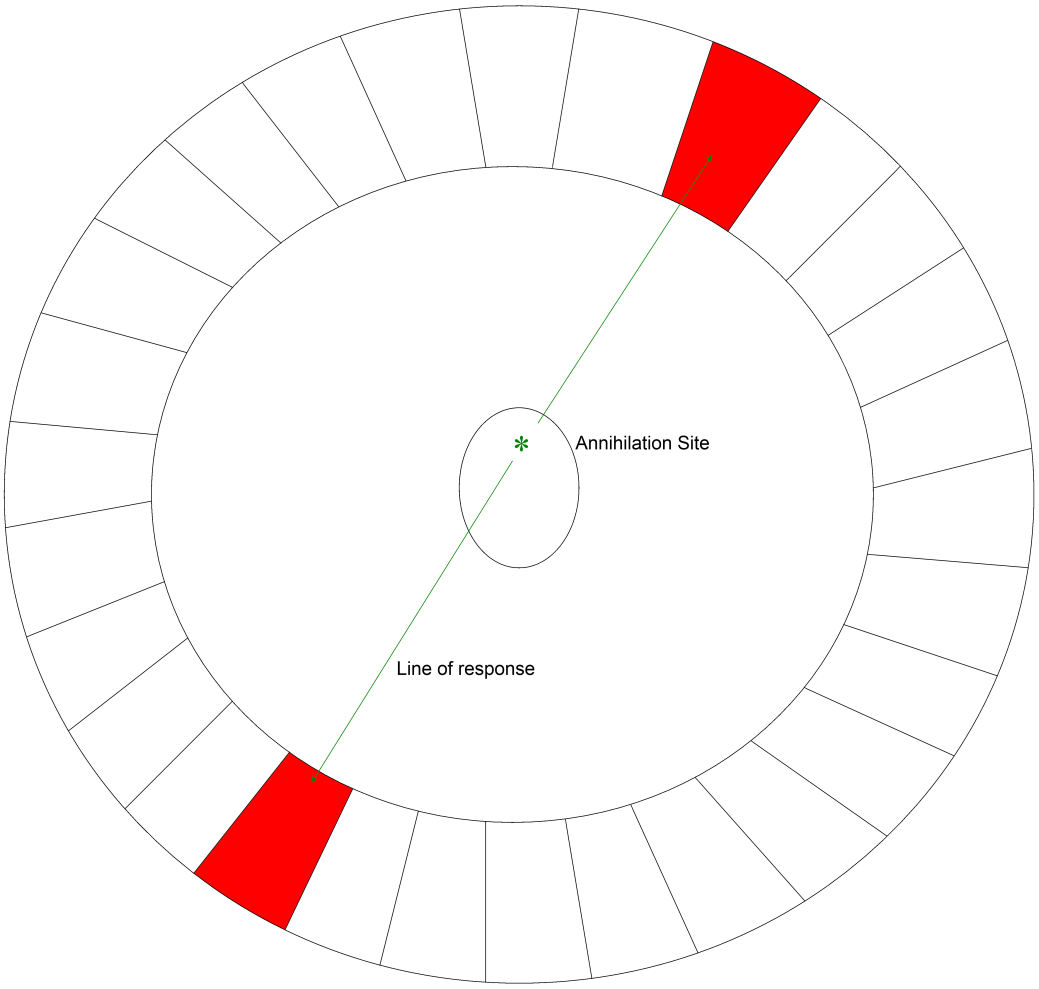

Figure 1.2: Coincidence detection in PET. After a positron annihilates with an electron in the imaging subject event a pair of annihilation photons are emitted that can be detected by a ring of detectors placed around the imaging subject.

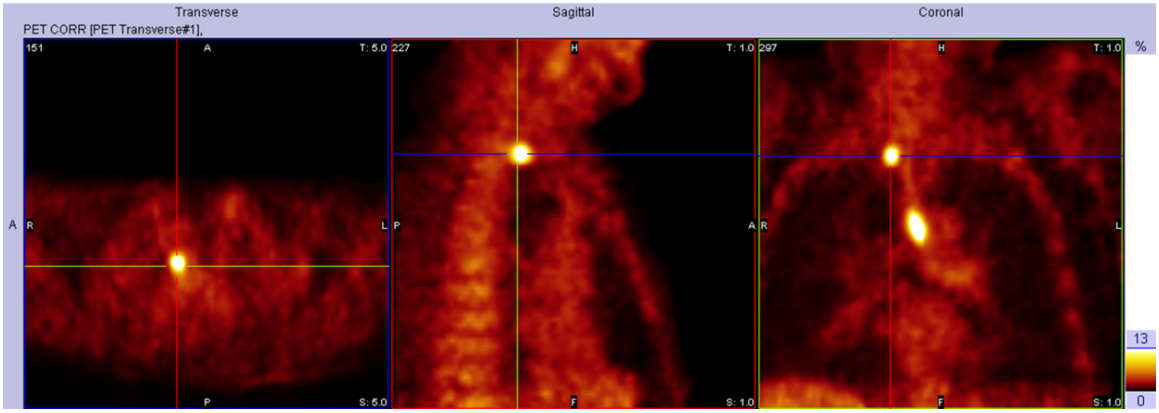

Figure 1.3: Clinical image from a ^{18}F-FDG PET scan. Brighter areas indicate more tracer uptake. Two esophegeal tumours (bright spots) are visible in this image; the upper tumour was not visible on conventional CT. Image courtesy of Great-West Life PET-CT Imaging Centre.

The most popular radiotracer used in PET imaging is fluorodeoxyglucose (^{18}F-FDG) [4], a glucose analog. This radiotracer has been used to perform comparison studies of brain function in patients with and without degenerative diseases such as Alzheimer's disease. It has also been used to evaluate the viability of cardiac tissue and hence detect signs of coronary artery disease [1]. However, PET with ^{18}F-FDG is most widely used clinically in cancer imaging to provide information about tumour size and activity. Since glucose is taken up preferentially by tumour cells at a considerably higher rate than that seen in normal cell metabolism [5], this leads to a large amount of FDG tracer uptake in cancer lesions and a concurrently high sensitivity for finding small lesions with PET. (Figure 1.3). When combined with computed tomography (CT) scanning to provide anatomical reference points, PET has been found to provide superior sensitivity, specificity, and accuracy in diagnosing cancers such as small cell lung cancer than traditional planar x-ray or CT scanning alone [6].

While it's technically possible to detect annihilation photons with a traditional

single-photon emission computed tomography (SPECT) system, SPECT detectors are ill-suited for detecting 511 keV photons. The thick collimators needed to reduce scatter would also severely reduce the sensitivity of the system, and SPECT systems lack the coincidence logic to take advantage of the electronic collimation that is possible from the anti-parrallel trajectory of annihilation photons. PET's coincidence logic data processing allows collimation to be done electronically based on coincidence timing so that the narrow line of response drawn between the coincidence events eliminates the need for absorptive collimation. By using a ring of detectors around the patient PET can also acquire data from all possible emission angles around the patient simultaneously while SPECT systems can only acquire data within a limited set of angles at each angular position. These factors allow PET to have a higher sensitivity than SPECT for a given midplane resolution (roughly 49 000 cps/MBq vs 100 cps/MBq [7][8]), as well as allowing data to be gathered more quickly and with fewer motion artifacts [9].

PET's ability to perform dynamic and quantitative studies of tracer uptake [10] has led to the development of new tracer compounds beyond FDG. Some of these can track dopamine levels [11], breast cancer-related factors [12], and custom labelling of candidate drugs to discover the pharmacodynamics of a drug in a non-invasive manner [13]. The development of these tracers has traditionally been carried out in laboratory animals such as rats, mice, rabbits, and other small mammals before reaching human clinical trials. Imaging small animals requires higher spatial resolution (millimetres as opposed to centimetres) than a clinical PET system can provide due to the large difference in size between humans and animals, as well as equal sensitivity compared to a clinical PET system to maintain image quality [14].

The development of small animal (also known as "preclinical") PET systems began as strictly research devices and one-off prototypes, but began to become commercial-

ized with the release of the Concorde Microsystems microPET system in the late 1990s [15]. This system was the first commercially available scanner to use the high performance scintillator lutetium oxyorthosilicate (LSO, discussed in 1.2.2), allowing it to achieve both high spatial resolution with small crystals and high energy resolution due to the crystals' high light output. The past decade has seen the development of several other preclinical PET systems both commercial [16] and non-commercial [17][18], each attempting to increase spatial resolution or correct artifacts that are endemic to PET scanners with small detector ring diameters. This increase in availabilty of imaging devices as well as the rapid development of new radiotracers and spread of awareness of PET in the life science research community has led to PET taking its place as a highly valued imaging modality when performing studies in molecular medicine [19].

1.2 PET Scanner Design

1.2.1 Overview

As stated before, PET scanners are composed primarily of a ring of photon detectors placed around an imaging bed. The most common detector designs use inorganic scintillator crystals to convert annihilation photons to visible light which is picked up by a photodetector, usually a photomultiplier tube (PMT). The crystals are often coupled to the PMT using fibre-optic or glass light guides, especially if the scintillator area exceeds the area of the PMT face. Signals from the PMTs are processed by readout electronics to determine whether detected events are within a given energy and timing window to be considered a pair of coincidence photons from the same annihilation event. If the signals are valid coincidence events they are then digitized and sent to a computer system where they are stored in a sinogram and can be

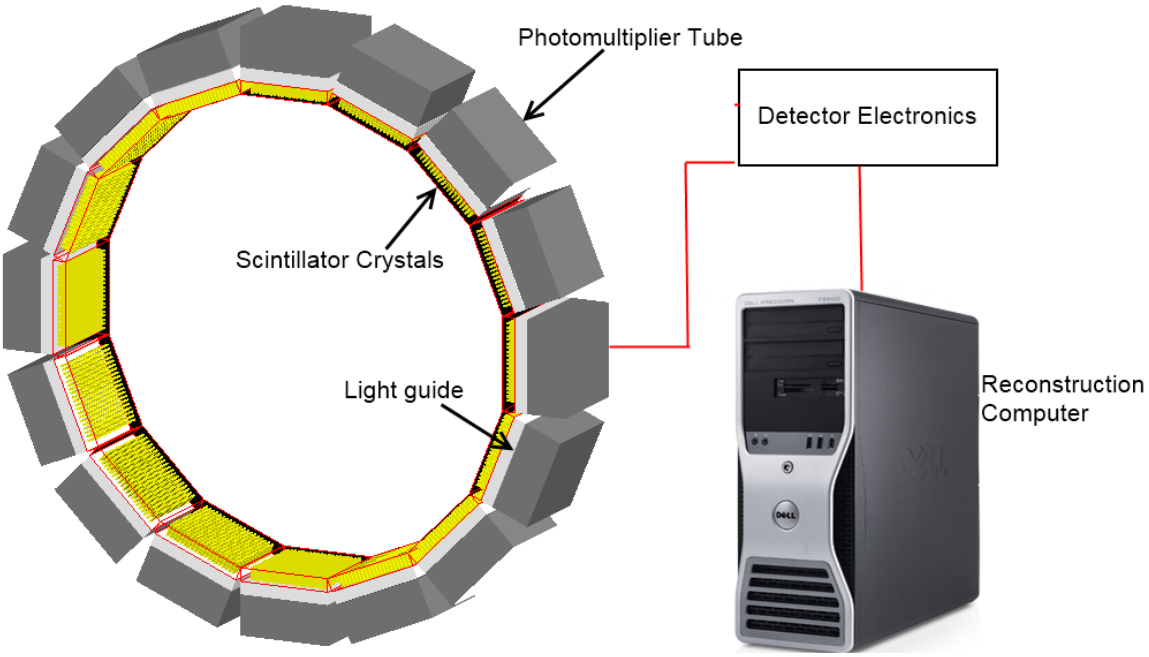

Figure 1.4: Schematic of a simple PET system. The detector ring's raw output signals are first processed by detector electronics to apply energy and coincidence windows before digitizing the signals. The digitized signals are processed using a computer to record and create the final PET image.

subsequently reconstructed into a transaxial image. A schematic of a basic PET system is shown in Figure 1.4.

1.2.2 Scintillator Crystals

Scintillator crystals convert high energy radiation such as gamma rays or annihilation photons into visible light that has a brightness proportional to the energy of the incoming radiation. This occurs mainly through interactions in the crystal lattice as the radiation elevates electrons from their normal positions in the valence band to the conduction band of the material [20]. The two most common methods of photon-electron interaction at the energies seen in PET (511 keV and lower) are via the photoelectric effect and Compton scattering.

The photoelectric effect occurs predominantly at energies below 0.5 MeV, and is a result of photons interacting with a tightly bound inner shell electron. An absorbed photon gives up all of its energy E to the electron, ejecting it from its orbit at an angle θ and a kinetic energy

$$T = E - E_b \tag{1.2}$$

where E_b is the binding energy of the electron with the nucleus. The nucleus recoils slowly to conserve momentum, but its kinetic energy is very nearly zero compared to the electron (Figure 1.5). For a given number of photons N_0 interacting with a thickness x of material with a density ρ, the number of attenuated photons N due to the photoelectric effect can be expressed as

$$N = N_0(1 - e^{-\frac{\tau}{\rho}x}) \tag{1.3}$$

where $\frac{\tau}{\rho}$ is the material's mass attenuation coefficient for the photoelectric effect. This quantity is proportional to

$$\frac{\tau}{\rho} \propto \frac{Z^3}{E^3} \tag{1.4}$$

where Z is the atomic number of the material and E is the photon energy. Photoelectric events occur much more frequently in higher Z materials, but also diminish rapidly as photon energy rises [21].

Compton scattering occurs when the incident photon transfers only a portion of its energy to a loosly bound electron that it collides with. In this case, a photon is scattered at an angle θ with a lower energy E'. The scattering angle and energy of the scattered photon are related by the equation

$$E' = \frac{E}{1 + (E/511\text{keV})(1 - \cos\theta)} \tag{1.5}$$

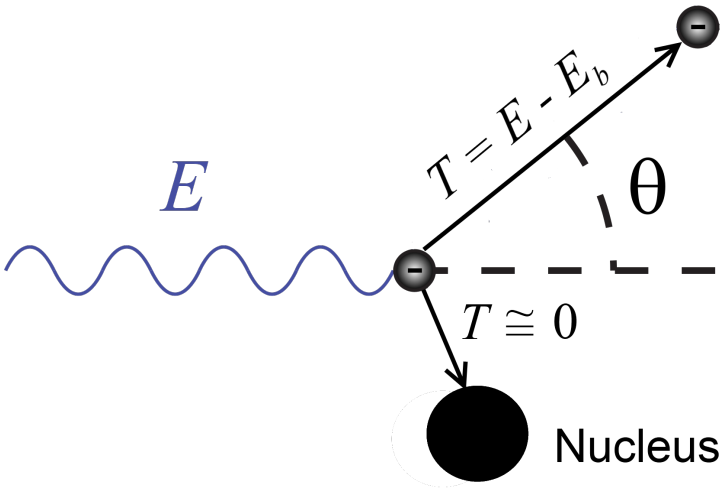

Figure 1.5: Kinematics of the photoelectric effect. An incoming photon gives up all of its energy E to an inner shell electron; the electron is ejected at an angle θ with energy T, while the nucleus recoils slowly to conserve momentum.

where 511 keV is the rest mass energy of the electron impacted by the photon. The energy T imparted to the electron is simply the difference in energy between the incident and scattered photons ($E-E_b$) (Figure 1.6). The Compton mass attenuation coefficient of a material $\frac{\sigma}{\rho}$ is given by

$$\frac{\sigma}{\rho} = \frac{N_A Z}{A}\sigma_e \tag{1.6}$$

where N_A is Avogadro's number, Z is the atomic number, and A is the molecular mass of the material. σ_e is the Klein-Nishina scattering cross-section that is determined by

$$\sigma_e = 2\pi r_e^2 \left[\frac{1+\alpha}{\alpha^2}\left(\frac{2(1+\alpha)}{1+2\alpha} - \frac{ln(1+2\alpha)}{\alpha}\right) + \frac{ln(1+2\alpha)}{2\alpha} - \frac{1+3\alpha}{(1+2\alpha)^2}\right] \tag{1.7}$$

where r_e is the classical electron radius and $\alpha = E/m_0c^2$; E is the energy of the photon and m_0c^2 is the electron rest mass energy of 511 keV [22]. Compton interactions are more dominant at photon energies above 600 keV for high-Z materials such as

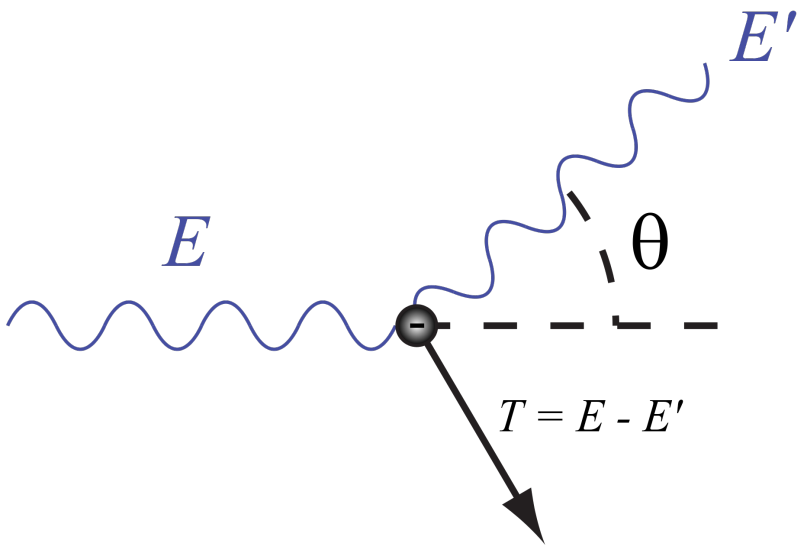

Figure 1.6: Kinematics of the Compton effect. An incoming photon with energy E collides with an electron and scatters at an angle θ with a reduced energy E'. The kinetic energy of the electron is equal to the difference in photon energy before and after the scatter event ocurs. Figure modified from Compton-scattering.svg by Wikipedia user Jabberwok, released under Creative Commons Attribution ShareAlike License 3.0.

scintillator crystals, but it is still a significant interaction at energies seen in PET.

Electrons excited during a photoelectric or Compton interaction ultimately excite the electrons in the scintillator crystal lattice into the conduction band. As the excited lattice electrons return to their original energy state in the valence band they release this energy in the form of another photon. The energy gap is usually near the same energy as the incoming radiation that excited the electron initially, causing the re-emitted photon from pure inorganic scintillator crystals to have an energy above the visible range . To overcome this issue, a small amount of a compound known as an activator (or dopant) is added to the crystal structure that allows electrons to decay to an energy state between the conduction and valence bands. The gap between the activator's excited and ground states is small enough to produce photons in the visible

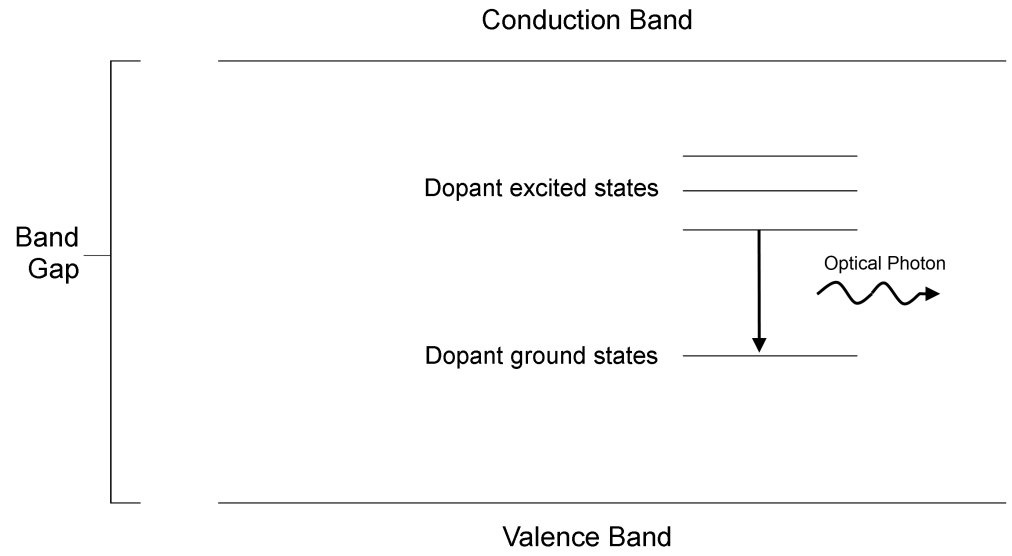

Figure 1.7: Example of how an activator allows the energy of a captured gamma-ray to be converted to visible light. An electron is lifted from the valence band to the conduction band of the crystal, where it decays to a lower energy state of the dopant material. This then decays to a dopant ground state whose separation from the dopant excited state is sufficient to produce optical photons.

range and become the basis of the scintillation process (Figure 1.7).

As the electron-hole (e-h) pairs produced by ionizing radiation travel through the crystal, the positively charged holes drift quickly to activator sites and ionize them. The electron from the pair can then drop down to the ionized activator, creating a new activator state. If this state is excited, it quickly decays while emitting a visible photon; the half-life of these excited states is roughly 50-500 ns. It usually takes roughly 3 times the band gap energy to create an electron-hole pair, and not every new activator state leads to a useful visible light photon, resulting in incomplete conversion of ionization energy to visible light. The ratio of energy out via visible photons to the incoming energy of the ionization radiation is the conversion efficiency, and is often expressed as the light yield in photons/MeV. The scintillation light can

Table 1.1: Properties of some common PET scintillators. Data taken from [23] and [24]

Scintillator	NaI(Tl)	BGO	LSO	GSO	$LaBr_3$	LuAP
Effective atomic no. (Z_{eff})	51	74	66	59	47	65
Density (gm/cm^3)	3.67	7.13	7.4	6.7	5.3	8.34
Linear attenuation coeff. (cm^{-1})	0.34	0.92	0.87	0.62	0.47	0.9
Light yield (photons/MeV)	41 000	6 150	30 750	12 300	65 600	6 560
Decay constant (ns)	230	300	40	65	25	18

escape the crystal as its energy is too low to create an e-h pair; the emission spectrum from the activator states is in the range of wavelengths that are not in the crystal's absorption band [20].

Scintillator crystals used for detecting annihilation photons in PET must have a high effective atomic number (Z_{eff}) to provide a high stopping power for 511 keV photons and reduce the crystal thickness needed in a detector; photon absorption due to the photoelectric effect increases roughly proportionally to Z_{eff}^3 as discussed earlier. The crystal should also have a high light output to maximize the energy resolution of the system as well as a short decay time to optimize data acquisition at high count rates. Most scintillators involve a compromise between these different goals (Table 1.1). Bismuth germanate (BGO) crystal has a very high stopping power but low light output and a long decay time, while sodium iodide (NaI) crystal has a very high light output but relatively poor stopping power and a decay time that is still considered long for PET applications. Lutetium oxyorthosilicate (LSO) crystal offers a very good balance of these properties, which has led to it becoming the scintillator of choice for PET systems [24]. LSO combines a high light output with a very high attenuation coefficient (Figure 1.8) that reduces the length of crystal needed to detect 511 keV photons compared to other scintillators, allowing detector crystals to be made smaller while maintaining the small attenuation length and high light output needed

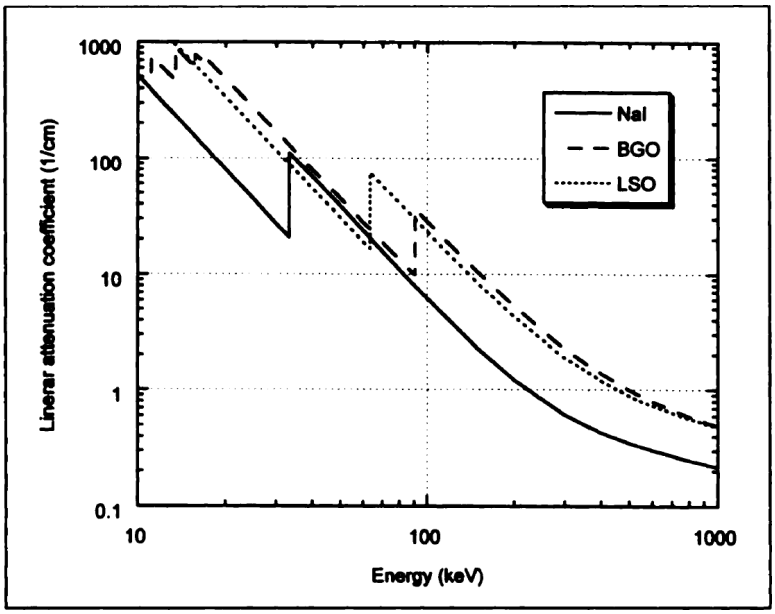

Figure 1.8: Plot of attenuation coefficients for NaI, BGO, and LSO scintillators. LSO's attenuation coefficient is nearly as high as BGO at 511 keV, while NaI's attenuation coefficient drops significantly at energies above 200 keV. Reprinted by permission of the Society of Nuclear Medicine from [25], Figure 1.

to maintain the spatial and energy resolution of the system respectively. Its fast decay time also allows for a high count rate with reduced dead time after each detected event. This leads to a better system timing resolution than slower scintillators, resulting in a lower chance of seeing random coincidence events than is seen when using scintillators with long decay times.

1.2.3 Photomultiplier Tubes

Photomultiplier tubes (PMTs) are the most commonly used devices for detecting light from scintillator crystals. A photocathode on the PMT face releases electrons via the photoelectric effect when struck by photons. These electrons are then accelerated in a high voltage electric field into a metallic surface inside the PMT known as a dynode. As the electrons smash into the dynode they liberate a greater number of electrons

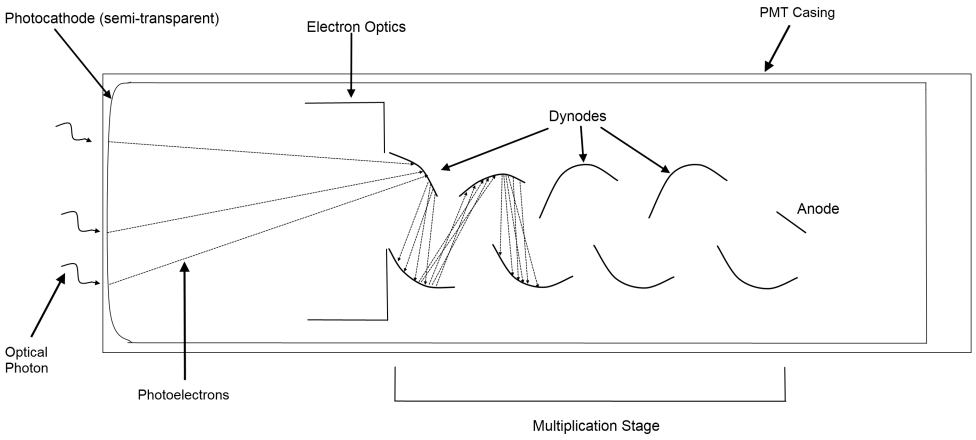

Figure 1.9: Cut-away view of a traditional photomultiplier tube (PMT). Photoelectrons are liberated when photons strike the photocathode, and are accelerated under high voltage. Electron optics steer them to the dynodes which are also held at a high voltage bias. Each collision liberates more photoelectrons than were incident due to the high velocity at impact, resulting in more total electrons at each dynode stage. At the end of the multiplier chain is the anode where charge is collected and read out.

which are accelerated into another dynode further along the PMT length (Figure 1.9). The number of electrons continues to increase at each dynode stage until they reach the tube anode, where the gathered charge is read out (usually by a preamplifier to convert charge to voltage). The size of the output pulse is proportional to the number of photons that struck the PMT face and is typically on the order of 10^6 [26].

