MCNPX–PoliMi Variance Reduction Techniques for Simulating Neutron Scintillation Detector Response

by

Shikha Prasad

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Nuclear Engineering and Radiological Sciences)
in The University of Michigan
2012

Doctoral Committee:

Associate Professor Sara A. Pozzi, Co–Chair
Professor Edward W. Larsen, Co–Chair
Professor William R. Martin
Assistant Professor David D. Wentzloff
Assistant Research Scientist Shaun D. Clarke

© Shikha Prasad

2012

To my family: my mother Sangeeta Prasad, my father Vijay K. Prasad,

and my brother Ishan Prasad.

Acknowledgement

I would like to thank my mother, Sangeeta Prasad and my father, Capt. Vijay K. Prasad, and my brother Ishan Prasad for their confidence in my abilities and their continued support to help me pursue my ambitions. Without the support of my parents and their sacrifices, I would not have been able to pursue higher education.

It goes without saying that this thesis work would not have been possible if it were not for the innovative ideas of my advisers Professor Sara A. Pozzi and Edward W. Larsen. Professor Pozzi has always welcomed my ideas, and carefully guided me to develop them. She has also provided me with many unique opportunities to showcase my work. Professor Larsen has spent many hours at a stretch with me to derive solutions from first principles, to teach me a way to think about a problem clearly and logically. He has taught me the importance of finding simple solutions to complex problems. I am also thankful to Dr. Shaun Clarke, without whose suggestions I could not have made timely progress. Shaun has helped me look at problems from angles that help diagnose the problems efficiently. He has also provided me valuable feedback on my writing and presentation skills.

I would also like to thank Professor William R. Martin and Peggy Jo. Gramer without whose efforts I would not have been able to come back to complete my Ph.D studies. Professor Martin has also provided my valuable feedback on my research. I must also thank Professor David D. Wentzloff who has inspired me to think about my thesis

from an applications point of view. I would also like to thank Professor John C. Lee, Professor Ronald Gilgenbach, and Professor James P. Holloway for their continued guidance and support during the course of my studies.

I thank my friends at the University of Michigan (many of them have graduated): Gaurav Pandey, Troy Becker, Kaushik Banerjee, Greg Davidson, Shashi Bhushan Singh, Gokhan Yesilyurt, Jesse Cheatham, Chris Wahl, Natallia Pinchuk, Brian Wagner, Seth Johnson, Eric Miller, Eva Sunny, Sonal Joshi, Jennifer Dolan, Willy Kaye, Nick Touran, Rona Oran, Darskhan Karwat, Jinan Yang, Crystal Thrall, Brian Kitchen, Andreas Enqvist, Zhu Kaya, and Brock Palen for standing by me whenever and wherever I have needed them. Over the course of the many years I been at the University of Michigan, I have interacted with several other friends at numerous exciting and intellectually stimulating occasions. I am also grateful to them for making the journey of my higher education so special and invigorating.

Last but not the least, I take this opportunity to thank all my teachers, professors and mentors who have helped me throughout my studies and my career. Thank You!

Table of Contents

List of Figures

Chapter 1

Chapter 2

Chapter 5

Chapter 6

Chapter 7

List of Tables

Chapter 1

Chapter 2

Chapter 3

Chapter 5

Chapter 6

Chapter 7

List of Appendices

List of Abbreviations

DNNG	Detection for Nuclear Nonproliferation Group, University of Michigan
eVee	electron–volt electron equivalent
EJ–309	Organic liquid scintillation detector by Eljen Techolgy
FOM	figure–of–merit
IPOL	MCNPX–PoliMi input card
MRI	magnetic resonance imaging
PGT	pulse generation time
PHD	pulse height distribution
PMT	photo–multiplier tube
PSD	pulse shape discrimination
RMM	response matrix method
SB	source bias
SNM	special nuclear material
TOF	time of flight

Chapter 1
Neutron Detection for Nuclear Nonproliferation and Safeguards Problems

1.1 Introduction

Detection techniques are being developed widely and aggressively to meet nonproliferation–safeguard scenarios, especially as developing countries acquire nuclear power production capabilities. These techniques are being implemented to (i) identify and interdict special nuclear material (SNM) crossing borders, (ii) characterize SNM by determining its mass and composition, and (iii) locate the SNM in challenging scenarios, such as, in the presence of thick shields or large source–detector distances. A distinct characteristic of SNM is that its nuclei undergo fission. Therefore, detection of fission is a logical way of detecting SNM.

1.2 Detection of Fission for Nonproliferation

Uranium–235, primarily from the front end of the nuclear fuel cycle (such as in uranium enrichment facilities), and plutonium–239, primarily from the back end of the fuel cycle (such as in spent nuclear fuel) are both isotopes of major concern. These elements can be used to make thermo–nuclear devices and are therefore the focus of nonproliferation detection.

Nuclear fission is a major source of neutrons and gamma–rays from both uranium and plutonium. In the fission process, a given nuclide splits into two fission fragments, neutrons, gamma–rays and other particles. Fission can occur spontaneously for an unstable nuclide, or it can be induced with the capture of a neutron (or in some cases with the capture of a photon). In nuclear fission, multiple neutrons and gamma–rays are released whose directions of flight are correlated [1]. This characteristic of fission is the foundation of detection analysis techniques, such as multiplicity measurements and cross–correlation measurements; these will be discussed in the subsequent chapters of this thesis.

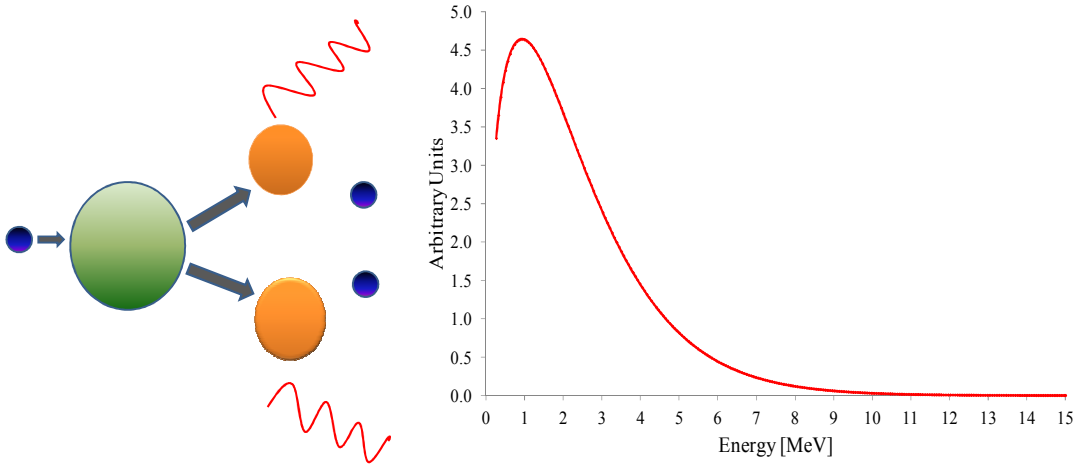


Figure 1– 1 A Fission event shown on left, and a typical fission spectrum approximated by Watt spectrum on right.

Special nuclear material (SNM), such as uranium and plutonium, undergo spontaneous fission. Each SNM isotope has a unique rate of fission, an average number of neutrons emitted per fission, an average number of gamma–rays emitted per fission, and other properties, some of which are shown in Table 1–1 [1].

Table 1– 1: Neutron emission properties for prominent SNM material from Ref. 1.

Isotope	Spontaneous fission rate (neutrons/s.kg)	Average Multiplicity for Spontaneous Fission	Average Multiplicity for Induced Fission
^{235}U	3.0×10^{-1}	1.7	2.4
^{238}U	1.4×10^{1}	2.0	2.3
^{240}Pu	9.2×10^{5}	2.2	2.8
^{242}Pu	1.7×10^{6}	2.1	2.8

1.2.1 Neutron Detection Applications

Neutron detectors can be used to detect the presence of neutrons from fission, to locate the neutron source by finding the distribution of source neutrons in space, and to help characterize neutron sources (composition, isotopic ratios, and mass). Fortunately, the first task is simplified by the diminutive background neutron population [1]. In fact, for the measurements done in the Detection for Nuclear Nonproliferation Group (DNNG) lab at the University of Michigan, the measured neutron background population was small enough to be ignored. Detection of neutrons can be complicated by introducing neutron shields. Methods such as active interrogation systems that can induce nuclear reactions in SNM using an external source are being studied for heavily shielded scenarios. The remaining tasks (to locate and identify the material) are also challenging, and have attracted considerable attention from the scientific community.

The ability to locate SNM is complicated by large stand–off distances or by shielding. For large stand–off distances (typically greater than 10 m), fast neutrons which have greater mean–free–paths in air, O(10 m), than thermal neutrons are more promising [1]. For measurements done in the safeguards regime (counting pin–diversion in spent

fuel assemblies), or in the context of the verification regime (counting warheads), the source–detector distance can be fixed, but finding relevant data from the superfluity of data surrounding the object of interest is a critical but daunting task.

When the location of a source has been determined, the next task is to characterize and indentify the isotopic compositions, mass, and geometry of the source in question. It is here that neutron spectral and multiplicity information measured with cross–correlation techniques can provide potent evidence. These techniques can be challenging, due to their inherently small efficiencies (thus requiring long measurement times) and the presence of competing interactions: induced fission, spontaneous fission, and alpha–neutron reactions [1].

1.2.2 Neutron Detection Principles

Neutron detectors utilize three main nuclear interactions: elastic scattering, inelastic scattering, and neutron capture. In the range of fission energies, elastic scattering with detector material is generally the principle energy loss mechanism for neutrons [1]. Therefore, neutron detector designs are being developed that utilize materials that have large scattering cross–sections (such as scintillation detectors). But large scattering cross–sections do not suffice, detector materials should be such that energy deposition from each scatter is also maximized to achieve greater detector efficiency. The energy deposition in a scatter event is maximized when the neutron is left with the least amount of energy possible. In elastic scattering, the least amount of energy possible, E_{min}, is simply a fraction of the initial energy E_0, This fraction, α, depends on the atomic mass, A of the nucleus with which the neutron scatters as [2]:

$$\alpha = \frac{(A-1)^2}{(A+1)^2} \quad . \tag{1}$$

Thus, it is clear that smaller value of A (atomic mass of the target) will lead to greater energy deposition for detectors based on scatter mechanism. For instance, in the case of a neutron elastically scattering with hydrogen, the neutron can be left with no energy as its α factor is zero. The average energy loss in the case of hydrogen scatters is half of the incident neutron energy [2].

Large neutron capture cross–sections of isotopes in the thermal neutron energy range can be advantageous in neutron detection; however, neutron capture cannot be used with fission neutrons, which tend to have significantly higher energies. Nevertheless, using effective moderators, fast neutrons from fission can be slowed down to energy regions in which they can be captured by target material more easily. In order to slow down fission neutrons before they reach the detector (capture) region, moderator material is chosen considering Eq. (1), such that the energy loss in each scatter with the moderator is maximized (materials with small α are chosen) and neutrons eventually slow down to thermal energy regions.

Lastly, it is important to design durable detectors that can be deployed and used in fields easily. Also, a detector that can be made into large sizes is desirable because for a given measurement, a larger detector will subtend a larger solid angle, allow more particles to enter the detector, and be detected.

1.2.3 Helium–3 Detection and Challenges

Proportional counters based on helium–3 are neutron capture detectors that have manifold advantages and have thus become the "gold standard" for neutron detection [3]. Helium–3 detectors have large thermal neutron capture cross sections while being relatively insensitive to gamma–rays. They have intrinsic efficiencies close to 77% (efficiency after neutrons have been moderated). Proportional counters based on He–3 are simple, mechanically stable, operable in rugged conditions, and last for many years [3]. Due to these reasons, He–3 detectors are not only used for homeland security and international safeguards applications, but also at nuclear plants, for well–logging in the oil and gas industry, for medical applications (MRI lung imaging), and for basic research in nuclear and condensed matter physics [3]. The increased demand and acute supply shortage of He–3 has necessitated the need to find neutron detection alternatives. Kouzes et. al. have estimated that the He–3 demand is approximately 65,000 1itres/y, while the supply is only 15,000 1itres/y [3].

1.2.4 Thermal neutron detection

Thermal neutron detectors, including He–3 detectors, take advantage of the high Q–value (amount of energy released in a nuclear reaction) of neutron capture reactions. The charged particles emitted from a capture reaction have greater energy than recoil protons from elastic scattering (such as in scintillation detectors), and therefore are highly capable of discriminating gamma–rays from neutrons; this is an important criterion in SNM detection [1]. Helium–3, lithium–6, boron–10, and gadolinium–157 are isotopes with large thermal cross–sections for neutron capture. With the shortage of He–3

detectors, these other isotopes have emerged as viable candidates for thermal neutron detection. Presently, the main thermal neutron detector types are: gaseous detectors, water–based Cherenkov detectors, conversion layer detectors, homogeneous compound semiconductors, microchannel plates and scintillators [1]. While these detector types have their advantages and challenges, they require fission neutrons to become moderated before they interact with the detector material, which is an inefficient process. The advancement of fast neutron detection technology eliminates the need for this step.

1.2.5 Fast neutron detection

In this thesis, fast neutrons are defined as neutrons greater than 100 keV. By directly measuring the fast neutrons from fission, one can assess their angular distribution (directionality), unlike the moderated neutrons which have scattered many times and reached the detector at various different angles. Since fast neutrons have not spent times on the order of micro–seconds thermalising in a moderator coupled to a detector, they can be measured on the order of nanoseconds after their emission. Thus, techniques taking advantage of time–dependent cross–correlation measurements of fission chains can be used with fast neutron detectors.

To maximize neutron scatter with the detector material and energy deposition, the typical choice of detector material consists of materials with a high concentration of hydrogen. Hydrogen, due to having a similar mass as a neutron, is the best choice for momentum and energy transfer in a scatter collision [2]. Therefore, an exclusive choice for fast neutron detection is hydrogen.

Typically, fast neutron detection is based on recording the response of the material when the nuclei of the material recoil after scattering with neutrons. There are two main

types of fast neutron detection: scintillation detectors and threshold detectors. Scintillation detectors convert the energy of the recoil nuclei into light that is generally converted into charge using a photo–multiplier tube (PMT) [4]. Organic liquid or solid scintillation detectors, and liquid argon scintillation detectors are the major candidates of these types of detectors. Threshold detectors only measure the response of neutrons above a specific energy. With these, one can exclusively detect fission neutrons by using an appropriate detector material with interactions cross–sections that occur at fission neutron energies.

Of the detector types discussed above, scintillation detectors have emerged as a viable candidate as they [1]:

1. Have good intrinsic efficiency for both neutrons and gamma–rays.
2. Have response times on the order of nanoseconds, which is the same time–scale as the length of the fission chains.
3. Can provide detailed energy information.
4. Allow good discrimination between neutrons and gamma–rays.
5. Are practical as they are portable, relatively inexpensive, and can be made into large sizes.

1.3 Simulation of Scintillation Detector Response

Scintillation detector response has been directly measured and stored in the form of matrices [5–8]. However, the scintillation detector response mechanism is complicated and cannot be easily simulated using analytical methods. Scintillation detector response requires accurate geometry information, detailed particle–nucleus interactions, and event–by–event tracking. Therefore, Monte Carlo codes have been used to simulate

scintillation detector response [9–14]. In the two methods that are discussed below variance reduction techniques are used to speedup up Monte Carlo speedup. These two methods are expected to:

1. Provide computational speedup (savings in computational time)
2. Produce accurate results that can be validated by measurements
3. Be general and applicable to different types of detector response, including time–dependent detector response.

Variance reduction techniques are used to speedup the convergence of mean answers estimated by statistical methods, such as Monte Carlo. In an analog Monte Carlo simulation, the problem is simulated as close as possible to the physical reality. In a nonanalog Monte Carlo simulation, the physics is altered to focus the computation time only on those aspects of the phase–space that contribute to the user–desired means. For instance, particles that do not contribute to a detector response need not be followed, if the user is only interested in estimating the detector response for a given problem. To use variance reduction techniques, there is a need to develop appropriate tally mechanisms such that the final results are accurate. However, before using variance reduction techniques it important to understand the all the physical aspects of the problem such that phase–space that contributes to the mean answer is not ignored. There are several automated methods that a user may find useful when using variance reduction techniques; these are discussed in Ch. 4.

1.3.1 Response Matrix Methods

In the 1950s and 1960s, extensive work was done on scintillation detector response function formulation [6, 7]. Authors have also represented scintillation detector response

functions in form of response matrices [8, 9]. In neutron detector problems, response matrix decomposition has been used for unfolding neutron spectra [10]. Response matrices have also been used to measure the detector response of ^{3}He and ^{6}LiI detectors located at the center of polyethylene Bonner spheres [11, 12]. In a recent paper, the EJ–309 scintillation detector response was studied for very high energy cosmic neutron radiation using MCNPX [13]. However, none of these previous studies was applied to give a detailed non–linear scintillation detector response.

MCNPX–PoliMi and its associated post–processor allow simulation of detailed non–linear scintillation detector response specifically for nonproliferation and safeguards applications [14]. The MCNPX–PoliMi code is unique because it contains built–in correlated fission sources that have multiplicity–dependent energy distributions, and light fission fragment direction–dependent neutron flight directions [15]. Also, the code provides detailed event–by–event collision information in data files that can be used to model scintillation light response to incident radiation [15]. However, MCNPX–PoliMi has traditionally been run in analog mode. In this thesis, MCNPX–PoliMi is run with traditional MCNP variance reduction techniques, and appropriate post–processing models are developed to speed up simulations for neutron scintillation detector response.

In the first approach developed in the thesis, MCNPX–PoliMi is used in analog mode to pre–compute a response matrix for a neutron detector. This matrix is combined with incident neutron current (on the face of the detector) to generate neutron pulse height distributions. In this method, the response matrix method acts like a variance reduction tool in itself. But unlike standard variance reduction techniques, the response matrix has a truncation error associated with the finite energy binning. The matrix

elements also have a small statistical error due to the use of a finite number of Monte Carlo particles to simulate them. The incident neutron current can be estimated using existing variance reduction techniques. In fact, it is possible to estimate the incident neutron current using a deterministic method, and combine that with the MCNPX–PoliMi–generated response matrix.

1.3.2 Direct Monte Carlo

Pulse height tallies provide distributions of detector response (as a function of energy deposited, light produced, voltage etc). These tallies are non–Boltzman tallies because they cannot be obtained from the standard Boltzman transport equation [16, 17]. In the past, authors have developed methods to calculate pulse height distribution tallies, but they have not included the complete scintillation detector response mechanisms [16, 17]. For a long time, the neutron pulse height tally in MCNP was not recommended in the user's manual [18]. Recently, MCNPX capabilities were improved to simulate pulse height distributions; however, the code does not model scintillation detector mechanisms for accurate neutron scintillation detector pulse height response [19].

In the second approach discussed in this thesis, MCNPX–PoliMi is run directly with existing variance reduction techniques to calculate neutron pulse height distributions, time–of–flight curves, and cross–correlations. To run MCNPX–PoliMi with variance reduction, the nonanalog and the analog parts of the Monte Carlo problem are separated. Since the detailed scintillation physics occurs inside the detector, the weight of the neutron is kept constant inside the detector by keeping the neutron transport analog. Nonanalog techniques are only used outside the detector: in the phase–space with thick shields and through large source–detector distances.

1.4 Thesis Outline

The content of this thesis is outlined below with short descriptions. Overall, the thesis provides background and theory in Chapters 1–4; it then presents new analyses and results in Chapters 5–8. The contents of each chapter are described briefly as follows:

- Chapter 1 provides an introduction and motivation for this thesis work. It also provides background on nuclear fission, neutron detection principles, neutron detection applications, present challenges, and metrics for new detector types.
- Chapter 2 provides the theory on the scintillation detector mechanism. Some sensitivity analysis and results are shown in this chapter because they are useful when validating simulations with measurements.
- Chapter 3 provides a background and discussion of MCNPX–PoliMi and its capabilities. This is helpful in understanding why the implementation of nonanalog techniques in MCNPX–PoliMi is nontrivial.
- Chapter 4 provides a background on the concept of weights and importances, development of nonanalog Monte Carlo techniques, and other variance–reducing techniques are used in this thesis.
- Chapter 5 discusses the response matrix method (RMM), which is used to calculate pulse height distributions. It also provides discussion on a radial leakage correction factor, which is necessary to generalize the method for cases in which the incident particle direction is not normal to the front face of the detector. Finally, nonanalog results are compared with analog results,

the computation speedup is discussed, and simulation results are validated with laboratory measurement results.

- Chapter 6 introduces a method that can be used directly with MCNPX–PoliMi to speed up the Monte Carlo simulations. The method separates the nonanalog components of the problems (outside the detector), from the analog component (inside the detector). Comparisons between nonanalog and analog results are provided for pulse height distributions and time–of–flight tallies.
- Chapter 7 furthers the method introduced in Ch. 6 to compute tallies for correlated sources, in which multiple neutrons are emitted from a single source event. Cross–correlation nonanalog tallies are compared with analog tallies for various geometries. Finally, cross–correlation simulations are validated with measurement results, and the speedup is calculated.
- Chapter 8 provides a summary of the thesis work and identifies ways in which this work can be progressed in the future. It also discusses provisions that can be made in MCNPX–PoliMi for its users to allow an easier and more accurate use of nonanalog techniques.

1.5 References

[1] R. C. Runkle, A. Bernstein and P. E. Vanier, "Securing special nuclear material: Recent advances in neutron detection and their role in nonproliferation," *Journal of Applied Physics,* **108**, 111101–1(2010).

[2] J. J. Duderstadt and L. J. Hamilton, Nuclear Reactor Analysis, John Wiley & Sons,

Inc. (1976).

[3] R. T. Kouzes, J. H. Ely, L. E. Erikson, W. J. Kernan, A. T. Lintereur, E. R. Siciliano, D. L. Stephens, D. C. Stromswold and R. M. Van Ginhoven, "Neutron detectionalternatives to ^{3}He for national security applications," *Nuclear Instruments and Methods in Physics Research A,* **623**, 1035 (2010).

[4] G. F. Knoll, *Radiation Detection and Measurements*, Third ed., Hoboken: John Wiley & Sons Inc. (2000).

[5] P. R. Bell, "The Use of Anthracene as a Scintillation Counter," *Physical Review,* **73:11**, 1405 (1948)

[6] C. J. Taylor, W. K. Jentschke, M. E. Remley, F. S. Eby and P. G. Kruger, "Response of Some Scintillation Crystals to Charged Particles," *Physical Review,* **84**, No. 5 1034 (1951)

[7] J. B. Birks, *The Theory and Practice of Scintillation Counting*, A Pergamon Press Book; The Macmillan Company, New York (1964).

[8] V. V. Verbinski, W. R. Burrus, T.A. Love, W. Zobel and N. W. Hill, "Calibration of an Organic Scintillator for Neutron Spectrometry," *Nuclear Instruments and Methods*, **65**, 8 (1968)

[9] H. Klein and S. Neumann, "Neutron and Photon Spectrometry with Liquid Scintillation Detectors in Mixed Fields," *Nuclear Instruments and Methods to Physics Research A*, **476**, 132 (2002).

[10] J. Pulpan and M. Kralik, "The Unfolding of Neutron Spectra Based on Single Value Decomposition of the Response Matrix," *Nuclear Instruments and Methods to Physics Research A*, **325**, 314 (1993).

[11] H. R. Vega–Carrillo, E. Manzanares–Acuña, V. Martín, Hernánde–Dávila, and G.A. Mercado Sánchez, "Response Matrix of a Multisphere Neutron Spectrometer with an ^{3}He Proportional Counter," *Revisita Mexicana de Fisica S*, **51**, 47 (2005).

[12] H. R. Vega–Carrillo, I. Donaire, E. Gallego, E. Manzanares–Acuña, A. Lorente, M. P. Iñiguez, A. Martin–Martin , and J. L.Gutierrez–Villanueva , "Calculation of Response Matrix of a BSS with ^{6}LiI Scintillator," *Revisita Mexicana de Fisica S*, **54**, 57 (2008).

[13] M. Takada, K. Yajma, H. Yasuda, T. Nakamura, M. Baba, T. Honma, A. Endo and Y. Tanimura, "Response Functions of Phoswich–Type Neutron Detector for High–Energy Cosmic Ray Neutron Measurement," *Journal of Nuclear Science and Technology*, **47**:**10**, 917 (2010).

[14] S. A. Pozzi, E. Padovani, and M. Marseguerra, "MCNP–PoliMi: A Monte Carlo Code for Correlation Measurements," *Nuclear Instruments and Methods to Physics Research A*, **513**, 550 (2003).

[15] S. A. Pozzi, S. D. Clarke, W. Walsh, E. Miller, J. Dolan, M. Flaska, B. Wieger1, A. Enqvist, E. Padovani, J. K. Mattingly, D. Chichester, and P. Peerani, "MCNPX–PoliMi for Nuclear Nonproliferation Applications," accepted by *Nuclear Instruments and Methods to Physics Research A* (2012).

[15] R. T. Kouzes, J. H. Ely, L. E. Erikson, W. J. Kernan, A. T. Lintereur, E. R. Siciliano, D. L. Stephens, D. C. Stromswold, and R. M. Van Ginhoven, "Neutron detectionalternatives to ^{3}He for national security applications," *Nuclear Instruments and Methods in Physics Research A,* **623**, 1035 (2010).

[16] T. E. Booth, "A Monte Carlo Variance Reduction Approach for Non–Boltzman

Tallies," *Nuclear Science and Engineering*, **116**, 113 (1993).