1.2.4 PET Detectors

Most modern PET systems use block detectors with arrays of pixellated crystals to provide higher spatial resolution than was possible with single monolithic crystals coupled to PMT arrays. Light sharing among the group of PMTs is used to determine which crystal the event occurred in [27]. To avoid the high cost and space requirements of having each crystal block read out by a group of PMTs, a single position sensitive

PMT (PSPMT) is often used in preclinical PET systems instead. The location on the PMT face where light from a scintillation event struck the surface is determined by using multiple dynode channels (Figure 1.10A) cause photoelectrons to be accelerated in a path parallel to their incidence location on the PMT surface. A series of wires with a resistor network (Figure 1.10B) are used to determine the position of an interaction based on the voltage at each readout point (X^+,X^-,Y^+,Y^-); the voltage in the X/Y direction corresponds to events occuring further to the right/top of the network respectively since the voltage decreases as the signal travels through more resistors on its way to the readout point [28]. The total energy of an event can be determined by summing the voltage at all four readout points [29]. The position of the event X, Y is determined using modified Anger logic [30]:

$$X = \frac{X^+ - X^-}{X^+ + X^-} \tag{1.8}$$

$$Y = \frac{Y^+ - Y^-}{Y^+ + Y^-} \tag{1.9}$$

X and Y both range from -1 to 1, with a value of zero corresponding to the an event in the centre of the PSPMT.

If a crystal array is uniformly irradiated and a large number of counts (a few hundred thousand to a few million) are recorded, the position information of each event can be used to create a flood histogram where the brightness of each pixel in the histogram is proportional to the number of counts that occurred at that location (Figure 1.11). A flood histogram is split into a series of x and y bins, usually a square array. The X and Y positions of each event calculated in equations 1.8 and 1.9 are assigned to flood histogram bins x or y via

$$x = X \left[\frac{\text{flood width}}{2}\right] + \left[\frac{\text{flood width}}{2}\right] \tag{1.10}$$

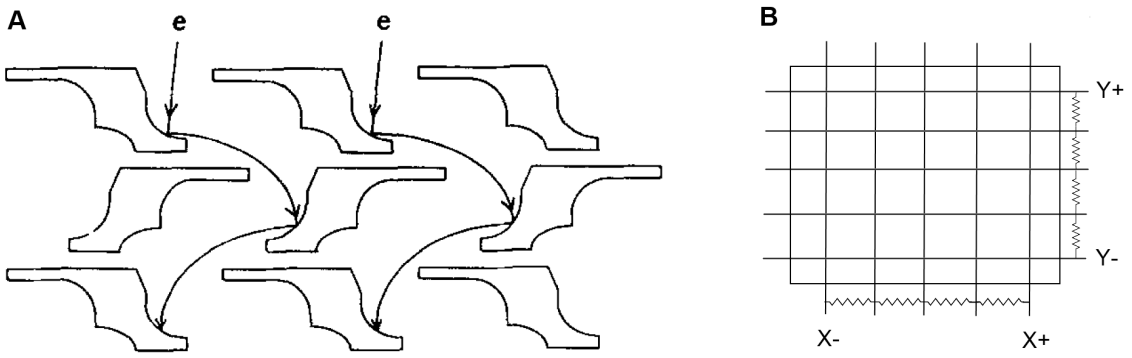

Figure 1.10: Typical position sensitive photomultiplier tube (PSPMT) functionality. A series of multi-channel anodes (Part A) are used to funnel electrons parallel to their interaction point on the PMT face. These anodes are read out with a resistor network (Part B) to determine where along the anode surfaces the electrons stopped, corresponding to where they struck the PMT surface. Part A is reprinted from [29], Copyright 1997, with permission from Elsevier.

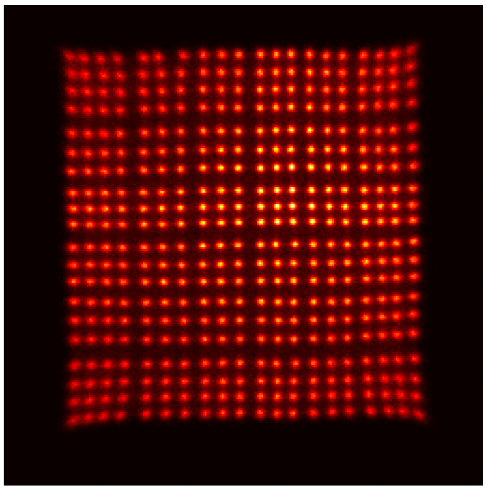

Figure 1.11: Flood histogram created using a Siemens Inveon detector with a 20 $\times$ 20 crystal array. This flood contains 3 million counts from a ^{22}Na point source.

$$y = Y \left[\frac{\text{flood height}}{2} \right] + \left[\frac{\text{flood height}}{2} \right] \tag{1.11}$$

The histogram bins x and y are rounded to an integer value that correspond to the x and y positions in the flood histogram image respectively, and range in value from 1 to the flood height or width as appropriate for that dimension. The width and height of the flood are usually set to values such as 128, 256, or 512 pixels; higher values require more counts to fill the histogram. The flood histogram is usually segmented to assign events located in a particular region to a particular crystal. This allows each crystal in the array to have its energy response calibrated individually removing any effects of variation in response for each detector element. The spatial location of the detected event in the scanner is calculated via the flood histogram as well since the position of each crystal in the scanner should be well defined if the detector modules are installed correctly.

1.3 PET Data Processing

Once an annihilation photon is detected, it is processed by the data acquistion electronics of the PET system. The energy of each event is then examined by adjustable energy discrimination circuits to determine if the coincidence falls within an energy window defined by the lower level discriminator (LLD) and the upper level discriminator (ULD). Ideally the window would be set to match the system's energy resolution at 511 keV, rejecting detected photons at other energy levels that correspond to scatter. However, as mentioned earlier even unscattered photons may interact in the scintillator crystal via Compton interactions that deposit less than 511 keV. For this reason, setting the energy window narrowly around the 511 keV photopeak would drastically reduce the system's sensitivity [31]. The energy window is usually set to

values of 425-650 keV in clinical scanning [7] to provide a compromise between scatter rejection and system sensitivity. In small animal studies a lower LLD setting (200 or 250 keV) can be used to increase system sensitivity as the considerably smaller thickness of a rat or mouse compared to a human provides a much lower chance of 511 keV annihilation photons scattering before exiting the imaging subject. This leads to a greater chance that a detected event with an energy less than 511 keV is actually a 511 keV photon that did not deposit its full energy in the detector.

If two detected events are within the energy window and both occur within a very small time window (usually 6 to 10 ns), they are considered to be a valid coincidence event and a LOR is drawn between the two detectors the events were recorded in. Coincidence events are classified as either true, scattered, or random. A true coincidence event simply detects a pair of annihilation photons from a single annihilation event and the LOR drawn between them passes through the annihilation location (Figure 1.12, left diagram). If one or both of the annihilation photons are Compton scattered a "scatter" coincidence is recorded since both photons come from the same annihilation. The LOR drawn between the two photons will be incorrect, however, resulting in the annihilation being positioned incorrectly in the image (Figure 1.12, centre diagram). Random coincidences result when two photons (or other particles) are detected within the coincidence window but are not from the same annihilation event (Figure 1.12, right diagram). This results in a LOR whose position is not related to the tracer distribution in the imaging subject, and occurs at a rate proportional to the single event count rates S_1 and S_2 in each detector:

$$R_{randoms} = 2\tau S_1 S_2 \tag{1.12}$$

where τ is the coincidence window length. A correction for random coincidences is

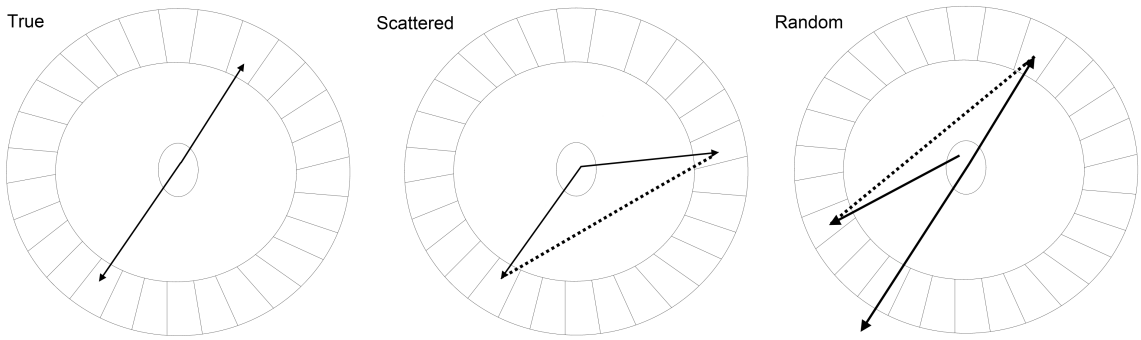

Figure 1.12: Examples of true, scattered, and random coincidence events. Lines of response (dotted lines) do not cross the annihilation location in scattered or random coincidences, resulting in image noise.

often performed by estimating the number of randoms from the previous equation and subtracting them from the data set. Scattered and random coincidences are visible in the uncorrected image as background noise, which reduces image contrast [32].

Each LOR is stored in a data structure known as a sinogram, which represents tomographic data as a 2D array with the y-axis corresponding to the LOR's azimuthal angle and the x-axis representing the LOR's displacement from the scanner's central axis. The number of events at each angle and displacement is recorded over the scanning period, creating an image after collecting many thousands to millions of events. Simple objects such as line or point sources appear as sine waves (Figure 1.13), while more complex objects have sinograms that are the result of many overlapping sine waves. Since most PET systems contain multiple crystal rings, a separate sinogram "slice" is created for each ring in coincidence with itself and with any other possible ring pairing; These slices correspond to the position along the scanner axis. The slices are generally organized by the difference between the ring numbers where the coincidence events are recorded. A ring difference of 0 corresponds to coincidences within a single crystal ring, +1 and -1 are with the adjacent ring, +2 and -2 are between rings 2 apart, and so on (Figure 1.14) [33]. For the 80 rings of the Siemens

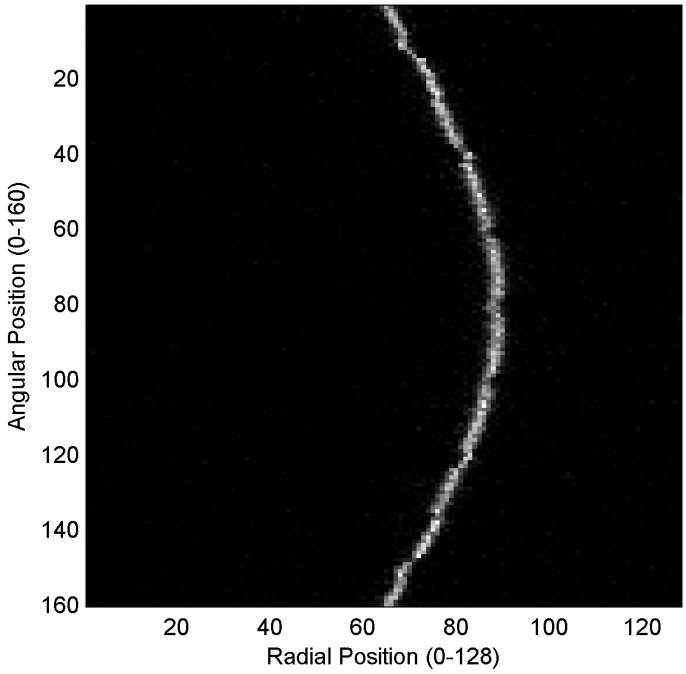

Figure 1.13: Central sinogram from a line source in a PET scanner. The source is offset slightly from the center of the field of view; the amount of offset is equal to the amplitude of the "wave."

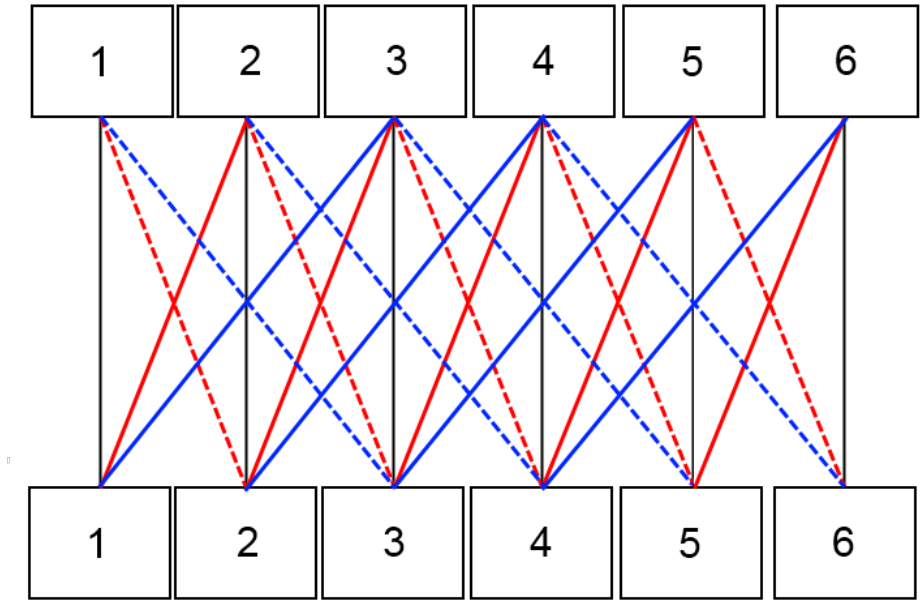

Figure 1.14: Side view of a simple 6-ring PET scanner showing only top and bottom detectors. A ring difference of 0 corresponds to the solid black lines which show coincidences only between detectors within the same detector ring. Dashed red and blue lines show coincidences between detectors with ring differences of +1 and +2 respectively, while solid red and blue lines correspond to ring differences of -1 and -2.

Inveon scanner used in this study, the total number of sinograms with a maximum ring difference of 79 is 6400 (80×80).

Many scanners use large numbers of small crystal rings to increase axial sampling and therefore axial resolution. This can result in a loss of sensitivity per axial slice, however, as each of these smaller crystals will detect fewer counts than a system with fewer larger crystals covering the same area during an acquisition with a fixed number of total counts. Directly reconstructing images that are acquired with all allowed ring coincidences (3D acquisition mode) is also more time consuming than reconstructing data that only allows coincidences within each individual crystal ring. This is mitigated by combining axial data via a process known as single slice rebinning

(SSRB), which is described in [34]. SSRB combines the data from multiple sinogram slices that correspond to a similar axial position to create a new sinogram that has fewer axial slices but a greater number of events per slice; this sinogram can then be reconstructed using traditional filtered back-projection. SSRB loses accuracy when the source distribution is farther away from the scanner's central axis; in this case more advanced algorithms such as Fourier rebinning are used [35].

1.4 Intrinsic Activity in Lu-based scintillators

One complication of LSO and other lutetium-based scintillators is that roughly 2.6% of the lutetium in the crystal is made up of the radioactive isotope ^{176}Lu, which decays via β^- emission followed by a prompt gamma-ray cascade at energies of 307 and 202 keV (Figure 1.15) [36]. The beta particle is almost always detected in the crystal it originated from and if one of the prompt gamma rays is detected in an opposing detector a true coincidence event can be registered (Figure 1.16) that adds a background signal to the PET image since it has no correlation to the tracer distribution in the imaging subject. This intrinsic activity is at a low level (276 Bq/cm^3) and is not noticable during routine scanning with activity levels on the order of several MBq, but it can become an issue when imaging at low activity levels such as in cell tracking studies in small animal imaging; to increase system sensitivity the energy window is commonly set to a wider range than normal, allowing the 307 and 202 keV intrinsic gamma rays to be within the energy window [37][38] (Figure 1.17). Intrinsic activity in the crystals increases the rate of random coincidences by roughly 8% during NEMA standards testing [39], requiring corrections to measurements of a scanner's spatial resolution since these tests require a low count rate to reduce random coincidences, and their results are impaired if additional random coincidences are present in the

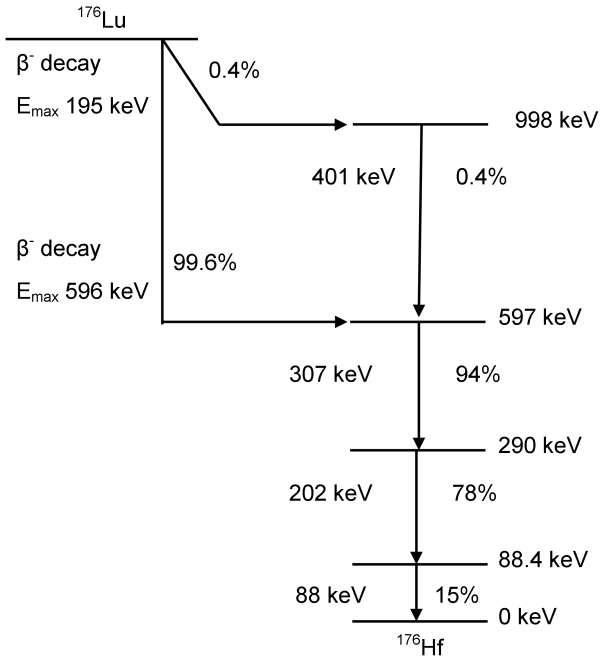

Figure 1.15: ^{176}Lu decay scheme. β^- emission is followed by a prompt gamma ray cascade. Based on data from [36]

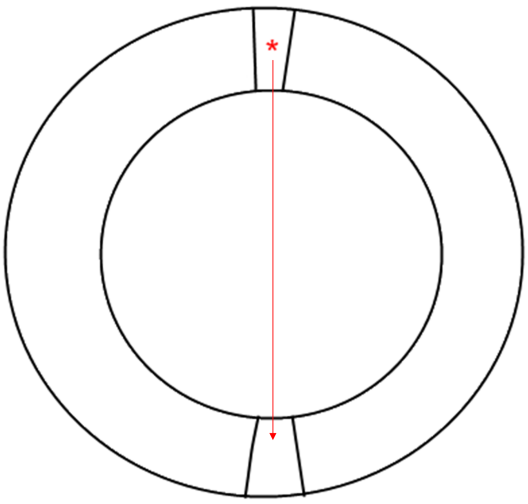

Figure 1.16: Coincidence event caused by intrinsic LSO events. If a β^- (top asterisk) is detected in its originating crystal and a prompt gamma event is picked up in an opposing detector, a true coincidence event is recorded that adds background signal to the PET image.

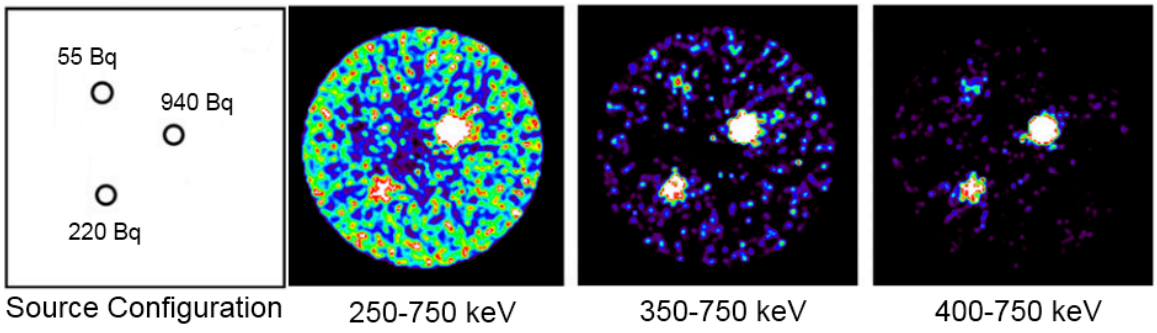

Figure 1.17: Effects of intrinsic activity while imaging a weak source distribution in a microPET R4 scanner. At lower LLD settings (250 or 350 keV), the weakest source is overwhelmed by intrinsic activity. Figure from [38], Copyright 2007 with permission from Elsevier Science

data set. Intrinsic LSO activity can also lead to count losses in a PET system due to increased dead time, since the scanner's data channels and processing time must now process the energy and timing information of intrinsic events as well as detected annihilation photons [39].

Small animal PET systems are more prone to the effects of intrinsic LSO activity than clinical scanners since their comparitively small detector ring diameter causes intrinsic coincidence events to make up a greater fraction of the total number of prompt coincidence events in an acquisition since the amount of crystal used to make up the ring decreases linearly with the detector ring diamater r and the probability of detecting an intrinsic γ-ray *increases* proportionally to the decrease in r^2. LSO-produced gamma rays are also attenuated far less when travelling through animals such as rats and mice than humans, considerably increasing the number of β-γ coincidences especially when a lower energy window setting (such as 200 or 250 keV) is chosen to increase sensitivity. Perhaps the most detrimental effect, though, is that the ability to quantify the activity of a source in the image is severely impaired; as shown in [38], the percentage error in the measured ratio of source activities for a 220 Bq and a 940 Bq source ranges from 27.1% to 32.8% when imaging with a 250-750 keV energy window. This error grows to anywhere from 108.6% to 134.7% when comparing 55 Bq and 940 Bq sources with the same energy window. Raising the LLD setting can dramatically reduce these errors, but at the expense of greatly reduced system sensitivity. Since PET studies may involve precisely quantifying the concentration of a weak radiotracer, the potential effects of ^{176}Lu activity should be carefully considered when designing a new preclinical PET scanner.

1.5 Monte Carlo Simulations

The high cost of detector materials such as scintillator crystals and readout electronics makes the development of novel PET scanner configurations very expensive. To better understand the performance of a scanner that is still in the design stage, computer simulations are often used to create a model to predict the behaviour of the real scanner.

Monte Carlo simulations use a computational model that employs a random number generator to create a model of particle transport based on the statistical probability of an interaction occurring in a medium. The scanner and radioactive source are modelled in software to define geometry volumes and materials according to known physical properties such as density, atomic structure, and x-ray attenuation coefficient. Each radioactive particle produced by a radioactive source has its path divided into a series of steps, and if the particle intersects with a material in the simulation at a particular step a random number is generated. The random number is compared to the statistical probability of an interaction (and of the type of interaction) occurring based on the particle's energy and the properties of the material it is interacting with. If the random number meets the criteria for an interaction to occur, then the interaction (scattering event, photoelectric absorption, etc.) is logged in the simulation results. If no interaction occurred or the particle was not stopped in the interaction then it is moved to the next distance step and the process is repeated until the particle is "killed" or it reaches a pre-defined cutoff point. Averaging many particle histories leads to an approximation of how real particles will interact in a medium, with the statistical uncertainty depending on the number of particles simulated N. Since radiation counting statistics are modelled as a Poisson process, the uncertainty in the model decreases on the order of $N^{-1/2}$ [40].

Increasing the number of histories also leads to a linear increase in computation time, however, as each particle (and the histories of all other particles that it disturbs) must be calculated. This can be mitigated by enforcing cutoffs of particles below a certain energy level or travelling beyond a certain distance (ie. their track would extend beyond the simulated volume), or by ignorning physical processes such as Rayleigh scatter that result in minimal loss of energy and rarely occur at energies of interest in nuclear medicine [41]. Other approximations such as modelling a radioactive source that emits positrons directly (rather than going through the nuclear decay processes that produce positrons) are often made to decrease the number of interactions that must be modelled and hence decrease simulation time. The time needed for Monte Carlo simulations has been mitigated somewhat with continued increases in computer speed as well as the rapid decrease in price of parallel processing systems such as multi-core CPUs and programmable GPUs [42].

GATE (Geant4 Application for Tomographic Emission) is a popular Monte Carlo simulation package for simulating PET systems [43]. It is a macro-based language that interfaces with the well-validated Geant4 particle simulation package [44], providing a relatively user-friendly way of quickly defining a simulation environment without the hassle of learning how to produce Geant4's often ponderous configuration scripts. GATE is specially designed to model the detector electronics used in PET and other nuclear medicine systems, including detailed models of common scintillator materials and the ability to model effects such as energy resolution and windowing, pulse pile-up and detector dead time, and coincidence processing. It also provides easy modelling of the radioactive decay of sources with short half-lives such a ^{18}F and ^{99m}Tc and includes the properties of many common scanner and phantom materials found in PET systems. Changes to the system's geometry are visible immediately using Geant4's robust visualization tools as well. GATE has been extensively validated against phys-

ical results from various clinical and preclinical PET systems [45]. This well validated simulation model combined with its relative ease of use and freeware status has led to GATE becoming a very popular platform for simulating not only conventional PET systems but also new designs that use detectors with novel readout methods [46-50]. An accurate simulation of a scanner can prove very useful when testing a new scanner geometry and/or detector technology, as changes can be made relatively quickly and without the considerable cost of building and rebuilding a prototype system.

1.6 Modelling of ^{176}Lu Intrinsic Activity in GATE

As was discussed in section 1.4, the effects of the intrinsic activity of ^{176}Lu in PET systems have been well documented, and have been shown to be more dramatic when imaging weak sources in small animal scanners. These effects should be taken into account as our research group designs a new scanner that will be used in these types of imaging studies (Figure 1.18), but by default GATE does not include the effects of ^{176}Lu intrinsic activity in its model of LSO or other lutetium-containing scintillator crystals. This can be remedied by adding a ^{176}Lu source inside the crystal volume as was seen in [51], but to date there has been no systematic comparison of a simulation model against measurements from a physical PET system. This project seeks to validate a GATE model of intrinsic ^{176}Lu activity against measured results from a bench-top system with two detector modules from a Siemens Inveon dedicated preclincal PET system as well as against measurements from a complete Inveon scanner. Once this model is successfully validated, future scanners that may potentially be greatly affected by this intrinsic activity can have these effects modelled with confidence. This model may also be used to help develop imaging protocols for existing preclinical scanners to better optimize settings for weak-source acquisitions.

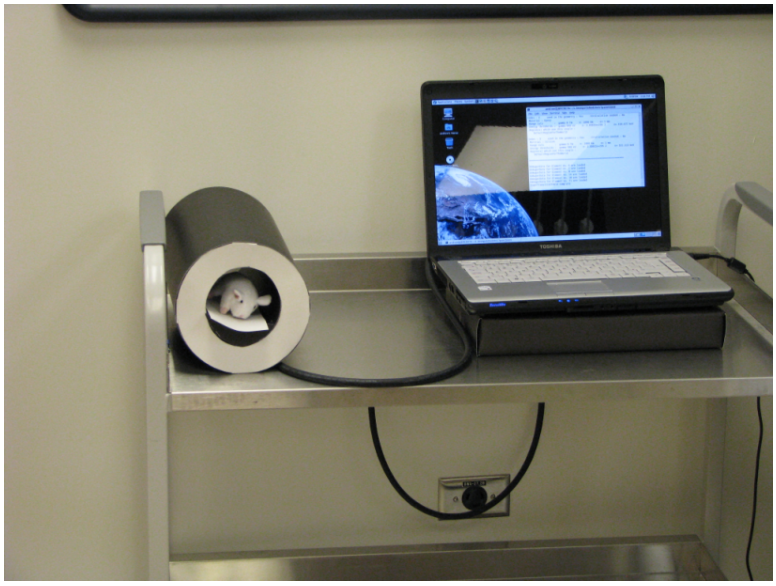

Figure 1.18: Mock-up of proposed ultra-portable PET scanner. Output from the scanner is processed by a small readout electronics system (black box), and images are stored and reconstructed on a notebook computer.

This thesis will first describe the hardware used to acquire data from the bench-top PET detector system, as well as the manner in which the data were analyzed. Details of the data acquisition settings will follow, as well as a description of the setup of the GATE simulations of the bench-top apparatus. After a discussion of the results of the bench-top comparison, the setup of the measurements and simulations of the complete Inveon system will be detailed and the results from the full system comparison will be presented.

Chapter 2

Bench-Top Data Acquisition

2.1 Data Acquisition Hardware

Commercially produced PET systems are optimized for producing images for the end user and for calibrating the system via a procedure designed by the manufacturer. This can make it difficult to gather detailed information about detector performance, especially raw coincidence event data as most systems use singles mode to generate energy spectra during calibration. Using a bench-top system that acquires data directly from the detectors themselves allows us to avoid these limitations and gather detailed data about each detected event. This project's bench-top system used a pair of detector modules from a Siemens Inveon dedicated preclinical PET system [52] to create a setup similar to a stationary 2-head PET system. The detectors were read out using nuclear instrumentation module (NIM) electronics via a custom-built readout board, and data from the detectors were then sampled using a PC-based data acquisition (DAQ) card controlled by custom software developed using LabWindows CVI. Multiple projections were not acquired in this study, precluding the possibility of creating a tomographic image, but the direct readout of the detectors allowed us to

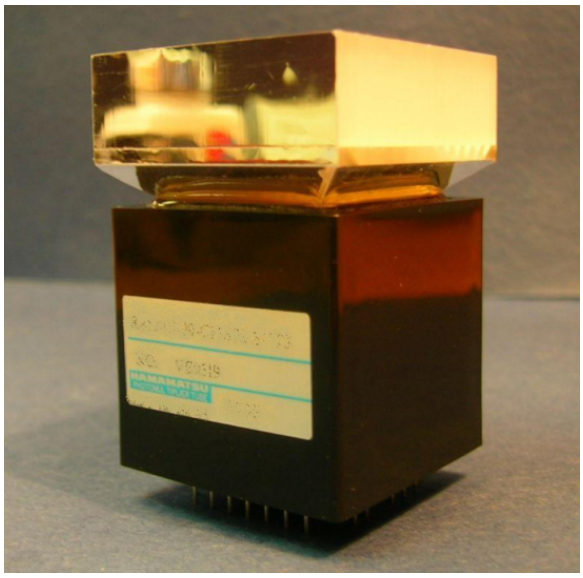

Figure 2.1: Sample assembled Inveon block detector. A LSO crystal array (top) is coupled to the PMT face using a glass light guide with minimal distortion. Photo from [52], ©2007 IEEE.

do a thorough study of the coincidence energy spectrum from LSO intrinsic activity. This allowed a carefully controlled comparison to data acquired from a simulation of this bench-top setup, as all data acquisition parameters were well-defined.

2.1.1 Inveon Block Detectors

Block detectors from the Siemens Inveon system [52] (Siemens Preclinical Imaging, Knoxville, TN, USA) are composed of a 20 x 20 array of 1.51 x 1.51 x 10 mm^3 LSO crystals. A glass light guide is used to couple the array to a Hamamatsu R8900-C12 PSPMT (Hamamatsu Photonics K.K., Iwata City, Shizuoka Pref., Japan); without this light guide the relatively large crystal array (32 $\times$ 32 mm) would exceed the size of the PSPMT face (23.5 $\times$ 23.5 mm) (Figure 2.1). A preamplifier stack supplied by Siemens uses a resistor chain to multiplex the PSPMT's 12 anode outputs (6 x-direction, 6 y-direction) into 4 signals corresponding to X^+, X^-, Y^+, and Y^- in a

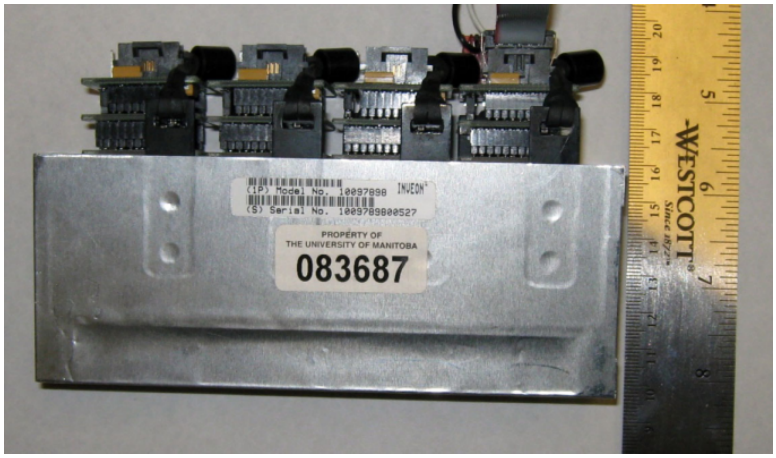

Figure 2.2: Inveon detector module that contains 4 block detectors and their preamplifier stacks in a light-tight aluminum enclosure.

manner described in subsection 1.2.4. The preamplifier stack is also used to provide the -800 V detector bias high voltage supplied by an Ortec 556H power supply. Four detector blocks are placed side-by-side in a light-tight aluminum enclosure to create one complete detector module (Figure 2.2). A 20-pin ribbon cable provides a 5V readout bias voltage to the detector and also carries the multiplexed detector output channels. These 4 output signals are split out to BNC connectors using custom readout boards (AP Circuits, Calgary, AB, Canada) that plug into a prototype board (Figure 2.3). Signal inverters (Phillips 460, Phillips Scientific, Mahwah, NJ, USA) convert the positive pulses from the detectors into negative pulses that can be used by the NIM equipment.

2.1.2 NIM Signal Processing

Each channel of detector output is first processed by a fast 16-channel preamplifier (Phillips 778). The fan-out of the preamplifier allows the signal to be split into two identical output paths without any voltage drop; one path is used for generating timing signals, while the other is used for pulse shaping and sampling. This amplifier

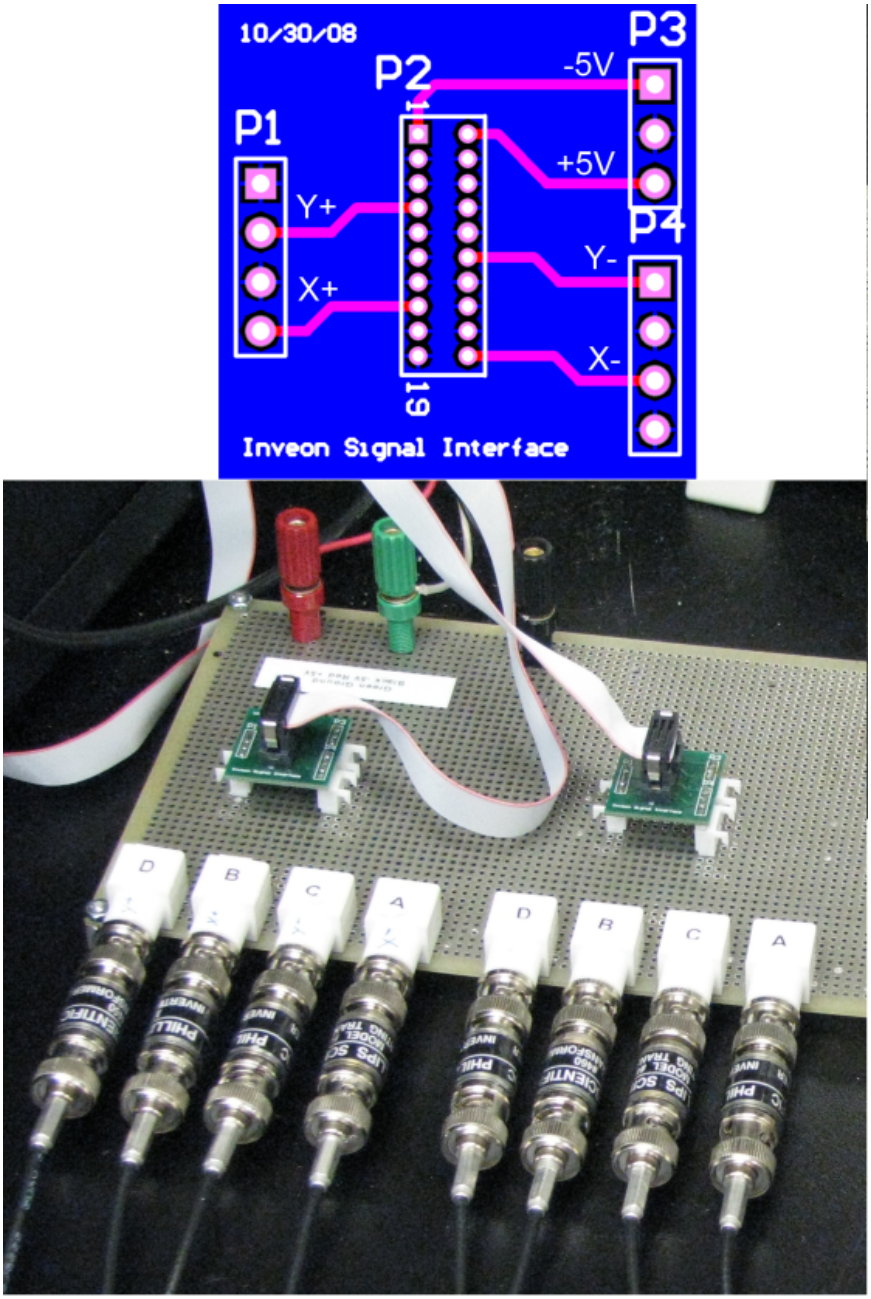

Figure 2.3: Custom readout board circuit diagram (top). These boards receive the ribbon cable connection from the detectors and split it out into 4 signals for each detector via the prototype board (bottom).

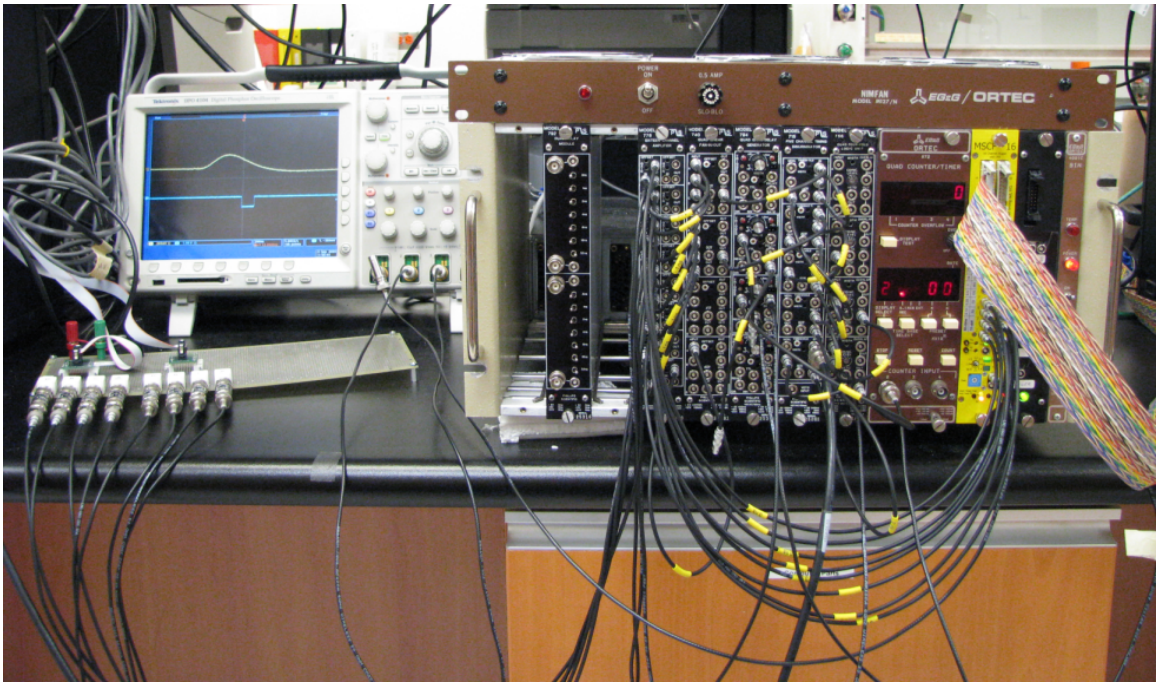

Figure 2.4: NIM detector electronics used for this experiment.

performs minimal shaping on the signal (Figure 2.5), increasing the amplitude without significantly modifying the pulse width. This amplifier was set to its minimal gain of 2X for this study and has a 200 MHz bandwidth to ensure that it can process fast signals without noticable attenuation.

The timing path (red line, Figure 2.6) leads first to a fan-in/fan-out module (Phillips 740). This module is used to sum the signals from all four channels of each detector block, giving an output pulse that is proportional to the total energy of a detected event. This output is fed to a constant fraction discriminator (CFD, Phillips 715). Constant fraction timing is a technique that first makes an inverted copy of the original signal pulse. This inverted pulse is delayed slightly (1 ns in this study), and then added to the original pulse. The sum of these two pulses will then have a zero crossing at a particular fraction of the original pulse amplitude (Figure 2.7). A NIM logic pulse is produced by the CFD at this zero-crossing point, providing

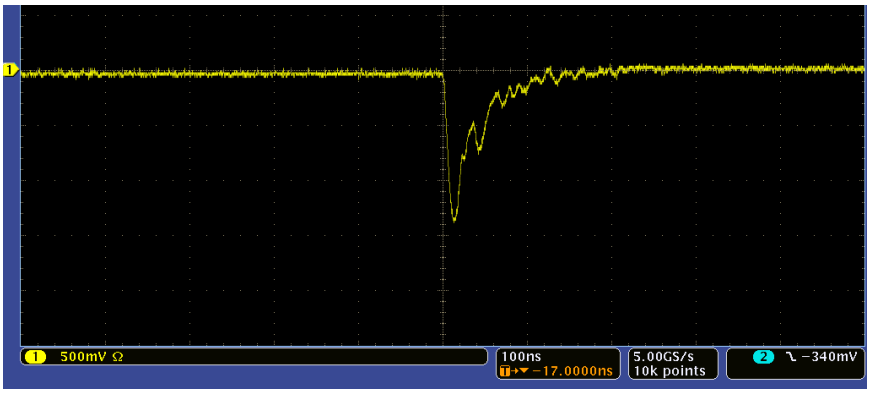

Figure 2.5: Example of a single detector output after processing by the Phillips 778 preamplifier. The 100 ns wide pulse is minimally shaped, and has had its amplitude raised to roughly -1.5 V.

a trigger that occurs at the same time to provide a trigger that is independent of the signal amplitude. The length of the delay time determines what the fraction of the original pulse is, and this delay length is selected by connecting a cable with a known signal propogation time between the delay terminals on the CFD. This is in contrast to leading edge timing methods where the logic pulse is produced once a particular voltage amplitude is reached; two signals of differing amplitudes that have the same rise and fall times would have triggers at different points in time, missing the peak of the larger pulse if the trigger point were set for a smaller one. If the leading edge trigger is set to 10 V and an appropriate delay is chosen before sampling the pulse, a pulse with a larger amplitude would not sample at the pulse peak (Figure 2.8).

When performing measurements in coincidence mode, a coincidence logic unit (Phillips 756) is used to create the trigger signal. This device takes the CFD output from two detectors, and would produce a NIM logic pulse if the two CFD signals overlapped. The size of the coincidence window was equal to the sum of the widths of both CFD output pulses, in this case 10 ns each for a total 20 ns coincidence window.

As the pulse from the detectors is extremely short (roughly 100 ns) and has a

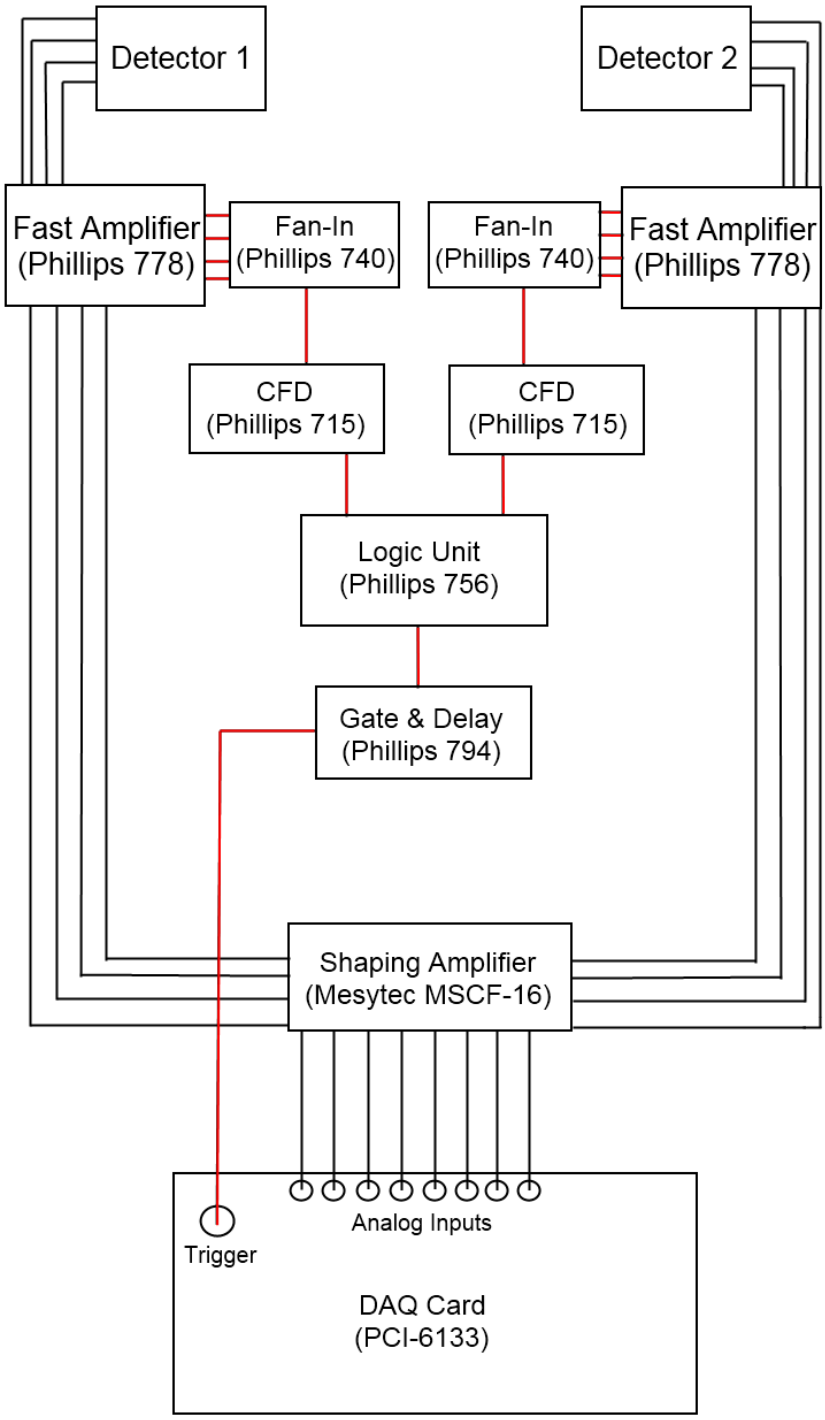

Figure 2.6: Block diagram of bench-top NIM electronics. Once the signals from the 2 detectors are processed by the fast amplifier, they are split to a shaping path leading to the shaping amplifier (black lines) and a timing path (red lines). The DAQ card samples from all 8 analog inputs simultaneously when a trigger pulse is received.

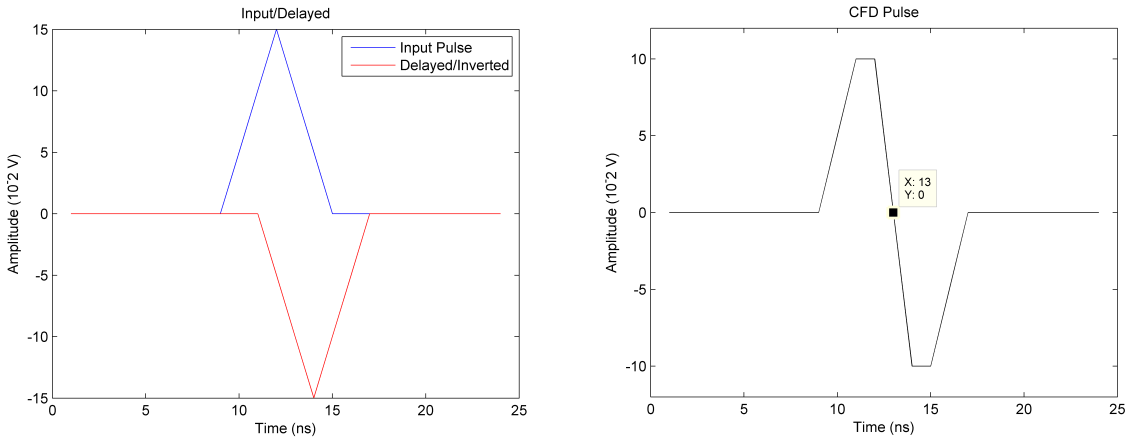

Figure 2.7: Constant fraction pulse timing. The original pulse (positive pulse, blue) is inverted and delayed (negative pulse, red line) before summing the two together (black line, right plot). The CFD produces a trigger pulse at the zero-crossing point (marked with a square data point), which occurs at a given fraction of the original pulse amplitude.

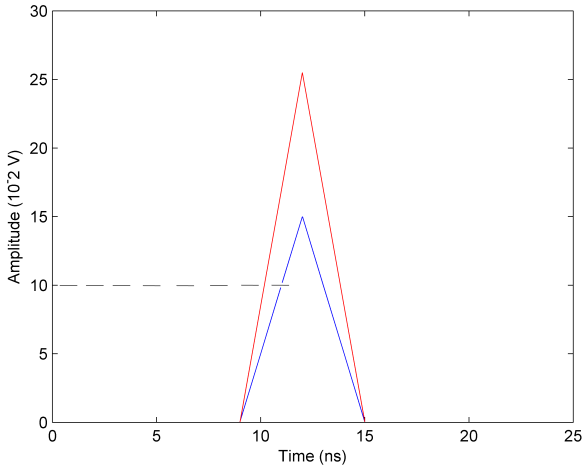

Figure 2.8: An example of how leading edge timing can lead to timing walk. If the leading edge trigger is set to 10 V (dashed line) and an appropriate delay is chosen to sample a given pulse (blue line), a pulse with the same width but a higher amplitude (red line) will be sampled at a point that is not at its peak since the trigger amplitude is reached sooner than the smaller pulse.

much shorter rise time than fall time, additional pulse shaping and amplification is necessary to achieve good sampling with the 1.3 MHz bandwidth of the DAQ card. A multi-channel charge integrating shaping amplifier (MSCF-16, Mesytec GmbH & Co., Putzbrunn, Germany) is used to shape the pulse, increasing the amplitude and pulse width to provide the best possible signal to our sampling hardware. The charge integration acts as a low-pass filter removing high-frequency noise components in the pulse as charge builds in the circuit's capacitors. Allowed shaping times on the MSCF-16 are 120, 250, 500, and 1000 ns, and the gain settings vary from 1X to 20X in 16 steps of 1.22X. The MSCF-16 was set to a shaping time of 250 ns and a gain of 13.42X (setting 11 out of 16).

The length of time needed to shape the pulse is much longer than the time needed to produce the timing pulse in the CFD or coincidence unit (a few dozen ns). For this reason the trigger pulse is run into a gate/delay generator (Phillips 794). This module creates a user adjustable variable-width gate pulse when it receives a trigger input from the CFD. At the end of the gate pulse a positive TTL pulse is generated that is used as the sampling signal for the DAQ hardware. In this way, the sampling signal can be adjusted so that it appears simultaneously with the peak of the shaped pulse. A 1 μs lockout window is also applied after each TTL pulse to prevent a new trigger from reaching the DAQ card before the previous acquisition was complete.

2.1.3 DAQ Card

The DAQ card used for this project was a National Instruments PCI-6133 (National Instruments, Austin, TX, USA). This card is capable of simultaneously sampling 8 analog input channels with 14-bit resolution and a 1.3 Mhz bandwidth at a maximum rate of 2.5 million samples per second, allowing the simultaneous readout of two detectors. The card is equipped with a very large (16 million sample) first-in-first-out

(FIFO) hardware buffer, and uses a PCI-interface to transmit data directly to the acquisition computer's memory bus. The card digitizes the voltage on each channel simultaneously whenever a trigger signal from the gate/delay generator is received on the PFI3 trigger input and stores the recorded voltage at that time in the hardware buffer.

During testing it was found that to accurately sample the peak of the voltage pulse from the shaping amplifier the trigger signal needed to be delayed to a point 160 ns after the pulse peak. This is due to the signal being modulated by the 1.3 MHz bandwidth of the PCI-6133; the faster higher-frequency components of the pulse were filtered out, leaving the slower low-frequency components which peaked at a slightly later time than the complete pulse.