[17] D. Légrády, J. E. Hoogenboom, "Feasibility of an Adjoint Monte Carlo Pulse Height Spectrum Calculation*," Proceedings of Nuclear Mathematical and Computational Sciences,* Gatlingburg, Tennessee, 6–11 April, 2003 (2003).

[18] "MCNP—A General N–Particle Transport Code, Version 5, Volume I: Overview and Theory," X–5 MONTE CARLO TEAM, LA–UR–03–1987, Los Alamos National Laboratory (2003).

[19] D. Pelowitz (ed.). "MCNPX User's Manual, version 2.6.0," LA–CP–07–1473, Los Alamos National Laboratory (2008).

Chapter 2
Scintillation Detectors

2.1 Background

The scintillation process is one of the oldest and most useful physical processes used for the detection and spectroscopy of various forms of radiation [1]. The scintillation response to ionizing radiation was first measured by H. Kallman in 1947 [2]. Scintillation light in detectors is produced when incident radiation excites the molecules or the electrons of the detection material. Upon their de–excitation, light (belonging generally in the visible range) is produced. This light can be converted into charge and collected, using a photomultiplier tube coupled to the scintillation detector [1].

2.2 EJ–309 Scintillation Detector

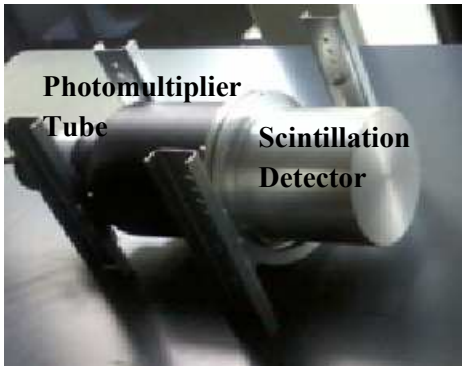


Figure 2– 1: An EJ–309 scintillation detector with its photomultiplier tube.

The EJ–309 liquid scintillation detector shown in Fig. 2–1 is comprised of hydrogen and carbon nuclei with a ratio of 1.25 to 1. Light is produced as gamma–rays Compton scatter on electrons, or as neutrons elastically scatter on hydrogen and carbon nuclei. The energy deposited by the gamma–ray or the neutron is then converted into light. The light produced in a given time window called the pulse–generation–time (PGT) is summed to form a pulse. For this study, the PGT is 10 ns long. The light created in the scintillation detector is converted into charge in the photomultiplier tube (PMT). Bigger light pulses result in greater charge production and collection in the PMT. A typical pulse is shown in Fig.2–2.

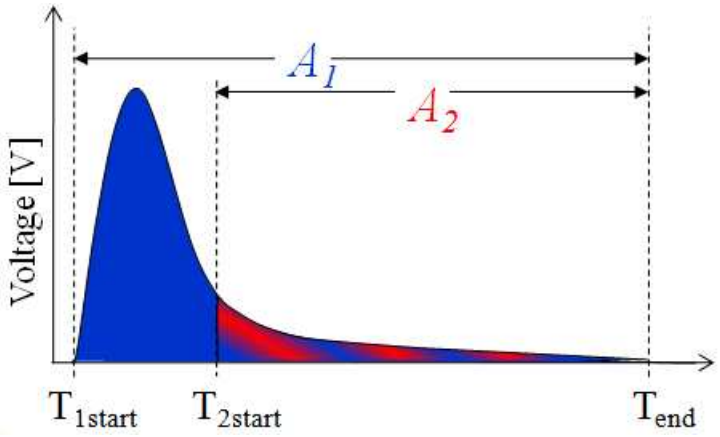


Figure 2– 2: An EJ–309 scintillation pulse and its features.

In Fig 2–2, the pulse maximum or the height of the pulse is called the pulse height. A distribution of the pulse heights, commonly known as a pulse height distribution (PHD), is a common scintillation detector response measureable. The acquisition time window shown above is generally 400 ns long. Data is acquired every 4 ns, thus, there are a total of 100 points in a typical acquisition time window. Typically the maximum of the pulse occurs between 50 ns and 100 ns in the acquisition time window. The start time of the

pulse is marked by T_{1start}, the start time of tail area of the pulse is marked by T_{2start} (this is 20 ns after the maximum of the pulse occurs), and the end time of the pulse is marked by T_{end} (which is 220 ns after T_{2start}). As marked in Fig.2–2, a pulse has two distinct regions. The first region, A_1, is the total integral of the pulse or the area under the entire pulse. The second region, A_2, is the tail integral of the area under the tail of the pulse. A ratio of A_2 to A_1, or tail–to total is calculated for each pulse. The ratio of tail–to–total is larger for heavier charged particles and can be used to distinguish neutron pulses from gamma–ray pulses, as will be shown in Ch 5 and Ch 7 [1].

For the case of gamma rays, the energy–to–light conversion is a one–to–one conversion process. However, in the case of neutrons this process is more complicated. For neutrons, the light produced is not only a function of energy deposited but is also dependent on the nucleus on which the collision has occurred. In the case of carbon, it has been empirically found that the energy deposited by a neutron, T, is transformed to light, L, with a rate of 2% of that for gamma–rays [3]. In the case of hydrogen, it is found that energy to light conversion is a nonlinear process and can be given by a quadratic fit [4]:

$$L = aT^2 + bT + c \quad . \tag{1}$$

Here, a and b are empirically determined constants for the quadratic and the linear terms respectively, whereas, c is the empirically determined constant term. Recently, an exponential fit for the neutron–hydrogen scatter light conversion process has also been found as [5]:

$$L = aT + b(1 - e^{-cT}) \, . \tag{2}$$

In this chapter, sensitivity studies are performed using the quadratic fit.

It has also been found that the order of collisions in the scintillation detector matters. In Fig. 2–3 from Ref. 4, we can see that the PHDs are different for different permutations of collisions. To take the simplest case where the neutron pulses are created only due to two scatters, the plot in the middle, Fig. 2–3 (b) shows: the PHD resulting from pulses with the first collision on hydrogen and the second collision on carbon (HC), is different from the PHD resulting from pulses with the first collision on carbon and the second collision on hydrogen (CH).

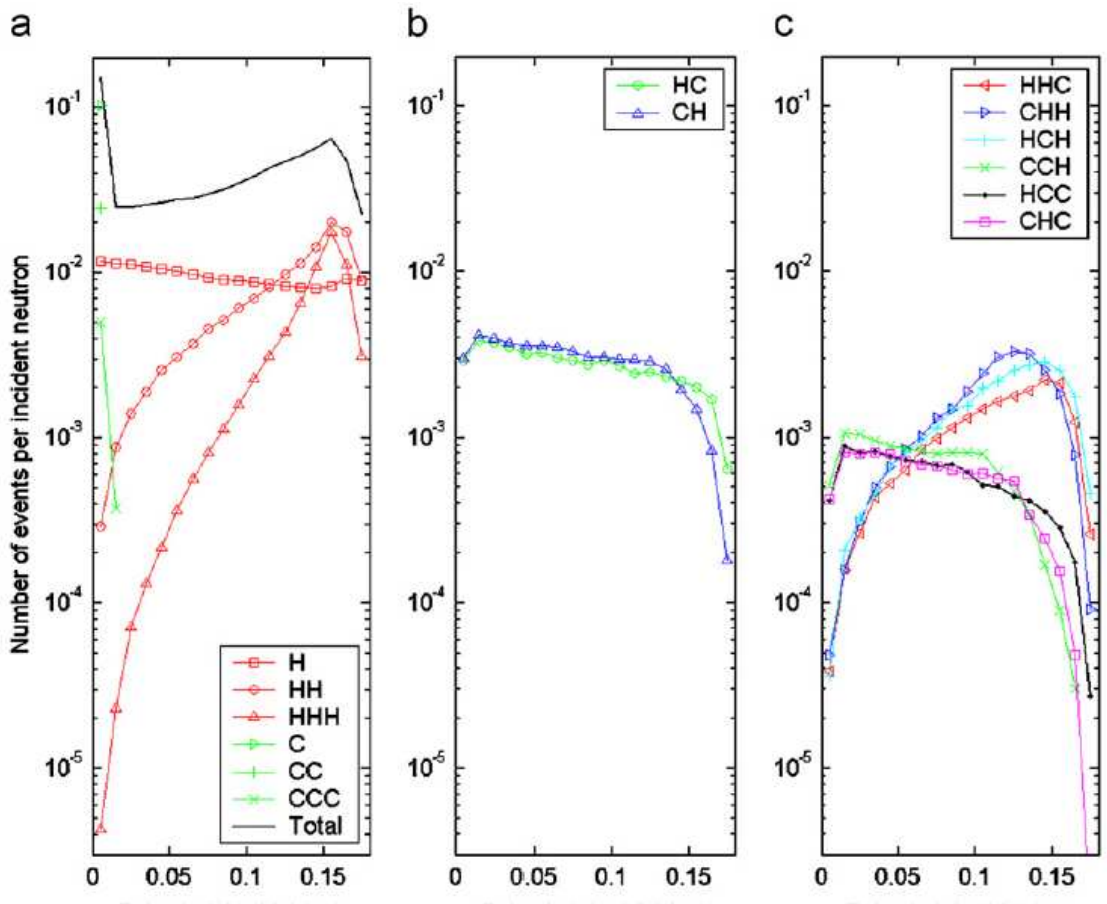


Figure 2– 3: Pulse height distributions resulting from different order of collisions from Ref. 4

Time–of–flight (TOF) measurements were used to measure the constants shown in Eq. 1 and Eq. 2. In the measurement, a ^{252}Cf neutron source is placed at 100 cm from the

face of an EJ–309 scintillation detector [6]. The EJ–309 detector is a cylinder with a radius of 6.34 cm and a length of 12.51 cm. The chemical composition of the EJ–309 liquid is 55.5% hydrogen and 44.5% natural carbon by atom. Because the radius of the detector is relatively small compared to the source detector distance, one may assume that all neutrons travel nearly the same distance of 100 cm. Having made this assumption, one can now divide the TOF into several narrow time bins, such that each time bin, t_n, corresponds to a particular energy bin, E_n, of the incident neutron energy spectrum through the kinetic energy formula shown in Eq. (3). The mass of the neutron, m, and the distance travelled by the neutron until it is detected, d (100 cm in this case), are fixed in Eq (3):

$$E_n = \frac{1}{2} m \left(\frac{d}{t_n} \right)^2 . \tag{3}$$

Thus, several pulse height distributions are created for these time bins. With small enough time bins these pulse height distributions correspond to quasi mono–energetic neutrons. Theoretically, the pulse heights should have a maximum value, which corresponds to the maximum energy deposited. In the presence of hydrogen, this will occur when a previously uncollided neutron elastically scatters on a hydrogen nucleus and transfers all of its energy. Thus, ideally, the edge of the pulse height distribution should correspond to the maximum energy deposited (i.e. the energy of the incident neutron). Hence, using the different pulse height distributions for all the time bins, we can create pairs of energy deposited (incident energy) and light produced (edge of pulse heights). This is shown as a plot in Fig. 2–4.

However, with experimental data which includes noise, the edge of the pulse height distribution is not a distinct feature and needs to be carefully determined from the

measured data. For the present fit of energy–to–light conversion, as discussed in the next section, 20% of the maximum value of the pulse height distribution is chosen to be 'the edge'. This value agrees with the edge value chosen in Ref. 7 when considering the differences in experimental setup. In the subsequent sections, a sensitivity analysis of this parameter is shown.

2.3 Measurement Determined Fits

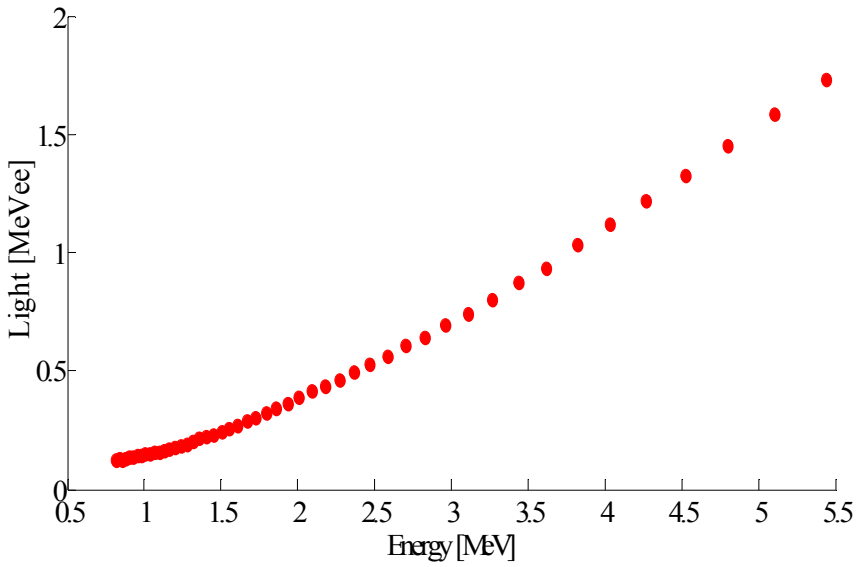


Figure 2– 4: Maximum neutron energy deposition in scatters with hydrogen and the subsequent light production curve for EJ–309 scintillation detectors.

The experimental data shown above was fit to a quadratic polynomial as in Eq. 1 to yield:

$$L = 0.03495T^2 + 0.1424T - 0.0362 \,. \tag{4}$$

The data was also fit to yield an exponential form shown in Eq. 2:

$$L = 0.74787T - 2.4077(1 - e^{-0.29866T}) \,. \tag{5}$$

The sensitivity analysis presented in the next section is carried out utilizing the coefficients given in Eq. 4.

2.4 Sensitivity Analysis

The determination of the edge of a measured pulse height distribution of a nearly monoenergetic case is not simple, since it is distorted by noise and measurement uncertainties. In our analysis, a value of 20% of the maximum is chosen as the edge. We vary this percentage of the edge value to find the impact on simulated pulse height distributions. Furthermore, the total distance travelled by neutrons before they interact and deposit their energies (*d* in Eq. 3) is varied, and the subsequent impact on the pulse heights distributions is analyzed. Next, we investigate the impact of changing the ^{137}Cs calibration to convert the pulse heights in voltage units to pulse heights in light units. The term MeV *electron equivalent* (MeVee) will be used to quantify light. One MeVee is defined as the amount of light produced by one MeV energy deposited by a recoil electron (following Compton scattering) [1].

2.4.1 Varying Edge Values and Interaction Depth

We consider energy to light conversion process by studying pulse heights for a given TOF bin that corresponds to a quasi–monoenergetic energy bin. The edge values are calculated at 20% and 50% of the maximum of the pulse height distribution. Neutron flight paths to the face of the detector and including an interaction depth of 4 cm inside the detector are considered. There are some noticeable trends: making variations does not impact the distribution in the low light output bins as much as for higher light output bins. As the edge values are increased from 20% to 50%, for a given interaction depth,

the neutron pulse height distributions are reduced for higher light outputs. As the interaction depth is increased the pulse heights distribution are also reduced.

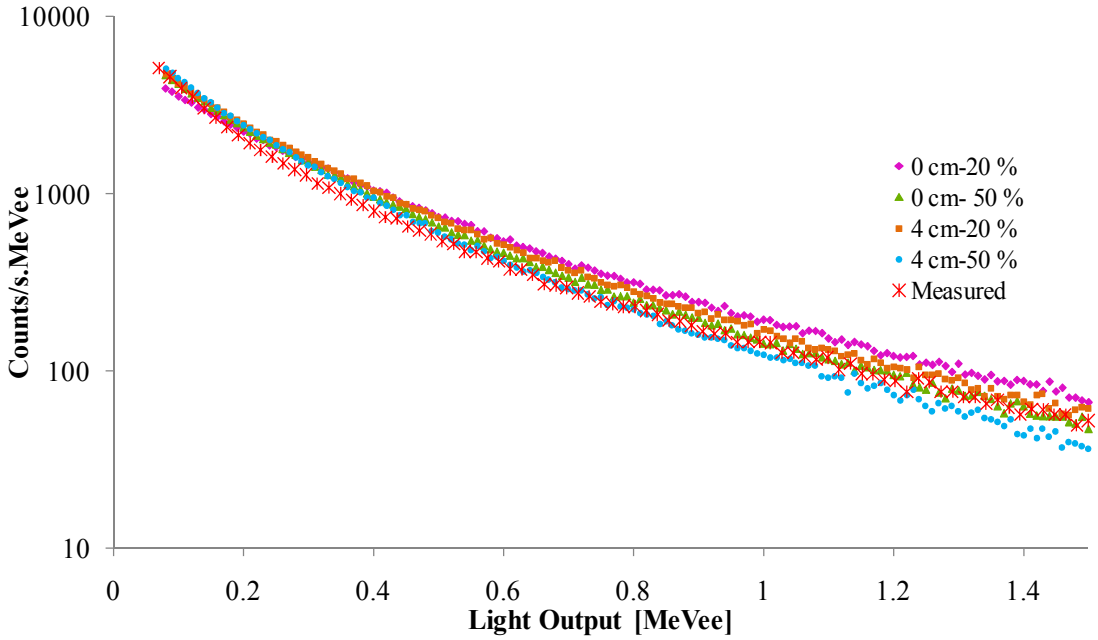


Figure 2– 5: Pulse height distributions for bare ^{252}Cf for varying edge values and interaction depth

The constants *a, b* and *c* as shown in Eq. 1 are provided in Table 2–1 for the cases plotted above. Table 2–1 shows that the intercept, *c*, becomes less negative as we increase the edge value to 50% of the maximum.

Table 2– 1: Coefficients for Varying Depth and Edge Value

Case	*a*	*b*	*c*
0 cm – 20%	0.048448	0.17849	–0.061007
0 cm – 50%	0.042940	0.13635	–0.043883
4 cm – 20%	0.045067	0.14741	–0.043161
4 cm – 50%	0.037450	0.12096	–0.036679

2.4.2 Varying ^{137}Cs Calibration to Determine Light Output

A ^{137}Cs calibration is performed to convert the charge recorded into light produced (volts–MeVee conversion). This is done by estimating the Compton edge of the ^{137}Cs spectrum as a feature that occurs at 75% of the maximum. Other studies have also considered this value, which typically falls within the range of 60% to 90% [8].

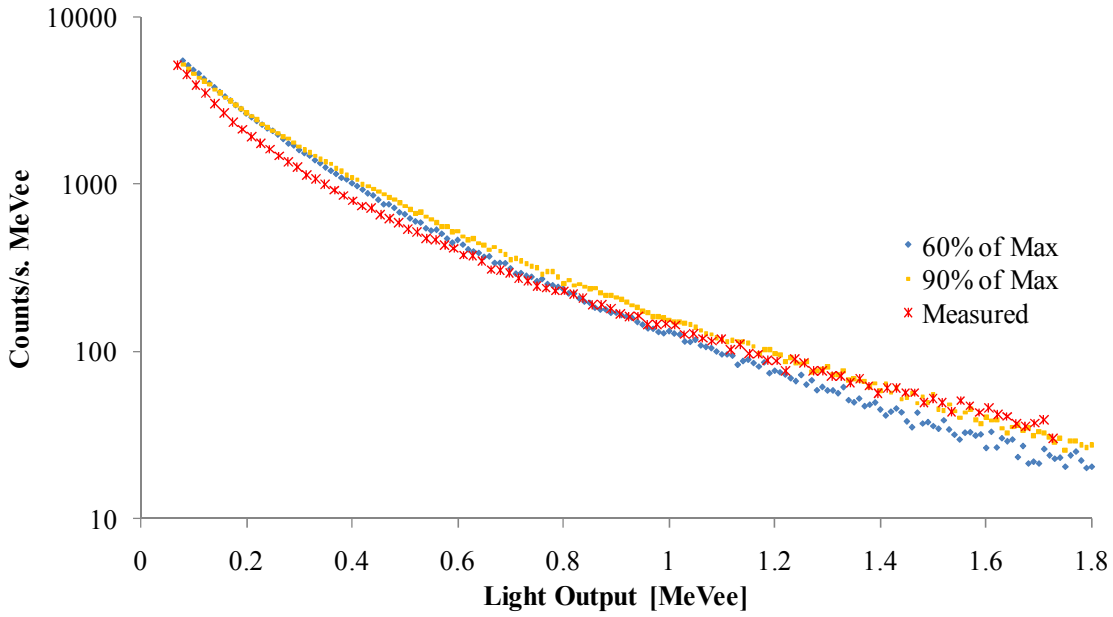


Figure 2– 6: Pulse height distributions for bare ^{252}Cf by varying the Compton edge in ^{137}Cs calibration

If we move the estimated Compton edge to the right or to the higher voltages, the volts–MeVee conversion would yield more light produced at a given energy; similarly, shifting the edge to the left indicates less light produced. This can be understood from the comparisons in Fig. 2–6 and coefficients listed in Table 2–2 for the quadratic Eq. 1. The coefficients in Table 2–2 correspond to the case of 90% of maximum taken for ^{137}Cs calibration. In the 90% case, the volts–MeVee conversion would lead to greater light

production despite the slightly more negative intercept. Therefore, in Fig. 2–6 the simulation of pulse height distribution using 90% of maximum as the Compton edge has higher counts than the simulation using 60% of maximum.

Table 2– 2 Coefficients for varying Cs Edge

Case	*a*	*b*	*c*
60 % max Compton edge	0.0326	0.1335	–0.0337
90 % max Compton edge	0.0368	0.1506	–0.0380

2.5 Validity of Assumptions

We have made some assumptions in the process of evaluating energy deposited and light produced. Consider the energy distributions are simulated using MCNPX–PoliMi data file, as shown in Fig.2–7 and 2–8.

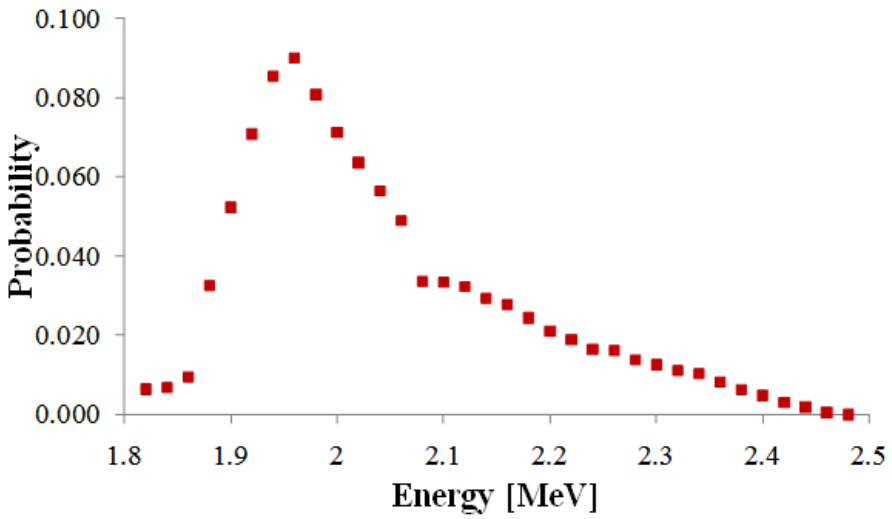


Figure 2– 7: Distribution of contributing energies for the time window corresponding to 1.95 MeV – 2.05 MeV energy bin.

When narrow TOF bins are taken to correspond to certain energies, the TOF bins may also capture some of the neutrons from adjacent energy bins. For instance, a given time–bin corresponding to a certain energy bin can get a contribution from a higher energy neutron that interacts deep in the detector and thus travels a greater distance than the assumed TOF distance. Therefore, in Fig. 2–7, although the expected energy window is between 1.95 MeV and 2.05 MeV, we observed contributions from neutrons between 1.8 MeV and 2.5 MeV. Similarly, in Fig. 2–8, for the expected energy window between 3.95 MeV and 4.05 MeV, the actual distribution lies between 3.5 MeV and 4.7 MeV.

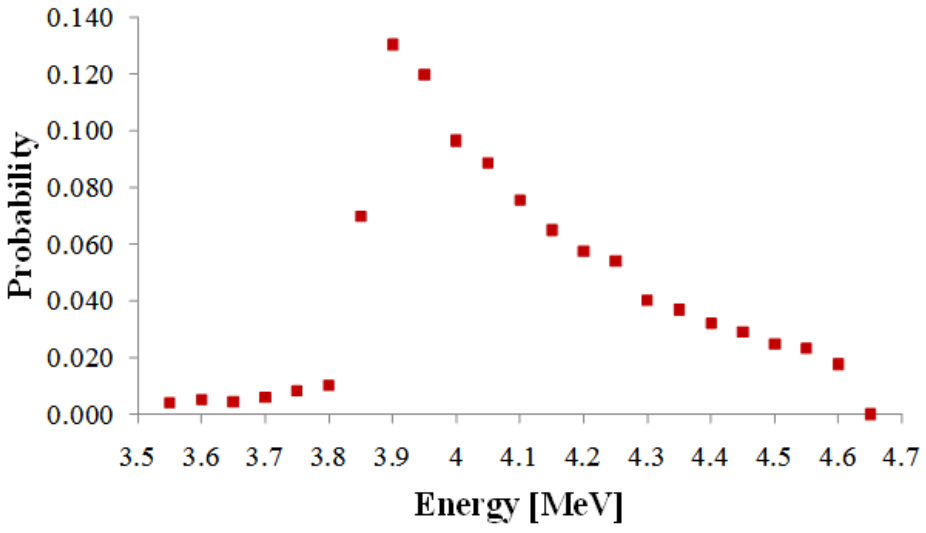


Figure 2– 8: Distribution of contributing energies for the time window corresponding to 3.95 MeV – 4.05 MeV energy bin.