2.2 Data Acquisition Software

The data acquisition software for this project was created using National Instruments LabWindows CVI 8.5.1. This development environment allows a user to build a graphical user interface (GUI) by simply dragging and dropping elements such as numerical displays and command buttons onto a window "canvas." The GUI for the bench-top data acquisition (Figure 2.9) is able to acquire data from up to 8 channels simultaneously, and is capable of being set to stop acquisition after a given number of counts or a preset amount of time has elapsed. A multi-channel analyzer function is capable of showing an energy spectrum for all acquired channels simultaneously, and has user definable axis limits. Flood histograms (described in subsection 1.2.4) from multi-channel detectors can also be shown on a separate pop-up panel. The computer used for data acquisition was a Dell Precision T3400 workstation (Dell Computers, Austin, TX) with 4 GB of RAM and an Intel Q6700 CPU clocked at 2.67 GHz, with

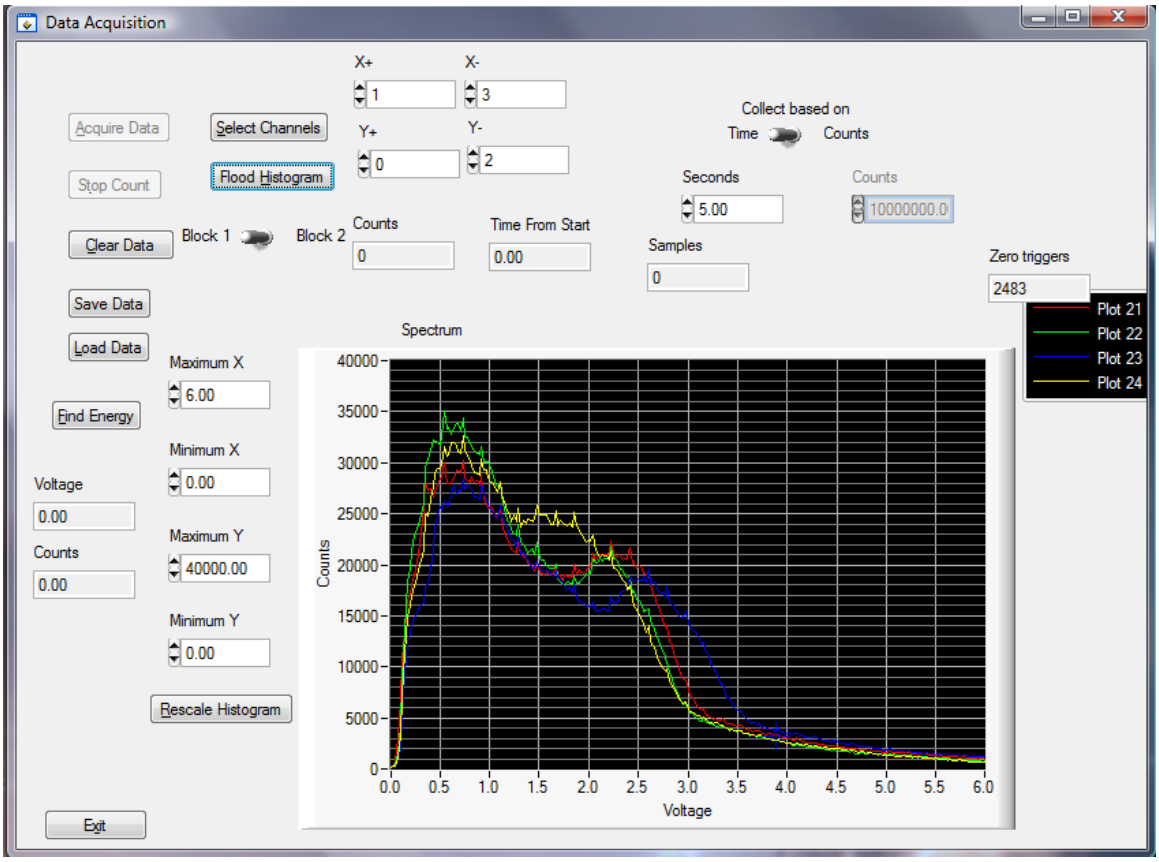

Figure 2.9: Data acquisition software for the bench-top detector system. Energy spectra from a 4 channel acquisition are shown on the program's display.

Windows Vista Business SP2 (64-bit version) used as the host OS. Source code for this program is included in Appendix A.

2.2.1 Acquisition Details

Data is acquired using the DAQmx libraries provided by National Instruments. An invisible Timer control on the front panel of the program ticks every 0.01 seconds, reading any events in the DAQ card's hardware buffer and storing them in memory on each timer tick. This process continues until either the maximum number of counts is reached or the specified amount of acquisition time has elapsed. The program then automatically creates an energy spectrum by histogramming the data into 256 voltage bins and displays it on the large graph on the GUI. Data is saved in comma-separated ASCII format when the user clicks the "Save Data" button, and can be loaded into memory later on using the "Load Data" button.

If a 4-channel position sensitive detector was used for data acquisition, the program can display a flood histogram by clicking the "Flood Histogram" button. A new interface panel pops up over the main GUI, displaying a gray-scale 256x256 image where pixel brightness corresponds to how many counts were present at that location in the histogram.

2.3 Bench-Top Data Processing

Data analysis was performed using scripts developed in MATLAB R2007b (The Mathworks Inc., Natick, MA, USA). After first calibrating the detectors' energy response, energy spectra are created by histogramming the energy of each detected event.

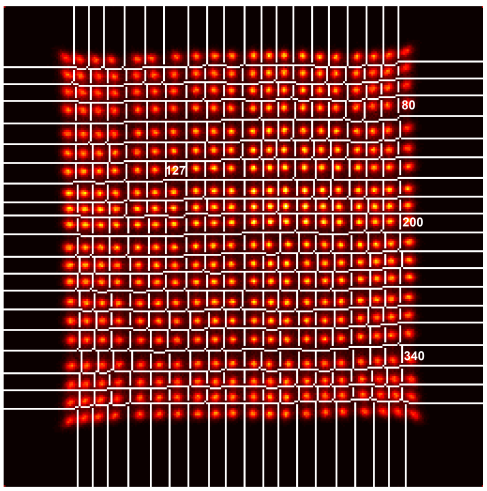

Figure 2.10: Segmented flood histogram from calibration with a ^{22}Na source. All 400 crystals are visible, and the flood histogram is split into regions corresponding to each one (white lines)

2.3.1 Calibration

Data from a flood irradiation of the detector with a 511 keV photon source and the detector operating in singles detection mode is used to determine the energy response of the detector and the location of each detector crystal on a flood histogram. To provide good statistics, at least 3 million counts were collected in this study for each calibration acquisition. The data is first processed into a flood histogram to determine the x and y position of each event on the detector face using the method described in subsection 1.2.4. A table of crystal centre locations is created manually, with the user clicking on the centre of each crystal on the flood histogram in order from left to right in each consecutive row (Figure 2.10). Crystal region boundaries are placed at the halfway point between two adjacent peaks, with the edge rows/columns bounded by the very edge of the flood histogram. Once this crystal map is made, the x and y position of each event that was used to make the flood histogram is referenced, and the crystal lookup table is used to determine which crystal the event

occurred in. The energy of each event is determined by summing the voltage from all 4 readout channels X^+,X^-,Y^+, and Y^-, and the energy of all events occurring in a particular crystal are recorded. After creating an energy spectrum for each crystal using MATLAB's histogram function, the voltage that corresponds to the highest peak of the spectrum is taken to be the position of the 511 keV photopeak. This is the reason that the number of counts needed for calibration is quite high; enough counts must be detected in each crystal to produce a good quality energy spectrum with a well-defined photopeak (Figure 2.11). The voltage of the 511 keV photopeak is used to determine the voltage per keV of energy deposited in each individual scintillator crystal. This is necessary due to the variability of energy response between each crystal due to impurities in the LSO crystal, the variability of the PMT response across the crystal array, and differences in crystal coupling across the lightguide.

This procedure is repeated for the other detector, and its crystal segmentation map and energy response are saved for use when performing analysis of arbitrary data. The crystal segmentation map is valid for use between experiments as long as the detector high voltage and amplifier gain settings are the same, but the location of the 511 keV photopeak should be determined again for each power-on cycle of the detectors as the detector response can change slightly as the detectors reach a stable operating temperature. Code for these routines is in Appendix B (B.1.1 to B.1.4).

2.3.2 Energy Spectrum Generation

After calibration is complete, energy spectra from any arbitrary acquisition can be created quickly. A flood histogram must be created for the data set to determine the x and y position of each event as in the calibration step, but the existing segmentation and energy response data generated during the calibration is used to determine the energy of each event recorded by the detectors rather than having to re-determine

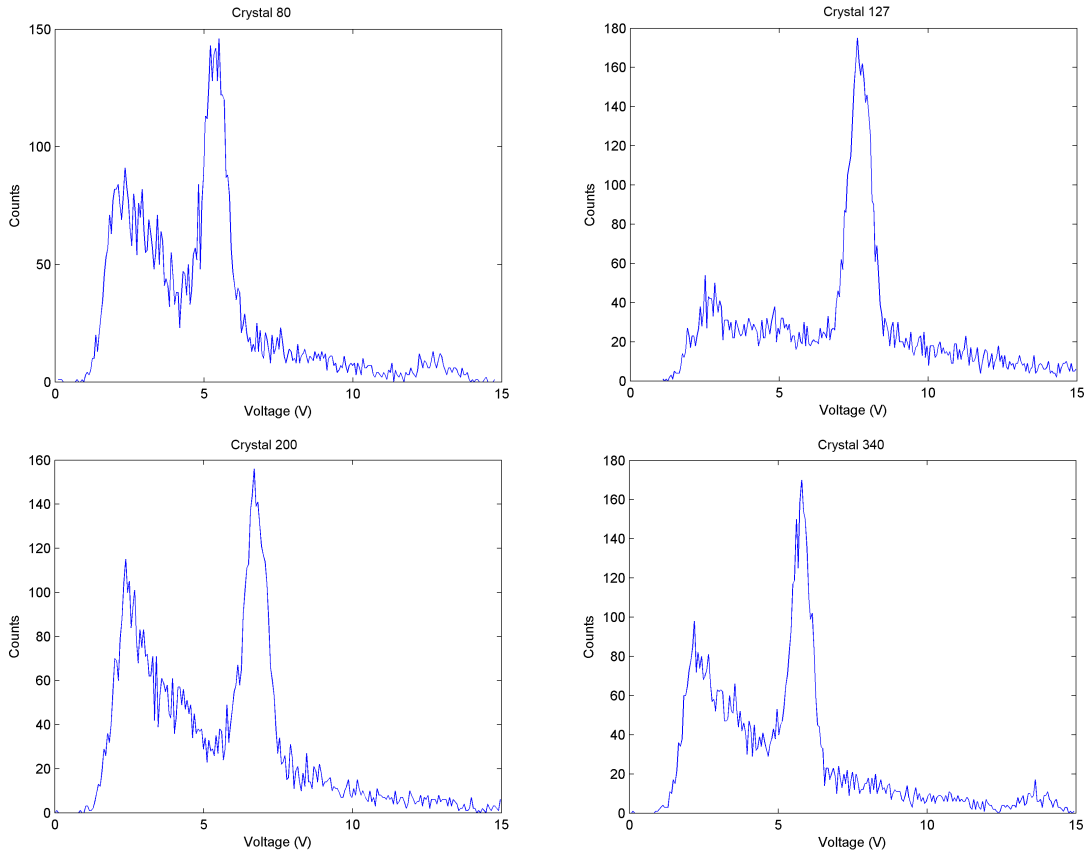

Figure 2.11: Individual energy spectra for four crystals in a Siemens Inveon block detector. Each crystal's 511 keV photopeak has a different voltage, necessitating the crystal-by-crystal characterization of the detector's energy response. Note also that edge crystals (80, 200, 340) have more lower-energy Compton scatter events than a crystal from the detector centre (crystal 127).

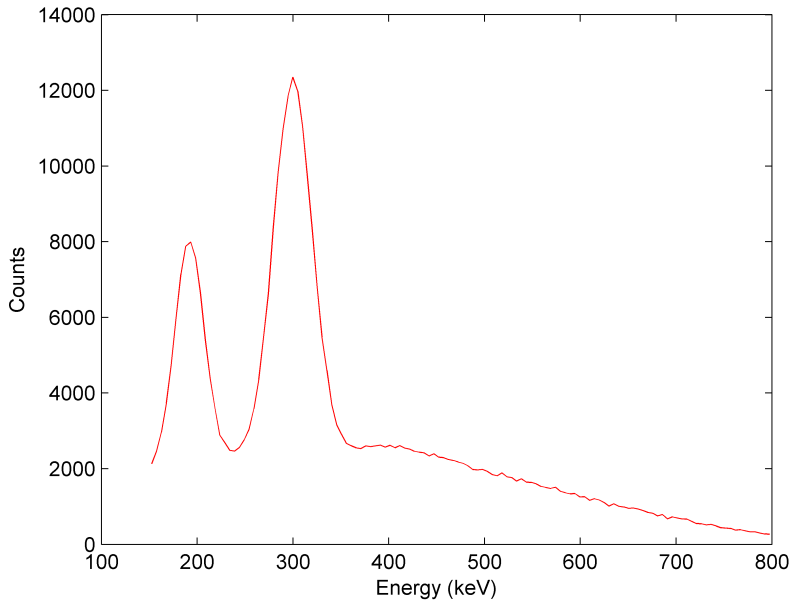

Figure 2.12: Sample coincidence energy spectrum from the bench-top system, created using intrinsic LSO activity only. Note the prominent 307 and 202 keV intrinsic gamma-ray photopeaks.

the 511 keV peak in each individual crystal. The measured energies of all events in all crystals are used to create a single large array which is histogrammed to create an energy spectrum for either a complete detector block. If coincidence data is acquired, a separate analysis script exists to create a coincidence energy spectrum using data from both detectors (Figure 2.12, code described in Appendix B.1.5).

Chapter 3

Experimental Details

3.1 Bench-top Measurements

Measurements using the two Siemens Inveon detector modules (described in subsection 2.1.1) were made reading out one detector block from each module. The detector block pair at the far edge was used to minimize the length of the readout cables needed for the experiment (Figure 3.1).

Two main sets of bench-top measurements were made. The sensitivity of the system was measured by separating the two detectors by 15.5 cm and placing a 740 kBq ^{22}Na point source suspended in a $10 \times 10 \times 10$ mm^3 plastic cube between the two. The source was centred across both detector faces to ensure uniform flood irradiation. A 3 minute acquisition was performed, and an energy spectrum was created using the MATLAB analysis scripts described in subsection 2.3.2. The energy resolution at 511 keV was determined by first finding the photopeak full width at half maximum ($FWHM$) and then using the formula

$$\text{Energy Resolution} = \frac{FWHM}{\text{Peak Centre Position}} \times 100 \tag{3.1}$$

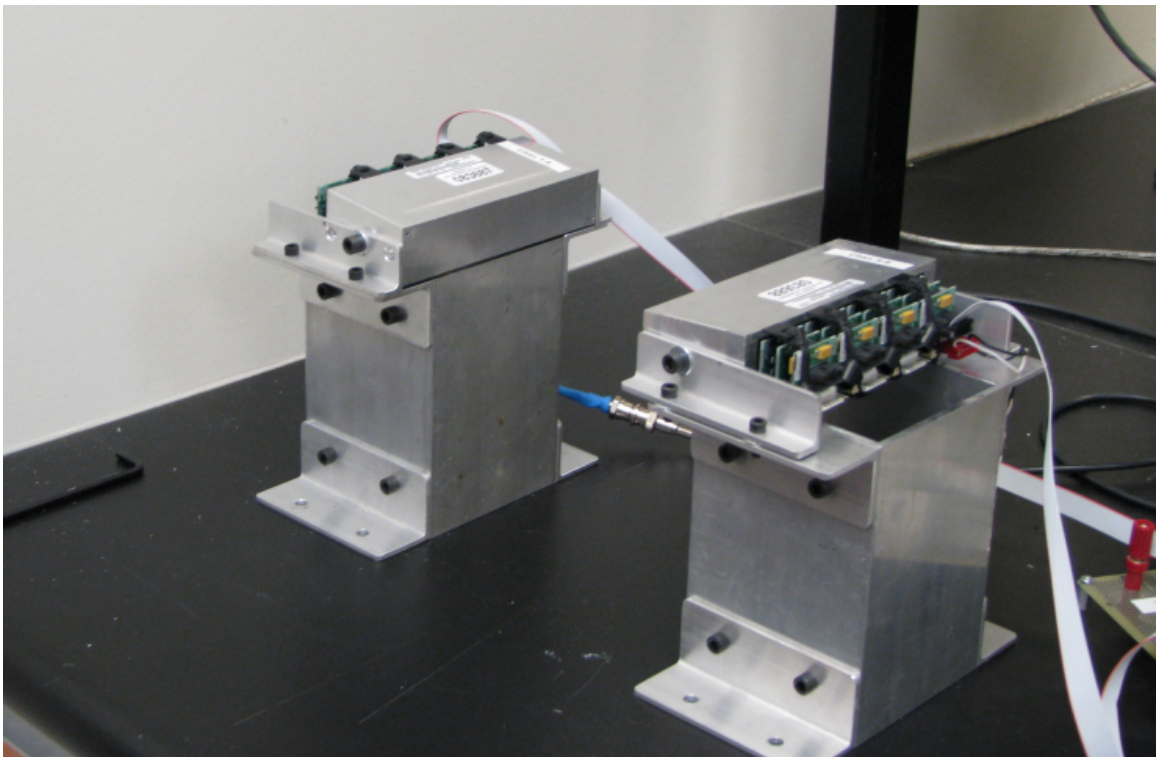

Figure 3.1: Bench-top setup of two opposing Siemens Inveon detectors. Detector stands courtesy of CancerCare Manitoba Medical Devices.

to calculate the energy resolution. A coincidence timing window of 20 ns was used during data acquisition, and an energy window of 150-800 keV was set in software.

The second set of measurements took place using intrinsic activity only. The detector modules were placed at distances of 25.5, 15.5, and 8 cm to approximate the detector ring diameters of a microPET P4 [53], a Siemens Inveon [54], and a proposed compact PET system respectively. Coincidence energy spectra were calculated on a crystal-by-crystal basis using the MATLAB analysis scripts, and an energy window was set in software with various lower level discriminator (LLD) settings to examine the effects of stepping the LLD on the intrinsic count rate. The LLD was stepped from 150 to 400 keV in 50 keV increments with a fixed upper level discriminator (ULD) setting of 800 keV. A coincidence window of 20 ns was used for this measurement as well.

3.2 Bench-Top Simulations

The bench-top simulations were performed using GATE 5.0.0 (patch 1) and Geant4 9.2 (patch 2). A generic cylindrical PET system was defined as the scanner type, and two opposing Siemens Inveon detector modules were created with the same specifications (1.51 x 1.51 x 10 mm^3 crystals, 1.59 mm crystal pitch, identical spacing between detector blocks) as the physical bench-top detector apparatus (Figure 3.2). The detector casing was modelled as a 1 mm thick sheet of aluminum covering the face of each block detector. X-rays and δ-rays (secondary scattered electrons) below 1 keV were ignored in the simulation to speed computation time as these energy levels were below the noise floor in the physical system, while electrons were not modelled beyond 0.1 mm of travel since electrons in LSO crystal are stopped in a very short distance. The more accurate "lowenergy" physics packages were chosen to model

photoelectric, Compton, and Rayleigh interactions in materials down to energies of 250 eV, while the gamma conversion model that models gamma ray interactions was set to the standard model that deals with energies at 10 keV and above. The intrinsic LSO activity was calculated to be 277 Bq/cm^3 by determining the number of ^{176}Lu atoms in 1 cm^3 of LSO using NIST-sourced atomic masses and a half-life of 4 x 10^{10} years as found in [36], as well as the 2.6% natural abundance of ^{176}Lu in all Lu. A large cylindrical ^{176}Lu source with this activity was placed in the simulation that encompassed the entire scanner; this activity was then confined to the crystal volume using GATE's "confine" command designed to limit motion. The ^{176}Lu source was defined as an "ion source" in GATE which simulates the radioactive decay and atomic de-excitation as accurately as possible and is the most realistic (but slowest) method for modelling a source in GATE. This method ensured that the activity was uniformly distributed throughout all crystals in the scanner. Simulation code for these runs is in Appendix C.1 (C.1).

System sensitivity measurements were performed to compare against the results from the physical bench-top system. A simulated 740 kBq ^{22}Na sealed 10 $\times$ 10 $\times$ 10 mm^3 source was placed directly between the two detector modules and centred across the face of the two edge detector blocks for 3 minutes of simulated time. The coincidence output of the simulation was filtered to only include coincidences from the two detector blocks of interest. The simulated system's digitizer was set with an energy window of 150-800 keV and a coincidence window of 20 ns to match the bench-top measurements. The system's energy resolution was set to 13.3% at 511 keV to match the results seen in the measurements from the bench-top system.

Intrinsic activity simulations were performed by running a simulation of the scanner with intrinsic activity only. The detector spacing was set to 25.5 cm, 15.5 cm, and 8 cm to match the distances used for the physical measurements. Two simulations

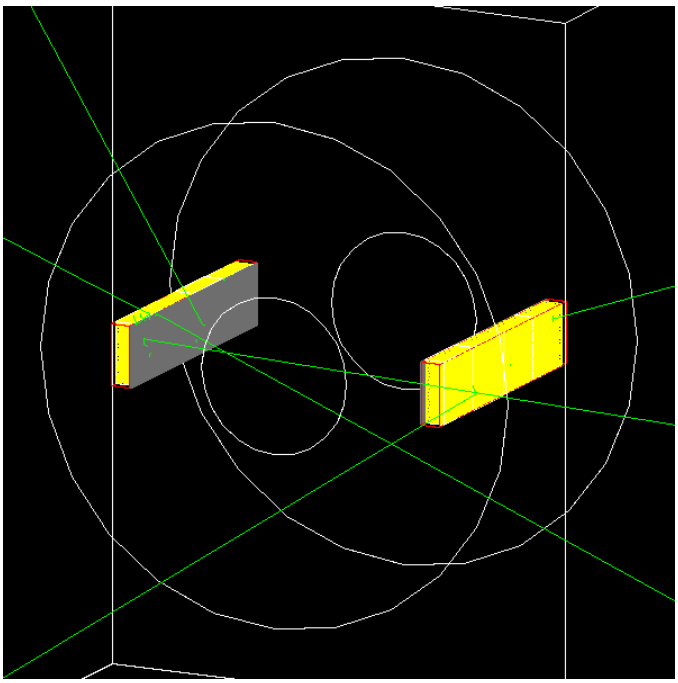

Figure 3.2: GATE simulation of the bench-top detector setup. The two detectors are modelled as LSO crystal arrays with aluminum casing on the detector faces. Intrinsic activity gamma rays are shown as green lines.

with different automatically generated random number seeds were run concurrently for 3 hours each and their results combined to create 6 hours of simulated time. The simulated energy window was set to 150-800 keV and was stepped during analysis to compare coincidence count rates to those seen in the bench-top measurements. Energy spectra were created using the same method as in the physical measurements, but without the requirement of creating crystal lookup tables or characterizing each crystal's energy response; each crystal's energy response was identical in the simulation and the energy of each coincidence event was recorded appropriately by the simulation. Simulations took roughly 4 hours to run using two cores of an Intel Q6700-based PC with 4 GB of DDR2-667 RAM.

3.3 Results

3.3.1 Sensitivity

The sensitivity measurements for the bench-top system showed very good agreement between the measured and simulated results. The position and height of the 511 keV photopeaks match extremely well (Figure 3.3), and the sensitivity during the 3 minute acquisition was found to be 0.3177% for the simulated model compared to 0.3108% from the physical detectors, with a margin of error of $\pm 0.0005\%$. While the two values do not agree within error since the margin is so small, for all practical purposes the values are close enough (within less than 0.01%) to be considered equal.

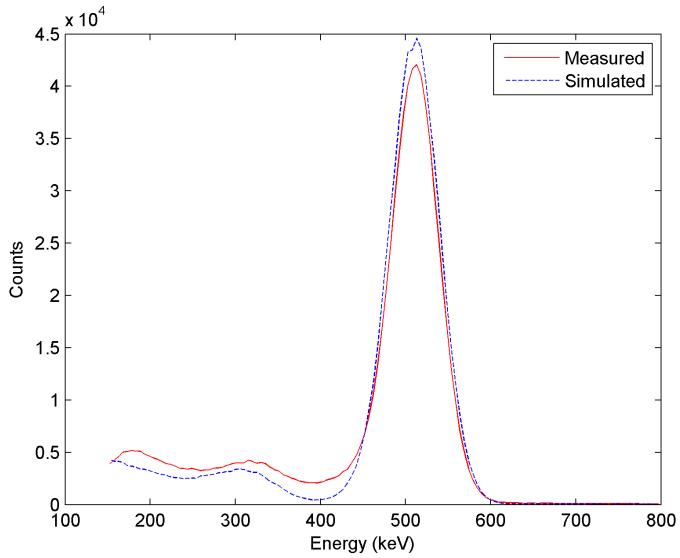

Figure 3.3: Bench-top sensitivity measurement energy spectra. Results from the simulated system (dashed blue line) match those measured from the bench-top detectors (red line) very well, with the two photopeaks lining up closely.

3.3.2 Intrinsic Activity Measurements

Simulations with intrinsic activity only also agreed well with the measured results from the bench-top system. Energy spectra acquired with the detectors separated by 8 cm (Figure 3.4) overlapped very closely, with only minor differences in the location of the 202 and 307 keV intrinsic gamma-ray photopeaks (4.1% and 1.7% respectively). These discrepancies are likely due to a non-linearity in the energy response of the detectors at lower energy levels. The light output of LSO is ideally a constant 30,750 photons/MeV of incident photon energy, but it has been found that this light output is non-linear at energies below 500 keV. Figure 3.5 shows that at 300 and 200 keV LSO's light output is 4% to 6% lower respectively compared to the quoted light output which was taken at 662 keV [55]. Since the energy of the detected event is proportional to the the number of optical photons incident on the PMT and the system's energy response is calibrated at 511 keV, the energy of events

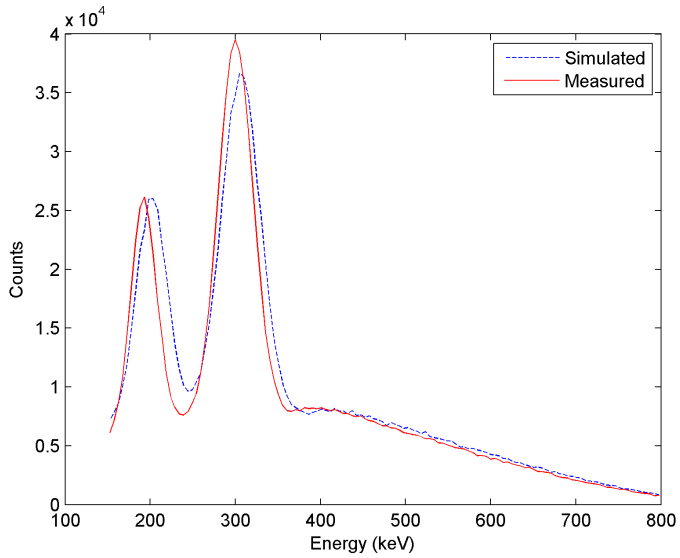

Figure 3.4: Intrinsic activity energy spectra from the bench-top detector analysis. The simulated results show very good agreement with those measured from the physical detectors, with only slight differences in photopeak position.

at 307 and 202 keV will be underestimated when using a linear model of light output. These non-linearities are not modelled in the GATE simulation, and GATE does not have a non-linear detector response model that matches that seen with LSO crystals. Energy spectra acquired at distances of 15.5 and 25.5 cm were virtually identical to those gathered at 8 cm, but with a lower number of total counts due to the greater detector separation.

Table 3.1 shows the count rates observed with various LLD settings at the 3 distances that data were acquired at. The count rates agree very well when the LLD is set to 150 and 250 keV, but begin to diverge when the LLD is set close to the energy of an intrinsic gamma-ray photopeak such as at 200 or 350 keV. This is likely due to the slight misalignment of the peaks mentioned earlier. Since more of the simulated photopeak lies above the LLD value than the measured peak due to the underestimation of the measured system's peak energy, more counts are seen for the

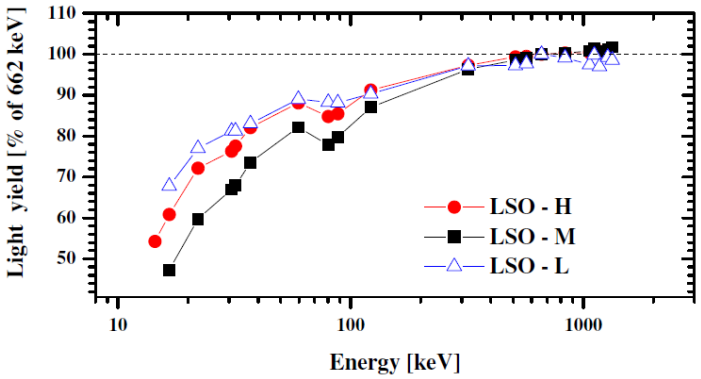

Figure 3.5: Plot of LSO light output as a function of energy. At energies above 500 keV the light output is directly proportional to incident photon energy, but can be 4% to 6% lower than expected at energies of 300 and 200 keV respectively. Plot taken from [55], ©2004 IEEE, used with permission.

Table 3.1: Bench-Top Intrinsic Count Rates (counts/second)

Distance							
	LLD Setting (keV)	150	200	250	300	350	400
8cm	Measured	26.5	19.3	14.7	6.3	0.23	0.09
	Simulated	27.9	21.5	14.9	7.7	0.43	0.12
15.5cm	Measured	9.18	6.07	4.63	1.98	0.14	0.09
	Simulated	8.62	6.66	4.67	2.47	0.21	0.10
25.5 cm	Measured	3.40	2.47	1.88	0.83	0.10	0.07
	Simulated	3.41	2.65	1.89	1.03	0.15	0.10

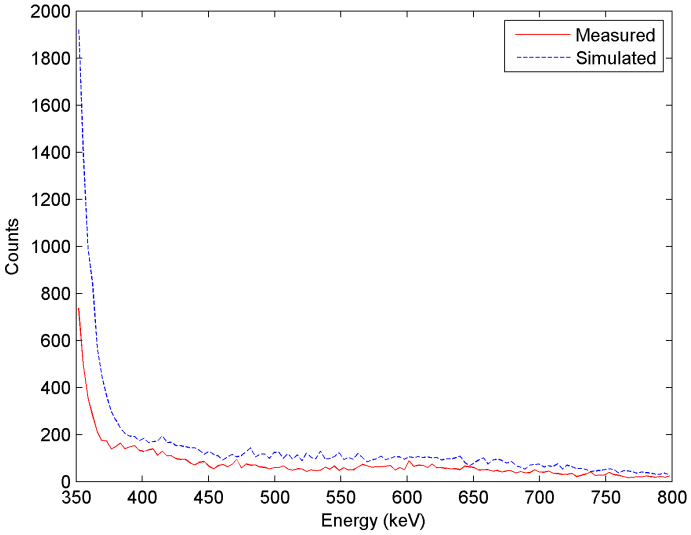

Figure 3.6: Close-up of Figure 3.4 showing the region from 350 to 800 keV. More counts are under the simulated (blue dashed) curve than the measured curve when the LLD is set to 350 keV due to misalignment of the photopeaks.

simulated system (Figure 3.6).

Chapter 4

Inveon System Comparison

4.1 Full System Measurements

Physical measurements on a complete preclinical scanner were performed on the Siemens Inveon preclinical PET system [54] (Siemens Preclinical Solutions, Knoxville, TN, USA) at the Crump Institute for Molecular Imaging (University of California Los Angeles, Los Angeles, CA, USA). This system has a 16 cm diameter LSO crystal ring constructed from the detector modules used in the bench-top study, and has a 12.7 cm axial field of view (FOV). Raw scan list-mode data were histogrammed as a set of 128x160x159 sinograms created using SSRB with a maximum ring difference of 39 and a span of 79. A series of 9 minute acquisitions with a 3.432 ns coincidence window and intrinsic activity only were taken to generate an energy spectrum using a procedure first described by Goertzen et al [57]. The LLD was stepped in 10 keV increments from 150 to 790 keV with a fixed ULD of 800 keV, and in a separate set of measurements the ULD was stepped in 10 keV increments from 800 to 150 keV with a fixed LLD of 50 keV. To determine the average number of counts S_{avg} at each

energy level E with energy bin width ΔE, we use the equation

$$S_{avg}(E + \frac{\Delta E}{2}) = \frac{\Delta C_{LLD} + \Delta C_{ULD}}{2\Delta E} \tag{4.1}$$

In this case, ΔC_{LLD} is the difference between the number of true counts with the LLD set to E and $E - \Delta E$ and the ULD set to a fixed value. ΔC_{ULD} is calculated similarly, but with the ULD varied and the LLD set to a fixed value. This allows an energy spectrum to be created by plotting S_{avg} versus energy.

To further investigate the performance of the Inveon system, a water-filled 50 mL centrifuge tube (28.5 mm diameter, 115 mm tall) was imaged for one hour with no source present aside from intrinsic activity to obtain a "transmission" sinogram. The tube's attenuation of the intrinsic activity gamma rays gave a sinogram similar to that seen from a transmission source rotated around the field of view. Three separate acquisitions were performed with the LLD set to 200, 250, and 300 keV with the ULD fixed at 800 keV. The central 20 sinogram slices were summed to create a high statistics sinogram for each acquisition, and line profiles were created by summing all of this sinogram's rows at each LLD setting.

4.2 Full System Simulations

Simulations of a complete Siemens Inveon system were performed using GATE 5.0.0 (patch 1) with Geant4 9.2 (patch 2). The detector modules used in the bench-top simulations were duplicated to create a complete detector ring that contained 16 modules (64 block detectors total) with a crystal ring radius of 8.054 cm to match the physical Inveon system. The total activity of the simulated ^{176}Lu source was increased to match the increased volume of simulated scintillator crystal. Since the

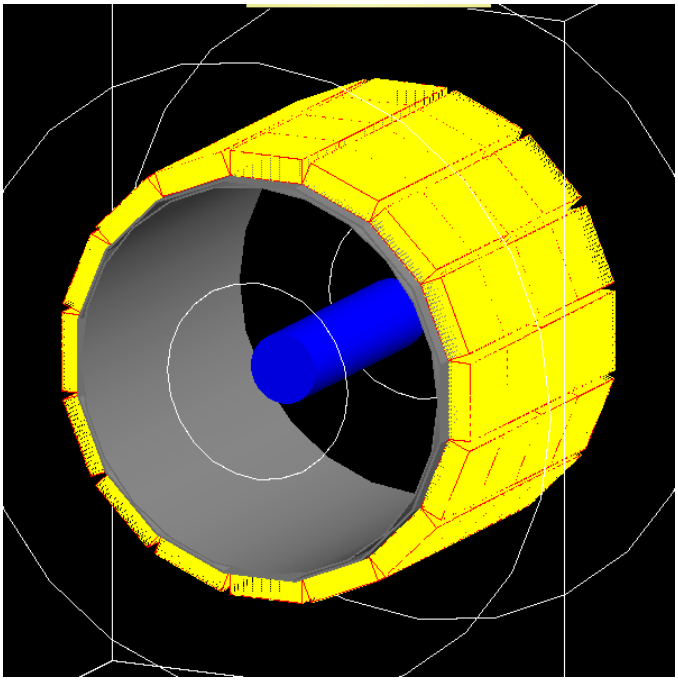

Figure 4.1: Simulated Inveon scanner with a water-filled centrifuge tube placed in the centre of the scanner's field of view.

complete scanner's gantry attenuates the intrinsic gamma rays slightly, a 0.25 mm ring of aluminum and a 0.5 mm ring of carbon fibre were added to the system to reproduce the gantry material as specified by the manufacturer. The generic CylindricalPET system in GATE is unable to directly create sinogram output, requiring the system to be respecified as a Siemens ECAT-type scanner in GATE's readout configuration that can create sinograms directly. Simulated system output was a series of 128x160x6400 sinograms, with each slice representing all allowed cross-plane coincidences. These sinograms were then processed using SSRB with a maximum ring difference of 39 and a span of 79 to create 159 sinogram slices. A 3.432 ns coincidence window was set to match that used on the physical system. A series of 9 minute acquisitions with intrinsic activity only were simulated while stepping the ULD and LLD to generate energy spectra using the same method used for the data from the physical system. A

50 mL water-filled centrifuge tube was also simulated for one hour of acquisition time to generate "transmission" sinograms similar to what would be seen if a radioactive source of similar activity were moved around the detector ring. The LLD was stepped to match the measurements performed on the physical scanner (Figure 4.1). These one hour simulations took roughly 30 hours each to run using an Intel Q9550 CPU with 4 GB of DDR2-800 RAM. Code for these simulations is shown in Appendix C.2.

To confirm the sensitivity of the simulated system, an acquisition with a simulated 198 kBq ^{22}Na point source was simulated to replicate the measurements taken by Bao et al [54]. The simulated system's energy window was set to 300-625 keV and the number of true coincidence counts was compared to the total number of decay events taking into account ^{22}Na's positron branching ratio of 0.899 [58].

4.3 Results

System sensitivity measurements showed good agreement with those from Bao et al [54]. The simulated sensitivity with an energy window of 300-625 keV was 8.487% ($\pm$0.005%) with the source at the centre of the scanner FOV, compared to 7.86% as reported in their results. This does not agree within error limits since the error is so small, but is within 0.6% and may be explained by the simulated system having a greater sensitivity to lower-energy events than the physical system as discussed below.

The intrinsic activity only measurements resulted in energy spectra that do not agree as well with the simulation as the results from the bench-top system. Figure 4.2 shows that the energy spectrum from the simulated system matches the measured data well at higher energies, especially in the beta-coincidence spectrum beyond 350 keV. The intrinsic gamma-ray photopeaks are found at energy levels that are consistently lower in the measured results than those obtained from the simulated results

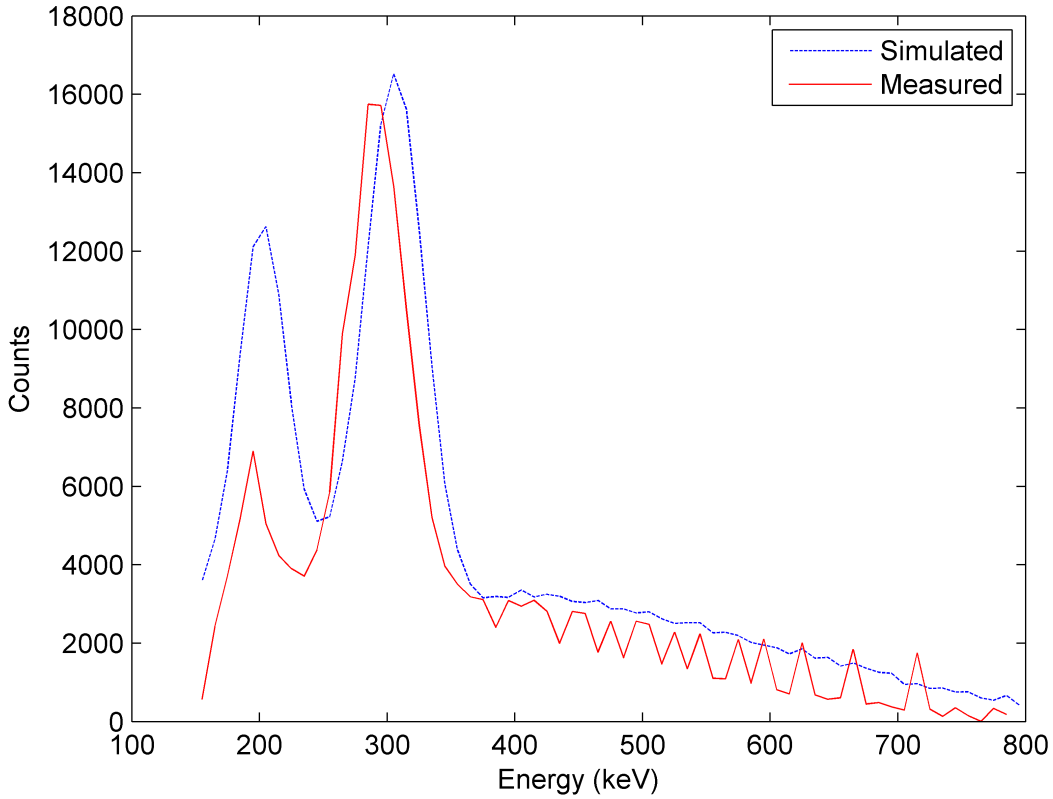

Figure 4.2: Energy spectra comparing the simulated Inveon system with those from a physical scanner. The energy of the measured intrinsic photopeaks are consistently 5% lower than their nominal values.

with the 307 keV photopeak located at an energy 5.1% lower than its simulated counterpart, while the 202 keV photopeak is 5.2% lower. The height of the simulated 202 keV photopeak is also significantly higher than the measured photopeak. These results are not unusual, however, as previous studies have shown a consistent underestimation of intrinsic gamma-ray photopeak energy by 2% to 5% of the nominal energy [57].

The underestimation of the photopeak energy can be explained by the same non-proportional response of LSO crystal as was seen in the bench-top study, but the physical Inveon system's much lower sensitivity for events below 250 keV can not be

explained by this alone. The source of this large discrepancy is likely due to the fact that the Inveon detectors are triggered with an energy threshold set on a block level. As was seen in figure 2.11 the voltage output of edge crystals is significantly lower than that seen in crystals closer to the centre of the detector, likely due to greater light loss along the edges of the light guide that couples the scintillator crystals to the PSPMT. This can cause lower energy events that would otherwise be detected to fall below the voltage threshold for the detector block.

This effect could likely be reproduced in GATE by fully modelling the light guides within the block detectors and using GATE's optical photon tracking routines to model light transport for each detected event. However, optical photon simulation is extremely computationally intensive since each high energy event produces large numbers of optical photons. Simulations of individual crystals take days to run, making this procedure currently unfeasible for a complete PET system given our current computer resources. It may be possible to modify GATE to include a more simple model of non-proportional energy response in edge crystals, however this would require a significant overhaul of the code that simulates detector response.

Results from the centrifuge tube sinogram profile measurements show mixed agreement with simulated results. Figure 4.3 shows that the profile shapes are nearly identical to each other, but the number of counts recorded in the simulated results are slightly higher than the measured counts at all LLD settings. The effect is more apparent when the LLD is set near a photopeak position, echoing the count rate effects seen while stepping the LLD in the bench-top analysis. This is likely due to the misalignment of the intrinsic photopeak positions as mentioned before. The simulated profiles match those seen from the physical system much better when the LLD is set to 250 keV, however.

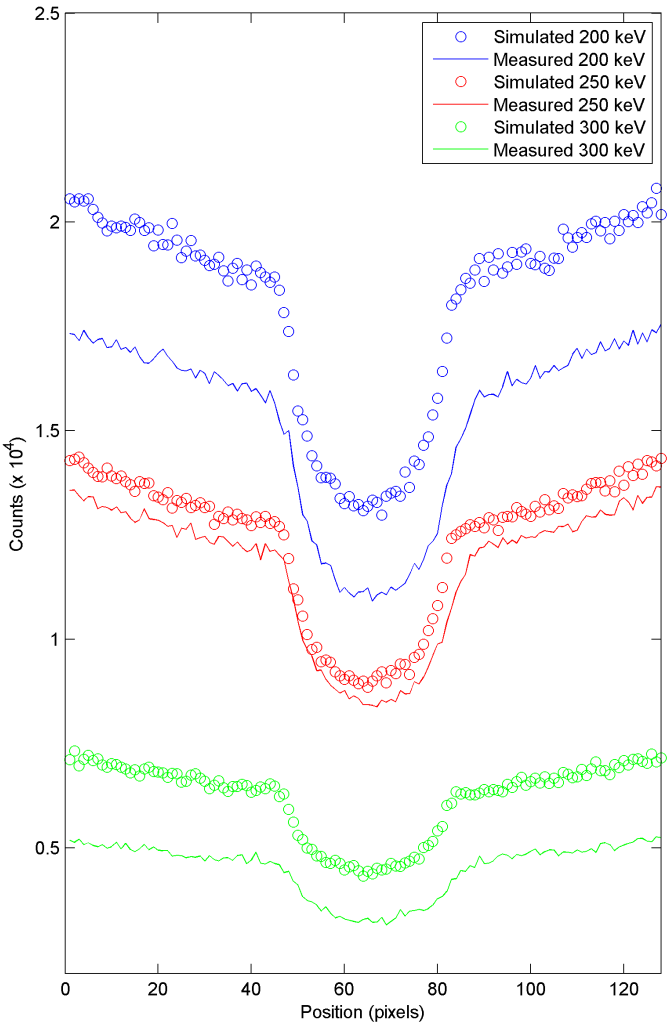

Figure 4.3: Sinogram line profiles from the central 20 sinogram slices with a centrifuge tube in the FOV and intrinsic activity only.

Chapter 5

Conclusions

This project successfully validated a model of ^{176}Lu intrinsic activity in preclinical PET systems. The bench-top detector simulations matched the bench-top measurements very well, with only small discrepancies in photopeak position. Simulated intrinsic coincidence count rates also matched well with the measured count rates, but showed some differences when stepping the LLD of the system to a point near an intrinsic photopeak. This is due to the slight differences in photopeak position exaggerating the count differences between the measured and simulated systems' energy spectra, and is most likely caused by the non-proportional light output of LSO at energies below 500 keV. Results from the simulation of a full Inveon system did not agree as well with the measurements from a real Inveon scanner, partly due to the non-proportionality effects seen in the bench-top evaluation and also due to overly aggressive cutoff of events below 250 keV in the scanner's digitization electronics.

This simulation model provides very good reproduction of the effects of ^{176}Lu in LSO-based PET systems so long as the behaviour of the digitization hardware can be modelled accurately in GATE. These simulations could be used not only to predict how many coincidences are due to intrinsic activity, but also to determine

how much of a tradeoff exists between maintaining system sensitivity and mitigating the background signal in a reconstructed PET image. This work will also allow other GATE users to more accurately model LSO in future preclinical PET systems with greater accuracy than is currently available, at the expense of adding the simulation time of the LSO decay process itself.

References

1. M. Phelps, "PET: The Merging of Biology and Imaging into Molecular Imaging," *Journal of Nuclear Medicine* vol. 41, pp. 661-681, 2000

2. G. F. Knoll (2000), "Fast Electron Sources" in *Radiation Detection and Measurement*, 3rd ed. Hoboken, NJ: John Wiley & Sons, Inc., pp. 3-5

3. A. P. Patro, P. Sen, "Parapositronium lifetime," *Journal of Physics A: General Physics* vol. 4, pp. 856-858, 1971

4. T. J. Tewson et al, "[18F]-labeled 3-deoxy-3-fluoro-D-glucose: synthesis and preliminary biodistribution data," *Journal of Nuclear Medicine* vol. 19, pp. 1339-1345, 1978

5. G. Weber, "Enzymology of cancer cells," *New England Journal of Medicine* vol. 296, pp. 541-551, 1977

6. T. Y. Jeon et al, "Incremental Value of PET/CT Over CT for Mediastinal Nodal Staging of Non-Small Cell Lung Cancer: Comparison Between Patients With and Without Idiopathic Pulmonary Fibrosis," *American Journal of Roentgenology* vol. 195, pp. 370-376, 2010

7. Marco Brambilla et al, "Performance Characteristics Obtained for a New 3-Dimensional Lutetium Oxyorthosilicate-Based Whole-Body PET/CT Scanner

with the National Electrical Manufacturers Association NU 2-2001 Standard," *Journal of Nuclear Medicine* vol. 46, pp. 2083-2091, 2005

8. N. U. Schramm et al, "High-Resolution SPECT using Multi-Pinhole Collimation," 2002 Nuclear Science Symposium Conference Record, pp. 774-777

9. S. R. Cherry, J. A. Sorenson, M. E. Phelps (2003), "Annihilation Coincidence Detection" in *Physics in Nuclear Medicine*, 3rd ed. Philadelphia, PN: Elsevier Science (USA), pp. 325-327.

10. E. J. Hoffman, M. E. Phelps (1986), "Positron emission tomography: Principles and quantitation" in *Positron Emission Tomography and Autoradiography: Principles and Applications for the Brain and Heart*, New York, Raven Press, pp. 237-286.

11. The MICAD Research Team, "3-*N*-(2-[^{18}F]Fluoroethyl)spiperone," Molecular Imaging and Contrast Agent Database [Internet], Bethesda (MD): National Center for Biotechnology Information (US), 2004-2010.

12. K. Leung, "16α -[^{18}F]Fluoro-17β-estradiol[[^{18}F]FES]," Molecular Imaging and Contrast Agent Database [Internet], Bethesda (MD): National Center for Biotechnology Information (US), 2004-2010.

13. R. J. Hargreaves, "The Role of Molecular Imaging in Drug Discovery and Development," *Clinical Pharmacology & Therapeutics* vol. 83, pp. 349-353, 2008

14. A. F. Chatziioannou, "Molecular imaging of small animals with dedicated PET tomographs," *European Journal of Nuclear Medicine* vol. 29, pp. 98-114, 2002

15. S. R. Cherry et al, "MicroPET: a high resolution PET scanner for imaging small animals," *IEEE Transactions in Nuclear Science* vol. 44, pp. 1161-1166, 1997

16. M. C. Huisman et al, "Performance evaluation of the Philips MOSAIC small animal PET scanner," *European Journal of Nuclear Medicine and Molecular Imaging* vol. 48, pp. 532-540, 2007

17. T. Dumouchel et al, "Performance Evaluation of the LabPET APD-based digital PET scanner," *IEEE Transactions on Nuclear Science* vol. 56, pp. 10-16, 2009

18. R. de Leo et al, "The AX-PET project: Demonstration of a high resolution axial 3D PET," *Nuclear Instruments and Methods in Physics Research A* vol. 617, pp. 214-216, 2010

19. C. S. Levin, H. Zaidi, "Current Trends in Preclinical PET System Design," *PET Clinics 2* pp. 125-160, 2007

20. G. F. Knoll (2000), "Inorganic Scintillators" in *Radiation Detection and Measurement*, 3rd ed. Hoboken, NJ: John Wiley & Sons, Inc., pp. 231-234

21. F. H. Attix (1986), "Photoelectric Effect" in *Introduction to Radiological Physics and Radiation Dosimetry*, Weinheim, Germany: WILEY-VCH Verlag GmbH & Co., pp. 138-140,

22. F. H. Attix (1986), "Compton Effect" in *Introduction to Radiological Physics and Radiation Dosimetry*, Weinheim, Germany: WILEY-VCH Verlag GmbH & Co., pp. 125-127, 134-135

23. T. K. Lewellen, "Recent developments in PET detector technology," *Physics in Medicine and Biology* vol. 53, pp. R287-R317, 2008

24. J. Humm, A. Rosenfel, A. Del Guerra, "From PET detectors to PET scanners," *European Journal of Nuclear Medicine and Molecular Imaging* vol. 30, pp.

1574-1597

25. C. L. Melcher, "Scintillation Crystals for PET," *Journal of Nuclear Medicine* vol. 41, pp. 1051-1055, 2000

26. G. F. Knoll (2000), "Photomultiplier Tubes and Photodiodes" in *Radiation Detection and Measurement*, 3rd ed. Hoboken, NJ: John Wiley & Sons, Inc., pp. 265-273

27. S. R. Cherry, J. A. Sorenson, M. E. Phelps (2003), "Block Detectors" in *Physics in Nuclear Medicine*, 3rd ed. Philadelphia, Pa: Elsevier Science (USA), pp. 342-344

28. S. B. Siegel, S. R. Cherry, "Readout circuit configuration" in *An Investigation of Detector Modules for a High Resolution Animal PET Tomograph*, pp. 59-61, 1996

29. Y. Yoshizawa, J. Takeuchi, "The latest vacuum photodetector," *Nuclear Instruments and Methods In Physics Research A*, vol. 387, pp. 33-37, 1997

30. H. O. Anger, "Scintillation Camera," *Review of Scientific Instruments* vol. 29, pp. 27-33, 1958

31. J. T. Bushberg et al, "Design of a PET Scanner" in *The Essential Physics of Medical Imaging*, 2nd ed. Philadelphia, Pa: Lippincott Williams & Wilkins, pp. 722, 2002

32. J. T. Bushberg et al, "True Versus Random Coincidences, Scatter Coincidences" in *The Essential Physics of Medical Imaging*, 2nd ed. Philadelphia, Pa: Lippincott Williams & Wilkins, pp. 724-725, 2002

33. F. H. Fahey, "Data Acquisition in PET Imaging," *Journal of Nuclear Medicine Technology* vol. 30, pp. 39-49, 2002

34. M. E. Daube-Witherspoon, G. Muehllehner, "Treatment of axial data in three-dimensional PET," *Journal of Nuclear Medicine* vol. 28, pp. 1717-1724, 1987

35. M. Defrise et al, "Exact and Approximate Rebinning Algorithms for 3-D PET Data," *IEEE Transactions on Medical Imaging* vol. 16, pp. 145-158, 1997

36. E. Browne, H. Junde, "Nuclear Data Sheets for A = 176," *Nuclear Data Sheets*, vol. 84, pp. 337-486, 2002

37. S. Yamamoto et al, "Investigation of single, random, and true counts from natural radioactivity in LSO-based clinical PET," *Annals of Nuclear Medicine* vol. 19, pp. 109-114, 2005

38. A. L. Goertzen, J. Y. Suk, C. J. Thompson, "Imaging of Weak-Source Distributions in LSO-Based Small-Animal PET Scanners," *The Journal of Nuclear Medicine* vol. 48, pp. 1692-1698, 2007

39. C. C. Watson et al, "NEMA NU 2 Performance Tests for Scanners with Intrinsic Radioactivity," *Journal of Nuclear Medicine* vol. 45, pp. 822-826, 2004

40. I. Kawrakow et al, "Radiation transport in EGSnrc" in *The EGSnrc Code System: Monte Carlo Simulation of Electron and Photon Transport*, pp. 27-28, February 17, 2010

41. The OpenGATE Collaboration, "Physics process" in *GATE 5.0.0 Users Guide*, pp. 85-90, June 10, 2009

42. A. Badal, A. Badano, "Accelerating Monte Carlo simulations of photon transport in a voxelized geometry using a massively parallel graphics processing unit," *Medical Physics* vol. 36, pp. 4878-4881, 2009

43. S. Jan et al, "GATE: a simulation toolkit for PET and SPECT," *Physics in Medicine and Biology* vol. 49, pp. 4543-4561, 2004.