2.6 Conclusion

Scintillation detectors are a reliable and practical means of measuring radiation. The scintillation detector response is nonlinear for neutrons in the case of EJ–309 detectors. It is important to accurately measure and characterize this response with laboratory measurements, to be able to simulate scintillation detector response accurately. This response characterization is impacted by several quantities as discussed in the sensitivity analysis. MCNPX–PoliMi is capable of including this detector response in the form of light output coefficients to accurately simulate scintillation detector response. In the next chapter, MCNPX–PoliMi and its capabilities and features for nonproliferation detection are discussed.

2.7 References

[1] G. F. Knoll, *Radiation Detection and Measurements*, John Wiley & Sons, Inc., Hoboken (2010).

[2] P. R. Bell, "The Use of Anthracene as a Scintillation Counter," *Physical Review*, **73**, No. 11 1405 (1948).

[3] K. Schweda, D. Schmidt, "Improved Response Function Calculations for Scintillation Detectors Using an Extended Version of the MCNP Code" *Nuclear Instruments and Methods in Physics Research A,* **47**6, 155 (2002).

[4] S. A. Pozzi, M. Flaska, A. Enqvist, and I. Pázsit, "Monte Carlo and Analytical Models of Neutron Detection with Organic Scintillators" *Nuclear Instruments and Methods in Physics Research A,* **582**, 629 (2007).

[5] A. Enqvist, C. C. Lawrence, T. N. Massey, S. A. Pozzi, "Neutron Light Output

Functions Measured for EJ309 Liquid Scintillation Detectors," *Transactions of the Institute of Nuclear Materials Management 53rd Annual Meeting*, Orlando, Florida, 15–19 July, 2012 (2012).

[6] S. Prasad, A. Enqvist, S. D. Clarke, S. A. Pozzi, E. W. Larsen, "A Response Function Study for Energy to Light Conversion in Organic Liquid Scintillators," *Transactions of the Institute of Nuclear Materials Management 52nd Annual Meeting*, Palm Desert, California, 17 – 21 July, 2011 (2011).

[7] N. V. Kornilov, I. Fabry, S. Oberstedt, H. J. Hambsch, "Total Characterization of Neutron Detectors with a ^{252}Cf Source and a New Light Output Determination," *Nuclear Instruments and Methods in Physics Research A,* **599**, 226 (2009).

[8] J. Yan, R. Liu, C. Li, L. Jiang, X. X. Lu, T. H. Zhu, "Energy Calibration of a BC501A Liquid Scintillator Using a γ–γ Coincidence Technique," *Chinese Physics C*, **24:7** (2010).

Chapter 3
MCNPX–PoliMi Capabilities and Features

3.1 Background

The MCNPX–PoliMi Monte Carlo code was developed to perform enhanced simulations of the physical behavior of SNM [1, 2]. It has built–in multiplicity distributions for neutrons, photons, and correlated photon production to neutron interactions. Its salient features are: i) built–in cross–correlations for fission events from different isotopes, ii) event–by–event particle tracking that can be printed and post–processed by the user to calculate the tallies of interest, and iii) conservation of energy and momentum. As a result of these features, MCNPX–PoliMi is able to accurately simulate laboratory measurement of SNM [3, 4]. The present version of MCNPX–PoliMi runs with analog tracking to keep the tallying process of the complicated detector response simple. This chapter will elaborate on the above features.

3.2 Source treatment in MCNPX–PoliMi

MCNPX–PoliMi models neutron fission events by utilizing available nuclear data [5]. For a fission event, the multiplicity of the neutrons and photons is dependent on the isotope. In MCNPX–PoliMi, the multiplicities are available for ^{252}Cf, ^{238}U, ^{238}Pu, ^{240}Pu, ^{242}Pu, ^{242}Cm, ^{244}Cm and others [6].

For fission source events in MCNPX–PoliMi, first the isotope dependent multiplicity is sampled, then the energies of the neutrons and the photons are sampled depending on the multiplicity of the source event. As the multiplicity increases, the spectrum for the neutrons becomes softer. This is plausible because, when the same excitation energy gets distributed among more neutrons, each neutron will carry less energy as compared to a case in which fewer neutrons are emitted. It is also known that the prompt neutrons carry some of the momentum of the fission fragments from which they are produced, and their direction of flight is correlated to the fission fragment's direction of flight [5]. Thus, in addition to modeling the multiplicity–dependent energy distribution, MCNPX–PoliMi also models fission fragment correlated flight directions for neutrons. It is shown in Ref. 5 that neutrons are more likely to travel along, or opposite to the direction of the lighter fission fragment. The lighter fission fragment's flight direction is isotropically sampled [6].

Furthermore, MCNPX–PoliMi can be used to output event–by–event details as shown in Table 3–1. The data in Table 3–1 can be treated individually to simulate a detailed detector response, such as nuclide–dependent energy–deposition to light–conversion for all collisions in the detector.

3.3 Explicit Modeling and Energy Conservation

In standard MCNP/MCNPX, secondary photons are released independent of the type of neutron collision [6]. For instance, in the case of an inelastic neutron–carbon collision, the carbon nuclide can be left in any one of its several excitation states. The photon released subsequently will depend on the excitation level of the carbon nuclide.

Table 3– 1: An excerpt (first twelve columns) from the MCNPX–PoliMi data file with collision history

History No.	Particle No.	Interaction Type	Nuclide	Cell	Energy [MeV]	Time [shakes]	X [cm]	Y [cm]	Z [cm]	Weight	Generation No.	Scatter No.
3	1	–99	6000	6	0.287478	1.99	2.19	–4.37	4.48	1.000E+00	0	0
3	1	–99	1001	6	0.740393	2.03	2.03	–3.80	4.63	1.000E+00	0	1
3	1	–99	1001	6	0.146215	2.16	2.67	–2.96	4.66	1.000E+00	0	2

In standard MCNP/MCNPX, the photon produced from the inelastic scatter is determined always by only one of its excited states [6].

In MCNPX–PoliMi, secondary photon production is related to the type of collision and the corresponding energy conservation models. The photon–production subroutine is called after the collision type has been determined [6]. The possible neutron–nucleus interaction types modeled are capture, inelastic, elastic, fission, (n, xn), charged particle production reactions, neutron and charged particle production reactions.

Since the objective of this thesis is to speedup simulations of neutron detector response a detailed discussion of photon production will not be given here. Nonetheless, the interested reader is encouraged to see Ref. 6 for further details.

3.4 Event–by–Event Particle Tracking

In addition to performing detailed modeling and conservation of energy at each type of neutron collision, MCNPX–PoliMi also tracks every particle as a whole (analog mode) and prints its associated collision information in a data file for user specified cells (geometric regions) of interest. An example of the data file information is shown in Table. 3–1. It is evident from the table that MCNPX–PoliMi keeps a record of all collisions, their types, the nuclides involved in collisions, the energy deposited, the time of collisions, the location of collisions, the weight of the particles, the generation of the particles, etc. All this information is provided for *each* collision. In traditional MCNP/MCNPX, an average quantity is reported to the user in the tallies. With MCNPX–PoliMi, a user can post–process the data file using MPPost and obtain individual event–by–event detector response for SNM [7]. In the case of scintillation detectors, as

discussed in Ch. 2, it is important to convert energy deposited into light emitted by the scintillation detector for each neutron scatter. This is not feasible with a mean energy deposition reporting as in the MCNP/MCNPX pulse height (F8) tally [8].

As discussed in Ch. 2, the energy–to–light conversion is a nonlinear process for neutron–hydrogen scatters. Particles scatter to produce light–flashes, and those occurring in a PGT (a time–window which is typically 10 ns long) are summed to produce a pulse. Furthermore, it was also shown in Ch. 2 that light conversion is nuclide–dependent, and the order of collisions to produce a light pulse is important in determining the amplitude or the pulse height (Fig. 2–3). Due to these reasons, it is important for the user to treat each individual collision and its energy deposition in order. Additionally, it is important to track the timing of the collisions because light pulses are the sum of light outputs from energy depositions that occur within the PGT. Only if the user knows the timing of these light outputs can s/he determine if the collision and it associated light output will fall within the PGT for a given pulse. All these features of nuclide dependent and time characterized energy–to–light conversion is feasible with MCNPX–PoliMi.

3.5 Conclusion

In the above discussion, we have seen that MCNPX–PoliMi is capable of simulating correlated fission sources, performing detailed energy–to–light conservations of various interactions, and tracking each particle interaction–by–interaction. However, for simplicity when performing detailed simulation and tracking, MCNPX–PoliMi has avoided the use of nonanalog techniques. In this thesis, ways of implementing nonanalog techniques with MCNPX–PoliMi have been developed. The next chapter provides some

background on nonanalog or variance reduction techniques that can be helpful for speeding up MCNPX–PoliMi simulations.

3.6 References

[1] S. A. Pozzi, E. Padovani, M. Marseguerra, "MCNP–PoliMi: A Monte Carlo Code for Correlation Measurements," *Nuclear Instruments and Methods to Physics Research A*, **513**, 550 (2003)

[2] S. A. Pozzi, E. Padovani, M. Flaska, S. D. Clarke, "MCNP–PoliMi Post–Processing Code," ORNL/TM–2007/33, Version 1.9, Oak Ridge National Laboratory (2007).

[3] S. Prasad, S. D. Clarke, S. A. Pozzi, E. W. Larsen, "A Response–Matrix Method for the Calculation of Neutron Pulse Height Distributions in MCNP," *International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering,* Rio de Janeiro, Brazil, 8–12 May, 2011 (2011).

[4] S. Prasad, S. D. Clarke, S. A. Pozzi, E. W. Larsen, "Simulation of Neutron Pulse Height Distribution with a Response Matrix Method," *Nuclear Science and Engineering,* **172,** 78 (2012).

[5] S. A. Pozzi, S. D. Clarke, W. Walsh, E. Miller, J. Dolan, M. Flaska, B. Wieger1, A. Enqvist, E. Padovani, J. K. Mattingly, D. Chichester, and P. Peerani, "MCNPX–PoliMi for Nuclear Nonproliferation Applications," accepted by *Nuclear Instruments and Methods to Physics Research A* (2012).

[6] Enrico Padovania, Sara A. Pozzib, Shaun D. Clarkeb, and Eric C. Miller, "MCNPX–PoliMi User's Manual," (2012).

[7] E. C. Miller, S. D. Clarke, M. Flaska, S. Prasad, S. A. Pozzi, and Enrico Padovani, "MCNPX–PoliMi Post–Processing Algorithm for Detector Response Simulations," *Journal of Nuclear Materials Management,* **40***,* 34 (2012).

[8] D. Pelowitz (ed.). "MCNPX User's Manual, version 2.6.0," LA–CP–07–1473, Los Alamos National Laboratory (2008).

Chapter 4
Variance Reduction

4.1 Introduction

Monte Carlo simulations stochastically model physical scenarios by repeated random sampling from appropriate distributions to find an estimate of a mean [1]. This calculated mean has a statistical uncertainty; smaller uncertainty provides greater confidence in the result. A simple way to decrease the uncertainty is to increase the sample size; however, this can result in excessively long computational times for complicated configurations. Variance reduction techniques can reduce the statistical uncertainty in the mean for Monte Carlo simulations by increasing the likelihood of scoring events that are infrequent, but meaningfully contribute to the desired result.

4.2 Figure–of–Merit

The figure–of–merit (*FOM*), defined in the following way, is used to measure how efficiently a Monte Carlo simulation is performed [2]:

$$FOM \equiv 1/(R^2 T) \quad . \tag{1}$$

Here, R is the sample relative standard deviation of the mean, and T is the computation time. A high *FOM* value indicates an efficient use of computational resources, whereas a small *FOM* value indicates an inefficient use. The *FOM*, as it is defined in Eq. (1) is a way to perform a cost– (increased computational time) benefit (reduced uncertainty)

analysis by a user. For a given Monte Carlo simulation, if the number of source particles is increased such the statistical uncertainty in the mean is reduced (reduce R^2), then the computer must perform more work for a greater number of source particles, that is the computation time is increased (increase T). Trade–off occurs in a way such that the *FOM* for a given simulation stays approximately constant as the number of source particles is increased or decreased. The Central Limit Theorem gives that R^2 is inversely proportional to the sample size (in this case the number of tallied particles), N [2]. Also, there is a direction relation between T and N. If these definitions of N are applied to Eq. 1, it is clear that the *FOM* remains constant if only the number of Monte Carlo source particles is changed.

There are ways to modify the Monte Carlo simulation using variance reduction techniques such that computation time is focused on those particles that will meaningfully contribute to a desired system response. By using these techniques, it is possible to reduce the statistical uncertainty in the mean without significantly increasing the computation time. Variance reduction techniques, when used appropriately, help increase the *FOM* for a Monte Carlo simulation. If variance reduction techniques are chosen without much thought or understanding, it is possible that the computation time will be focused on particles not contributing to the detector response of interest. This will decrease the *FOM,* or make the simulation slower!

In MCNP, there are four main types of variance reduction techniques: truncation methods, population control methods, modified sampling methods, and partially deterministic methods [2]. Truncation methods simply truncate parts of phase–space not important to the problem. Population–control methods use splitting or Russian roulette to

control the sample size from different regions of phase–space (energy, space, direction). Modified sampling methods are used to sample from modified statistical models with appropriate weights such that the final answer is unbiased. Partially deterministic methods combine deterministic and stochastic methods to reduce variance; these are often referred to as "hybrid" methods.

4.3 Particle Weight and Importance

The weight of a particle is a concept that can be related to the importance of a particle to the desired system response. The concept of weight is used to let simulated particles represent a different number of physical particles, as the simulated particle is transported in the phase–space. Mathematically, weights allow accurate accounting of particles to yield an unbiased system response or tally for stochastic methods. In analog Monte Carlo simulations, the particle weight is always unity. In nonanalog Monte Carlo simulations, the particle weight can change such that the nonanalog particle represents a quantity of particles different from unity (a fraction or a multiple of particles).

The basis for the weight of a particle is its importance to a user–desired system response or tally. Particles that are more likely to contribute to the system response have higher importances than those less likely to contribute. This is because particles that contribute to a system response will increase confidence in the desired statistical mean (by decreasing statistical uncertainty), whereas unimportant particles consume computational time without any impact on the confidence in the given mean [3].

Nonanalog techniques or variance reduction techniques may be applied to change the actual particle distribution based on their importances. The importances help guide the

particles towards regions of phase–space important to the tally. As particles are guided and their population is modified, particle weights must be changed simultaneously such that the modified particle distribution and the modified weight together preserve the physics of the problem and yield an unbiased answer (discussed in Subsection 4.4.2).

Much work has been done to determine the biasing parameters efficiently [4–14]. Early work was done by Dwivedi [4] and Gupta [5] using the zero–variance methods. Adjoint fluxes commonly referred to as "importance functions" have been used to develop several other production–level methods such as MCBEND [6], TRIPOLI [7], AVATAR [8], LIFT [9], Cooper's weight window [10], and FW–CADIS [11–14].

4.4 Variance Reduction Techniques Used

MCNP provides several different variance reduction options [2]. In this thesis, two techniques are used to show that MCNPX–PoliMi can be run using variance reduction techniques. These techniques are source biasing and geometry splitting with Russian roulette.

4.4.1 Geometry Splitting with Russian Roulette

In geometry splitting with Russian roulette, the simulated geometry is divided into several different regions or cells. Each cell is given an importance by the user. It is the responsibility of the user to define importances for the problem, but this process can be helped by the automated methods listed in the previous section. As the particle moves from a region of lower importance, $imp_{L,}$ to region of higher importance, imp_H, it splits into daughters determined by the ratio imp_H/imp_L. Conversely, if the particle moves from a region of higher importance to lower importance it survives with a probability of

imp_L/imp_H [2]. Figure 4–1 illustrates geometry splitting. In Fig. 4–1(a), no variance reduction is used, the importance of the particle remains one as it travels through space, thus the weight of the particle remains one. However, in Fig. 4–1(b), geometry splitting is used to split the particle when the importance of the space changes beyond the splitting plane in the middle. After splitting, the weights of the two daughter are divided by two (in a way described above), such that if both particles scored towards a tally, the sum of their weights (½ + ½) will still yield one.

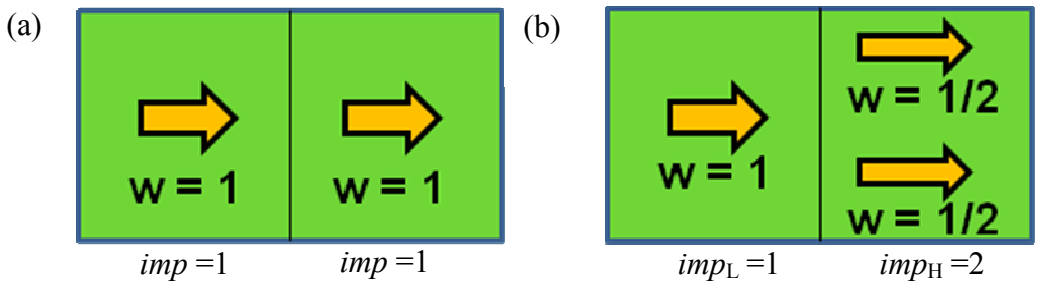


Figure 4– 1: Analog Monte Carlo transport shown on the left. Geometry splitting in space shown on the right.

Geometry splitting with Russian roulette can improve the *FOM* by increasing the population of particles in phase–space, after the particle distribution decreases due to reasons such as thick shielding. In such a case, computational work will be done for an increased number of particles only in certain parts of the phase–space (in Fig. 4–1 (b), beyond the splitting plane). Geometry splitting with Russian roulette (will be referred to as simply geometry splitting from here onward) is used in Ch. 6 and Ch. 7.

4.4.2 Source Biasing

Source biasing works by modifying the original distribution of particles in energies, space, or directions specified by the user, when the particles are born. In this method, the user specifies the desired probability distribution of source neutrons, $\hat{p}$ or the biases. Based on these biases, weights $\hat{w}$ are calculated such that the product of original weight $w_0 = 1$, and original probability p_0 are preserved [2]:

$$w_0 \cdot p_0 = \hat{w} \cdot \hat{p} \ . \tag{2}$$

Source biasing can improve the *FOM* for cases in which the neutron source introduces a large number of particles into the system that have relatively low importance. By biasing the source so that higher–importance source particles are preferentially born, more Monte Carlo particles are likely to score in the detector, and the variance in the detector response is reduced. This technique is used in Ch. 5, 6, and 7 to distribute source particles equally in all energy bins.

4.5 References

[1] L. L. Carter, E. D. Cashwell, "Particle–Transport Simulation with the Monte Carlo Method," ERDA Critical Review Series TID 26607, U.S. Department of Energy, Washington D.C. (1975).

[2] "MCNP—A General N–Particle Transport Code, Version 5, Volume I: Overview and Theory," X–5 MONTE CARLO TEAM, LA–UR–03–1987, Los Alamos National Laboratory (2003).

[3] J. Lewins, *IMPORTANCE The Adjoint Function*, Pergamon Press, Norwich (1965)

[4] S. R. Dwivedi, "A New Importance Biasing Scheme for Deep–Penetration Monte

Carlo," *Annals of Nuclear Energy*, **9:7,** 359 (1982).

[5] H. C. Gupta, "Importance Biasing Scheme for Use With the Expectation Estimator in Deep–Penetration Problems," *Annals of Nuclear Energy*, **11:6,** 283 (1984).

[6] S.J. Chucas, I. Curl, T. Shuttleworth, G. Morrell, "Preparing the Monte Carlo Code MCBEND for the 21st Century," *Proceedings of the Eighth International Conference on Radiation Shielding*, Arlington, Texas, 24–28 April, 1994, Vol. 1, 381 (1994).

[7] J. C. Nimal, T. Vergnaud, "TRIPOLI: A General Monte Carlo Code, Present State and Future Prospects," *Progress in Nuclear Energy*, **24,** 195 (1990).

[8] K. A. Van Riper, T. J. Urbatsch, P. D. Soran, D. K. Parsons, J. E. Morel, G. W. McKinney, S.R. Lee, L.A Crotzer, F.W. Brinkley, J.W. Anderson, and R.E. Alcouffe, "AVATAR – Automatic Variance Reduction in Monte Carlo Calculations," *Proceedings of Joint International Conference on Mathematical Methods and Supercomputing in Nuclear Applications,* Saratoga Springs, New York, 6–10 October, 1997, Vol. 1, 661 (1997)

[9] S. A. Turner, E. W. Larsen, "Automatic Variance Reduction for Three–Dimensional Monte Carlo Simulations by the Local Importance Function Transform". *Nuclear Science and Engineering*, **22**, 127 (1997)

[10] M. A. Cooper, E. W. Larsen, "Automated Weight Windows for Global Monte Carlo Particle Transport Calculations" *Nuclear Science and Engineering*, **137**, 1 (2001)

[11] J. C. Wagner, E. D. Blakeman, D.E. Peplow, "Forward–Weighted CADIS Method for Global Variance Reduction," *Transaction of the American Nuclear Society*, **97**, 630 (2007).

[12] S. W. Mosher, T. M. Miller, T. M. Evans, J. C. Wagner, "Automated Weight Window Generation for Threat Reduction Applications using ADVANTG," *Proceedings of the International Conference on Mathematics, Computation Methods and Reactor Physics,* Saratoga Springs, New York, 3 –7 May, 2009 (2009)

[13] S. A. Turner, "Automatic Variance Reduction for Monte Carlo Simulations via the Local Importance Function Transform" Technical Report LA–13119–T, Los Alamos National Laboratory, 1996.

[14] T. L. Becker, "Hybrid Monte Carlo/ Deterministic Methods for Radiation Shielding Problems," *Thesis –Ph.D., University of Michigan*, (2009).

Chapter 5
Simulation of Neutron Pulse Height Distributions with a Response Matrix Method

5.1 Introduction

Valuable information regarding the spectrum can be obtained from PHDs measured with organic scintillation detectors. Unfortunately, standard Monte Carlo codes (such as MCNP) do not have the ability to estimate PHDs from organic liquid scintillation detectors, as we discussed in Ch. 3. Firstly, standard Monte Carlo codes do not incorporate nonlinear detector response needed to generate PHDs as detailed in Ch. 2 and Ch. 3. Secondly, they do not correctly simulate individual collisions and the subsequent correlated light production, such as in organic liquid scintillation detectors [1, 2].

The method proposed in this chapter utilizes a single pre–computed detector response matrix that operates on the energy–dependent neutron current incident on the detector (estimated by Monte Carlo) to calculate the PHD. This method is general and can be applied to any source–detector configuration, including the presence of shielding: after the response matrix for the detector has been obtained, one only needs to calculate the neutron energy spectrum incident on the detector face. Any standard MCNP variance reduction technique may be applied to calculate this incident neutron energy spectrum. In this chapter, the response matrix method (RMM) is applied as a variance reduction

technique to shielded neutron sources; these results are compared to analog MCNPX–PoliMi results and to measured data.

5.2 Numerical Method

The RMM makes use of an MCNPX–PoliMi calculated response matrix to estimate the neutron PHD in an organic liquid scintillation detector. To use the response matrix, the neutron energy spectrum incident on the detector must be known. This spectrum may be estimated using Monte Carlo with any standard variance reduction technique applicable to the problem. The formulation of the method is described below.

5.2.1 MCNPX–PoliMi Response Matrix

The response matrix used in the RMM calculation contains information concerning the intrinsic efficiency of the organic liquid scintillation detector, which is sensitive only to gamma–rays and fast neutrons [3]. One MeVee is defined as the amount of light produced by one MeV energy deposited by a recoil electron (following Compton scattering) [4]. The rows of the response matrix correspond to incident neutron energies, ranging from 0.2 MeV to 15 MeV, with increments of 20 keV (741 energies), while the columns correspond to light output, ranging from 0.01 MeVee to 9.85 MeVee with increments of 10 keVee (985 light output bins). The energies and light bins are user–specified and can be different from the values discussed here. The sum of all light outputs (or columns) for a given energy (row) gives the total intrinsic detection efficiency at that energy. Thus, for *m* energy rows and *n* light output columns, the response matrix is:

$$R = \begin{bmatrix} R_{E1,L1} & \cdots & R_{E1,Ln} \\ \vdots & \ddots & \vdots \\ R_{Em,L1} & \cdots & R_{Em,Ln} \end{bmatrix}.$$

This response matrix was obtained using analog MCNPX–PoliMi simulations that calculated the energy deposited by each neutron collision; this energy was converted to light at each collision using appropriate coefficients. A separate MCNPX–PoliMi simulation was performed for each incident energy, corresponding to a single row in the matrix. The simulated neutrons were monoenergetic, monodirectional, and normally incident on the detector face. The neutrons started from a circular surface source placed 30 cm away from the EJ–309 detector. The EJ–309 detector is a cylinder with a radius of 6.34 cm and a length of 12.51 cm. The chemical composition of the EJ–309 liquid is 55.5% hydrogen and 44.5% natural carbon by atom. The simplified models of the photomultiplier tube, the Pyrex window, and the aluminum casing were also included.

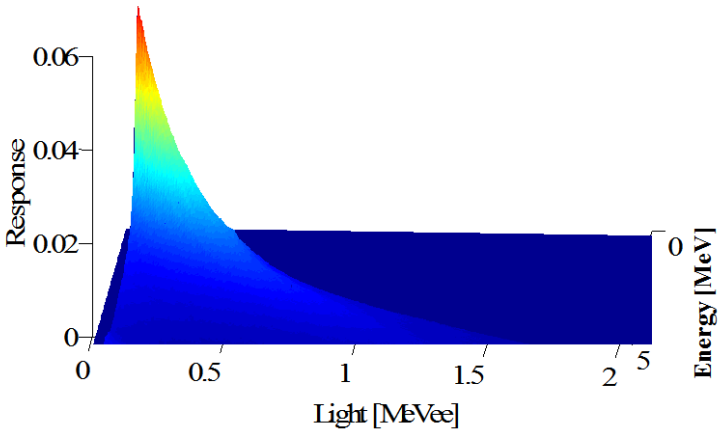


Figure 5– 1: EJ–309 liquid scintillation detector response matrix as a function of energy and light output.