44. S. Agostinelli et al, "GEANT4 - a simulation toolkit," *Nuclear Instruments and Methods in Physics Research A* vol. 506, pp. 250-303, 2003

45. List of Peer-reviewed publications, retrieved from http://www.opengatecollaboration.org/publications/peer-reviewed-papers.html October 4, 2010

46. J. D. Leroux et al, "Fast, accurate and versatile Monte Carlo method for computing system matrix," 2007 IEEE Nuclear Science Symposium Conference Record M18-386, pp. 3644-3648

47. R. A. Ramirez et al, "High Resolution L(Y)SO Detectors Using PMT-Quadrant-Sharing for Human and Animal PET Cameras," *IEEE Transactions on Nuclear Science* vol. 55, pp. 862-869, 2008

48. A. Braem et al, "AX-PET: A novel PET detector concept with full 3D reconstruction," *Nuclear Instruments and Methods in Physics Research A* vol. 610, pp. 192-195, 2009

49. S. St. James et al, "Simulation study of spatial resolution and sensitivity for the tapered depth of interaction PET detectors for small animal imaging," *Physics in Medicine and Biology* vol. 55, pp. N63-N74, 2010

50. F. ur-Rehman et al, "Observations on dual-ended readout of 100 mm long LYSO crystals," *Nuclear Instruments and Methods in Physics Research A*, DOI 10.1016/j.nima.2010.08.015, in press, 2010

51. N. Karakatsanis et al, "Investigation of the Minimum Detectable Activity Level of a Preclinical LSO PET Scanner," 2007 Nuclear Science Symposium Conference Record, pp. 3133-3338

52. R. A. Mintzer, S. B. Siegel, "Design and Performance of a New Pixelated-LSO/PSPMT Gamma-Ray Detector for High Resolution PET Imaging," 2007 IEEE Nuclear Science Symposium Conference Record M18-142, pp. 3418-3422

53. Y. C. Tai et al, "Performance evaluation of the microPET P4: a PET system dedicated to animal imaging," *Physics in Medicine and Biology* vol. 46, pp. 1845-1862, 2001

54. Q. Bao et al, "Performance Evaluation of the Inveon Dedicated PET Preclinical Tomograph Based on the NEMA NU-4 Standards," *The Journal of Nuclear Medicine* vol. 50, pp. 401-408, 2009

55. M. Kapusta et al, "Non-proportionality and thermoluminescence of of LSO:Ce," 2004 IEEE Nuclear Science Symposium Conference Record, pp. 822-828

56. W. W. Moses, S. E. Derenzo, "Design studies for a PET detector module using a PIN photodiode to measure depth of interaction," *IEEE Transactions on Nuclear Science* vol. 41, pp. 1441-1445, 1994

57. A. L. Goertzen, D. B. Stout, C. J. Thompson, "A method for measuring the energy spectrum of coincidence events in positron emission tomography," *Physics in Medicine and Biology* vol. 55, pp. 535-549, 2010.

58. MG Stabin, LC du Luz, "Decay data for internal and external dose assessment," *Health Physics*, vol. 83, pp. 1527-1535, 2002.

Appendix A

Appendix A: Data Acquisition Code

This code was developed in LabWindows CVI 8.5, and is used for acquiring data from the bench-top detector system as described in section 2.2. The commands used to configure the DAQ hardware are designed to work with the PCI-6133 interface card, triggered on inputs PFI0 or PFI1 and with signals on analog inputs AI0 to AI7.

```
#include "MCA.h"
#include <utility.h>
//==============
//
// Title:       MCA
// Purpose:     Multi-channel analyzer for a single input
//
// Created on:  10/07/2008 at 1:11:03 PM by Bryan McIntosh.
// Copyright:   . All Rights Reserved.
//

// Include files
#include <ansi_c.h>
#include <cvirte.h>
#include <userint.h>
#include "MCA.h"
#include "toolbox.h"
#include "math.h"
#include <NIDAQmx.h>
#include <formatio.h>
#include <analysis.h>
```

```
// Constants
#define maxentries 10000000 //should be at least 8X as much as the
    maximum number of events in the GUI
#define buffersize 20000
#define bins 257
#define dim 256
#define blocks 1
#define crystals 400

// Static global variables
static int panelHandle,panelChannel,panelFlood;
// Global variables
int stopon,voltsamples,i,n,num,status,index=0,total=0,histdata[8][bins];
int quitting=0,channels=0;
int chan[8]={0},plotcolor[8],floodblock=0;//used to identify which
    channels are active
double xmax,xmin,ymax,ymin; //variables used for histogramming
uInt32 counts = 0;
double newtime,maxtime,maxcounts,oldtime,voltbuffer0[buffersize],
    histaxis[8][bins];
double *sumvoltage, **voltage;
int imagearray[dim][dim];
TaskHandle Counter = 0,A0input=0;
char pathname[10000];
int block[4][256][256]; //Added by AG, 3/13/2011 to make program compile
    .
//Create file handles for saving data
FILE *outputhandle1;

// Global functions
/// HIFN The main entry-point function.
int main (int argc, char *argv[])
{
    int error = 0;

    /* initialize and load resources */
    nullChk (InitCVIRTE (0, argv, 0));
    errChk (panelHandle = LoadPanel (0, "c:/test/MCA_mk2/MCA.uir", PANEL
        ));
  panelChannel = LoadPanel (panelHandle, "c:/test/MCA_mk2/MCA.uir",
      PANEL_CHAN);
  panelFlood = LoadPanel(panelHandle, "c:/test/MCA_mk2/MCA.uir",
      PANEL_FLOO);
    /* display the panel and run the user interface */
    errChk (DisplayPanel (panelHandle));
    errChk (RunUserInterface ());
Error:
    /* clean up */
    DiscardPanel (panelHandle);
    return 0;
}
```

```
// UI callback function prototypes
/// HIFN Exit when the user dismisses the panel.
int CVICALLBACK panelCB (int panel, int event, void *callbackData,
        int eventData1, int eventData2)
{
    if (event == EVENT_CLOSE)
        QuitUserInterface (0);
    return 0;
}
int CVICALLBACK StartCount (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{     int channel,shaswitch;
  double result,vmin=-10.0,vmax=10.0;
  switch (event)
  {
    case EVENT_COMMIT:
      SetCtrlAttribute(panelHandle,PANEL_START_COUNT,ATTR_DIMMED,1);
      SetCtrlAttribute(panelHandle,PANEL_STOP_COUNT,ATTR_DIMMED,0);
      //Determine which channels are selected
      GetCtrlVal(panelChannel,PANEL_CHAN_CH1,&chan[0]);
      GetCtrlVal(panelChannel,PANEL_CHAN_CH2,&chan[1]);
      GetCtrlVal(panelChannel,PANEL_CHAN_CH3,&chan[2]);
      GetCtrlVal(panelChannel,PANEL_CHAN_CH4,&chan[3]);
      GetCtrlVal(panelChannel,PANEL_CHAN_CH5,&chan[4]);
      GetCtrlVal(panelChannel,PANEL_CHAN_CH6,&chan[5]);
      GetCtrlVal(panelChannel,PANEL_CHAN_CH7,&chan[6]);
      GetCtrlVal(panelChannel,PANEL_CHAN_CH8,&chan[7]);
      //Populate the array that contains the histogram colour data
      plotcolor[0]=VAL_RED;
      plotcolor[1]=VAL_GREEN;
      plotcolor[2]=VAL_BLUE;
      plotcolor[3]=VAL_YELLOW;
      plotcolor[4]=VAL_MAGENTA;
      plotcolor[5]=VAL_WHITE;
      plotcolor[6]=VAL_CYAN;
      plotcolor[7]=VAL_DK_MAGENTA;
      //initialise the array that stores voltage info
      voltage= (double**)malloc(8*sizeof(double*)); //dynamically
          allocate an array with support for all 8 channels
      for(i=0;i<8;i++){
        voltage[i] = (double*)malloc(maxentries*sizeof(double));
      }
      GetCtrlVal(panelHandle,PANEL_MAX_TIME,&maxtime);
      GetCtrlVal(panelHandle,PANEL_MAX_COUNTS,&maxcounts);
      //Create a DAQmx task to gather data
      DAQmxCreateTask("A0input",&A0input);
      //Select the channels to gather data from
    for(i=0;i<8;i++){
      if(chan[i]==1){ //makes sure that we choose the proper channel to
          add
```

```
      channel = i;
    }
    else{channel = -9;}
    switch (channel)
    { case 0:
        DAQmxCreateAIVoltageChan (A0input, "Dev1/ai0", "analog0",
            DAQmx_Val_Diff, vmin, vmax, DAQmx_Val_Volts, "");
        channels++;
        break;
      case 1:
        channels++;
        DAQmxCreateAIVoltageChan (A0input, "Dev1/ai1", "analog1",
            DAQmx_Val_Diff, vmin, vmax, DAQmx_Val_Volts, "");
        break;
      case 2:
        DAQmxCreateAIVoltageChan (A0input, "Dev1/ai2", "analog2",
            DAQmx_Val_Diff, vmin, vmax, DAQmx_Val_Volts, "");
        channels++;
        break;
      case 3:
        DAQmxCreateAIVoltageChan (A0input, "Dev1/ai3", "analog3",
            DAQmx_Val_Diff, vmin, vmax, DAQmx_Val_Volts, "");
        channels++;
        break;
      case 4:
        DAQmxCreateAIVoltageChan (A0input, "Dev1/ai4", "analog4",
            DAQmx_Val_Diff, vmin, vmax, DAQmx_Val_Volts, "");
        channels++;
        break;
      case 5:
        DAQmxCreateAIVoltageChan (A0input, "Dev1/ai5", "analog5",
            DAQmx_Val_Diff, vmin, vmax, DAQmx_Val_Volts, "");
        channels++;
        break;
      case 6:
        DAQmxCreateAIVoltageChan (A0input, "Dev1/ai6", "analog6",
            DAQmx_Val_Diff, vmin, vmax, DAQmx_Val_Volts, "");
        channels++;
        break;
      case 7:
        DAQmxCreateAIVoltageChan (A0input, "Dev1/ai7", "analog7",
            DAQmx_Val_Diff, vmin, vmax, DAQmx_Val_Volts, "");
        channels++;
        break;
    }
}
    GetCtrlVal(panelChannel,PANEL_CHAN_SHASWITCH,&shaswitch);
    if (shaswitch == 1){
    //used when using the SHA
    DAQmxCfgSampClkTiming (A0input, "/Dev1/PFI0", 2000000,
        DAQmx_Val_Rising, DAQmx_Val_ContSamps, 2);
```

```
        }
        else{
        //used when NOT using the SHA
        DAQmxCfgSampClkTiming (A0input, "/Dev1/PFI3", 2000000,
            DAQmx_Val_Rising, DAQmx_Val_ContSamps, 2);
        }
        //start the tasks to improve performance
        DAQmxStartTask(A0input);
        SetCtrlVal(panelHandle,PANEL_MISSED,channels); //debug what this
            is
        //Start the timer
        SetCtrlAttribute(panelHandle,PANEL_TIMER,ATTR_ENABLED,1);
        oldtime = Timer();
        break;
      case EVENT_RIGHT_CLICK:
        break;
    }
    return 0;
}
int CVICALLBACK ResetCount (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
      case EVENT_COMMIT:
        SetCtrlAttribute(panelHandle,PANEL_TIMER,ATTR_ENABLED, 0);
        SetCtrlAttribute(panelHandle,PANEL_XPOS,ATTR_VISIBLE,0);
        SetCtrlAttribute(panelHandle,PANEL_YPOS,ATTR_VISIBLE,0);
        SetCtrlAttribute(panelHandle,PANEL_START_COUNT,ATTR_DIMMED,0);
        SetCtrlAttribute(panelHandle,PANEL_RESCALEHIST,ATTR_DIMMED,1);
        DeleteGraphPlot(panelHandle,PANEL_GRAPH,-1,VAL_IMMEDIATE_DRAW);
        SetCtrlVal(panelHandle,PANEL_COUNTS,0);
        SetCtrlVal(panelHandle,PANEL_ELAPSED,0.00);
        SetCtrlVal(panelHandle,PANEL_SAMPLES,0);
        index=0;
        total=0;
        counts=0;
        channels = 0;
        SetCtrlAttribute(panelHandle,PANEL_RESET,ATTR_DIMMED,1);
        SetCtrlAttribute(panelHandle,PANEL_SAVEDATA,ATTR_DIMMED,1);
        //Deallocate the memory for the large array; put it in the exit
            part of the timer callback, perhaps?
        for(i=0;i<8;i++){
          free(voltage[i]);
        }
        free(voltage);
        free(sumvoltage);
        break;
      case EVENT_RIGHT_CLICK:
        break;
    }
```

```
  return 0;
}
int CVICALLBACK StopCount (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{   int i,j,k;
  switch (event)
  {
    case EVENT_COMMIT:
      SetCtrlAttribute(panelHandle,PANEL_TIMER,ATTR_ENABLED,0);
      DAQmxClearTask(A0input);
      SetCtrlAttribute(panelHandle,PANEL_STOP_COUNT,ATTR_DIMMED,1);
      SetCtrlAttribute(panelHandle,PANEL_RESET,ATTR_DIMMED,0);
      //Create a sum of all voltages for a rough ULD/LLD?
      sumvoltage = (double*)calloc(maxentries,sizeof(double));
      for(j=0;j<total;j++){
        sumvoltage[j] = 0;
        for(k=0;k<channels;k++){
          sumvoltage[j] = sumvoltage[j] + voltage[k][j];
        }
      }
      //histogram the data you have
      //set the graph axis limits
      GetCtrlVal(panelHandle,PANEL_XMAX,&xmax);
      GetCtrlVal(panelHandle,PANEL_XMIN,&xmin);
      GetCtrlVal(panelHandle,PANEL_YMAX,&ymax);
      GetCtrlVal(panelHandle,PANEL_YMIN,&ymin);
      SetAxisScalingMode(panelHandle,PANEL_GRAPH,VAL_LEFT_YAXIS,
          VAL_MANUAL,ymin,ymax);
      for(i=0;i<channels;i++){
        Histogram(voltage[i],total,xmin,xmax,histdata[i],histaxis[i],
            bins);
      }
      DeleteGraphPlot(panelHandle,PANEL_GRAPH,-1,VAL_IMMEDIATE_DRAW);
      for(i=0;i<channels;i++){
        PlotXY(panelHandle,PANEL_GRAPH,&histaxis[i],&histdata[i],bins,
            VAL_DOUBLE,VAL_INTEGER,VAL_THIN_LINE,VAL_EMPTY_SQUARE,
        VAL_SOLID,1,plotcolor[i]);
      }
      SetCtrlAttribute(panelHandle,PANEL_XPOS,ATTR_VISIBLE,1);
      SetCtrlAttribute(panelHandle,PANEL_YPOS,ATTR_VISIBLE,1);
      SetCtrlAttribute(panelHandle,PANEL_RESCALEHIST,ATTR_DIMMED,0);
      SetCtrlAttribute(panelHandle,PANEL_SAVEDATA,ATTR_DIMMED,0);
      break;
    case EVENT_RIGHT_CLICK:
      break;
  }
  return 0;
}

int CVICALLBACK ModeSwitch (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
```

```
{
  switch (event)
  {
    case EVENT_COMMIT:
      GetCtrlVal(panelHandle,PANEL_MODE,&stopon);
      if(stopon == 1){
        SetCtrlAttribute(panelHandle,PANEL_MAX_COUNTS,ATTR_DIMMED,0);
        SetCtrlAttribute(panelHandle,PANEL_MAX_TIME,ATTR_DIMMED,1);
      }
      else{
        SetCtrlAttribute(panelHandle,PANEL_MAX_TIME,ATTR_DIMMED,0);
        SetCtrlAttribute(panelHandle,PANEL_MAX_COUNTS,ATTR_DIMMED,1);
        SetCtrlVal(panelHandle,PANEL_MAX_COUNTS,10000000.00);
      }
      break;
    case EVENT_RIGHT_CLICK:
      break;
  }
  return 0;
}
int CVICALLBACK CallbackTimer (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{ int i,j,k;
  newtime = Timer()-oldtime;
  if(quitting == 1){ //prevents histogram errors when the program exits
      and frees memory
  }
  else if(stopon == 0 && newtime > maxtime || counts > maxcounts){//If
      we've gone to max time, stop
      SetCtrlAttribute(panelHandle,PANEL_TIMER,ATTR_ENABLED,0);
      DAQmxClearTask(A0input);
      //create a sum of all voltages to set a rough ULD/LLD
      sumvoltage = (double*)calloc(maxentries,sizeof(double));
      for(j=0;j<total;j++){
        sumvoltage[j] = 0;
        for(k=0;k<channels;k++){
          sumvoltage[j] = sumvoltage[j] + voltage[k][j];
        }
      }
      //histogram the data you have
      //set the graph axis limits
      GetCtrlVal(panelHandle,PANEL_XMAX,&xmax);
      GetCtrlVal(panelHandle,PANEL_XMIN,&xmin);
      GetCtrlVal(panelHandle,PANEL_YMAX,&ymax);
      GetCtrlVal(panelHandle,PANEL_YMIN,&ymin);
      SetAxisScalingMode(panelHandle,PANEL_GRAPH,VAL_LEFT_YAXIS,
          VAL_MANUAL,ymin,ymax);
      for(i=0;i<channels;i++){
        Histogram(voltage[i],total,xmin,xmax,histdata[i],histaxis[i],
            bins);
      }
```

```
    DeleteGraphPlot(panelHandle,PANEL_GRAPH,-1,VAL_IMMEDIATE_DRAW);
    //plot the data from all channels
    for(i=0;i<channels;i++){
      PlotXY(panelHandle,PANEL_GRAPH,&histaxis[i],&histdata[i],bins,
          VAL_DOUBLE,VAL_INTEGER,VAL_THIN_LINE,VAL_EMPTY_SQUARE,
      VAL_SOLID,1,plotcolor[i]);
    }
    SetCtrlAttribute(panelHandle,PANEL_XPOS,ATTR_VISIBLE,1);
    SetCtrlAttribute(panelHandle,PANEL_YPOS,ATTR_VISIBLE,1);
    SetCtrlAttribute(panelHandle,PANEL_RESET,ATTR_DIMMED,0);
    SetCtrlAttribute(panelHandle,PANEL_STOP_COUNT,ATTR_DIMMED,1);
    SetCtrlAttribute(panelHandle,PANEL_RESCALEHIST,ATTR_DIMMED,0);
    SetCtrlAttribute(panelHandle,PANEL_SAVEDATA,ATTR_DIMMED,0);
}
else if(stopon == 1 && counts > maxcounts){
  SetCtrlAttribute(panelHandle,PANEL_TIMER,ATTR_ENABLED,0);
    DAQmxClearTask(A0input);
    //create a sum of all voltages to set a rough ULD/LLD
    sumvoltage = (double*)calloc(maxentries,sizeof(double));
    for(j=0;j<total;j++){
      sumvoltage[j] = 0;
      for(k=0;k<channels;k++){
        sumvoltage[j] = sumvoltage[j] + voltage[k][j];
      }
    }
    //histogram the data you have
    //set the graph axis limits
    GetCtrlVal(panelHandle,PANEL_XMAX,&xmax);
    GetCtrlVal(panelHandle,PANEL_XMIN,&xmin);
    GetCtrlVal(panelHandle,PANEL_YMAX,&ymax);
    GetCtrlVal(panelHandle,PANEL_YMIN,&ymin);
    SetAxisScalingMode(panelHandle,PANEL_GRAPH,VAL_LEFT_YAXIS,
        VAL_MANUAL,ymin,ymax);
    for(i=0;i<channels;i++){
      Histogram(voltage[i],total,xmin,xmax,histdata[i],histaxis[i],
          bins);
    }
    DeleteGraphPlot(panelHandle,PANEL_GRAPH,-1,VAL_IMMEDIATE_DRAW);
    for(i=0;i<channels;i++){
      PlotXY(panelHandle,PANEL_GRAPH,&histaxis[i],&histdata[i],bins,
          VAL_DOUBLE,VAL_INTEGER,VAL_THIN_LINE,VAL_EMPTY_SQUARE,
      VAL_SOLID,1,plotcolor[i]);
    }
    //show the cursor info
    SetCtrlAttribute(panelHandle,PANEL_XPOS,ATTR_ENABLED,1);
    SetCtrlAttribute(panelHandle,PANEL_YPOS,ATTR_ENABLED,1);
    SetCtrlAttribute(panelHandle,PANEL_RESET,ATTR_DIMMED,0);
    SetCtrlAttribute(panelHandle,PANEL_STOP_COUNT,ATTR_DIMMED,1);
    SetCtrlAttribute(panelHandle,PANEL_RESCALEHIST,ATTR_DIMMED,0);
    SetCtrlAttribute(panelHandle,PANEL_SAVEDATA,ATTR_DIMMED,0);
    }
```

```
    else{
      SetCtrlVal(panelHandle,PANEL_ELAPSED,newtime);
      //Read out the input on AI0
      DAQmxReadAnalogF64(A0input,-1,10.0,DAQmx_Val_GroupByChannel,
          voltbuffer0,20000,&voltsamples,NULL);
      for(index=index,i=0;i<voltsamples;index++,i++){
        for(n=0;n<channels;n++){
          voltage[n][index]=voltbuffer0[(i+n*voltsamples)];
        }
        total++;
      }
      counts = total;//needed if counter not running
      SetCtrlVal(panelHandle,PANEL_COUNTS,counts);
      SetCtrlVal(panelHandle,PANEL_SAMPLES,total);

    }
    return 0;
}
int CVICALLBACK RescaleHist (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
      case EVENT_COMMIT:
        GetCtrlVal(panelHandle,PANEL_XMAX,&xmax);
        GetCtrlVal(panelHandle,PANEL_XMIN,&xmin);
        GetCtrlVal(panelHandle,PANEL_YMAX,&ymax);
        GetCtrlVal(panelHandle,PANEL_YMIN,&ymin);
        //Populate the array that contains the histogram colour data
        plotcolor[0]=VAL_RED;
        plotcolor[1]=VAL_GREEN;
        plotcolor[2]=VAL_BLUE;
        plotcolor[3]=VAL_YELLOW;
        plotcolor[4]=VAL_MAGENTA;
        plotcolor[5]=VAL_WHITE;
        plotcolor[6]=VAL_CYAN;
        plotcolor[7]=VAL_DK_MAGENTA;
        SetAxisScalingMode(panelHandle,PANEL_GRAPH,VAL_LEFT_YAXIS,
            VAL_MANUAL,ymin,ymax);
        for(i=0;i<channels;i++){
          Histogram(voltage[i],total,xmin,xmax,histdata[i],histaxis[i],
              bins);
        }
        DeleteGraphPlot(panelHandle,PANEL_GRAPH,-1,VAL_IMMEDIATE_DRAW);
        for(i=0;i<channels;i++){
          PlotXY(panelHandle,PANEL_GRAPH,&histaxis[i],&histdata[i],bins,
              VAL_DOUBLE,VAL_INTEGER,VAL_THIN_LINE,VAL_EMPTY_SQUARE,
          VAL_SOLID,1,plotcolor[i]);
        }
        SetCtrlAttribute(panelHandle,PANEL_XPOS,ATTR_VISIBLE,1);
        SetCtrlAttribute(panelHandle,PANEL_YPOS,ATTR_VISIBLE,1);
```

```
      break;
    case EVENT_RIGHT_CLICK:
      break;
  }
  return 0;
}
int CVICALLBACK QuitCallback (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
  switch (event)
  {
    case EVENT_COMMIT:
      quitting = 1;
      if(voltage){
        for(i=0;i<8;i++){
            free(voltage[i]);
        }
        free(voltage);
        free(sumvoltage);
      }
      QuitUserInterface(0);
      break;
    case EVENT_RIGHT_CLICK:
      break;
  }
  return 0;
}
int CVICALLBACK SaveData (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{   char addon[2];
  char newpath[10001];
  int z;
  switch (event)
  {
    case EVENT_COMMIT:
      //output sampled data to a file
      status = FileSelectPopup ("", "*.*", "", "", VAL_SAVE_BUTTON, 0,
         0, 1, 1, pathname);
      if(status != VAL_NO_FILE_SELECTED){
        outputhandle1 = fopen(pathname, "w+");
        for(z=0;z<total;z++){
          for (i=0;i<=channels-2;i++){
            num = fprintf(outputhandle1, "%f,",voltage[i][z]);
          }
          num = fprintf(outputhandle1, "%f,\n",voltage[channels-1][z]);
              //write the last channel, start new line
        }
        fclose(outputhandle1);
        MessagePopup ("Ready", "Data Saved");

      }
```

```
        //SetCtrlAttribute(panelHandle,PANEL_SAVEDATA,ATTR_DIMMED,1);
        break;
      case EVENT_RIGHT_CLICK:
        break;
    }
    return 0;
}
int CVICALLBACK SumEnergy (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{   int j,k;
    switch (event)
    {
      case EVENT_COMMIT:
        GetCtrlVal(panelHandle,PANEL_XMAX,&xmax);
        GetCtrlVal(panelHandle,PANEL_XMIN,&xmin);
        GetCtrlVal(panelHandle,PANEL_YMAX,&ymax);
        GetCtrlVal(panelHandle,PANEL_YMIN,&ymin);
        SetAxisScalingMode(panelHandle,PANEL_GRAPH,VAL_LEFT_YAXIS,
            VAL_MANUAL,ymin,ymax);
        Histogram (sumvoltage,total,xmin,xmax,histdata[0],histaxis[0],bins
            );
        DeleteGraphPlot(panelHandle,PANEL_GRAPH,-1,VAL_IMMEDIATE_DRAW);
        PlotXY(panelHandle,PANEL_GRAPH,&histaxis[0],&histdata[0],bins,
            VAL_DOUBLE,VAL_INTEGER,VAL_THIN_LINE,VAL_EMPTY_SQUARE,
          VAL_SOLID,1,VAL_RED);
        break;
      case EVENT_RIGHT_CLICK:
        break;
    }
    return 0;
}
int CVICALLBACK GraphCallback (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{   double xval,yval;
    int xpos,ypos;
    switch (event)
    {
      case EVENT_VAL_CHANGED:
        GetGraphCursor(panelHandle,PANEL_GRAPH,1,&xval,&yval);
        xpos = (int)xval;
        SetCtrlVal(panelHandle,PANEL_XPOS,xval);
        SetCtrlVal(panelHandle,PANEL_YPOS,yval);
        break;
      case EVENT_RIGHT_CLICK:
        break;
    }
    return 0;
}
int CVICALLBACK ChanSelect (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
```

```
  switch (event)
  {
    case EVENT_COMMIT:
      DisplayPanel(panelChannel);
      break;
    case EVENT_RIGHT_CLICK:
      break;
  }
  return 0;
}
int CVICALLBACK ChanHide (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{switch (event)
  {
    case EVENT_COMMIT:
      HidePanel(panelChannel);

      break;
    case EVENT_RIGHT_CLICK:

      break;
  }
  return 0;
}
int CVICALLBACK ShowFlood (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{   //const int limit = total;
  double xfloat,yfloat,xplus,yplus,xminus,yminus,x,y; //floating point
      temporary storage for converting from voltage to position
  int row,i,j,k,image1,edgeimage,blockrow = 0,width=dim,height=dim,xint,
      yint,missed=0,*colortable,temp=0,scale,spots=0;//Note that
      colortable needs to have at least as many values as you have
      shades in your colormap!
  int xpluschan,xminuschan,ypluschan, yminuschan;
  int TESTA,TESTB;//Used for testing only
  unsigned char *imagedata;
  int limit, avg; //Added by AG to make it compile, 3/13/2011.
  switch (event)
  {
    case EVENT_COMMIT:
      //changes needed to make limit and avg global:
      limit = 0;
      //reset the image array to zeros
      for(i=0;i<dim;i++){
        for(j=0;j<dim;j++){
          imagearray[i][j] = 0;
        }
      }
      CanvasClear(panelFlood,PANEL_FLOO_CANVAS1,VAL_ENTIRE_OBJECT);
      imagedata =(unsigned char*)calloc(width*height,sizeof(unsigned
          char)); //allocate memory for the long array storing the image
```

```
    data
//Get channel assignments
GetCtrlVal(panelHandle,PANEL_XPLUS,&xpluschan);
GetCtrlVal(panelHandle,PANEL_XMINUS,&xminuschan);
GetCtrlVal(panelHandle,PANEL_YPLUS,&ypluschan);
GetCtrlVal(panelHandle,PANEL_YMINUS,&yminuschan);
//Allow for block 2 to be used
if(floodblock == 1){
  xpluschan = xpluschan + 4;
  ypluschan = ypluschan + 4;
  xminuschan = xminuschan + 4;
  yminuschan = yminuschan + 4;
}
//convert the voltage data to coordinates
for(k=0;k<total;k++){
  xplus = voltage[xpluschan][k];
  xminus = voltage[xminuschan][k];
  yplus = voltage[ypluschan][k];
  yminus = voltage[yminuschan][k];
  x = ((xplus-xminus)/(xplus+xminus))*128 + 128;
  y = ((yplus-yminus)/(yplus+yminus))*128 + 128;
  xint = (int) x;
  yint = (int) y;
  //code to kludge out the bad values caused by negative voltages
  if(xint > 255){
    xint = 255;
    missed++;
  }
  if(yint > 255){
    yint = 255;
    missed++;
  }
  if(yint < 0){
    yint = 0;
    missed++;
  }
  if(xint < 0){
    xint = 0;
    missed++;
  }
  imagearray[xint][yint] = imagearray[xint][yint] + 1;
}
SetCtrlVal(panelHandle,PANEL_MISSED,missed);
//find maximum of flood histogram
for(i=0;i<255;i++){
  for(j=0;j<255;j++){
    if(imagearray[i][j] > limit){
      limit = imagearray[i][j];
    }
    if(imagearray[i][j] != 0){
      temp = temp + imagearray[i][j];
```

```
            spots++;
          }
        }
      }
      avg = temp/spots;
      for(i=0;i<dim;i++){ //Scale any abnormally high spots to keep them
          from clipping
        for(j=0;j<dim;j++){
          if(imagearray[i][j] > avg+126){
            imagearray[i][j] = avg + 126;
          }
        }
      }
      for(i=0;i<dim;i++){
        for(j=0;j<dim;j++){
          if(imagearray[i][j] > avg + 35){
            block[0][i][j] = imagearray[i][j]; //cut noise from the
                array used for peak/edge finding
          }
        }
      }
      //create the color array
      colortable = (int*)calloc(limit,sizeof(int));//Scale to maximum
          number of counts
      for(i=0,j=limit-255;j<limit;i++,j++){
        colortable[j] = MakeColor(i,i,i);
      }
      scale = limit/127;
      for(row=0;row<height;row++){
        for(i=0,j=0;i<width;i++,j++){
          imagedata[j+(row*dim)]=imagearray[i][row]*scale; //generate
              image data
        }
      }
      //draw the bitmap
      DisplayPanel(panelFlood);
      NewBitmap(width,8,width,height,colortable,imagedata,NULL,&image1);
      CanvasDrawBitmap(panelFlood,PANEL_FLOO_CANVAS1,image1,
          VAL_ENTIRE_OBJECT,MakeRect(0,0,dim,dim));
      free(colortable);
      free(imagedata);
      break;
    case EVENT_RIGHT_CLICK:
      break;
  }
  return 0;
}
int CVICALLBACK FloodClose (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{   int i,j;
  switch (event)
```

```
  {
    case EVENT_COMMIT:
      HidePanel(panelFlood);
      for(i=0;i<dim;i++){
        for(j=0;j<dim;j++){
          block[0][i][j] = 0;
        }
      }
      break;
    case EVENT_RIGHT_CLICK:
      break;
  }
  return 0;
}
int CVICALLBACK LoadData (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{   double **volttest,test1,test2,test3,test4;
  int totaltest,z,i,j,k;
  FILE *inputhandle;
  switch (event)
  {
    case EVENT_COMMIT:
      status = FileSelectPopup ("", "*.dat", "", "", VAL_LOAD_BUTTON, 0,
          0, 1, 1, pathname);
      if(status != VAL_NO_FILE_SELECTED){
        voltage= (double**)malloc(8*sizeof(double*)); //dynamically
            allocate an array with support for all 8 channels
        for(i=0;i<8;i++){
          voltage[i] = (double*)malloc(maxentries*sizeof(double));
        }//allocate memory to load data into
        inputhandle = fopen(pathname, "r");
        for(z=0;z<maxentries;z++){
          num = fscanf(inputhandle,"%lf,%lf,%lf,%lf",&voltage[0][z],&
              voltage[1][z],&voltage[2][z],&voltage[3][z]);
          total++;
          if(feof(inputhandle)){
              fclose(inputhandle);
              break;

          }
        }//end loop over lines of file
        SetCtrlAttribute(panelHandle,PANEL_RESCALEHIST,ATTR_DIMMED,0);
        SetCtrlAttribute(panelHandle,PANEL_START_COUNT,ATTR_DIMMED,1);
        SetCtrlAttribute(panelHandle,PANEL_RESET,ATTR_DIMMED,0);
        SetCtrlAttribute(panelHandle,PANEL_SAVEDATA,ATTR_DIMMED,0);
        channels = 4; //hard-coded for now; should it be otherwise?
        sumvoltage = (double*)calloc(maxentries+2,sizeof(double));
        for(j=0;j<total;j++){
          sumvoltage[j] = 0;
        for(k=0;k<channels;k++){
          sumvoltage[j] = sumvoltage[j] + voltage[k][j];
```

```
        }
      }
        MessagePopup ("Ready", "Data Loaded");
      }//End file selection conditional
      break;
    case EVENT_RIGHT_CLICK:
      break;
  }
  return 0;
}
int CVICALLBACK BlockSwitch (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
  switch (event)
  {
    case EVENT_COMMIT:
      GetCtrlVal(panelHandle,PANEL_BLOCKSWITCH,&floodblock);
      break;
    case EVENT_RIGHT_CLICK:
      break;
  }
  return 0;
}
```