The response matrix is created by simulating neutrons entering the detector one at a time; therefore, a key assumption is that only one neutron contributes to a pulse. If two or more neutrons arrive close in time (within the same PGT) they may contribute to the same pulse. The assumption of a pulse formed by a single neutron is only valid if less than 2×10^{8} neutrons are incident on the detector per second (based on the characteristic response time of this detector). This value is orders of magnitude higher than typically encountered nonproliferation and safeguards measurements; the source used for the measurements discussed in this chapter emitted approximately 3.3×10^{5} neutrons per second.

A 2.0 GHz, 2.0 GB RAM Intel Celeron single processor computer was used to perform the response matrix simulations. Nearly 29.5 hours were required to simulate all the 741 MCNPX–PoliMi simulations and another 14.5 hrs to post–process them. Thus, the total time to calculate the response matrix was nearly 44 hours. This time was not included in the figure of merit (*FOM)* calculations because the response matrix was created only once for a detector and does not need to change for different source–shielding configurations. The individual elements of the response matrix have statistical uncertainties ranging from 0.5% to 20%. The response matrix elements that are of practical interest to us measure the fission neutron energy spectrum (less than 2 MeVee), and have statistical uncertainties ranging from 0.5% to 5%. These uncertainties have been included in calculating the statistical uncertainties in PHDs as discussed in Subsection 5.5.2.

5.2.2 Pulse Height Distribution Formulation

A response matrix element represents the intrinsic efficiency for an incident energy E at light output L, *i.e.,* its product with the number of incident neutrons at a given energy E yields the total number of neutrons detected for that E, at light output L. Summing these products over all incident energies as in Eq. (1), one obtains the total number of counts for the light output bin L. Similarly, counts can be obtained for all light output bins, yielding a complete pulse height distribution. The formula governing this method using MCNP is given as follows:

$$N(L) = \sum_{i=1}^{m} \hat{n} \bullet J^{+}(E_i) \cdot A \cdot R(E_i, L) \cdot \Delta E_i \quad . \tag{1}$$

$N(L)$ = *total counts for a given light output L*
L = *a given light output bin*
E_i = *i^{th} energy bin*
$\hat{n}$ = *vector normal to the detector face*
J^{+} = *partial current towards detector*
A = *area of the detector*
$R(E_i, L)$ = *response matrix element at E_i and L*
ΔE_i = *energy bin width about E_i*

The F1 tally in MCNP counts the number of particles crossing a user–specified surface. Thus, the number of particles entering the detector face is calculated by the F1 tally and satisfies [5]:

$$F1_{MCNP}(E_i) = \hat{n} \bullet J^{+} \cdot A \cdot \Delta E_i \quad . \tag{2}$$

Substituting Eq. (2) into Eq. (1), we obtain the following relationship for counts in a light output bin:

$$N(L) = \sum_{i=1}^{m} F1_{MCNP}(E_i) \cdot R(E_i, L) \quad . \tag{3}$$

Equation (3) is used in the RMM to calculate the neutron PHD. The MCNP F1 tally used in this result can be calculated using MCNP5, MCNPX or MCNPX–PoliMi. The F1 tally

also does not need to be simulated in analog mode and can make use of applicable variance reduction techniques. In fact, the neutron current could also be solved with a deterministic code, but we are only considering Monte Carlo simulations in this chapter.

5.2.3 Radial Leakage Correction Factor

As mentioned in Subsection 5.2.1, the response matrix has been constructed for monoenergetic neutrons that travel monodirectionally, normal (perpendicular) to the detector face as shown in Fig. 5–2 (a). In Figs. 5–2 (a) and (b), the blue lines depict the approximate path taken by the neutrons in the cylindrical detector. In reality, neutrons are rarely monodirectional or normal to the detector face. In Fig 5–2 (b), a setup with an isotropic point source is shown, in which the neutrons enter the detector at different angles. As shown in Fig. 5–2 (b), the non–normal neutrons have a greater chance of leaking out of the sides of the detector, and in general will have a shorter pathlength (have a smaller likelihood to scatter) in the detector, and thus a smaller probability of being detected. As a result, the PHD calculated with the previously described RMM will be an overestimate as in Fig. 5–3. When the detector is directly pointed towards the source, for large source–detector distances this effect will be minimal because the angle subtended between the source and the detector will be small, thus the incident particle directions will be closer to perpendicular. However, as the source moves closer to the detector, the spread of the incident direction of a particle with respect to the detector face increases, and this effect becomes more important. Figure 5–3 illustrates this effect for a bare point ^{252}Cf source placed 10 cm away from the detector.

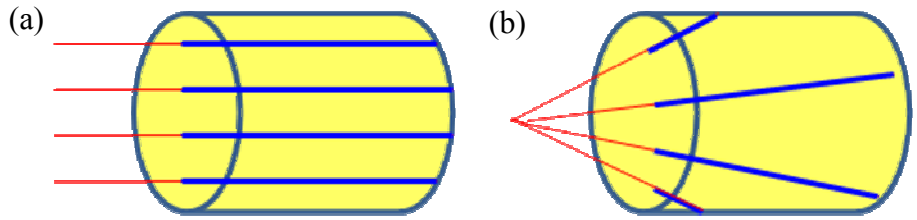


Figure 5– 2: On left source neutrons enter the detector parallel to the detector face. On right source neutrons from an isotropic point source enter the detector at various angles.

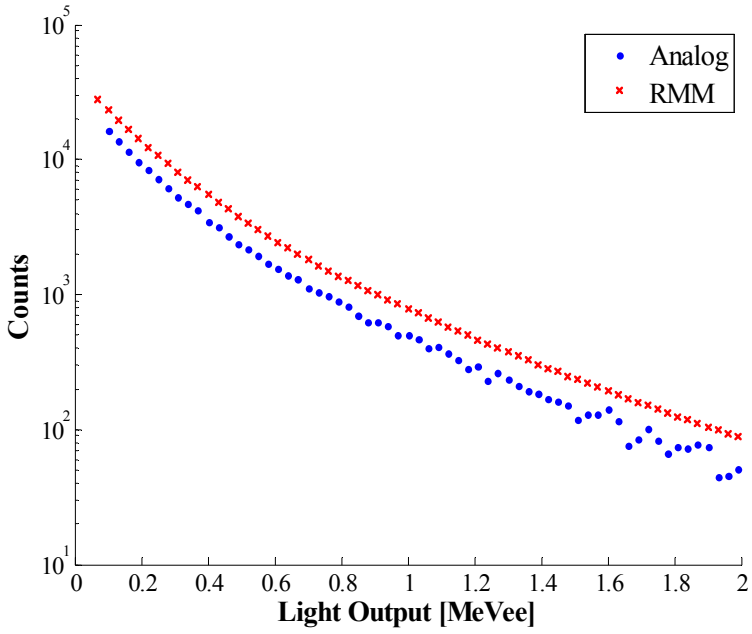


Figure 5– 3: Analog PHD and response matrix PHD without radial leakage correction.

In Fig. 5–3, the discrepancy is significant between the RMM PHD and the MCNPX–PoliMi analog PHD, which correctly models the incident angle of the neutron and the subsequent interactions. To correct for this discrepancy, we formulate a radial leakage correction factor:

$$\zeta \equiv \Phi_{true} / \Phi_{normal} \ . \qquad (4)$$

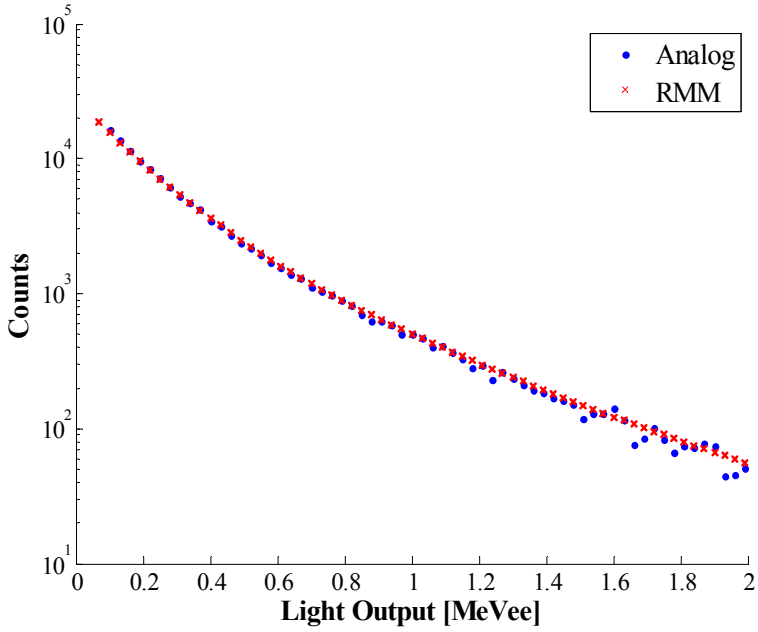


Figure 5– 4: Analog PHD and response matrix PHD with radial leakage correction.

The radial leakage correction factor is a ratio of the volume averaged flux (rate of pathlength creation in a given volume [5]) for the given setup with unmodified particle direction, Φ_{true}, to the case where the particles in the setup are made monodirectionally incident on the detector face, Φ_{normal}. Thus, ζ is problem–specific; however, there is no need to perform additional Monte Carlo simulations to calculate ζ. The desired parameter, Φ_{normal} can be calculated at the time of obtaining the response matrix, whereas Φ_{true} can be calculated in the problem–specific Monte Carlo simulation used to calculate the incident current on the detector face. In the problems discussed in this thesis, the correction factor accounts for greater leakage from the detector sides to reduce the intrinsic efficiency, and the overall counts. The ratio given in Eq. (4) is calculated for each energy group. The intrinsic efficiency, ε, of each light output in an energy group is multiplied by its corresponding ζ to yield the corrected intrinsic efficiency as:

$$\varepsilon_{corrected}(E_m, L_n) = \varepsilon_{normal}(E_m, L_n)\,\zeta(E_m) \quad . \tag{5}$$

The radial leakage correction factor was incorporated by modifying Eq. 3 as:

$$N(L) = \sum_{n=1}^{N} \zeta(E_n) \cdot F1_{MCNP}(E_n) \cdot R(E_n, L) \quad . \tag{6}$$

After this modification, the agreement between the RMM and the analog MCNPX PoliMi PHDs is significantly improved, as shown in Fig. 5–4. The radial leakage correction factor for the problems discussed in this thesis is less than unity; however, there may be cases where the factor can be greater than unity.

5.2.4 Variance Reduction Techniques

It is important to choose the variance reduction options that efficiently reduce uncertainty, while minimizing simulation run–times [5]. A typical fission spectrum has fewer high energy neutrons (beyond a few MeVs), so an F1 tally will tend to have greater uncertainty in these regions. Therefore, it is important to reduce the uncertainty in the F1 tally for higher energies. To do this, we applied a source biasing technique to calculate the F1 tally.

To obtain a uniform energy distribution throughout the spectrum, a flat distribution is chosen as the desired probability density distribution, and the weights are then calculated according to Eq. (2) in Ch. 4. Thus, given that w_0 =1, it can be gleaned from that $\hat{w}$ must follow the spectral shape of p_0 (if $\hat{p}$ is a flat distribution) such that the product $\hat{w} \cdot \hat{p}$ is preserved as shown in Eq (2) of Ch. 4. The success of these choices is demonstrated using MCNP results, as discussed later in Section 5.5.

5.3 Laboratory Measurement

Measurements were performed to validate simulation results; the experimental setup is shown in Fig. 5–5 through Fig. 5–7. A ^{252}Cf source emitting approximately 3.3×10^{5} neutrons per second, placed 30 cm from detector was shielded with different thicknesses of polyethylene and lead. The measurements were performed in the Detection for Nuclear Nonproliferation Group Laboratory (DNNG) at the University of Michigan.

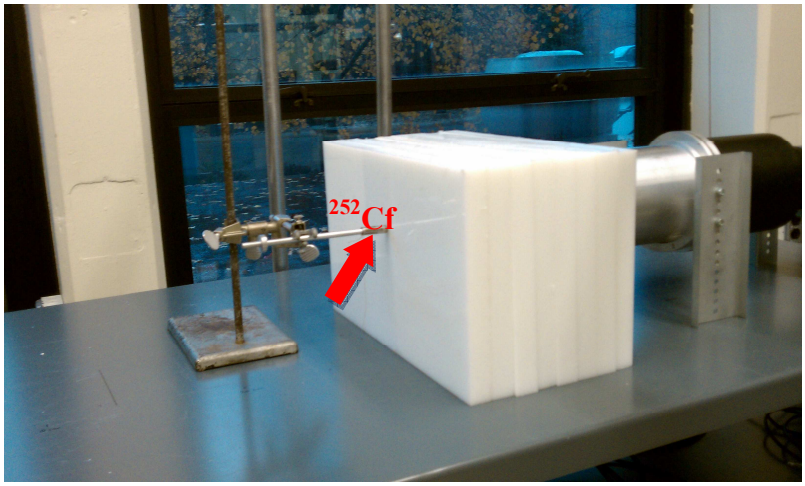


Figure 5– 5: Laboratory setup with a ^{252}Cf source, 20.64 cm thick polyethylene shield and an EJ–309 detector.

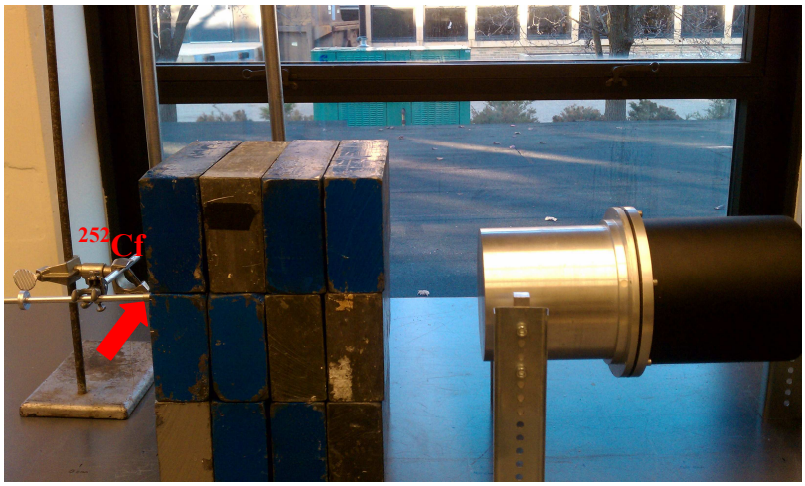
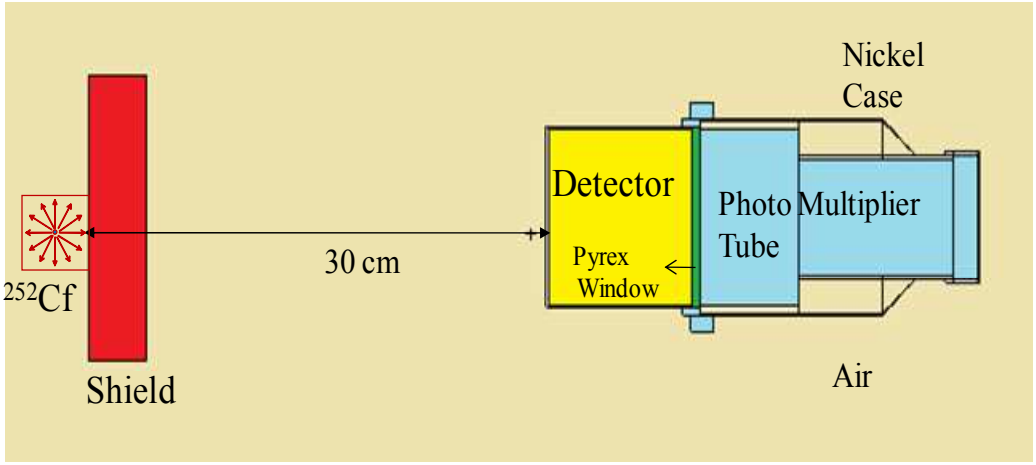


Figure 5– 6: Laboratory setup with a ^{252}Cf source, 20.64 cm thick lead shield and an EJ–309 detector.

A 1–μCi ^{137}Cs source was used for calibration, and a 12–bit, 250 MHz waveform digitizer was used for data acquisition. The MCNP simulation geometry is depicted in Fig. 5–7.



Polyethylene Shield = 5.08, 10.16, 15.24 and 20.32 cm
Lead Shield = 5.08, 10.16, 15.24 and 20.32 cm

Figure 5– 7: Shielded ^{252}Cf setup with an EJ–309 liquid scintillation detector.

The threshold for detection was determined to be 70 keVee (equivalent of 650 keV of neutron energy deposited on hydrogen). The acquisition window was 100 points long, with each point 4 ns. The pulses were discriminated between neutrons and gamma–rays using a standard charge integration method [6]. The pulse shape discrimination (PSD) results are shown in Fig. 5–8. Neutrons interact with the nuclei in the scintillation detector and result in larger tails compared to gamma–rays, which interact with electrons. In Fig. 5–8, points corresponding to higher tail integral values for the same total integral value of the pulse are from neutrons; those points are found above the discrimination line. In Fig. 5–8 the ^{137}Cs data have been overlaid to verify that the pulses below are indeed gamma–rays.

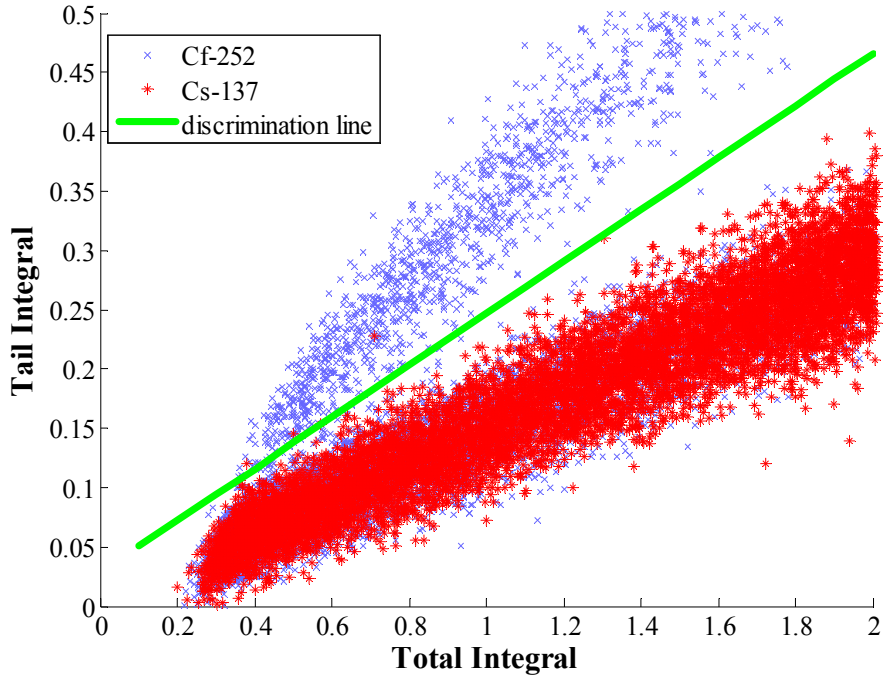


Figure 5– 8 Pulse shape discrimination between neutrons (above line) and gamma–rays (below line) for the setup with no shielding.

5.4 Monte Carlo Simulations

The measurement setup depicted in Fig. 5–7 contains an isotropic point–like ^{252}Cf source placed 30 cm from the face of an EJ–309 liquid scintillation detector. In addition to the features shown in Fig. 5–7, the iron table on which the detector rests and the concrete floor are also modeled. The source is shielded by lead or polyethylene rectangular blocks that are 5.08 cm, 10.16 cm, 15.24 cm and 20.32 cm thick. The composition and the dimensions of the detector have been specified in Subsection 5.2.1. The point ^{252}Cf source is modeled only to emit the fission spectrum neutrons simulated by the Watt spectrum (shown on the right in Fig. 1–1) using coefficients defined in

Ref. 19. This Watt spectrum is in good agreement with the Watt spectrum given by MCNP–DSP [8].

As discussed in the previous section, source biasing requires the user to input a desired distribution of source neutrons, $\hat{p}$. This desired distribution in our case is a uniform distribution of source neutrons throughout the spectrum such that a uniform uncertainty distribution is obtained in the F1 tally throughout the energy spectrum. Based on these biased probabilities, weights $\hat{w}$ are calculated such that the product of original weight w_0 and probability p_0 are preserved as given by Eq. (2) of Ch. 4. In Fig. 5–9, both the MCNP new weights $\hat{w}$ and those calculated independently using Eq. (2) are in good agreement. Since $\hat{p}$ is a flat distribution, it is expected that the spectral shape of $\hat{w}$ will be that of a Watt spectrum (Fig. 5–9), in order to preserve the original assumed probability distribution p_0 of the Watt spectrum.

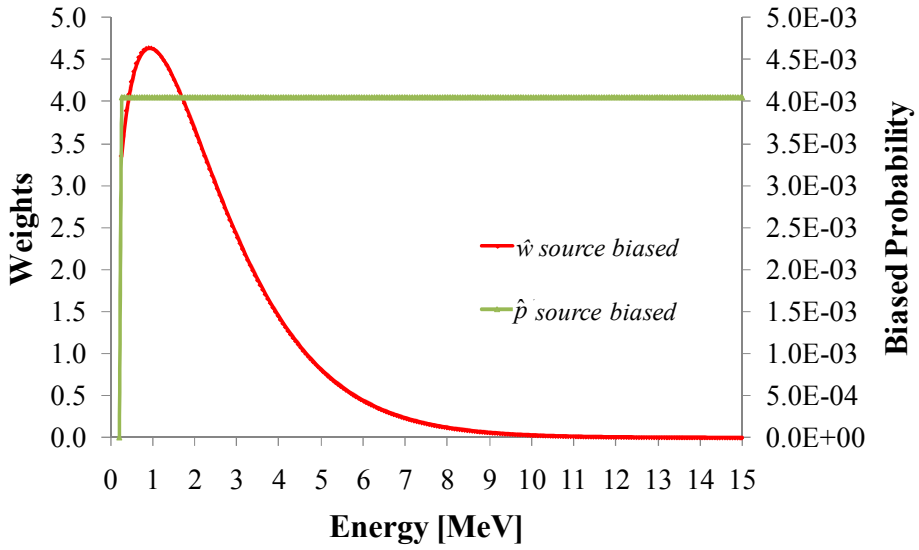


Figure 5– 9: Source biased new weights and biased new probability.

5.5 Results and Analysis

This section is divided in four subsections. Subsection 5.5.1 presents the source–biased F1 tally results that give the neutron energy spectrum entering the detector. This spectrum is combined with response matrix to yield pulse height distributions. Subsections 5.5.2 and 5.5.3 discuss the RMM results and their statistical uncertainties respectively. Finally, in Subsection 5.5.4, comparisons of the simulation results with the measurement data are given.

5.5.1 Incident Current on Detector Face Using Source Biasing

The F1 tally in MCNP was used to tally the neutron spectrum entering the detector face after leaving the 10.16–cm thick polyethylene or lead shields. These cases, depicted in Fig. 5–10, were simulated for 10^7 source particles and have been normalized to the source strength in the laboratory measurements. The neutron distribution from the polyethylene shield in Fig. 5–10 has been heavily moderated and has lost its original Watt spectrum shape, but the neutron distribution from the lead shield has still retained its basic Watt spectrum shape. The reason for the results in Fig. 5–10 is that polyethylene is a hydrogenous material and is very effective in moderating neutrons, while lead, as a heavy nucleus, is a less effective moderator. Lead also has several resonances in its elastic scatter cross–sections, contributing to the fluctuations seen in the peak of the distribution in Fig. 5–10 [9].

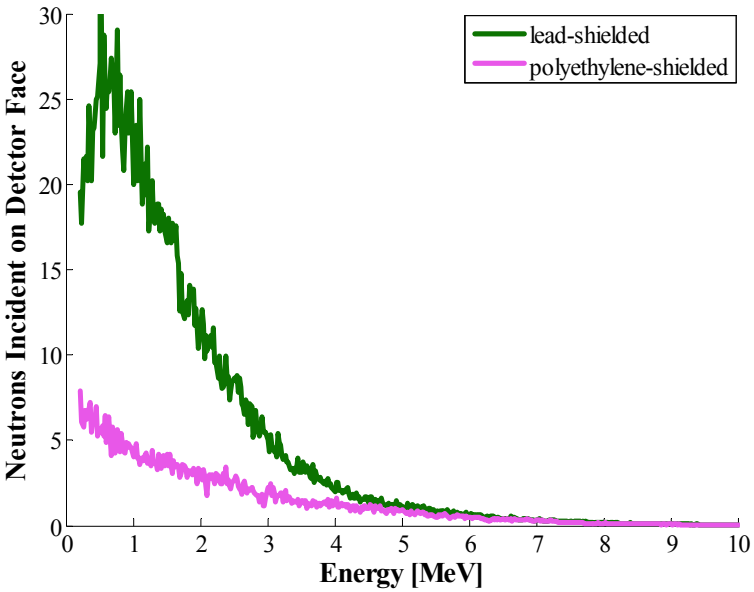


Figure 5– 10: ^{252}Cf neutron distributions out of 10.16 cm of polyethylene and lead shields incident on detector face.