Appendix B

Appendix B: Analysis Code

B.1 MATLAB Scripts

These scripts were developed using MATLAB 7.4, and are designed to be used with either measured data acquired using the LabWindows program mentioned above or the GATE simulations.

B.1.1 manualpeaks

```
%Manual peak location
%Manually locate the peaks and draw boundaries between them
%Set number of crystals in each row/column; only works for square
    matrices
iCr = 20;
imagesc(mFlood)
axis square
colormap hot
title(sprintf('Click on every peak from left to right on each row'));
[peakx,peaky] = ginput(iCr*iCr);
rightedge = zeros(400,1);
bottomedge = zeros(400,1);
%Now create crystal edges
for crystal = 1:iCr-1 %Do the first row separately
    bottomedge(crystal) = int16((peaky(crystal+iCr) - peaky(crystal))/2
        + peaky(crystal));%Take midpoint betweeen this crystal and the
        one below
    rightedge(crystal) = int16((peakx(crystal+1) - peakx(crystal))/2 +
        peakx(crystal));
end
rightedge(iCr) = 256;%Manually set the right edge for the top row
i = 0;
while i < iCr
    rightedge(iCr + i*iCr) = 256;%Assign the very right edge for the
        last column
    i = i + 1;
```

```
end
%Do the other rows now
crystalrow = 2;%Start on the second row
for crystal = iCr:iCr*iCr-iCr%Go from start of second row to second last
    row
    bottomedge(crystal) = int16((peaky(crystal+iCr) - peaky(crystal))/2
       + peaky(crystal));
end
for crystal = iCr*iCr - iCr + 1:iCr*iCr
    bottomedge(crystal) = 256;%Assign the very bottom to the bottom row
        as its bottom edge
end
%Now create the right edge
while crystalrow < iCr+1
    for crystal = 1:iCr-1
        rightedge(crystal+iCr*(crystalrow-1)) = int16((peakx(crystal +
            iCr*(crystalrow-1)+1) - peakx(crystal+ iCr*(crystalrow-1)))
            /2 + peakx(crystal + iCr*(crystalrow-1)));
    end
    crystalrow = crystalrow + 1;
end
%Assign each pixel to its appropriate crystal region
%Pre-allocate the array
mSegRegWarp = zeros(256,256);
crystalrow = 0;
while crystalrow < iCr
    %For each row, go from left to right
    for n = (iCr*crystalrow)+1:iCr*(crystalrow+1)
        %If you're at a left edge, start at 1 instead of the previous
        %crystal's right edge
        if n == crystalrow*iCr + 1
            left = 1;
        else
            left = rightedge(n-1);
        end
        if n <= iCr
            top = 1;
        else
            top = bottomedge(n-iCr);
        end
        for i = left:rightedge(n)
            for j = top:bottomedge(n)
                mSegRegWarp(j,i) = n;
            end
        end
    end
    crystalrow = crystalrow + 1;
end
```

B.1.2 Flood_MATLAB.m

```
function [xplus,xminus,yplus,yminus,x,y,imagearray] = Flood_MATLAB(
    rawcounts,block)
%Flood histogram generator
%Creates a 256x256 flood histogram from voltage values
%Input arguments: rawcounts = 4 or 8 element array
%block = 1 or 2
%Output: x,y = array of x,y values from each count
%imagearray = 256x256x2 array that contains the flood histogram
%Channel assignments: X+ = 2, X- = 4, Y+ = 1, Y- = 3
asize = 256;
%Re-assign raw data to X+,X-,Y+,Y-
if block == 1
yplus = abs(rawcounts(:,1));
yminus = abs(rawcounts(:,3));
xplus = abs(rawcounts(:,2));
xminus = abs(rawcounts(:,4));
else
    yplus = abs(rawcounts(:,5));
    yminus = abs(rawcounts(:,7));
    xplus = abs(rawcounts(:,6));
    xminus = abs(rawcounts(:,8));
end
counts = length(xminus); %Get the size of the array
x = zeros(counts,1);
y = zeros(counts,1);
imagearray = zeros(asize,asize,2);
for i = 1:counts
    x(i) = int16(((xplus(i))/(xplus(i)+xminus(i))*asize));
    y(i) = int16(((yplus(i))/(yplus(i)+yminus(i))*asize));
    %Get rid of values that are outside the possible range
    if x(i) < 1
        x(i) = 1;
    end
    if x(i) > asize
        x(i) = asize;
    end
    if y(i) < 1
        y(i) = 1;
    end
    if y(i) > asize
        y(i) = asize;
    end
    imagearray(y(i),x(i),block) = imagearray(y(i),x(i),block) + 1;
end
```

B.1.3 crystalenergy.m

For use when calibrating a single detector's response.

```
%function [eventindex,energy] = crystalenergy(crystal,x,y,xplus,xminus,
    yplus,yminus,mSegRegWarp)
%Energy histogram routine
%Generates an energy spectrum from a segmented flood histogram and a
    record
%of each x/y position as a function of scintillation event
%Set which crystals to generate energy spectra for
crystalstart = 1;
crystalfinish = 400;
counts = size(xplus);
index = ones(400,1);
%Make an index of which events correspond to which crystal
for k = 1:counts;
    currentcrystal = mSegRegWarp(y(k),x(k));
    eventindex(currentcrystal,index(currentcrystal)) = k;
    index(currentcrystal) = index(currentcrystal) + 1;
end
%Find the energy of each event in the crystal
endpoint = zeros(1,crystalfinish);
for crystal = crystalstart:crystalfinish
    i = 1;%Start a new index of energies for each crystal
    j = 1;%Prevents j from being 0 from the previous crystal
    crystalcounts = size(eventindex(crystal,:));
    for k = 1:crystalcounts(2) %Second dimension gives number of events
        for that crystal
        j = eventindex(crystal,k); %Determines which events are from
            that crystal
        if j ~= 0 %Needed because all arrays are the same size as the
            largest
            %energy(crystal,i) =
            %xplus(j)+xminus(j)+yplus(j)+yminus(j);%Changed May 6, 2009
            tempenergy = xplus(j)+xminus(j)+yplus(j)+yminus(j);
            %Kludge to keep abnormally high readings out; 4 channels x
                10 V
            %max leads to a max value of 40 V
            if tempenergy < 15 %We don't see anything above 15 V
                energy(crystal,i) = tempenergy;
                i = i+ 1;
            end
        else
            endpoint(crystal) = i-1;%Make the endpoint accurate
            break
        end
    end
end
%Create energy histograms
%Pre-allocate histogram array
ehistogram = zeros(crystalfinish,256);
```

```
for crystal = crystalstart:crystalfinish
    [ehistogram(crystal,:),histaxis(crystal,:)] = hist(energy(crystal,1:
        endpoint(crystal)),256);
end
```

B.1.4 calibrate511.m

```
keVolt = zeros(crystalfinish,1);
peakpos = zeros(crystalfinish,1);
for crystal = crystalstart:crystalfinish
    %Find the peak farthest to the right
    [C,peakpos(crystal)] = max(ehistogram(crystal,:));
    %Now determine how many keV are per volt
    keVolt(crystal) = 511/histaxis(crystal,peakpos(crystal));
end
```

B.1.5 crystalenergy_duo

For use when calibration data exists

```
%function [eventindex,energy] = crystalenergy(crystal,x,y,xplus,xminus,
    yplus,yminus,mSegRegWarp)
%Energy histogram routine
%Generates an energy spectrum from a segmented flood histogram and a
    record
%of each x/y position as a function of scintillation event
%ONLY to be used when calibration data already exists!!!
%Set which crystals to generate energy spectra for
crystalstart = 1;
crystalfinish = 400;
LLD = 150;
ULD = 800;
offset1 = 0;%.001;
offset2 = 0;%.001;
counts = size(xplus1);
energy1 = zeros(counts(1),1);
energy2 = zeros(counts(1),1);
%Find energy on a crystal-by-crystal basis if not done yet
%(done while stepping the LLD)
for k = 1:counts;
    crystal1 = mSegRegWarp1(y1(k),x1(k));
    crystal2 = mSegRegWarp2(y2(k),x2(k));
    energy1(k) = (xplus1(k)+xminus1(k)+yplus1(k)+yminus1(k)+offset1)*
        keVolt1(crystal1);
    energy2(k) = (xplus2(k)+xminus2(k)+yplus2(k)+yminus2(k)+offset2)*
        keVolt2(crystal2);
end
index = 1;
windowenergy = zeros(counts(1)*2,1);
for i = 1:counts
```

```
    if energy1(i) > LLD && energy1(i) < ULD && energy2(i) > LLD &&
       energy2(i) < ULD
        windowenergy(index) = energy1(i);
        index = index + 1;
        windowenergy(index) = energy2(i);
        index = index + 1;
    end
end
%Remove non-zero energies
if index < counts(1)*2
    j = 1;
    while windowenergy(j) ~= 0 && j < counts(1)*2;
        j = j + 1;
    end
    nonzeroenergy = windowenergy(1:j-1);
end
[sumhistogram,sumaxis] = hist(nonzeroenergy,128);
```

B.1.6 simcoincidence.m

This script generates an energy spectrum from simulated bench-top data.

```
%Simulated energy spectrum script
%Uses data from GATE ASCII output with A. Goertzen's coincidence mask
counts = size(simdata);
%Pre-allocate part of the array for speed
preall = int32(counts(1)*2);
simenergy = zeros(preall,1);
index = 1;
LLD = 0.35;
ULD = 0.80;
for i = 1:counts(1)
    if simdata(i,7) == 3 && simdata(i,17) == simdata(i,7)
        if simdata(i,4) > LLD && simdata(i,4) < ULD && simdata(i,14) >
           LLD && simdata(i,14) < ULD
            simenergy(index) = simdata(i,4);
            index = index + 1;
            simenergy(index) = simdata(i,14);
            index = index + 1;
        end

    end
end
%Determine where the non-zero points of simenergy end
if preall > index
    j = 1;
    while simenergy(j) ~= 0 && j < preall;
        j = j + 1;
    end
    nonzerosimenergy = simenergy(1:j-1);
end
```

```
%Create the energy spectrum
[simhistogram,simaxis] = hist(nonzerosimenergy,128);
simaxis = simaxis.*1000;
```

B.1.7 readonly.m

Reads a measured data set from the full Inveon system.

```
function [sinogram] = readonly(filename)
%Reads a mashed sinogram from an Inveon scanner
%Assumes data is row x column x slice int16
%Declare the dimensions of the sinogram
rows = 160;
columns = 128;
slices = 159;
%Pre-allocate the sinogram
sinogram = int16(zeros(rows,columns,slices));
%Open the file and read out the data
fid = fopen(filename);
for i = 1:slices
    tempa = fread(fid,[columns,rows],'int16');
    tempb = tempa';%Transpose the matrix since MATLAB stores it as row/
        column
    sinogram(:,:,i) = tempb;
end
fclose(fid);
```

B.1.8 loadmes_10keV.m

Loads a series of scan data from a Siemens Inveon system to calculate the coincidence energy spectrum in the manner described in [57]. Note that there are kludges here for peculiarities in the data set acquired in 2009!

```
sino_lld = int16(zeros(160,128,159,65));
sino_uld = int16(zeros(160,128,159,65));
counts_lld = int32(zeros(65,1));
counts_uld = int32(zeros(65,1));
%Load the LLD stepped data
j = 150;
for i = 1:65
pathname = ['Background_Run1_Window_9min_',num2str(j),'-800keV_em_v1.scn
    '];
[sino_temp] = readonly(pathname);
sino_lld(:,:,:,i) = sino_temp;
j = j + 10;
clear sino_temp%Free memory to speed execution
end
%Load the ULD stepped data
j = 150;
for i = 1:65
```

```
pathname = ['Background_Run1_Window_9min_50-',num2str(j),'keV_em_v1.scn'
    ];
[sino_temp] = readonly(pathname);
sino_uld(:,:,:,i) = sino_temp;
j = j + 10;
clear sino_temp%Free memory to speed execution
end
for i = 1:65
    temp = sum(sum(sum(sino_lld(:,:,:,i))));
    counts_lld(i) = temp;
    %kludge out negative LLD stepping values
    if counts_lld(i) < 0
        counts_lld(i) = 0;
    end
    temp2 = sum(sum(sum(sino_uld(:,:,:,i))));
    counts_uld(i) = temp2;
end
%Kludge out ULD stepping values that are multiplied by 1000
for i = 1:6
    counts_uld(i) = counts_uld(i)/1000;
end
counts_uld(8) = counts_uld(8)/1000;
%Create the energy spectrum
%Calculate the differences
difflld = zeros(64,1);
diffuld = zeros(64,1);
for i = 1:64
    difflld(i) = counts_lld(i)-counts_lld(i+1);
    diffuld(i) = counts_uld(i+1)-counts_uld(i);
end
%Calculate the spectrum
spectrum = zeros(64,1);
spectaxis = zeros(64,1);
deltaE = 10;
for i = 1:64
    spectrum(i) = (difflld(i)+diffuld(i))/(2*deltaE);
    spectaxis(i) = 150+deltaE*(i-1)+deltaE/2;
end
```

B.1.9 sinomash.m

Creates a Michelogram to determine how to rebin simulation data from the raw set of 2D sinograms.

```
michelogram = zeros(80,80);
sinoindex = 1;
ystart = 1;
yfinish = 80;
xstart = 1;
xfinish = 80;
ringmax = 39;
```

```
%Start populating the michelogram with the central diagonal line
j = ystart;
for i = xstart:xfinish
    michelogram(j,i) = sinoindex;
    j = j + 1;
    sinoindex = sinoindex + 1;
end
%Now populate the bottom half, using the ring difference as the stop
    point
ystart = ystart + 1;
xfinish = xfinish - 1;
x2start = xstart + 1;%Starting point for the upper half of the
    michelogram
y2start = 1;%Need to start the upper half at the top of the image
while ystart < ringmax+1 %We'll never start higher than this
    %Populate the lower half of the michelogram
    j = ystart;
    for i = xstart:xfinish
        michelogram(j,i) = sinoindex;
        j = j + 1;
        sinoindex = sinoindex + 1;
    end
    %Now do the upper half
    j = y2start;
    for i = x2start:80
        michelogram(j,i) = sinoindex;
        j = j + 1;
        sinoindex = sinoindex + 1;
    end
    %Iterate the starting positions for the next line
    ystart = ystart + 1;
    xfinish = xfinish - 1;
    x2start = x2start + 1;
end
mashindex1 = 2;%mashed sinogram number
mashindex2 = 1;%unmashed sinogram index
%Check to see which lines are being covered
mashplot = zeros(80,80);
%Do the first point separately
mashplot(1,1) = 10000;
mashsino(1,1) = michelogram(1,1);
%Now fill as many sinograms as possible starting from the top and going
%down diagonally then up once you hit the left side
i = 2;
while i < 81
    j = 1;
    %Go down if the spot you're at has no michelogram data
    if michelogram(j,i) == 0
        while michelogram(j,i) == 0
            j = j + 1;
        end
```

```
        %Check to make sure you haven't missed a sinogram
        if michelogram(j,i-1) ~= 0
            i = i - 1;
        end
    end
    %Go down diagonally until you hit the edge or empty space
    while i > 0
        mashplot(j,i) = mashplot(j,i) + 10000;
        mashsino(mashindex1,mashindex2) = michelogram(j,i);%Record which
            2D sinograms belong to this mashed sinogram
        mashindex2 = mashindex2 + 1;
        i = i - 1;
        j = j + 1;
        if i == 0 || j > 80 || michelogram(j,i) == 0
            break
        end
    end
    mashindex1 = mashindex1 + 1;
    mashindex2 = 1;%Start over again
    %Now go over one pixel to the right and go up to grab the other
        section
    i = i + 1;
    while j > 0
        mashplot(j,i) = mashplot(j,i) + 10000;
        mashsino(mashindex1,mashindex2) = michelogram(j,i);
        mashindex2 = mashindex2 + 1;
        i = i + 1;
        j = j - 1;
        if j == 0 || i > 80 || michelogram(j,i) == 0
            break
        end
    end
    mashindex1 = mashindex1 + 1;
    mashindex2 = 1;
    %i = i + 1;
end
%Fill in the remaining sinograms
i = 80;
j = j + 2;%Go down two spots to start your final line
flag = 0;
while flag == 0
    while j < 81%Go down diagonally from here
        mashplot(j,i) = mashplot(j,i) + 10000;
        mashsino(mashindex1,mashindex2) = michelogram(j,i);
        mashindex2 = mashindex2 + 1;
        j = j + 1;
        i = i - 1;
    end
    mashindex1 = mashindex1 + 1;
    mashindex2 = 1;
    if i >= 78 && j >= 80
```

```
        flag = 1;
    end
    %Now go up and to the right
    i = i + 2;%Needed to offset the final act of the loop before it
        broke
    j = 80;%Needed to get j back to j = 80
    while i < 81
        mashplot(j,i) = mashplot(j,i) + 10000;
        mashsino(mashindex1,mashindex2) = michelogram(j,i);
        mashindex2 = mashindex2 + 1;
        i = i + 1;
        j = j - 1;
    end
    %Move down to do the next row
    j = j + 2;
    if i > 80
        i = 80;
    end
    mashindex1 = mashindex1 + 1;
    mashindex2 = 1;
end
```

B.1.10 masher_direct.m

Converts data from GATE's sinogram output to match the format from the Inveon system. The script first loads the simulated data, generates a Michelogram to determine how to rebin the data set, and then performs the rebinning.

```
function [sino_mashed,sinogram] = masher_direct(filename)
%Sinogram mashing file
%Converts 2D sinogram output from GATE to the Inveon format
%Hard-coded to do 6400 sinograms
%Note that the working directory must be set to the location of the
    files!
%Load sinogram data into the file
fid = fopen(filename);
%Pre-allocate the sinogram data
sinogram = uint16(zeros(160,128,6400));
%Load in the data one sinogram at a time
for i = 1:6400
    tempa = fread(fid,[128,160],'uint16');
    tempb = tempa';
    sinogram(:,:,i) = tempb;
end
fclose(fid);
disp('Data loaded; generating michelograms')
%Call the script to generate the michelogram that will state how the
%sinograms are to be mashed
sinomash
disp('Mashing sinograms')
```

```
%Generate 3D sinograms
sino_mashed = uint16(zeros(160,128,159));
for index = 1:159
    i = 1;%Start the index for each mashing
    while i < ringmax + 1 && mashsino(index,i) ~= 0
        %Get the number of the 2D sinogram to add to the current mashed
            sinogram
        sinonumber = mashsino(index,i);
        %Add the sinogram data together
        sino_mashed(:,:,index) = sino_mashed(:,:,index) + sinogram(:,:,
            sinonumber);
        i = i + 1;
    end
end
disp('Complete!')
```