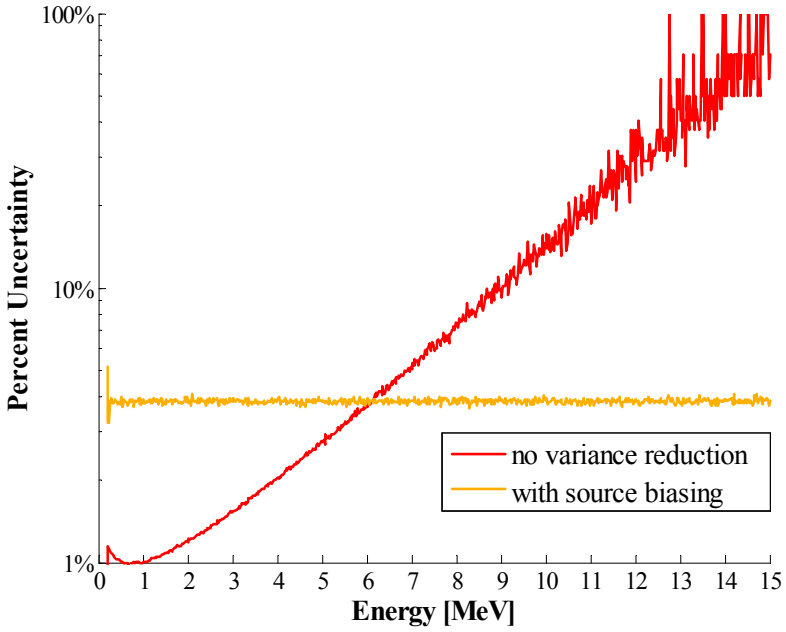


Figure 5– 11: Percent uncertainty in neutron energy distribution entering detector.

The MCNP simulations were performed with and without variance reduction techniques. The MCNP simulation with variance reduction utilized source biasing to produce uniform uncertainty throughout the energy distribution, as shown in Fig. 5–11.

5.5.2 Pulse Height Distribution Comparison

The current of neutrons entering the detector is combined with the response matrix as shown by Eq. 3 to yield pulse height distributions as shown in Figs. 5–12 and 5–13 for 10.16–cm thick polyethylene and lead shields. These figures show that the analog pulse height distribution obtained from MCNPX–PoliMi has large variances. However, in the case of the RMM, and the RMM with source biasing, the pulse height distributions have reduced variances because all neutrons incident on the detector face contribute to each of the light output bins of the solution.

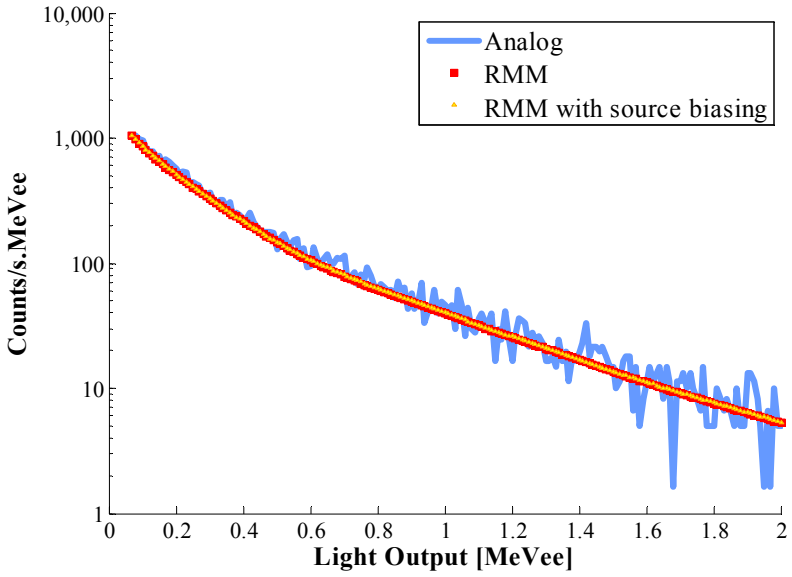


Figure 5– 12: Simulated ^{252}Cf neutron PHDs from 10.16 cm of polyethylene shielding (RMM and RMM with source biasing are indistinguishable to the eye in this figure).

The figures are shown only up to 2 MeVee because the measurement data were measured only up to 2 MeVee. For this light output range, both the response matrix PHD methods, with and without source biasing have small uncertainty and are difficult to differentiate. The differences between these methods are better illustrated by Figs. 5–14 and 5–15.

Figures 5–12 and 5–13 show that the lead–shielded neutron PHD has higher counts than the polyethylene–shielded neutron PHD, which is expected given the two energy distribution shown in Fig. 5–11 (the lead–shielded neutron PHD is less moderated than the polyethylene–shielded neutron PHD).

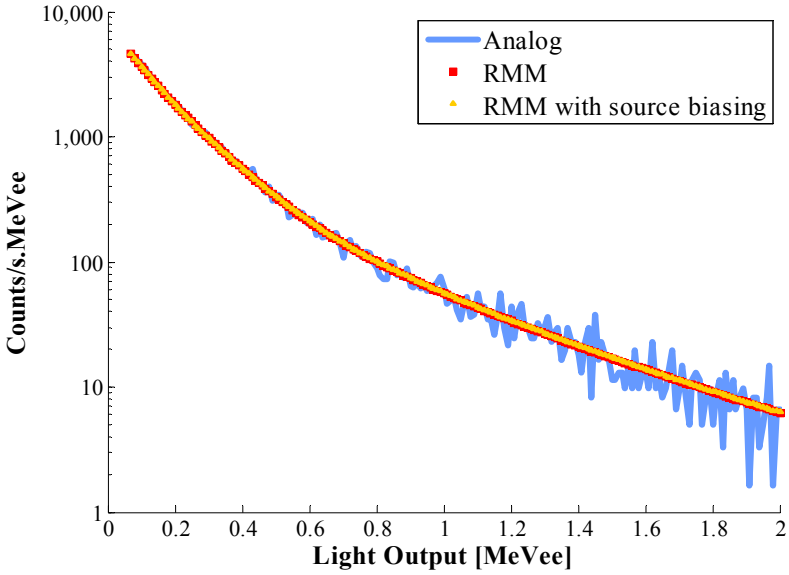


Figure 5– 13: Simulated ^{252}Cf neutron PHDs from 10.16 cm of lead shielding (RMM and RMM with source biasing are indistinguishable to the eye in this figure.

5.5.3 Pulse Height Distribution Uncertainty Comparison

The statistical uncertainty in the PHD for 10.16 cm thick polyethylene or lead shield was obtained using three methods: analog MCNPX–PoliMi, the RMM, and the

RMM with source biasing. The results in Figs. 5–14 and 5–15 include the statistical uncertainties in the incident neutron spectrum, the radial leakage correction factors, and the response matrix elements. The advantage of using the RMM is evident from these graphs. The analog PHD results in uncertainty over 10% at approximately 500 keVee and reaches nearly 100% at approximately 2 MeVee. The RMM PHD method is a significant improvement over the analog case: in this case, the uncertainty shows a gradually increasing trend, which crosses 10% around 5 MeVee.

However, the RMM with source biasing provides the best improvement: for this method the uncertainty remains constant between 1% and 2% up to 8 MeVee. This is nearly a three order–of–magnitude improvement for light output greater than 1 MeVee for the same number of source particles and slightly reduced simulation time in MCNP.

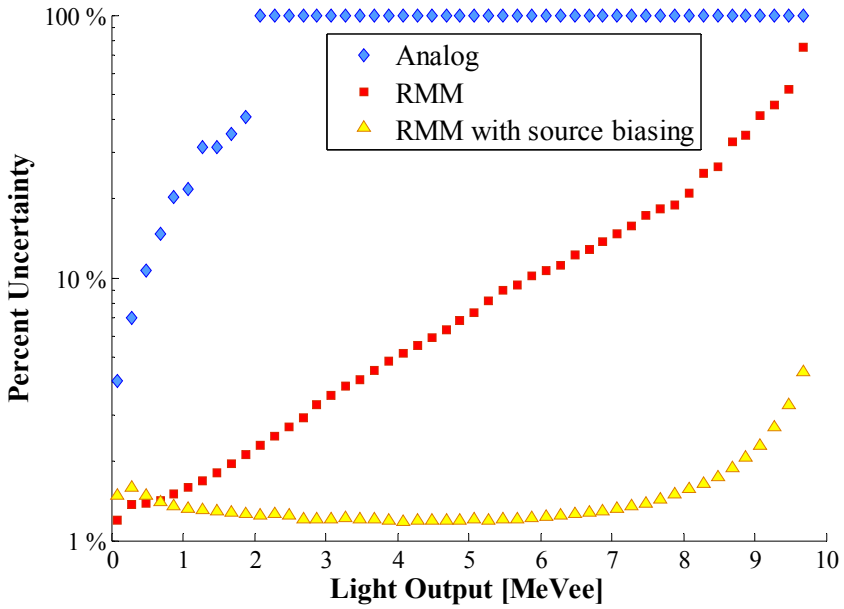


Figure 5– 14: Uncertainty in simulated ^{252}Cf PHDs from 10.16 cm of polyethylene shielding.

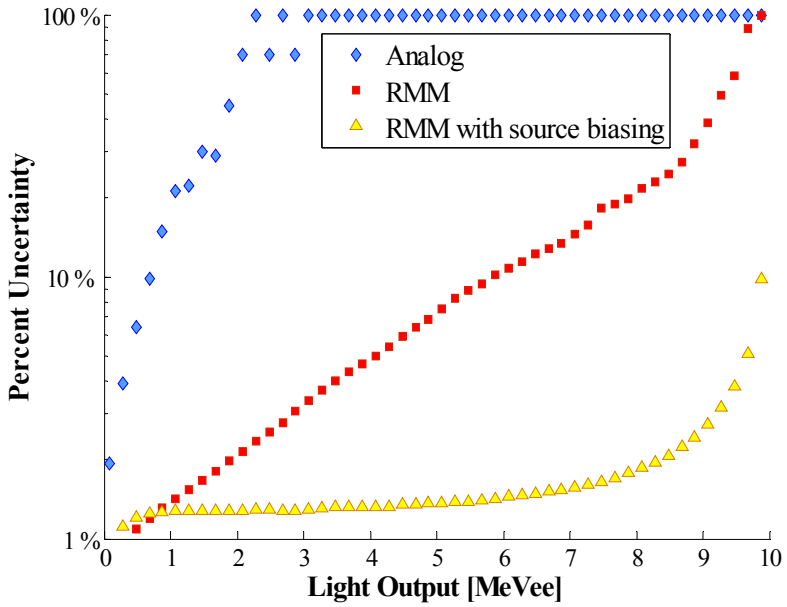


Figure 5– 15: Uncertainty in simulated ^{252}Cf PHDs from 10.16 cm of lead shielding.

A figure of merit (*FOM*) is calculated using the standard MCNP convention:

$$FOM \equiv 1/(R^2 T) \quad , \tag{7}$$

where R is the sample relative standard deviation of the mean and T is the computational time [5]. The *FOM* factor of improvement, I, is then calculated as

$$I = FOM_{RMM\ source\ biased} / FOM_{\ analog} \quad . \tag{8}$$

In Fig. 5–16, I factors are shown as a function of light output for 10.16 cm of polyethylene and lead–shielded cases. For individual low light output bins these factors are small; but as we move towards higher light output bins, I increases by greater than three orders of magnitude. For polyethylene shielding, the I factors are significantly higher than lead. This is because the number of neutrons exiting the polyethylene shield is significantly lower than the number of neutrons exiting the lead shield. Thus, for analog simulations, the relative error in the polyethylene–shielded case is higher for the

same number of source particles, resulting in a low *FOM*. With the RMM, the improvement in the *FOM* is more effective and noticeable for the heavily shielded scenarios.

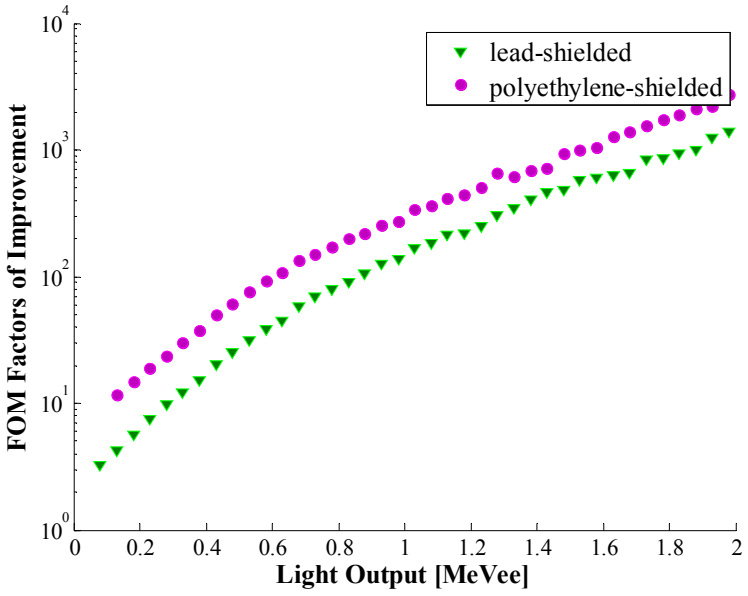


Figure 5– 16: Factors of improvement versus light output for 10.16 cm of lead and polyethylene shielding.

In Table I, average factors of improvement in the *FOM*, I_{avg} (average of *I* over all light bins), for different shield types and thicknesses are shown. The factor I_{avg} can be understood as the saving in computation time if the user is equally interested in counts from all light output bins. For lead–shielded cases, on average the user will save a factor of about 300 in computational time, but for polyethylene–shielded cases the saving is approximately 600. As discussed in Subsection 5.2.1, the simulation time for the response matrix is not included in the *FOM* since the response matrix only needs to be created once for a detector and does not change with different source–shielding configurations.

Table 5– 1 Average *FOM* Improvement with RMM.

Shielding Material	5.08 cm	10.16 cm	15.24 cm	20.32 cm
Polyethylene	479	660	765	605
Lead	282	329	339	327

The RMM differs from the MCNPX–PoliMi analog method in that processing an additional data bank is not needed for RMM; the detector response is already contained in the MCNPX–PoliMi calculated response matrix. Furthermore, because the response matrix has fixed dimensions (*m* by *n*), the number of floating point operations performed to process the data (Eq. 3) is independent of the number of source particles: it is always *2mn – n.* Given 741 energies and 985 light bins for cases discussed here, there are less than 1.5 million floating point operations performed for any number of source particles using the RMM. However, in the analog PHD case, even the post processor will do more work and run for longer times for a greater number of source particles, since there will be more collisions to follow. Additionally, the analog case will use greater storage space to save the data bank as the number of source particles is increased.

5.5.4 Comparison of Response Matrix PHD with Measured Data

The computational advantage of the RMM with and without standard MCNP variance reduction techniques has been verified above by comparing the results with MCNPX–PoliMi analog cases. In this section, the RMM is validated with measured data to assess its ability to simulate physical reality. The measurement technique has been discussed in Section 5.3; here we discuss the results.

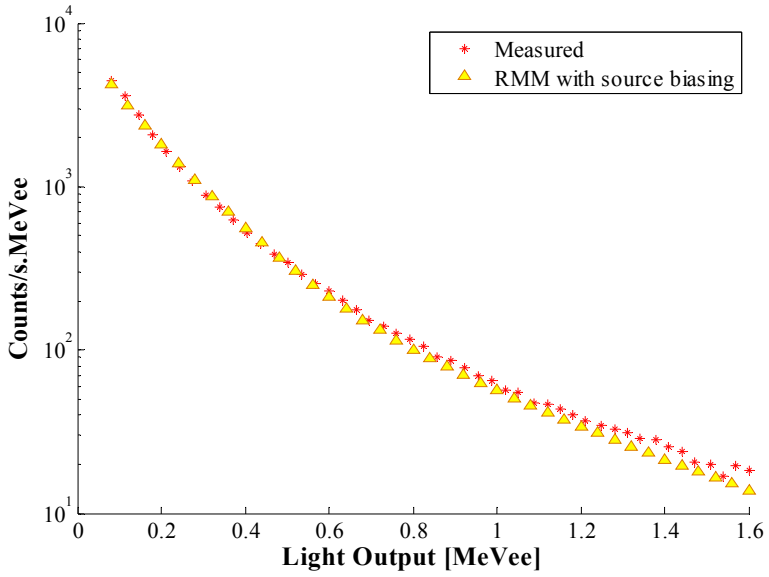


Figure 5– 17: Measured and simulated ^{252}Cf neutron PHDs from 10.16 cm of lead shielding.

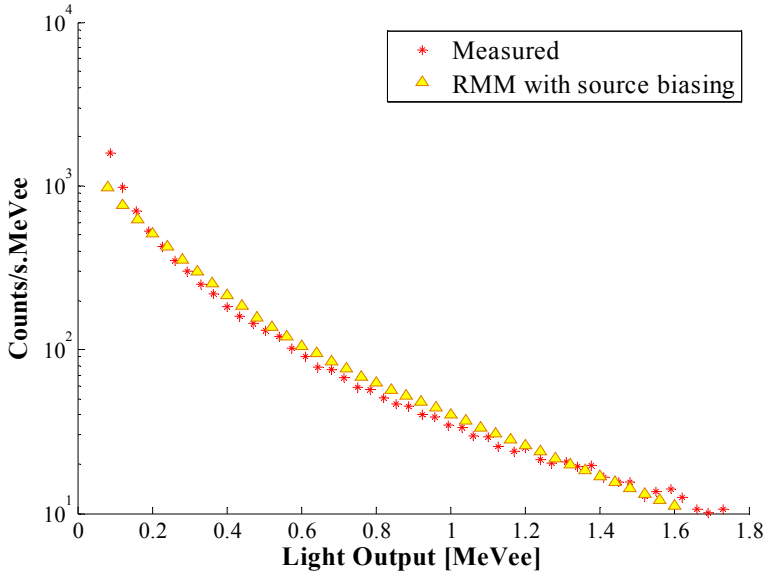


Figure 5– 18: Measured and simulated ^{252}Cf neutron PHDs from 10.16 cm of polyethylene shielding.

The RMM results for a lead–shielded ^{252}Cf source shown in Fig. 5–17 are within 20% of the measured data. The simulated RMM counts are slightly lower than the

measurement for light output greater than 0.80 MeVee. Measurements with polyethylene shields are shown in Fig. 5–18. The agreement in the case of polyethylene is generally good and within 20%. The simulated counts are lower than the measured values for light output below 0.2 MeVee. The disagreement in this low region is likely due to the misclassification of neutrons in the PSD process, as shown in Fig 5–8 and discussed in Ref. 18. For low light outputs, gamma–rays and neutrons can overlap in the PSD plot. This overlap becomes more important when the shield becomes increasingly moderating, in which case there are fewer neutrons, and even infrequent misclassifications can yield erroneous results. A more general reason for the discrepancy between the measured and the simulated could be the measured light conversion coefficients (Eq. 4 in Ch. 2) that convert the energy deposited in the detector into the light output [10]. The pulse height distributions are very sensitive to the measured coefficients used for converting energy deposited in an interaction into the pulse height in light units (MeVee). The detailed sensitivity analyses and discussions can be found in Ch. 2.

5.6 Conclusion

Standard Monte Carlo codes such as MCNP cannot calculate neutron pulse height distributions for organic scintillation detectors. MCNPX–PoliMi and its associated post processor can calculate pulse height distributions, but these calculations can be done only in analog mode, resulting in time–consuming simulations and post processing. The RMM utilizes a single detector response matrix, which operates on the incident neutron energy to calculate the detector pulse height distribution. This method allows the use of variance reduction techniques to estimate the neutron current incident on the detector face. The RMM also includes a radial leakage correction factor to correct for neutrons not normally

incident on the detector face. This correction factor is problem–specific, but it can be calculated using Monte Carlo calculations already employed.

A comparison of the analog method, the RMM, and the RMM with source biasing is made for 10^7 source neutrons. It is shown that the RMM greatly reduces the variance throughout the pulse height distribution. For RMM with source biasing, a *FOM* improvement (savings in computation time) of a factor of 600 is achieved in the case of polyethylene shields and a factor of 300 is achieved in the case of lead shields over the entire PHD, on average. Using the RMM with source biasing decreases the variance and keeps it nearly constant throughout the distribution. The statistical uncertainty in the RMM simulation with source biasing is usually constant between 1–2%, whereas in the analog simulation the uncertainty exceeds 10% at only 0.5 MeVee. Furthermore, the analog simulation requires storage of large data files and time consuming post processing of these files.

Comparisons with measured data for polyethylene and lead shields are promising and show good agreement in general. In the future, work can be done to further generalize this method by creating a response matrix solely to calculate the radial leakage correction factor. In conclusion, we note that the response matrix method as described in here is not limited to organic liquid scintillation detectors, but is applicable to other detector types as well.

Although RMM has provided good speedup and results validated by measurements, it is difficult to extend it to time–dependent problems. Thus, we proceed with the direct nonanalog MCNPX–PoliMi approach to develop a method which is efficient, accurate and applicable to time–dependent problems.

5.7 References

[1] S. A. Pozzi, E. Padovani, and M. Marsegeurra, "MCNP–PoliMi: A Monte Carlo Code for Correlation Measurements," *Nuclear Instruments and Methods to Physics Research A*, **513**, 550 (2003).

[2] S. A. Pozzi, E. Padovani, M. Flasks, and S. D. Clarke, "MCNP–PoliMi Post– Processing Code," ORNL/TM–2007/33, Version 1.9, Oak Ridge National Laboratory (2007).

[3] S. Prasad, S. D. Clarke, S. A. Pozzi and E. W. Larsen, "A Response Matrix Method for the Calculation of Neutron Pulse Height Distributions in MCNP," *Transactions of the Institute of Nuclear Materials Management 51st Annual Meeting*, Baltimore, Maryland, 11 – 15 July, 2010, (2010).

[4] G. F. Knoll, *Radiation Detection and Measurements*, John Wiley & Sons, Inc., New Jersey (2000).

[5] "MCNP—A General N–Particle Transport Code, Version 5, Volume I: Overview and Theory," X–5 MONTE CARLO TEAM, LA–UR–03–1987, Los Alamos National Laboratory (2003).

[6] S. D. Ambers, M. Flaska, and S. A. Pozzi, "A Hybrid Pulse Shape Discrimination Technique with Enhanced Performance at Neutron Energies Below 500 keV," *Nuclear Instruments and Methods to Physics Research A,* **638**, 116 (2011).

[7] D. Pelowitz (ed.). "MCNPX User's Manual, version 2.6.0," LA–CP–07–1473, Los Alamos National Laboratory (2008).

[8] T. E. Valemtine, "MCNP–DSP USERS MANUAL," ORNL/TM–13334, R2, Oak

Ridge National Laboratory (2001).

[9] M. B. Chadwick et al., "ENDFB–VII.0: Next Generation Evaluated Nuclear Data Library for Nuclear Science and Technology," *Nuclear Data Sheets*, **107**, 2931 (2006).

[10] S. Prasad, A. Enqvist, S. D. Clarke, S. A. Pozzi, and E. W. Larsen, "Response Function Study for Energy to Light Conversion in Organic Liquid Scintillator," *Transactions of the Institute of Nuclear Materials Management 52nd Annual Meeting,* Palm Desert, California, 17 – 21 July, 2011 (2011).

Chapter 6
Nonanalog MCNPX–PoliMi for Uncorrelated Sources

6.1 Introduction

MCNPX–PoliMi simulates detailed scintillation detector response in analog mode as discussed in Ch. 3 [1, 2]. In the previous chapter, it was shown that it is possible to capture detailed scintillation detector response as a response matrix, combine it with neutron current incident on the detector face, and produce a neutron pulse height distribution. This response matrix method (RMM) was shown to speed up PHD simulations. The method could employ variance reduction techniques to calculate the neutron current incident on the detector face. The MCNPX–PoliMi simulations to calculate the response matrix; however, were still performed in analog mode.

In this chapter, simulations of neutron detector response for liquid scintillation detectors using MCNPX–PoliMi are performed in the nonanalog mode; that is, traditional MCNP variance reduction techniques are used. The post–processing algorithm for nonanalog MCNPX–PoliMi, called VRPost, weighs the pulses to produce unbiased tallies. To test the method, uncorrelated sources are used, that is, only one neutron is emitted from a source event. The VRPost method is tested using a polyethylene shielded ^{252}Cf source for three different nonanalog MCNPX–PoliMi runs. This method is developed further in the next chapter for correlated sources (Ch. 7).

6.2 Nonanalog MCNPX–PoliMi

This section discusses: i) the reasons and circumstances in which variance reduction techniques can be applied to MCNPX–PoliMi enabling simulations in nonanalog mode, ii) quantification of improvement in computation time for nonanalog cases, and iii) development and implementation of the VRPost post–processing method that allows the use of nonanalog MCNPX–PoliMi. Detailed discussion of MCNPX–PoliMi code, the physics it simulates, and its present detection module can be found in Ch.3.

6.2.1 Separation of Phase–space for Direct Nonanalog MCNPX–PoliMi

In the analog MCNPX–PoliMi mode, each particle is explicitly tracked, thereby preserving the physics as though in an actual laboratory experiment. However, for the phase–space in which the source particle travels after it is born, but before the detector region, one may apply nonanalog Monte Carlo techniques as shown in Fig.6–1. Inside the detector region the particle is not allowed to change its weight because this is where it contributes to scintillation pulses based on energy deposition, nuclide of scatter and energy–to–light conversion coefficients.

For simulation methods that record the mean behavior of particles, such as the PHD and TOF, modifying the sample population of particles in a region of interest (in energy or in space) can still preserve the analog results while minimizing variance [3]. It is required, however, to appropriately adjust the weights to compensate for the unphysical modifications made to the sample population of particles. Thus, MCNPX-PoliMi is run

in nonanalog mode using applicable variance reduction techniques available in the standard MCNP code and as discussed in Ch. 4.

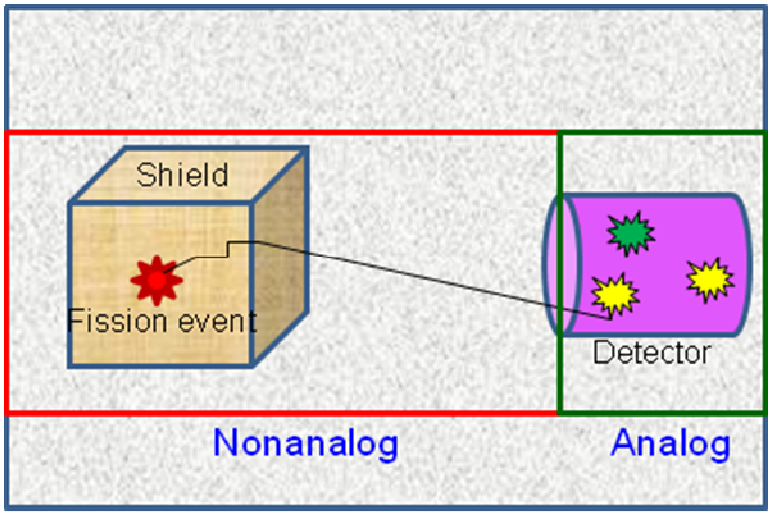


Figure 6– 1: Nonanalog MCNPX–PoliMi region after particle is born followed by analog MCNPX–PoliMi region once particle enters the scintillation detector.

6.2.2 Improvement Factors

The goal of using variance reduction techniques or performing nonanalog Monte Carlo simulations using MCNPX–PoliMi is to reduce simulation run times. Improvement factors are computed to quantify speedup, as in the RMM–PHD case (Ch.5), but here speedup is evaluated in comparison to the nonanalog cases by defining:

$$I \equiv \frac{FOM_{nonanalog}}{FOM_{analog}} \quad . \tag{1}$$

This improvement factor, *I,* is the factor by which computation run–time is reduced for the same relative error obtained for a tally.

6.2.3 Implementation of Nonanalog MCNPX–PoliMi

It is important to note that for the nonanalog MCNPX–PoliMi calculations introduced and discussed here, the particle weight can only change outside the detector, where particle collisions do not contribute to the scintillation detector pulses. The scintillation detector pulse for an incoming particle results from the energy depositions that occur during its collisions inside the detector. The total energy deposited by the particle is converted into total light produced by the particle, as discussed in Ch 3. For neutron–hydrogen collisions the light emitted is nonlinearly dependent on energy deposited. The light produced from collisions within a PGT is summed as pulse, and then compared to the detection threshold. If the pulse height or the light produced is greater than the threshold, the pulse is binned in the appropriate light output bin. The arrival time (given by the time of first scatter) and the particle type for each pulse are also stored. During this process, when light is produced, the entire pulse gets a weight equivalent to that of the incoming particle weight. Therefore, the weight of the particle must remain constant inside the detector [3]. The nonanalog MCNPX–PoliMi postprocessor, VRPost tallies the detected particle in the respective tally bin (light or time) by adding the weight. In a fully–analog case the tallies can only increment by one as a particle is followed without any change to its weight.

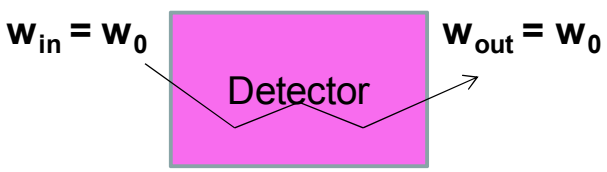


Figure 6– 2: Weight of the neutron remains constant inside the detector region.

The present MCNPX–PoliMi release version is not designed to run in nonanalog mode. Therefore, the data file output by MCNPX–PoliMi does not indicate the daughter number for particles. In Table 3–1 of Ch.3, for each collision the data file provides information on history number, particle number, scatter number and other interaction details. Therefore, when implementing the post–processing method or the detector module, the number of scatters is used to differentiate one daughter from another. As a particle scatters inside the detector, the number of scatters increases after each collision, as is shown in the data file (Table 3–1). However, for nonanalog cases, for a given particle one can observe the pattern shown in Table 6–1.

Table 6– 1: Discontinuous scatter numbers indicates the start of a new daughter.

History No.	Particle No.	Weight	Scatter No.
389678	12	0.5	7
389678	12	0.5	8
389678	12	0.5	0
389678	12	0.5	1

Table 6–1 shows only a part of the information given in a typical MCNPX–PoliMi data file. It can be seen that for the same history and same particle, the number of scatters first increases from 7 to 8 and then reduces to 0. Additionally, as seen from the weights in the above table, the particle has been split into two. We make the *assumption* that discontinuous scatter number indicates the start of another daughter. This assumption should be generally correct, except if the particle can leave the detector, scatter with a different object (e.g. a shield or another detector), and return to the detector. In this case,

there will be an error introduced. However, given the present limitation that the daughter number is not listed in the data file, we proceed with the present assumption to test the VRPost method.

6.3 Monte Carlo Simulation

This section describes the simulation cases, their MCNPX–PoliMi setup, and the variance reduction techniques used. The results obtained for these simulations will be discussed in Section 6.4.

6.3.1 Geometry Setup

The MCNPX–PoliMi model is shown in Fig.6–2. It contains an isotropic ^{252}Cf point source placed 15 cm from the face of an EJ–309 liquid scintillation detector. The source is shielded by 10.16 cm (4 inches) thick polyethylene rectangular blocks. The detector is a cylinder with a radius of 6.34 cm and a length of 12.51 cm.

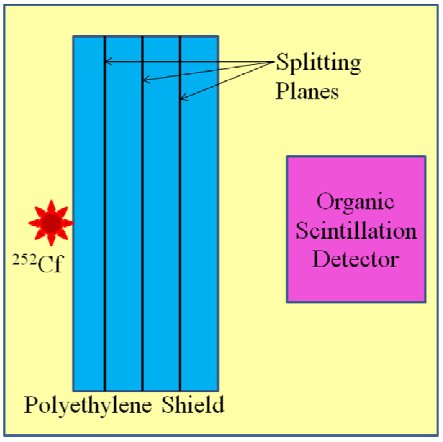


Figure 6– 3: **Shielded ^{252}Cf setup with an EJ–309 liquid scintillation detector**

The chemical composition of the EJ–309 liquid is 55.5% hydrogen and 45.5% natural carbon by atom. The ^{252}Cf source is modeled as a pure neutron source with energies sampled from a Watt spectrum using coefficients defined in Ref. 8 [7, 8]. For simplicity and ease of testing variance reduction techniques, the photomultiplier tube and detector casing have not been included.

6.3.2 Source Biasing in Energy

High energy fission neutrons are important because higher energies deposited result in larger light pulses. These pulses have a greater probability of being above the detection threshold than those generated by the more–abundant low–energy neutrons. Also, higher energy neutrons are more likely to leak through a thick polyethylene shield and reach the detector. Hence, we have applied the source biasing technique to obtain a uniform variance throughout the source distribution. The biasing function discussed in Ch. 5 has been used.

6.3.3 Geometry Splitting with Russian Roulette

Geometry splitting is a population control method which uses a particle's importance in space to determine its new weights for scoring. As a particle moves from a region in space of low importance (where it is less likely to score in the detector) to a region of high importance, it is split in more particles, based on user–specified importance ratios. The particle's weight is adjusted accordingly to preserve physics. Similarly, as a particle moves from a region of high importance to low importance it may be killed in an unbiased manner to avoid wasting computational time on particles that may not score in the detector [5].

For our case, the 10.16–cm of polyethylene shield was divided into three sub–regions to determine the importance of the neutrons as they move through the shield and towards the detector region. Based on their population it was observed that the population of neutrons dropped by a factor of two almost half–way through the shield, it had dropped by a factor of almost four ¾ of the way through the shield. Therefore, for the geometry splitting case, the importance of neutrons in the region until 5.08 cm in to the shield was one, between 5.08 cm and between 7.62 cm it was two, and beyond 7.62 cm in the shield it was four. Thus, neutrons would double in number after 5.08 cm and then double again after 7.62 cm in the shield.

6.4 Results and Analysis

This section first provides a comparison of PHDs and TOFs for the source biasing case, the geometry splitting case, and the source biasing with geometry splitting case, all with the analog MCNPX–PoliMi case. It then discusses the improvement factor; this is the factor by which a user can reduce the MCNPX–PoliMi simulation runtime.

6.4.1 Comparison of Pulse Heights Distributions and Time–of–Flights

To motivate the need of developing a separate post–processing algorithm for nonanalog MCNPX–PoliMi, comparisons of simulations between analog and nonanalog PHD have been made using MPPost which does not account for particle weights.

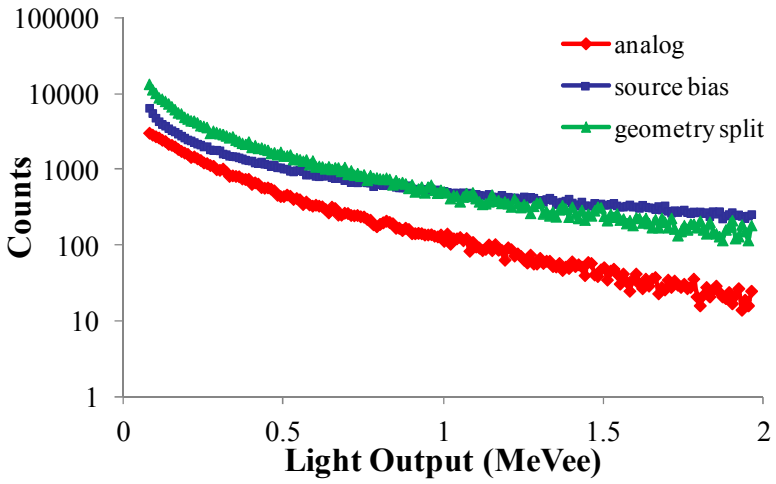


Figure 6– 4: Pulse height distributions for MCNPX–PoliMi analog, source bias and geometry split using MPPost.

As seen in Fig. 6–4, the nonanalog MCNPX–PoliMi data post–processing using the traditional MPPost inaccurately calculates the PHD because it does not tally using the appropriate weights. The MPPost tallies by adding unity for each score instead of the correct weight.

Next, comparisons of the VRPost post–processed nonanalog MCNPX–PoliMi case with the analog MCNPX–PoliMi case are presented. For each particle detected, its pulse height is first compared with the liquid scintillation detector threshold, which was 70 keVee for this case. Following this, if the pulse height is greater than the threshold, the pulse scores in its respective pulse height bin. The time of the first scatter in a pulse marks the TOF, the corresponding TOF bin is scored. In the analog case, the bin tally increases by one, whereas in the nonanalog case the bin tally increases by the weight of the pulse. The resulting analog and nonanalog results are shown in Fig. 6–5 through Fig. 6–10.

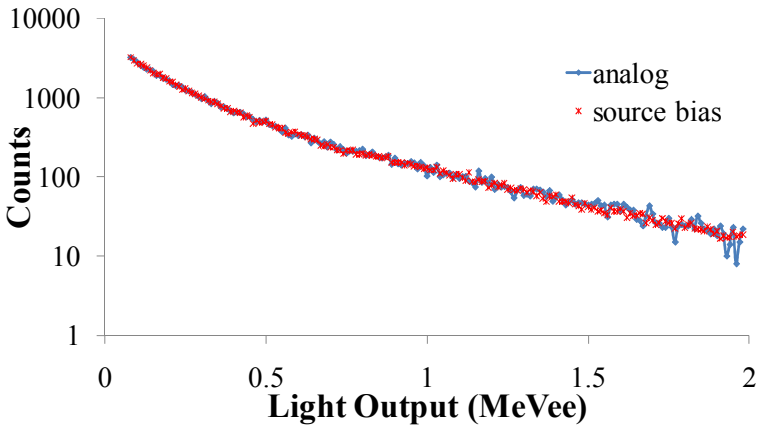


Figure 6– 5: Source bias and analog MCNPX–PoliMi PHDs post–processed with VRPost

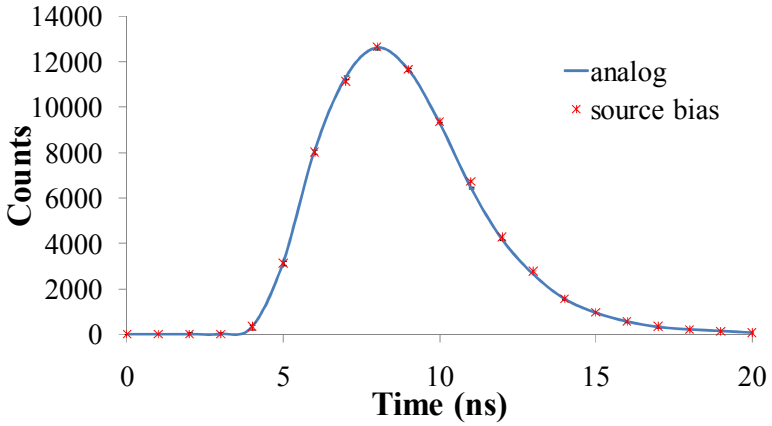


Figure 6– 6: Source bias and analog MCNPX–PoliMi TOF curves post–processed with VRPost

The weight distribution of particles has been shown in Fig. 5–9. As expected, in the given energy range from 0 to 15 MeV most high–energy neutrons (greater than 5 MeV) are produced with an increased biased probability and thus have smaller weights. These high energy neutrons are also more likely to pass through the polyethylene shield and enter the detector. Larger weights generally correspond to lower energy neutrons, which have a much lower distribution in the nonanalog case (as compared to the analog case). Due to

their low energies, few high weight neutrons leak through the shield; as a result, the likelihood of their detection is small.

In Fig. 6–5 and Fig. 6–6, the PHD and TOF plots obtained using source biasing show good agreement with the analog case. Furthermore, the nonanalog case using source biasing in the PHD plot converges to the analog results and has reduced stochastic errors. Due to the source biasing technique, which is sampled from modified flat particle distribution (Fig. 5–9), the convergence is good throughout the PHD for the nonanalog case, whereas for the analog case, the convergence is faster for the lower light–output bins than for the higher light–output bins (corresponding to higher energies).

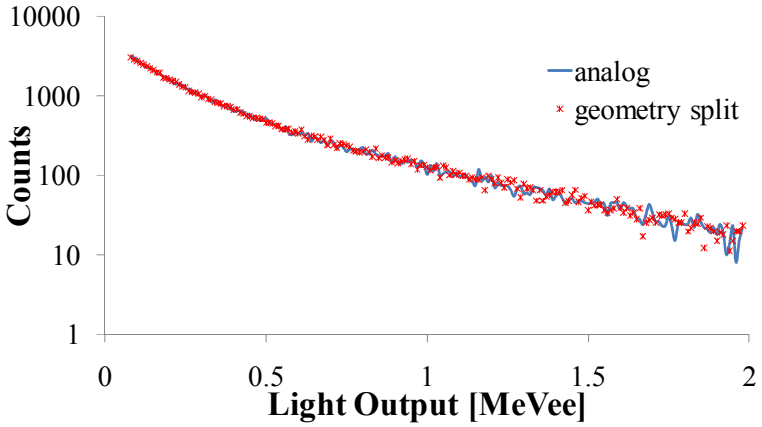


Figure 6– 7: Geometry split and analog MCNPX–PoliMi PHDs post–processed with VRPost

In Fig. 6–7 and Fig. 6–8, there is good agreement between the analog and the geometry split MCNPX–PoliMi simulations. However, comparing the PHD in Fig. 6–7 with that in Fig. 6–5, we can infer that geometry splitting does not lead to a uniform convergence throughout the pulse height spectrum. Nonetheless, it is shown in Fig. 6–11 that geometry splitting does lead to some improvement in the desired simulation run time.

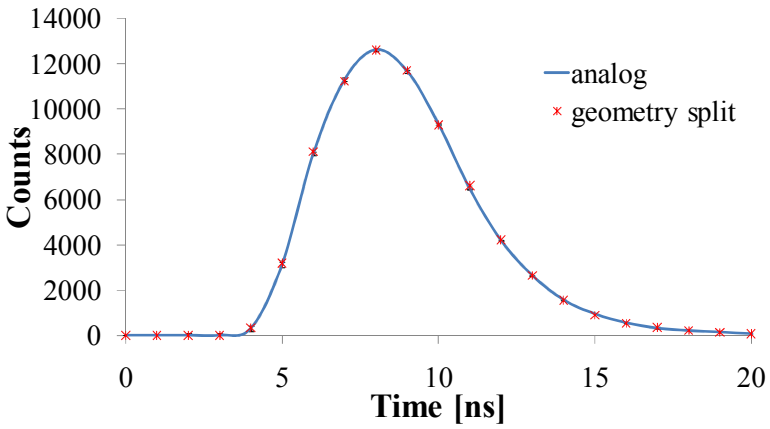


Figure 6– 8: Geometry split and analog MCNPX–PoliMi TOF curves post–processed with VRPost

In Fig. 6–9 and Fig. 6–10, results for the case which combined both the variance reduction techniques (source biasing and geometry splitting) are shown. The nonanalog PHD plot shows good agreement with the analog simulation. The TOF plot also shows good agreement. Due to the inclusion of the source biasing technique, we see that the PHD plot in Fig. 6–9 shows a faster convergence than the analog case for higher light outputs. The same trend is seen in the source biasing only case in Fig. 6–5.

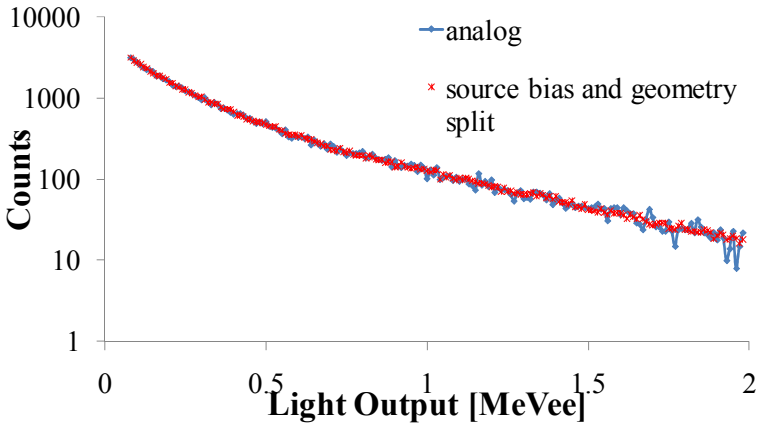


Figure 6– 9: Combined source bias and geometry split case with analog MCNPX–PoliMi post–processed using VRPost

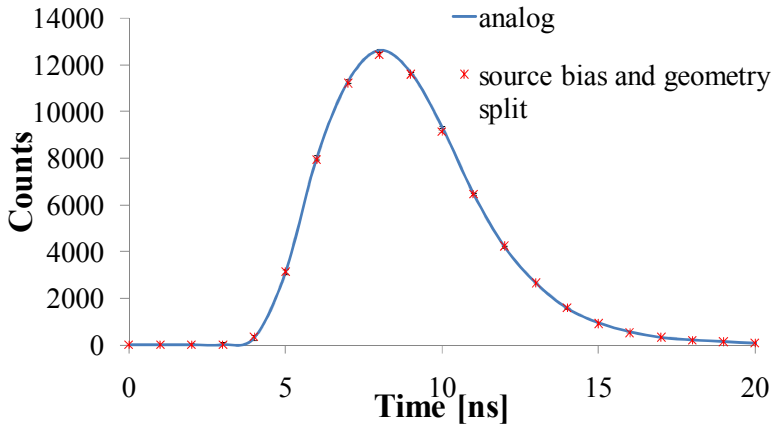


Figure 6– 10: Combined source bias and geometry split TOF curve with analog MCNPX–PoliMi post–processed with VRPost.

6.4.2 Speedup Determined by Improvement Factors

As discussed in Section 6.2, improvement factors are calculated for all bins for both the PHDs and the TOF curves for the three nonanalog cases discussed above. These results are shown in Fig. 6–11 and Fig. 6–12.

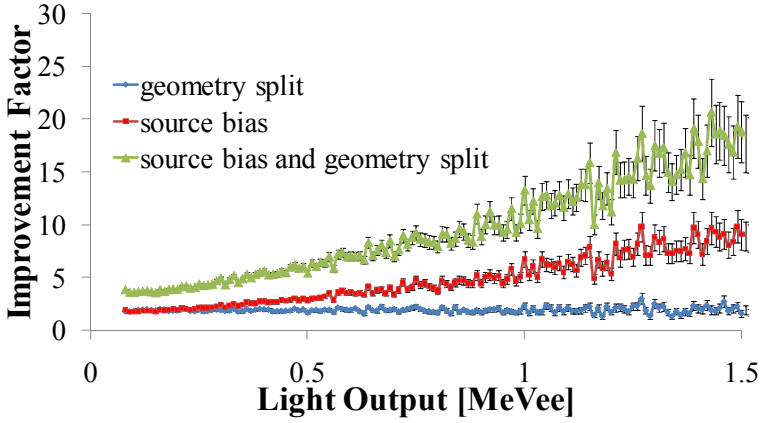


Figure 6– 11: Improvement factors for the nonanalog PHD simulations.

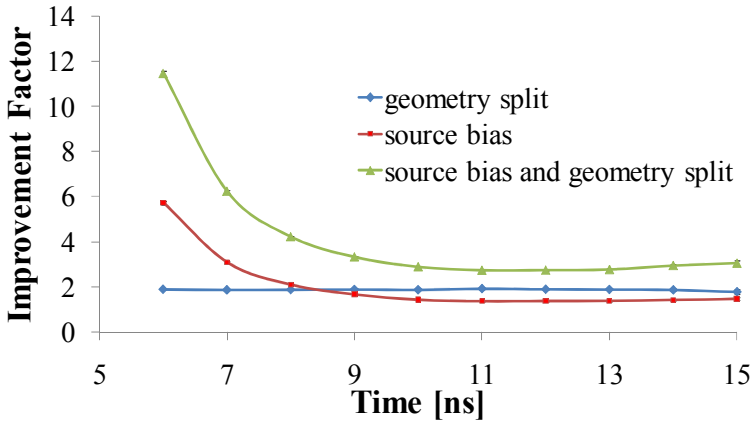


Figure 6– 12: Improvement factors for the nonanalog TOF simulations.

Figures 6–11 and 6–12 show that the best speedup is attained by combining the source biasing and geometry splitting techniques. For this case, the factors of improvement range from 5 at lower light outputs to greater than 25 at higher light outputs. From the improvement factor of the PHDs in Fig. 6–11, it is evident that source biasing provides faster convergence for higher light output bins (from faster neutrons). The geometry splitting method, in Fig. 6–11, changes the particle population based on their spatial importance only (as particles move in one direction, from the source towards the detector), with no regard to the energy of neutrons. Hence, the flat distribution of *I* (factor of approximately 2) as a function of light output (dependent on the energy variable) is seen. The combination of both techniques changes the particle distribution based on energy (source biasing) and spatial importance (geometry splitting), thus, gives the best results.

6.5 Conclusion

Liquid scintillation detectors are helpful in giving pulse height and time–of–flight information; however, MCNPX–PoliMi simulations of these need to be performed in fully analog mode and thus cannot utilize traditional MCNP variance reduction techniques. In this chapter it is shown that MCNPX–PoliMi simulations can be performed in nonanalog mode using variance reduction techniques to produce neutron detector response, but with the condition that the particle weight does not change inside the detector. The nonanalog MCNPX–PoliMi postprocessor, VRPost, tallies the pulses by their weights.

Comparisons of pulse height distributions and time–of–flight curves are made for nonanalog and analog MCNPX–PoliMi simulations. For the nonanalog cases: source biasing, geometry splitting with Russian roulette, and a combination of both are simulated. It is found that nonanalog results agree well with the analog cases. Furthermore, improvement factors are calculated to analyze the speedup in simulation time. It is found that source biasing and geometry splitting combination simulation provides the best results. Source biasing helps in modifying particle distributions for higher energies to obtain more scores, whereas geometry splitting maintains the same population of Monte Carlo neutrons throughout the shield. Neutron weights are adjusted as the neutron population is modified.

In the next chapter, multi–detector problems using MCNPX–PoliMi simulations run in nonanalog mode for correlated fission sources are discussed. These results are verified by laboratory measurements.

6.6 References

[1] S. A. Pozzi, E. Padovani, and M. Marseguerra, "MCNP–PoliMi: A Monte Carlo Code for Correlation Measurements," *Nuclear Instruments and Methods to Physics research A*, **513**, 550 (2003).

[2] S. A. Pozzi, E. Padovani, M. Flaska and S. D. Clarke, "MCNP–PoliMi Post–Processing Code Version 1.9," ORNL/TM–2007/33 (2007).

[3] S. Prasad, S. D. Clarke, S. A. Pozzi and E. W. Larsen, "Time–Dependent Neutron Detector Response Simulation for Shielded 252Cf," *Nuclear Science Symposium Conference Record (NSS/MIC)*, Knoxville, Tennessee, 30 October – 6 November 2010, 10.1109/ NSSMIC.2010.5873894, 911 (2010).