B.1.11 simspectrum_10keV.m

Creates a coincidence energy spectrum from the simulated Siemens Inveon system.

```
%Create a coincidence energy spectrum using info from
%pre-existing variables counts_lld and counts_uld
%Calculate the differences
%difflld = zeros(21,1);
%diffuld = zeros(21,1);
difflld = zeros(65,1);
diffuld = zeros(65,1);
%for i = 1:20
for i = 1:65
    difflld(i) = counts_lld(i)-counts_lld(i+1);
    diffuld(i) = counts_uld(i+1)-counts_uld(i);
    %difflld(i) = lld_clipped(i)-lld_clipped(i+1);
    %diffuld(i) = uld_clipped(i+1)-uld_clipped(i);
end
%Calculate the spectrum
%simspectrum = zeros(20,1);
%simspectaxis = zeros(20,1);
simspect = zeros(65,1);
simspectaxis = zeros(65,1);
deltaE = 10;
for i = 1:65
    simspect(i) = (difflld(i)+diffuld(i))/(2*deltaE);
    simspectaxis(i) = 150+deltaE*(i-1)+deltaE/2;
end
```

B.1.12 sum20.m

Creates a profile of the central 20 sinogram slices from both a data file from an acquisition on an Inveon scanner and a simulated acquisition. Requires an existing

simulated sinogram (sino_sim) to be loaded using masher_direct.m before running.

```
function [sinogram,measprofile,simprofile] = sum20(filename,sino_sim)
%Reads a mashed sinogram from an Inveon scanner
%Assumes data is row x column x slice int16

%Declare the dimensions of the sinogram
rows = 160;
columns = 128;
slices = 159;
%Pre-allocate the sinogram
sinogram = int16(zeros(rows,columns,slices));
%Open the file and read out the data
fid = fopen(filename);
for i = 1:slices
    tempa = fread(fid,[columns,rows],'int16');
    tempb = tempa';%Transpose the matrix since MATLAB stores it as row/
        column
    sinogram(:,:,i) = tempb;
end
%Sum the slices of the 20 central profiles (slices 69-89)
%Requires you to have a simulated sinogram loaded!
measprofile = int16(zeros(1,columns));
simprofile = uint16(zeros(1,columns));
for j = 69:89
    for i = 1:rows
        measprofile = measprofile + sinogram(i,:,j);
        simprofile = simprofile + sino_sim(i,:,j);
    end
end
```

Appendix C

Appendix C: GATE Simulations

C.1 Bench-Top Script

This script is for the measurement with 25.5 cm separation between the detectors. The simulations at 8.5 and 15.5 cm were similar, except with different values for the translation of the detectors.

```
#Simulation of 2 opposing detector blocks in a microPET Focus
#Visualization
/vis/disable
/vis/open OGLSX
/vis/viewer/reset
/vis/viewer/set/viewpointThetaPhi 30 30
/vis/viewer/zoom 2.6
/vis/viewer/set/style surface
/vis/drawVolume
/tracking/storeTrajectory 1
/vis/scene/endOfEventAction accumulate 1000
/vis/viewer/update
/gate/geometry/enableAutoUpdate
#Define World as a 50cm^3 box
/gate/world/geometry/setXLength 50. cm
/gate/world/geometry/setYLength 50. cm
/gate/world/geometry/setZLength 50. cm
#Insert a cylindrical PET system
/gate/world/daughters/name cylindricalPET
/gate/world/daughters/insert cylinder
/gate/cylindricalPET/setMaterial Air
/gate/cylindricalPET/geometry/setRmax 15 cm
/gate/cylindricalPET/geometry/setRmin 4 cm
/gate/cylindricalPET/geometry/setHeight 14.3 cm
/gate/cylindricalPET/vis/forceWireframe
#Visualization for the source
#/gate/cylindricalPET/daughters/name test
#/gate/cylindricalPET/daughters/insert cylinder
#/gate/test/setMaterial Air
```

```
#/gate/test/geometry/setRmax 10 cm
#/gate/test/geometry/setHeight 14.3 cm
#/gate/test/vis/setColor green
#Sector that holds block
/gate/cylindricalPET/daughters/name rsector
/gate/cylindricalPET/daughters/insert box
/gate/rsector/geometry/setXLength 10 mm
/gate/rsector/geometry/setYLength 32 mm
/gate/rsector/geometry/setZLength 14 cm
/gate/rsector/setMaterial Vacuum
/gate/rsector/placement/setTranslation 13.30 0 0 cm
/gate/rsector/vis/setColor red
/gate/rsector/vis/forceWireframe
#Block that holds crystals
/gate/rsector/daughters/name module
/gate/rsector/daughters/insert box
/gate/module/geometry/setXLength 10 mm
/gate/module/geometry/setYLength 31.9 mm
/gate/module/geometry/setZLength 31.9 mm
/gate/module/setMaterial Air
/gate/module/vis/forceWireframe
/gate/module/vis/setColor white
#Define the crystal
/gate/module/daughters/name crystal
/gate/module/daughters/insert box
/gate/crystal/geometry/setXLength 10 mm
/gate/crystal/geometry/setYLength 1.51 mm
/gate/crystal/geometry/setZLength 1.51 mm
/gate/crystal/setMaterial LSO
/gate/crystal/vis/setColor yellow
/gate/crystal/vis/forceWireframe
#Add aluminum casing to the front of the detectors
/gate/world/daughters/name shield
/gate/world/daughters/insert box
/gate/shield/geometry/setXLength 1 mm
/gate/shield/geometry/setYLength 32 mm
/gate/shield/geometry/setZLength 140 mm
/gate/shield/setMaterial Aluminium
/gate/shield/vis/setColor gray
/gate/shield/repeaters/insert linear
/gate/shield/linear/setRepeatNumber 2
/gate/shield/linear/setRepeatVector 25.5 0. 0. cm
#Repeat the crystal
/gate/crystal/repeaters/insert cubicArray
/gate/crystal/cubicArray/setRepeatNumberX 1
/gate/crystal/cubicArray/setRepeatNumberY 20
/gate/crystal/cubicArray/setRepeatNumberZ 20
/gate/crystal/cubicArray/setRepeatVector 0. 1.59 1.59 mm
#Repeat module
/gate/module/repeaters/insert linear
/gate/module/linear/setRepeatNumber 4
```

```
/gate/module/linear/setRepeatVector 0. 0. 33. mm
#Add reflector material
#/gate/rsector/daughters/name reflect
#/gate/rsector/daughters/insert box
#/gate/reflect/geometry/setXLength 10 mm
#/gate/reflect/geometry/setYLength 31.8 mm
#/gate/reflect/geometry/setZLength 1 mm
#/gate/reflect/setMaterial Plastic
#/gate/reflect/vis/setColor green
#/gate/reflect/placement/setTranslation 0. 0. -0.1 mm
#/gate/reflect/repeaters/insert linear
#/gate/reflect/linear/setRepeatNumber 3
#/gate/reflect/linear/setRepeatVector 0. 0. 33. mm
#Repeat rsector (and reflector)
/gate/rsector/repeaters/insert ring
/gate/rsector/ring/setRepeatNumber 2
#Attach system components
/gate/systems/cylindricalPET/rsector/attach rsector
/gate/systems/cylindricalPET/module/attach module
/gate/systems/cylindricalPET/crystal/attach crystal
/gate/crystal/attachCrystalSD
#Define Digitizer equipment
#Adder; regroups hits per volume into a pulse
/gate/digitizer/Singles/insert adder
#Readout
/gate/digitizer/Singles/insert readout
/gate/digitizer/Singles/readout/setDepth 2
#Blurring
#This will set a 13.3% energy resolution at 511 keV
/gate/digitizer/Singles/insert blurring
/gate/digitizer/Singles/blurring/setResolution 0.133
/gate/digitizer/Singles/blurring/setEnergyOfReference 511. keV
#Add crystal blurring when you move to a block
#Energy Window
/gate/digitizer/Singles/insert thresholder
/gate/digitizer/Singles/thresholder/setThreshold 150. keV
/gate/digitizer/Singles/insert upholder
/gate/digitizer/Singles/upholder/setUphold 1000. keV
#Coincidence sorter
/gate/digitizer/Coincidences/setWindow 20. ns
/gate/digitizer/Coincidences/MultiplesPolicy  takeWinnerOfGoods
/gate/digitizer/Coincidences/minSectorDifference 1
/gate/digitizer/name PETdelays
/gate/digitizer/insert coincidenceSorter
/gate/digitizer/PETdelays/setInputName Singles
/gate/digitizer/PETdelays/setWindow 10. ns
/gate/digitizer/PETdelays/setOffset 1. mus
/gate/digitizer/PETdelays/MultiplesPolicy  takeWinnerOfGoods
/gate/digitizer/PETdelays/setDepth 2
/gate/digitizer/PETdelays/minSectorDifference 1
#Define Physics
```

```
/gate/physics/gamma/selectPhotoelectric lowenergy
/gate/physics/gamma/selectCompton lowenergy
/gate/physics/gamma/selectGammaConversion standard
/gate/physics/gamma/selectRayleigh lowenergy
/gate/physics/setElectronCut 0.1 mm
/gate/physics/setXRayCut 1. keV
/gate/physics/setDeltaRayCut 1. keV
#Initialize Physics
/run/initialize
#Verbosity suppression
/gate/systems/cylindricalPET/verbose 0
/control/verbose 0
/grdm/verbose 0
/run/verbose 0
/event/verbose 0
/tracking/verbose 0
/gate/application/verbose 0
/gate/generator/verbose 0
/gate/stacking/verbose 0
/gate/event/verbose 0
/gate/source/verbose 0
#Single intrinsic source; confined to crystal
/gate/source/addSource intrinsic
/gate/source/intrinsic/gps/particle ion
/gate/source/intrinsic/gps/ion 71 176 0 0
/gate/source/intrinsic/gps/monoenergy 0. keV #Super important!
/gate/source/intrinsic/gps/angtype iso
/gate/source/intrinsic/gps/type Volume
/gate/source/intrinsic/gps/shape Cylinder
/gate/source/intrinsic/gps/radius 14.5 cm
/gate/source/intrinsic/gps/halfz 70.1 mm
/gate/source/intrinsic/setActivity 20192.56 Bq
#Confine to crystals
/gate/source/intrinsic/gps/confine crystal_P
#Output configuration
/gate/output/ascii/setOutFileHitsFlag 0
/gate/output/ascii/setOutFileSinglesFlag 0
/gate/output/ascii/setOutFileCoincidencesFlag 1
#/gate/output/ascii/setOutFileSizeLimit 5000000 #Sets file size to 5
    megs
/gate/output/ascii/setFileName 25.5cm_12hrb
/gate/output/ascii/setSingleMask 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 0 0 0
     0 0 0
/gate/output/ascii/setCoincidenceMask 1 1 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0
    0 1 1 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0
#Disable ROOT output
#/gate/output/root/disable
/gate/output/root/setFileName 25.5cm_12hrb
/gate/output/root/setRootCoincidencesFlag 1
/gate/output/root/setRootSinglesFlag 0
/gate/output/root/setRootHitFlag 0
```

```
/gate/output/lmf/disable
#Set random seed for runb
/gate/random/setEngineName MersenneTwister
/gate/random/setEngineSeed auto
#Start simulation
/gate/application/setTimeSlice 1 s
/gate/application/setTimeStart 0. s
/gate/application/setTimeStop 10800. s
/gate/application/startDAQ
```

C.2 Full System Simulations

C.2.1 Intrinsic Activity Only

This simulation is used to create intrinsic activity coincidence spectra for a complete Siemens Inveon system. The LLD and ULD were modified appropriately to provide a complete spectrum as described in 4.2.

```
#Simulation of a Siemens Inveon using ECAT output
#Visualization
/vis/disable
#/vis/open OGLSX
/vis/viewer/reset
/vis/viewer/set/viewpointThetaPhi 30 30
/vis/viewer/zoom 2.6
/vis/viewer/set/style surface
/vis/drawVolume
/tracking/storeTrajectory 1
/vis/scene/endOfEventAction accumulate 1000
/vis/viewer/update
/gate/geometry/enableAutoUpdate
#Define the world
/gate/world/geometry/setXLength 40. cm
/gate/world/geometry/setYLength 40. cm
/gate/world/geometry/setZLength 40. cm
/gate/world/daughters/name ecat
/gate/world/daughters/insert cylinder
/gate/ecat/setMaterial Air
/gate/ecat/geometry/setRmax 13. cm
/gate/ecat/geometry/setRmin 4. cm
/gate/ecat/geometry/setHeight 14.3 cm
/gate/ecat/vis/forceWireframe
#Define a block detector
/gate/ecat/daughters/name block
/gate/ecat/daughters/insert box
/gate/block/placement/setTranslation 8.554 0 0 cm
/gate/block/geometry/setXLength 10 mm
/gate/block/geometry/setYLength 32.0 mm
/gate/block/geometry/setZLength 32.0 mm
```

```
/gate/block/setMaterial Air
/gate/block/vis/forceWireframe
/gate/block/vis/setColor red
#Define the crystals
/gate/block/daughters/name crystal
/gate/block/daughters/insert box
/gate/crystal/geometry/setXLength 10 mm
/gate/crystal/geometry/setYLength 1.51 mm
/gate/crystal/geometry/setZLength 1.51 mm
/gate/crystal/setMaterial LSO
/gate/crystal/vis/setColor yellow
/gate/crystal/vis/forceWireframe
#Repeat the crystals
/gate/crystal/repeaters/insert cubicArray
/gate/crystal/cubicArray/setRepeatNumberX 1
/gate/crystal/cubicArray/setRepeatNumberY 20
/gate/crystal/cubicArray/setRepeatNumberZ 20
/gate/crystal/cubicArray/setRepeatVector 0. 1.6 1.6 mm
#Repeat the block linearly
/gate/block/repeaters/insert linear
/gate/block/linear/setRepeatNumber 4
/gate/block/linear/setRepeatVector 0. 0. 32. mm
#Construct the ring
/gate/block/repeaters/insert ring
/gate/block/ring/setRepeatNumber 16
#Add aluminum casing to the front of the detectors
/gate/world/daughters/name shield
/gate/world/daughters/insert box
/gate/shield/geometry/setXLength 1 mm
/gate/shield/geometry/setYLength 32 mm
/gate/shield/geometry/setZLength 140 mm
/gate/shield/setMaterial Aluminium
/gate/shield/vis/setColor gray
/gate/shield/repeaters/insert linear
/gate/shield/linear/setRepeatNumber 2
/gate/shield/linear/setRepeatVector 16. 0. 0. cm
/gate/shield/repeaters/insert ring
/gate/shield/ring/setRepeatNumber 16
#Inner aluminum shielding; 0.25 mm thick
/gate/world/daughters/name inshield
/gate/world/daughters/insert cylinder
/gate/inshield/geometry/setRmax 79.5 mm
/gate/inshield/geometry/setRmin 79.25 mm
/gate/inshield/geometry/setHeight 140 mm
/gate/inshield/setMaterial Aluminium
/gate/inshield/vis/setColor green
#Inner carbon fibre; 0.5 mm thick
/gate/world/daughters/name cfibre
/gate/world/daughters/insert cylinder
/gate/cfibre/geometry/setRmax 79.24 mm
/gate/cfibre/geometry/setRmin 78.75 mm
```

```
/gate/cfibre/geometry/setHeight 140 mm
/gate/cfibre/setMaterial Carbonfibre
/gate/cfibre/vis/setColor gray
#Attach system
/gate/systems/ecat/block/attach block
/gate/systems/ecat/crystal/attach crystal
/gate/crystal/attachCrystalSD
#Define Digitizer equipment
#Adder; regroups hits per volume into a pulse
/gate/digitizer/Singles/insert adder
#Readout
/gate/digitizer/Singles/insert readout
/gate/digitizer/Singles/readout/setDepth 1 #Depth 1 = block
#Blurring
#This will set a 13.3% energy resolution at 511 keV
/gate/digitizer/Singles/insert blurring
/gate/digitizer/Singles/blurring/setResolution 0.133
/gate/digitizer/Singles/blurring/setEnergyOfReference 511. keV
#Energy Window
/gate/digitizer/Singles/insert thresholder
/gate/digitizer/Singles/thresholder/setThreshold 50. keV
/gate/digitizer/Singles/insert upholder
/gate/digitizer/Singles/upholder/setUphold 800. keV
#Coincidence sorter
/gate/digitizer/Coincidences/setWindow 3.432 ns
/gate/digitizer/Coincidences/MultiplesPolicy  takeWinnerOfGoods
/gate/digitizer/Coincidences/minSectorDifference 1
/gate/digitizer/name PETdelays
/gate/digitizer/insert coincidenceSorter
/gate/digitizer/PETdelays/setInputName Singles
/gate/digitizer/PETdelays/setWindow 10. ns
/gate/digitizer/PETdelays/setOffset 1. mus
/gate/digitizer/PETdelays/MultiplesPolicy  takeWinnerOfGoods
/gate/digitizer/PETdelays/setDepth 1
/gate/digitizer/PETdelays/minSectorDifference 1
#Define Physics
/gate/physics/gamma/selectPhotoelectric lowenergy
/gate/physics/gamma/selectCompton lowenergy
/gate/physics/gamma/selectGammaConversion standard
/gate/physics/gamma/selectRayleigh lowenergy
/gate/physics/setElectronCut 0.1 mm
/gate/physics/setXRayCut 1. keV
/gate/physics/setDeltaRayCut 1. keV
#Initialize Physics
/run/initialize
#Verbosity suppression
/gate/systems/ecat/verbose 0
/control/verbose 0
/grdm/verbose 0
/run/verbose 0
/event/verbose 0
```

```
/tracking/verbose 0
/gate/application/verbose 0
/gate/generator/verbose 0
/gate/stacking/verbose 0
/gate/event/verbose 0
/gate/source/verbose 0
#Single intrinsic source; confined to crystal
/gate/source/addSource intrinsic
/gate/source/intrinsic/gps/particle ion
/gate/source/intrinsic/gps/ion 71 176 0 0
/gate/source/intrinsic/gps/monoenergy 0. keV #Super important!
/gate/source/intrinsic/gps/angtype iso
/gate/source/intrinsic/gps/type Volume
/gate/source/intrinsic/gps/shape Cylinder
/gate/source/intrinsic/gps/radius 10 cm
/gate/source/intrinsic/gps/halfz 70.1 mm
/gate/source/intrinsic/setActivity 161540.48 Bq
/gate/source/intrinsic/gps/confine crystal_P
#Output configuration
#Sinogram output
/gate/output/sinogram/RadialBins 128
/gate/output/sinogram/RawOutputEnable
/gate/output/sinogram/TruesOnly true
/gate/output/sinogram/setFileName uld800
/gate/output/ecat7/disable
/gate/output/ascii/disable
#ROOT output
/gate/output/root/disable
#Start simulation
/gate/application/setTimeSlice 540 s
/gate/application/setTimeStart 0. s
/gate/application/setTimeStop 540. s
/gate/application/startDAQ
```

C.2.2 Water Tube Simulation

This simulation script produces the simulated water phantom sinograms as described in 4.2. The LLD was stepped as appropriate to match the data generated from the real Inveon system.

```
#Simulation of a Siemens Inveon using ECAT output
#Visualization
/vis/disable
#/vis/open OGLSX
/vis/viewer/reset
/vis/viewer/set/viewpointThetaPhi 30 30
/vis/viewer/zoom 2.6
/vis/viewer/set/style surface
/vis/drawVolume
/tracking/storeTrajectory 1
```

```
/vis/scene/endOfEventAction accumulate 1000
/vis/viewer/update
/gate/geometry/enableAutoUpdate
#Define the world
/gate/world/geometry/setXLength 40. cm
/gate/world/geometry/setYLength 40. cm
/gate/world/geometry/setZLength 40. cm
/gate/world/daughters/name ecat
/gate/world/daughters/insert cylinder
/gate/ecat/setMaterial Air
/gate/ecat/geometry/setRmax 13. cm
/gate/ecat/geometry/setRmin 4. cm
/gate/ecat/geometry/setHeight 14.3 cm
/gate/ecat/vis/forceWireframe
#Define a block detector
/gate/ecat/daughters/name block
/gate/ecat/daughters/insert box
/gate/block/placement/setTranslation 8.554 0 0 cm
/gate/block/geometry/setXLength 10 mm
/gate/block/geometry/setYLength 32.0 mm
/gate/block/geometry/setZLength 32.0 mm
/gate/block/setMaterial Air
/gate/block/vis/forceWireframe
/gate/block/vis/setColor red
#Define the crystals
/gate/block/daughters/name crystal
/gate/block/daughters/insert box
/gate/crystal/geometry/setXLength 10 mm
/gate/crystal/geometry/setYLength 1.51 mm
/gate/crystal/geometry/setZLength 1.51 mm
/gate/crystal/setMaterial LSO
/gate/crystal/vis/setColor yellow
/gate/crystal/vis/forceWireframe
#Repeat the crystals
/gate/crystal/repeaters/insert cubicArray
/gate/crystal/cubicArray/setRepeatNumberX 1
/gate/crystal/cubicArray/setRepeatNumberY 20
/gate/crystal/cubicArray/setRepeatNumberZ 20
/gate/crystal/cubicArray/setRepeatVector 0. 1.6 1.6 mm
#Repeat the block linearly
/gate/block/repeaters/insert linear
/gate/block/linear/setRepeatNumber 4
/gate/block/linear/setRepeatVector 0. 0. 32. mm
#Construct the ring
/gate/block/repeaters/insert ring
/gate/block/ring/setRepeatNumber 16
#Add aluminum casing to the front of the detectors
/gate/world/daughters/name shield
/gate/world/daughters/insert box
/gate/shield/geometry/setXLength 1 mm
/gate/shield/geometry/setYLength 32 mm
```

```
/gate/shield/geometry/setZLength 140 mm
/gate/shield/setMaterial Aluminium
/gate/shield/vis/setColor gray
/gate/shield/repeaters/insert linear
/gate/shield/linear/setRepeatNumber 2
/gate/shield/linear/setRepeatVector 16. 0. 0. cm
/gate/shield/repeaters/insert ring
/gate/shield/ring/setRepeatNumber 16
#Inner aluminum shielding; 0.25 mm thick
/gate/world/daughters/name inshield
/gate/world/daughters/insert cylinder
/gate/inshield/geometry/setRmax 79.5 mm
/gate/inshield/geometry/setRmin 79.25 mm
/gate/inshield/geometry/setHeight 140 mm
/gate/inshield/setMaterial Aluminium
/gate/inshield/vis/setColor green
#Inner carbon fibre; 0.5 mm thick
/gate/world/daughters/name cfibre
/gate/world/daughters/insert cylinder
/gate/cfibre/geometry/setRmax 79.24 mm
/gate/cfibre/geometry/setRmin 78.75 mm
/gate/cfibre/geometry/setHeight 140 mm
/gate/cfibre/setMaterial Carbonfibre
/gate/cfibre/vis/setColor gray
#Water phantom
/gate/world/daughters/name wbox1
/gate/world/daughters/insert cylinder
/gate/wbox1/geometry/setRmax 14.25 mm
/gate/wbox1/geometry/setHeight 98.20 mm
/gate/wbox1/setMaterial Water
/gate/wbox1/vis/setColor blue
/gate/wbox1/placement/setTranslation 0. 0. 0. mm
#Attach system
/gate/systems/ecat/block/attach block
/gate/systems/ecat/crystal/attach crystal
/gate/crystal/attachCrystalSD
#Define Digitizer equipment
#Adder; regroups hits per volume into a pulse
/gate/digitizer/Singles/insert adder
#Readout
/gate/digitizer/Singles/insert readout
/gate/digitizer/Singles/readout/setDepth 1 #Depth 1 = block
#Blurring
#This will set a 13.3% energy resolution at 511 keV
/gate/digitizer/Singles/insert blurring
/gate/digitizer/Singles/blurring/setResolution 0.133
/gate/digitizer/Singles/blurring/setEnergyOfReference 511. keV
#Energy Window
/gate/digitizer/Singles/insert thresholder
/gate/digitizer/Singles/thresholder/setThreshold 300. keV
/gate/digitizer/Singles/insert upholder
```

```
/gate/digitizer/Singles/upholder/setUphold 800. keV
#Coincidence sorter
/gate/digitizer/Coincidences/setWindow 3.432 ns
/gate/digitizer/Coincidences/MultiplesPolicy  takeWinnerOfGoods
/gate/digitizer/Coincidences/minSectorDifference 1
/gate/digitizer/name PETdelays
/gate/digitizer/insert coincidenceSorter
/gate/digitizer/PETdelays/setInputName Singles
/gate/digitizer/PETdelays/setWindow 10. ns
/gate/digitizer/PETdelays/setOffset 1. mus
/gate/digitizer/PETdelays/MultiplesPolicy  takeWinnerOfGoods
/gate/digitizer/PETdelays/setDepth 1
/gate/digitizer/PETdelays/minSectorDifference 1
#Define Physics
/gate/physics/gamma/selectPhotoelectric lowenergy
/gate/physics/gamma/selectCompton lowenergy
/gate/physics/gamma/selectGammaConversion standard
/gate/physics/gamma/selectRayleigh lowenergy
/gate/physics/setElectronCut 0.1 mm
/gate/physics/setXRayCut 1. keV
/gate/physics/setDeltaRayCut 1. keV
#Initialize Physics
/run/initialize
#Verbosity suppression
/gate/systems/ecat/verbose 0
/control/verbose 0
/grdm/verbose 0
/run/verbose 0
/event/verbose 0
/tracking/verbose 0
/gate/application/verbose 0
/gate/generator/verbose 0
/gate/stacking/verbose 0
/gate/event/verbose 0
/gate/source/verbose 0
#Single intrinsic source; confined to crystal
/gate/source/addSource intrinsic
/gate/source/intrinsic/gps/particle ion
/gate/source/intrinsic/gps/ion 71 176 0 0
/gate/source/intrinsic/gps/monoenergy 0. keV #Super important!
/gate/source/intrinsic/gps/angtype iso
/gate/source/intrinsic/gps/type Volume
/gate/source/intrinsic/gps/shape Cylinder
/gate/source/intrinsic/gps/radius 10 cm
/gate/source/intrinsic/gps/halfz 70.1 mm
/gate/source/intrinsic/setActivity 161540.48 Bq
/gate/source/intrinsic/gps/confine crystal_P
#Output configuration
#Sinogram output
/gate/output/sinogram/RadialBins 128
/gate/output/sinogram/RawOutputEnable
```

```
/gate/output/sinogram/TruesOnly true
/gate/output/sinogram/setFileName tube_trues300a
/gate/output/ecat7/disable
/gate/output/ascii/disable
#ROOT output
/gate/output/root/disable
/gate/output/root/setFileName lld200root
/gate/output/root/setRootCoincidencesFlag 1
/gate/output/root/setRootHitFlag 0
/gate/output/root/setRootSinglesFlag 0
#Start simulation
/gate/application/setTimeSlice 1800 s
/gate/application/setTimeStart 0. s
/gate/application/setTimeStop 1800. s
/gate/application/startDAQ
```