[4] L. L. Carter and E. D. Cashwell, "Particle–Transport Simulation with the Monte Carlo Method," *ERDA Critical Review Series TID 26607*, U.S. Department of Energy, Washington D.C. (1975).

[5] "MCNP—A General N–Particle Transport Code, Version 5, Volume I: Overview and Theory," X–5 MONTE CARLO TEAM, LA–UR–03–1987, Los Alamos National Laboratory (2003).

[7] S. Prasad, A. Enqvist, S. D. Clarke, S. A. Pozzi and E. W. Larsen, "A Response Function Study for Energy to Light Conversion in Organic Liquid Scintillators," *Transactions of the Institute of Nuclear Materials Management 52nd Annual Meeting*, Palm Desert, California, 17 – 21 July, 2011 (2011).

[8] D. Pelowitz (ed.), "MCNPX User's Manual, version 2.6.0," LA–CP–07–1473, Los Alamos National Laboratory (200

Chapter 7
Nonanalog MCNPX–PoliMi for Correlated Sources

7.1 Introduction

In the previous chapter, uncorrelated responses such as pulse height distributions and time–of–flight were studied. We showed that nonproliferation problems can be divided into parts that must be run in analog MCNPX–PoliMi mode, and parts that can use variance reduction techniques. Such a targeted approach can lead to significantly reduced computation times for complex problems.

In this chapter, a time–dependent correlated response with nonanalog MCNPX–PoliMi is studied. In the case of correlated response such as cross–correlations, the efficiency of detection is extremely low. The likelihood of recording a cross–correlation event is extremely small, even after accounting for the solid angle of the problem geometry and the intrinsic efficiency of the detector. If the probability, p, that one neutron from a detector will score is small, then the probability that two particles will score is on the order of p^2, which is much smaller than p. For this reason, Monte Carlo simulations of these events will be costly; only a very small percentage (on the order of p^2) of Monte Carlo fission events will lead to two neutrons scoring in detectors. In this chapter, a time–dependent neutron cross–correlation response for liquid scintillation detectors, utilizing traditional MCNP variance reduction techniques, is studied. These techniques are used to mitigate the reduced efficiency due to small solid angles subtended

by the detector with respect to the source. Variance reduction techniques in MCNPX–PoliMi increase the neutron population closer to the detector. The method is tested using a bare ^{252}Cf source and a polyethylene shielded ^{252}Cf source.

7.2 Time Cross–Correlation with Nonanalog MCNPX–PoliMi

Cross–correlation measurements give a distribution of the differences in the times of detection of two or more different particles that reach two different detectors. An example setup for a cross–correlation detector response, in which a shield hiding a fission source is shown with two detectors, is shown in Fig. 7–1. The detection times of the three pulses generated in the detectors are given by t_1, t_2, and t_3 ($t_1 < t_2 < t_3$).

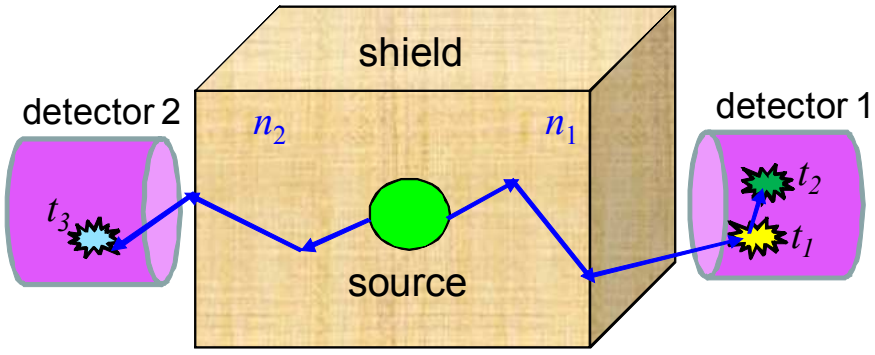


Figure 7– 1: An example setup for a cross–correlation scintillation detector response.

The order of subtraction is fixed, which can lead to negative times. For instance, if detector 2 pulses are being subtracted from detector 1 pulses, the difference between t_1 and t_3 will give a negative answer if t_1 occurs before t_3. Cross–correlation distributions are unique to the fission process, as they require multiple correlated particles emitted from an event. Only neutron–neutron cross–correlation measurement simulations will be

the focus of this discussion, since the neutron scintillation detector response is nonlinear, as discussed in Ch. 2.

7.2.1 Cross–Correlation Calculation

As explained in Ch. 3, MCNPX–PoliMi simulates nuclear nonproliferation measurements problems for complex scenarios. It models spontaneous and induced fission events in detail for different isotopes [5]. It has built–in source correlations for multiplicity–dependent energy distribution and light fission fragment dependent particle flight direction [5].

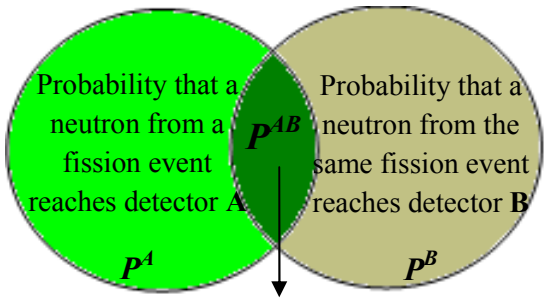
When we apply variance reduction techniques, we must ensure that the final answer remains unbiased. This is done by ensuring that the product of the true population, p_0 and the original weight w_0 (which is simply unity) is preserved by the new user–specified population, $\hat{p}$ and the corresponding weights $\hat{w}$:

$$w_0 \cdot p_0 = \hat{w} \cdot \hat{p} \quad . \tag{1}$$

For cross correlation measurements, two or more detectors are needed. In this study, only problems with two detectors are studied (but the method can be extended to more detectors with a similar approach). A count is produced when neutron pulses from the same fission event are recorded by both detectors.

In MCNPX–PoliMi, the energy distribution of neutrons from a fission event is dependent on the fission–multiplicity (the number of neutrons released in the fission event). Once the multiplicity has been sampled, the energy of the neutrons is sampled independently (from each another) from a distribution which is isotopic and multiplicity dependent [5]. Furthermore, the distribution of the flight directions of neutrons emitted

from a fission event depends on the flight direction of the lighter fission fragment emitted. Once the lighter fission fragment's flight direction has been sampled isotropically, the flight directions of the neutrons from that fission event are sampled independently of one–another, but dependent on the direction of the lighter fission fragment. As the sampling is done independently (there are no conditional probabilities), the following relationship in Fig. 7–2 can be shown:



P^{AB} = Probability that a neutron from a fission reaches detector **A** *and* another neutron from the same event reaches detector **B.**

$$= P^A \times P^B. \tag{2}$$

Figure 7– 2: An illustration of neutron probabilities from the same fission being detected by two detectors.

If we apply the relationship shown in Eq.(1) to the individual probabilities P^A and P^B, we can rewrite Eq. (2) as:

$$\begin{aligned} P^{AB} &= P^A \times P^B \\ &= (\hat{w}^A \cdot \hat{p}^A)\,(\hat{w}^B \cdot \hat{p}^B) = (\hat{w}^A \cdot \hat{w}^B)(\hat{p}^A \cdot \hat{p}^B). \end{aligned} \tag{3}$$

Thus, the product of the individual modified probabilities for the detectors is simply the combined modified probability. This modified combined probability should be weighed by the product of the weights of the neutrons reaching each detector. Thus each time cross correlation count will be incremented by the combined weight:

$$\hat{W} = (\hat{w}^A \cdot \hat{w}^B) \,. \tag{4}$$

This technique will only work if each neutron pulse is produced due to contributions from a single neutron. For a simple case we proceed with this *assumption* because the probability that two or more neutrons from the same fission will enter the same detector is much smaller than the likelihood of a single neutron contributing to a pulse (given the small solid angles subtended by the detectors in the problems considered here). This assumption will be violated in the event of a high splitting ratio, in which case it becomes more likely for daughters from different particles to enter the same detector within the same PGT.

7.2.2 Symmetric Case with Geometry Splitting

We test the above proposed methodology for the simple case in which the source–detector distance on each side of the source is 30 cm. The source and the detector are unshielded and contained in vacuum space as shown in Fig.7–3.

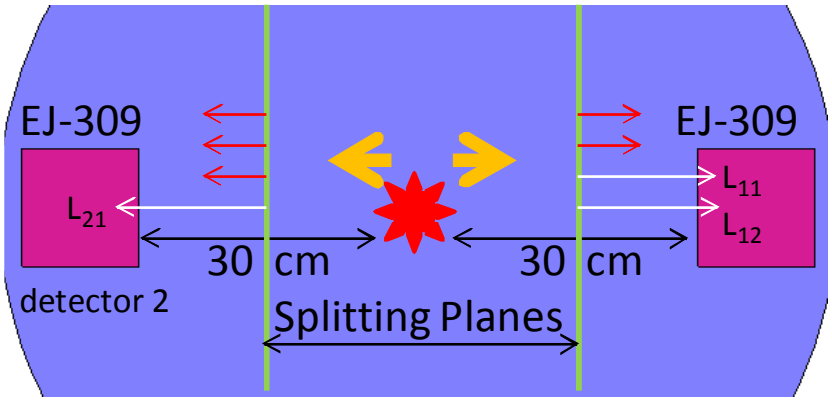


Figure 7– 3: An illustration of nonanalog cross–correlated tally setup in MCNPX–PoliMi

In the above case, as particles travel half–way towards the detector (15 cm), they split into four daughter particles. On the right hand side, two of the four split daughter particles enter detector 1. On the left hand side, only one of the four split daughter particles enters detector 2. Thus, the following pulses are formed.

Table 7– 1: List of pulses contributing to the tally for the example case in Fig. 7–3.

#	Pulse Heights [MeVee]	Weight
1	L_{11}	w_{11}
2	L_{12}	w_{12}
3	L_{21}	w_{21}

In Table 7–1, we can see that there are two light pulses formed in detector 1, L_{11} and L_{12}, whereas only one pulse formed in detector 2, L_{21}. The times of detection of these pulses are t_{11}, t_{12}, and t_{21} respectively. Each of the detector 1 pulses can be paired with the detector 2 pulse to form the following cross–correlated pairs shown in Table 7–2.

Table 7– 2: Cross–correlated events for the example case in Fig. 7–3.

#	Delta–Time [ns]	Weight
1	$t_{11} - t_{21}$	$w_{11} \times w_{21}$
2	$t_{12} - t_{21}$	$w_{12} \times w_{21}$

Each of the pairs is given a product of the weights as explained by Eq. 4. MCNPX–PoliMi simulations are performed to test this methodology. The MCNPX–PoliMi model for a symmetric bare and shielded ^{252}Cf source is shown in Fig. 7–4 and Fig. 7–5. The ^{252}Cf point source is an MCNPX–PoliMi built–in anisotropic spontaneous fission source (IPOL 1). The detector is a cylinder with a radius of 6.34 cm and a length of 12.51 cm. The chemical composition of the EJ–309 liquid is 54.8% hydrogen and

45.2% natural carbon by atom. The ^{252}Cf source is modeled as a pure neutron source. The photomultiplier tube and the detector casing have not been included for simplicity and ease of testing variance reduction techniques. The EJ–309 liquid scintillation detectors are placed at 30 cm on both sides of the source as shown in Fig. 7–4. The source and the detectors are surrounded by vacuum. In Fig. 7–4, the bare ^{252}Cf source is on the left, and the ^{252}Cf shielded by 10.16–cm–thick polyethylene on each side (20.32 cm or 8 inches total shielding thickness) is on the right.

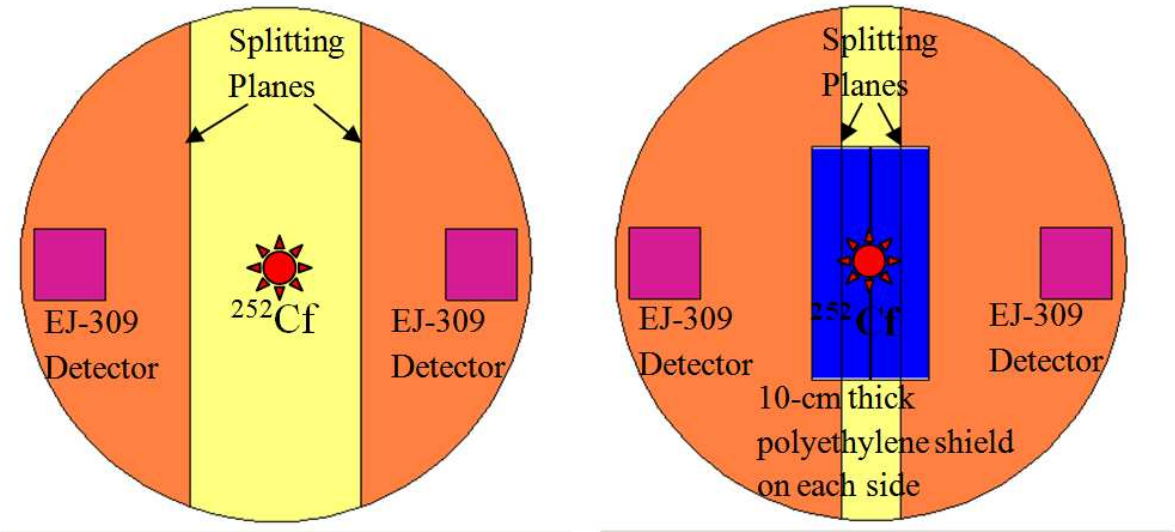


Figure 7– 4: Bare ^{252}Cf (left) and Polyethylene shielded (right) correlated ^{252}Cf fission source with two EJ–309 liquid scintillation detectors.

The cross–correlation simulation results for the above setup are presented for the analog and the geometry split cases of MCNPX–PoliMi in Fig. 7–5 and Fig. 7–6. The polyethylene–shielded case is normalized to the number of source particles run for the bare case. In Figs. 7–5 and 7–6, the y–axis scales indicating the counts differ by an order of magnitude. This is expected, since polyethylene is an effective moderator, which slows fission neutrons to energies below the detector threshold. For the case shown in Fig. 7–6,

a very thick polyethylene shield of 10.16–cm has been modeled on each side. The peak of the cross–correlation simulation curve for the bare ^{252}Cf occurs at about 15,000 counts; in contrast to 500 counts in the case of polyethylene shielded ^{252}Cf. This is a difference of a factor of approximately 30 between the two cases. Speedup factors were calculated using the same improvement factors as discussed in Ch. 5 and Ch. 6.

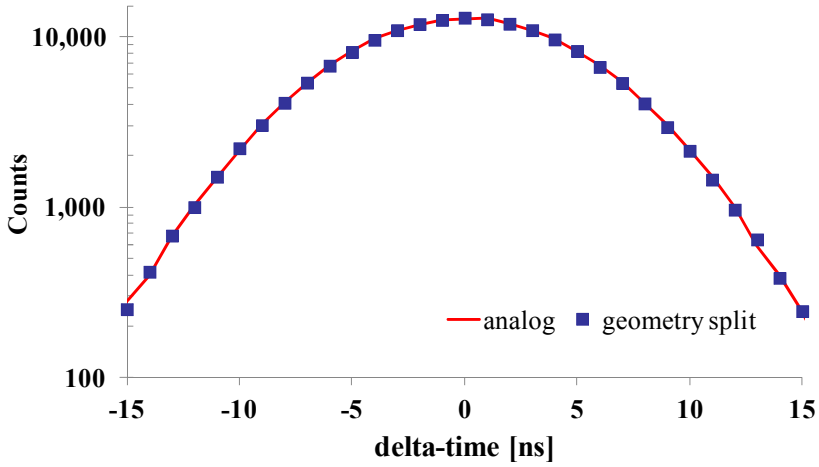


Figure 7– 5: Bare ^{252}Cf cross–correlation comparison for analog and geometry split simulations with MCNPX–PoliMi. Speedup factor 11.

As seen from Fig. 7–5 and 7–6, there is good agreement between analog and geometry split (nonanalog) MCNPX–PoliMi simulations. The average speedup for both of these cases was calculated by taking the ratio of the figure–of–merits [10]. The average speedup for the case of bare ^{252}Cf source is a factor of 11, whereas for the polyethylene shielded ^{252}Cf, the speedup is a factor of 4.6.

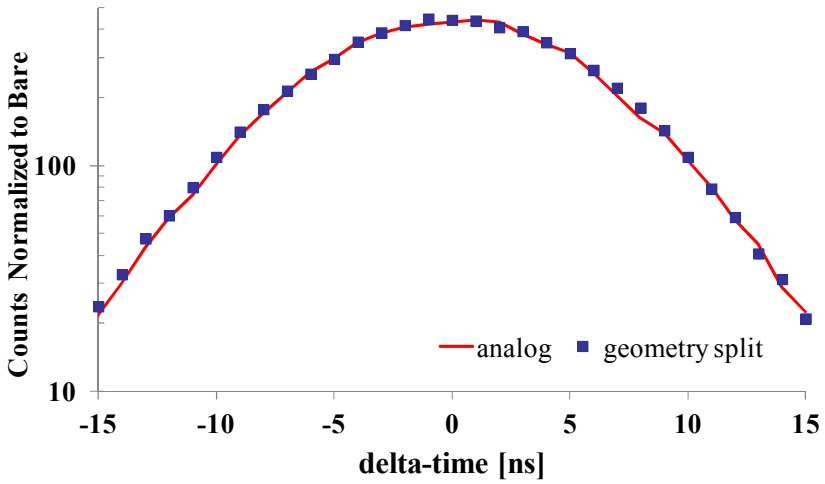


Figure 7– 6: Polyethylene–shielded ^{252}Cf cross–correlation comparison for analog and geometry split simulations with MCNPX–PoliMi. Speedup factor 4.6.

This difference in computational speedup can be explained by the splitting plane positions shown in Fig. 7–4. In the bare ^{252}Cf case, the splitting planes are simply placed in a vacuum space, thus after splitting are transported to the detector where they interact. In the the polyethylene–shielded case, the splitting plane is placed inside the shield, thus after splitting the daughter particles first interact in the thick shield and are then transported to the detector. It is shown above that the nonanalog Monte Carlo as tested by geometry splitting with MCNPX–PoliMi can be used to simulate neutron cross–correlation simulations for a symmetric case.

7.2.3 Symmetric Case with Source Biasing

Next, we implement the same source biasing technique as illustrated in Fig. 5–9 to check its compatibility with MCNPX–PoliMi. The cross–correlation curve is presented in Fig. 7–7.

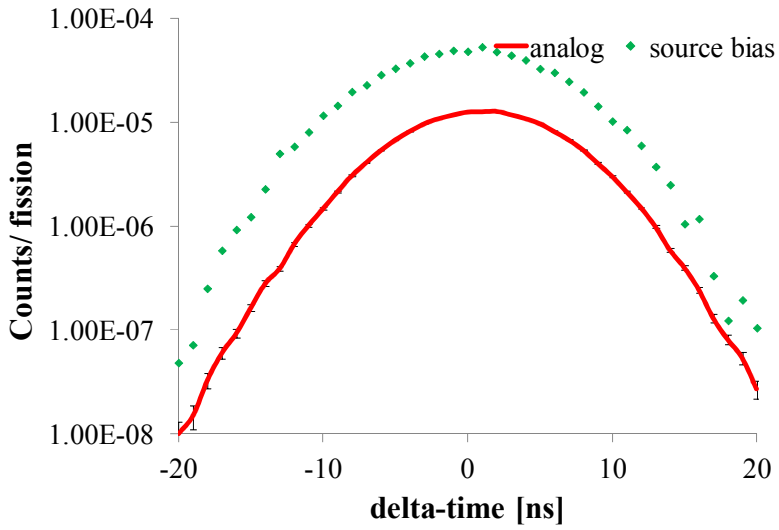


Figure 7– 7: Cross–correlation curves for analog and nonanalog MCNPX–PoliMi cases.

As seen above, the nonanalog MCNPX–PoliMi simulations do not converge to the correct answer given by the analog MCNPX–PoliMi simulation. The nonanalog curve was calculated using the same methodology as given in Table 7–2. Upon examination of the MCNPX–PoliMi data file it was found that the source–biasing method did not assign weights as expected (as shown in Fig. 5–9) when used with built–in correlated sources . This is because the developers of MCNPX–PoliMi did not intend the use of variance reduction techniques with simulations.

When using source–biasing in MCNP, each particle is assigned a weight depending on the specified biasing function. However, when specified in MCNPX–PoliMi for the built–in sources, source–biasing assigns the same weight to all the particles emitted in a fission event on an unknown basis. Furthermore, MCNPX–PoliMi weighs the entire fission event and not the individual particles. Thus, the methodology shown by Eq. 4 and in Table 7–2 will fail. In this scenario, all particles inherit the weight of the source event. This is because, once the fission event has been sampled with a given

probability and weight, it is certain (with a probability of unity) that all the particles emitted from that fission will exist.

Thus, the probability that a particle will exist is simply given by the probability that the fission will occur. Therefore, the probability that a cross–correlated pair will occur is simply given by the probability that the fission will occur (not by the probabilities of individual particles). Hence, the cross–correlated tally for the source–biasing case, as it is presently functioning in MCNPX–PoliMi, should only be weighed by a single weight (not the product of weights as shown in Eq.4) which is the weight of the source or fission event. The cross–correlation tally was modified to include this methodology for source–biasing, the results are shown in Fig. 7–8.

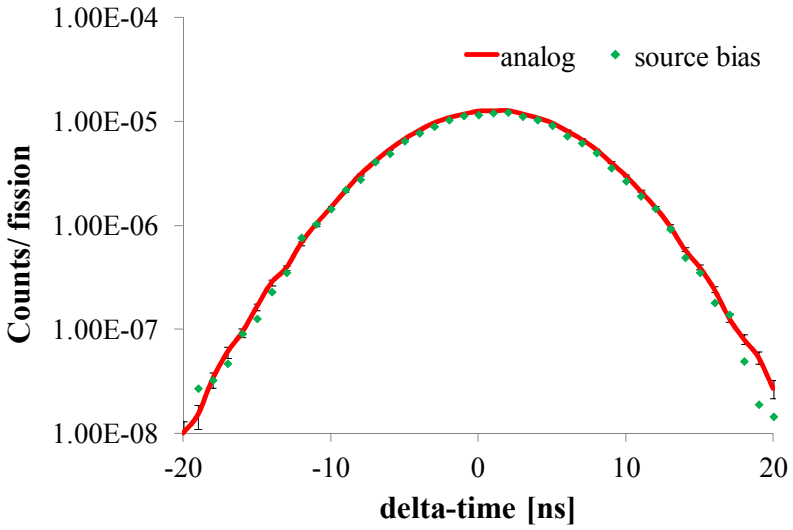


Figure 7– 8: Cross–correlation curves for analog and nonanalog MCNPX–PoliMi cases using a modified tallying technique.

As is evident from the above figure, the modified methodology proposed above regarding weighing cross–correlated pairs by the source or fission event gives a much

better agreement. However, source–biasing and its modified methodology will not be further discussed in this thesis since the weight of the source events is being determined on an unknown basis. In this thesis, we are interested in modifying the transport of the individual particles in a way such that the figure–of–merit for the cross–correlation tally is increased. This goal is unachievable given the present limitation of source–biasing in MCNPX–PoliMi. The MCNPX–PoliMi developers have been informed about this limitation, and they may find it helpful to implement source–biasing in MCNPX–PoliMi such that it is able to correctly bias individual particles.

7.2.4 Asymmetric Case with Geometry Splitting

Next, a bare ^{252}Cf source is placed with detector 2 much closer than detector 1, as shown in Fig. 7–9. Additionally, after going through the splitting planes the particles will split into 8 daughter particles. It is evident from Fig. 7–10, that the analog curve is being under–predicted by the geometry–split simulation.

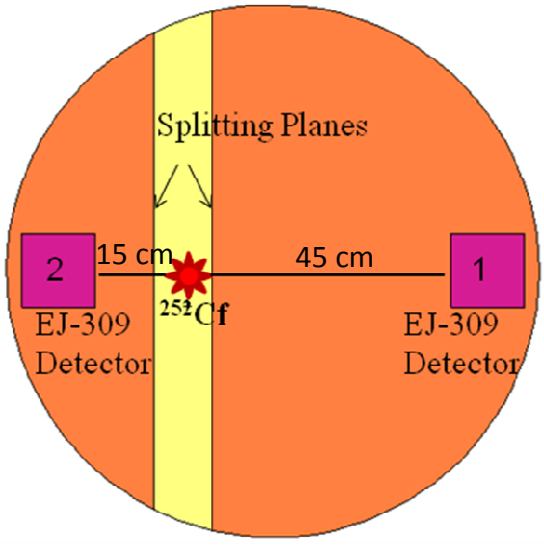


Figure 7– 9: Asymmetric setup with a ^{252}Cf source and two detectors using 1:8 splitting ratio at splitting planes.

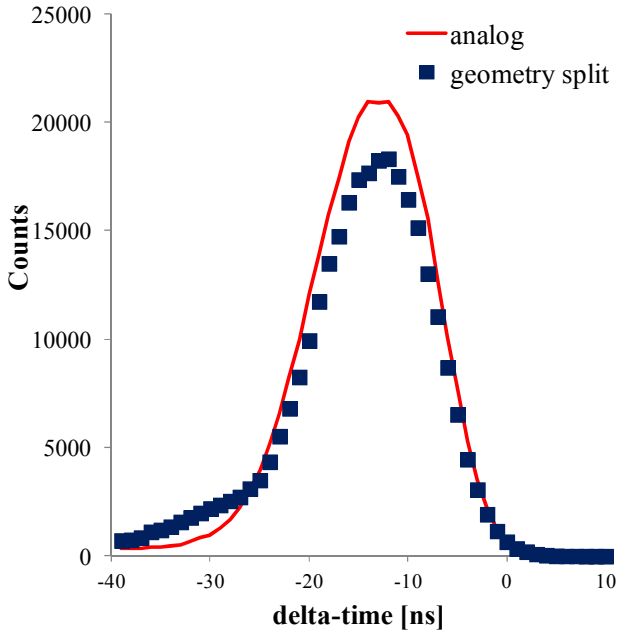


Figure 7– 10: Cross–correlation tally shown for the asymmetric setup utilizing analog and nonanalog MCNPX–PoliMi simulations. The splitting ration was greatly increased but the tally mechanism was kept simplified.

The disagreement in the above figure can be explained by the simplified tally mechanism that was developed in Section 7.2.1. An assumption was made that only one neutron from a fission event can contribute to a pulse formed in the detector. This assumption is violated when the splitting ratio increases: it becomes increasingly likely that daughters from two different neutrons of a fission event will enter the same detector within the same PGT (time–window), as shown in Fig. 7–10. This erroneous tally mechanism is modified to accommodate for the *multi–particle* behavior in the next section.

7.3 Multiple Particle Pulses in Nonanalog MCNPX–PoliMi

A method to correctly account for daughters from different neutrons, of the same fission event, entering the same detector, at similar times (within the same PGT) is formulated below. Consider an example case shown in Fig. 7–11.

7.3.1 Incorporating Multiple Particles

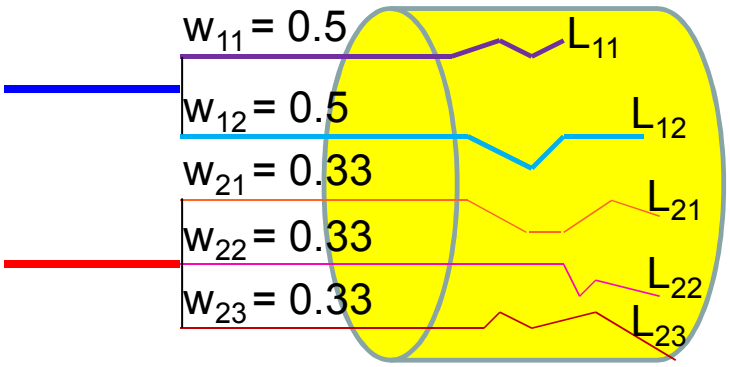


Figure 7– 11: An example case of split daughters from two different neutrons entering the same detector within the same PGT.

In Fig. 7–11, two different neutrons (red and blue) from the same fission event have split and entered the detector at the same time. The blue neutron has split into two different daughters (each with a weight of $^1/_2$), and the red neutron has split into three different daughters (each with a weight of $^1/_3$). Each daughter of a neutron will create its own independent track in the detector to represent a possible path that the original daughter could have taken. The daughters represent the different possibilities of the original neutron; therefore, it is clear that their tracks are mutually exclusive: that is, both daughters from the same neutron cannot occur at the same time. Hence, daughters from the same neutron cannot contribute to the same pulse. However, daughters from different

neutrons that also enter the same detector can contribute to the same pulse. Therefore, in Table 7–3, L_{11} and L_{12} cannot occur together, similarly, L_{21}, L_{22} and L_{23} cannot occur together. But, they can occur with the daughters of other neutrons, as shown below:

Table 7– 3: List of pulse formed in the detector for the example case.

#	Case	Pulse Height	Pulse Weight
1	w_{11} and w_{21}	$L_{11} + L_{21}$	$p_1 = w_{11} \times w_{21} = 0.167$
2	w_{12} and w_{21}	$L_{12} + L_{21}$	$p_2 = w_{12} \times w_{21} = 0.167$
3	w_{11} and w_{22}	$L_{11} + L_{22}$	$p_3 = w_{11} \times w_{22} = 0.167$
4	w_{12} and w_{22}	$L_{12} + L_{22}$	$p_4 = w_{12} \times w_{22} = 0.167$
5	w_{11} and w_{23}	$L_{11} + L_{23}$	$p_5 = w_{11} \times w_{23} = 0.167$
6	w_{12} and w_{23}	$L_{12} + L_{23}$	$p_6 = w_{12} \times w_{23} = 0.167$

In the above table the light pulses can only combine if they fall within the same PGT (time–window). The pulse–weights for all six cases add up to unity to yield an unbiased tally. In general, the number of cases or the possibilities of pulses from daughter of different neutrons of the same fission event, N is simple given as:

$$N = \prod_{p=1}^{P} d_p \ . \tag{5}$$

Here, P is the number of neutrons of the same fission event that enter the same detector. The number of daughters for each neutron entering the detector is represented by d_p. Once the cases have been formed, a pulse will be formed only for those collisions of the daughters that occur within the PGT of that pulse (time window). The pulses are finally checked against the detector threshold to determine if they can contribute to the tally. The algorithm is summarized in Fig.7–11. It was implemented to modify the original version of the code. The new results are discussed in the following section.

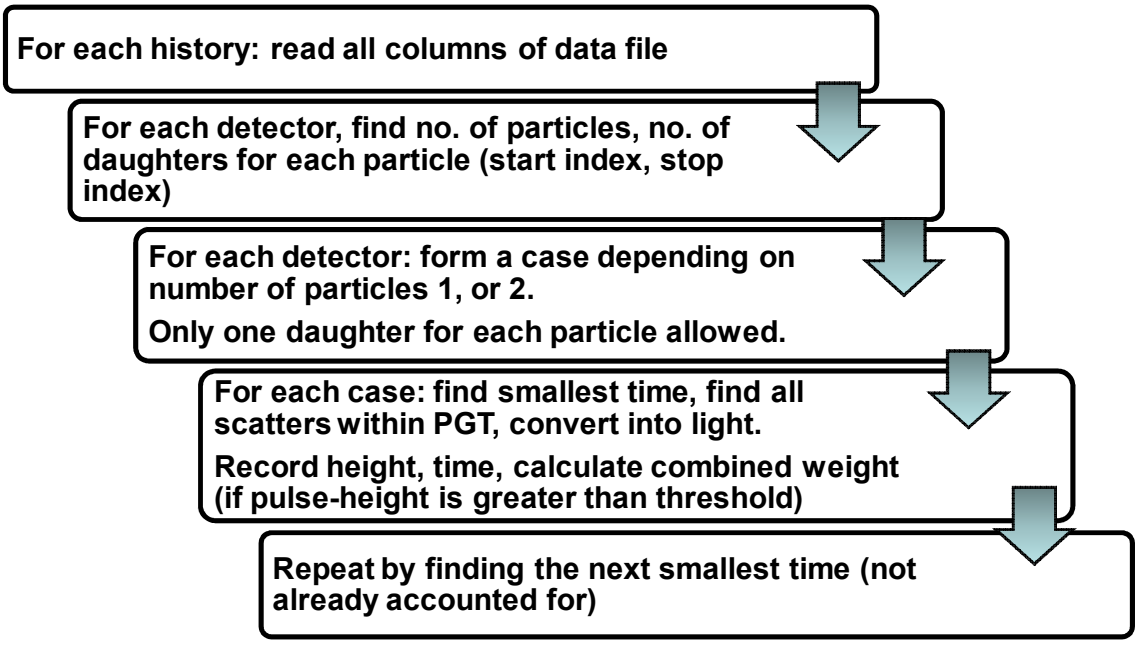


Figure 7– 12: Algorithm for incorporating multiple particle contribution in the pulses for 2 or more neutron entering the same detector.

The modified algorithm only includes up to two particles from the same fission event entering a detector, in the same pulse (if particles arrive within the same PGT). This is not exact: the algorithm will need to be modified to include more particles from a fission event for simulations in which large solid angles are subtended by detectors. However, the above algorithm is easy to modify to include the effect of more than two particles from a fission event entering the detector. In the simulations discussed in this thesis, the likelihood of three particles entering a detector within the same PGT is much smaller than one percent.

7.3.2 Asymmetric Case with Geometry Splitting

The MCNPX–PoliMi simulation for the setup shown in Fig.7–9 was re–tallied using the above modified algorithm to yield the results shown in Fig. 7–13.

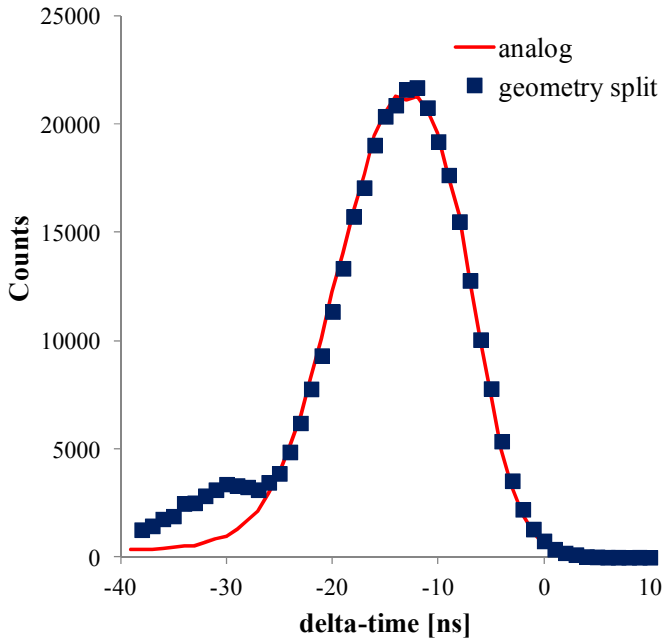


Figure 7– 13: Cross–correlation tally shown for the asymmetric setup utilizing analog and nonanalog MCNPX–PoliMi simulations. The splitting ration was greatly increased thus the tally mechanism was modified to incorporate the multi–particle behavior.

The results in Fig. 7–13 are a significant improvement over the results shown in Fig. 7–10. The nonanalog results agree with the analog results for most of the peak, except for the left side of the tail in which the nonanalog results greatly over–predict the analog solution. This error is a limitation of the post–processing methodology, in which the exact daughter numbers are not written in the data file. Therefore, the post–processing methodology determines the daughter number based on the number of scatters. If the number of scatters abruptly changes (by more than one scatter) the algorithm determines the following lines as belonging to the next daughter. The reader may look back at Table 3–1 to understand how the data is being written by MCNPX–PoliMi.

The above approximation will give erroneous answers when there is a highly scattering medium outside the detector. In this situation, a neutron can leave the detector,

scatter outside the detector, and return to the detector. Upon returning to the detector, the neutron will be incorrectly classified as a different daughter. This approximation will also not work where there is significant cross–talk between the detectors. That is, a neutron can enter one detector, deposit some energy, then leave that detector and enter the second detector. In the setup shown in Fig. 7–9, cross–talk is likely to occur, especially for neutrons entering detector 2 and then entering detector 1. These neutrons are incorrectly accounted for and will hence cause an error.

To eliminate the errors indicated above, it is necessary to add a column in the MCNPX–PoliMi data file that will indicate each daughter number explicitly. For cross talk events, the pulses created in the second detector should always be given a weight of unity, as their original weight has already been accounted for in the first detector.

In this section, a method was developed for tallying cross–correlation simulation that incorporated for multiple particle behavior. Next, we proceed to validate laboratory measurements with simulations and quantify speedup due to nonanalog techniques.

7.4 Speedup of Nonanalog MCNPX–PoliMi Simulation of Laboratory Measurements

Measurements were performed to validate simulation results for the setup shown in Fig. 7–14. The measurements were performed in the Detection for Nuclear Nonproliferation Group Laboratory (DNNG) at the University of Michigan.

7.4.1 Measurement Setup

A ^{252}Cf source emitting approximately 2.5×10^{5} neutrons per second (67,000 fissions per second), placed 30 cm from each detector was shielded with 5.08 cm (2 inches) of

lead and polyethylene shielding. A 1–μCi ^{137}Cs source was used for calibration, and a 12–bit, 250 MHz waveform digitizer was used for data acquisition.

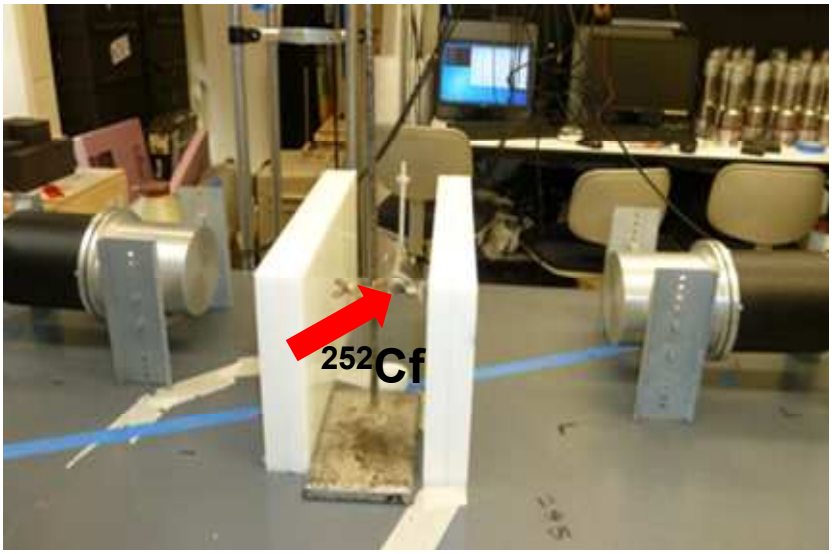


Figure 7– 14: Laboratory setup for the polyethylene–shielded ^{252}Cf and EJ–309 detectors.

The threshold for detection was determined to be 70 keVee (equivalent of 650 keV of neutron energy deposited on hydrogen). The acquisition window was 120 points long, with each point 4 ns. The pulses were discriminated between neutrons and gamma–rays using a standard charge integration method discussed in Ch. 2. The pulse shape discrimination (PSD) results are shown in Fig. 7–15. Neutrons interact with the nuclei in the scintillation detector and result in larger tails compared to gamma–rays, which interact with electrons. In Fig. 7–15, points corresponding to higher tail integral values for the same total integral value of the pulse are from neutrons; those points are found above the discrimination line. In Fig. 7–15 the ^{137}Cs data have been overlaid to verify that the pulses below are indeed gamma–rays.

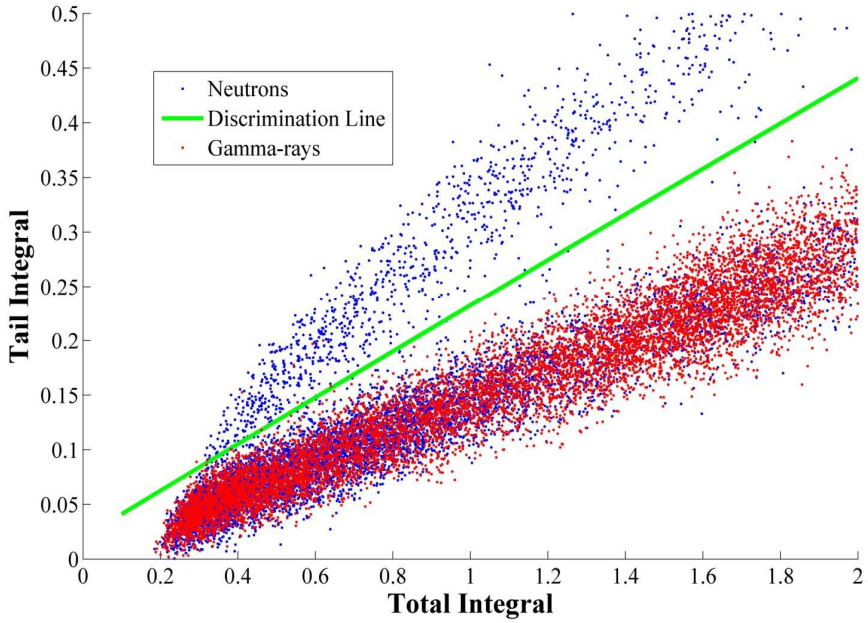


Figure 7– 15: Pulse shape discrimination between neutrons (blue dots above discrimination line) and gamma–rays (red dots below discrimination line)

7.4.2 Measurement Results

The cross–correlation results are shown in Fig. 7–16 through Fig. 7–18 for bare, lead–shielded, and polyethylene shielded cases. The neutron–neutron, neutron–photon, photon–neutron, and photon–photon time differences are shown. Bare and lead shielded measurements were taken for nearly one hour each, whereas the polyethylene measurement was taken for four hours. Therefore, the counts cannot be directly compared. As seen in Fig. 7–16, the photon–photon cross–correlation peak for a bare ^{252}Cf source is more than an order of magnitude greater than the neutron–neutron peak. This is because on average more gamma–rays come out of the ^{252}Cf fission event than neutrons. For a fission event, there may be 7 to 8 gammas–rays emitted on average, where as there may be only 3 to 4 neutrons emitted on average [11].

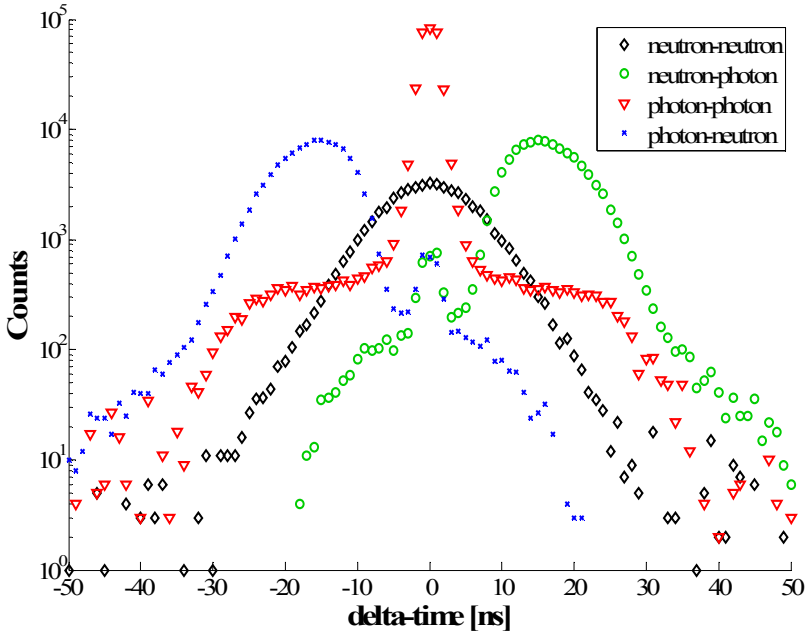


Figure 7– 16: Cross–correlation measurements with no shielding between the source and the EJ–309 detectors.

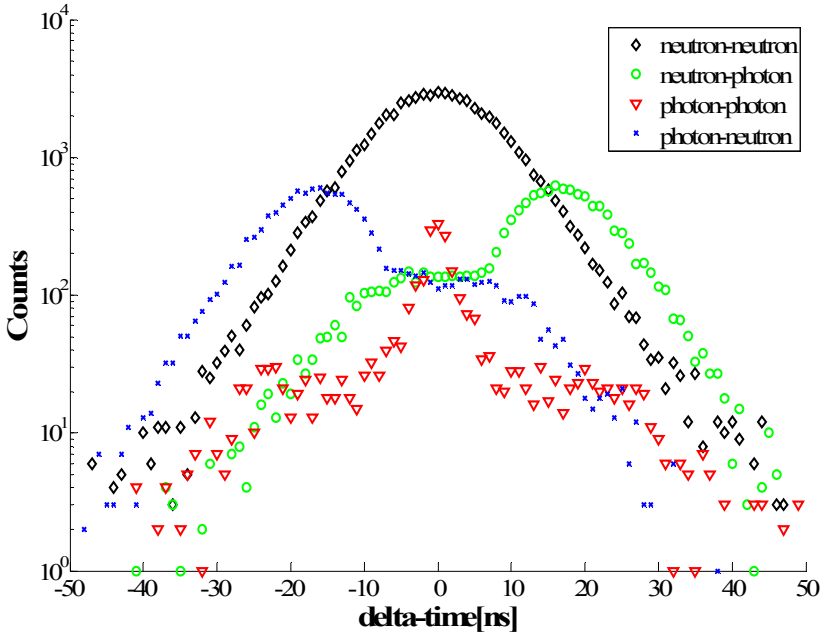


Figure 7– 17: Cross–correlation measurements with 5.08 cm thick lead shielding on both sides of the source and the EJ–309 detectors.

In Fig. 7–17, where the ^{252}Cf source is shielded by 5.08 cm of lead on both sides, the neutron–neutron peak is nearly an order of magnitude greater than the photon–photon peak. This is because lead, due to its high atomic number, is a very effective shield for gamma–rays, thereby causing the decrease in the magnitude of the photon–photon peak.

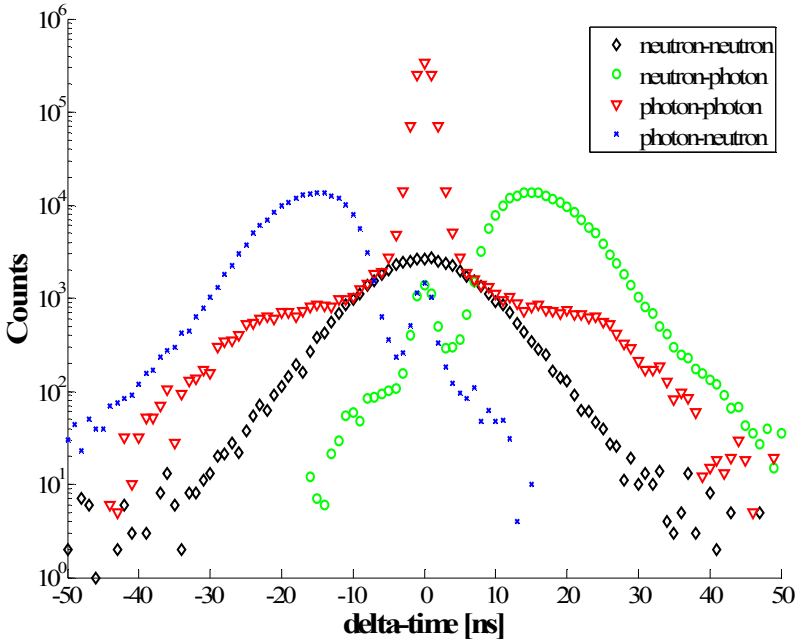


Figure 7– 18: Cross–correlation measurements with 5.08 cm of polyethylene shielding on both sides of the source and the EJ–309 detectors.

In Fig. 7–18, the cross–correlation curves from the 5.08–cm–thick polyethylene–shielded case are presented. Since polyethylene is a very good moderator of neutrons, the neutron–neutron peak is greatly reduced. In fact, this peak is nearly three orders of magnitude smaller than the photon–photon peak. In the next, subsection we simulate the measured results using nonanalog and analog MCNPX–PoliMi simulations.

7.4.3 MCNPX–PoliMi Nonanalog Setup

The measurement setup depicted in Fig. 7–19 contains an anisotropic point–like ^{252}Cf source placed 30 cm from the faces of each of the EJ–309 liquid scintillation detectors. In addition to the features shown in Fig. 7–19, the iron table on which the detectors rest, and the concrete floor are also modeled. The source is shielded by lead or polyethylene rectangular blocks that are 5.08 cm thick. The composition and the dimensions of the detectors are specified in Subsection 7.2.2. In addition to the detector, the detector casing and the PMTs have also been modeled. The energy–to–light conversion coefficients have been described in Eq. 5 of Ch 2. The orange boxes surrounding the detectors in Fig. 7–19 are lateral views of the cylinders containing the EJ–309 detectors. These cylinders merely serve the purpose of splitting particles as they enter the cylinder. For bare and lead shielded simulation the particles split 1:4, and for polyethylene shielded–case particles split 1:2.

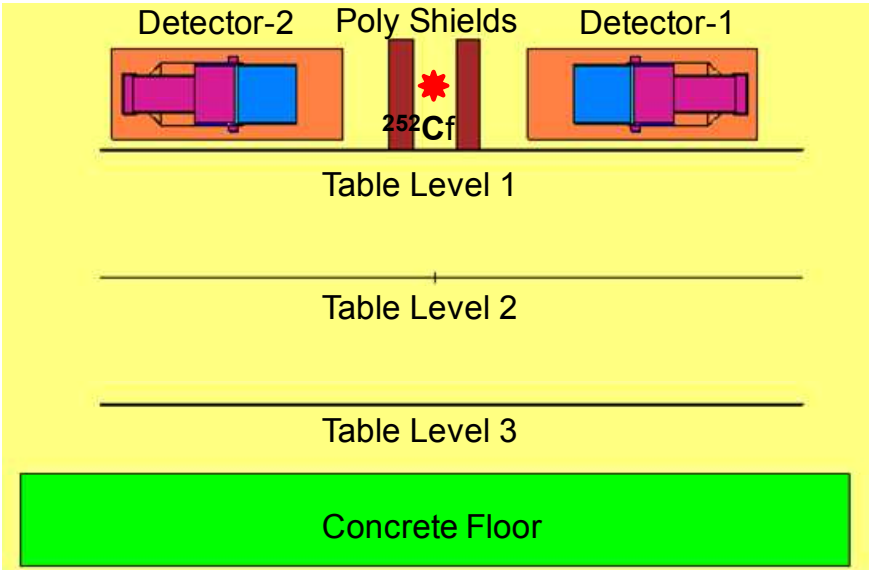


Figure 7– 19: MCNPX–PoliMi simulation setup for the polyethylene–shielded ^{252}Cf measurement.

7.4.4 Convergence and Speedup Results

In this subsection the speedup using geometry splitting is assessed by the same improvement factors, *I,* as discussed in Ch.6.

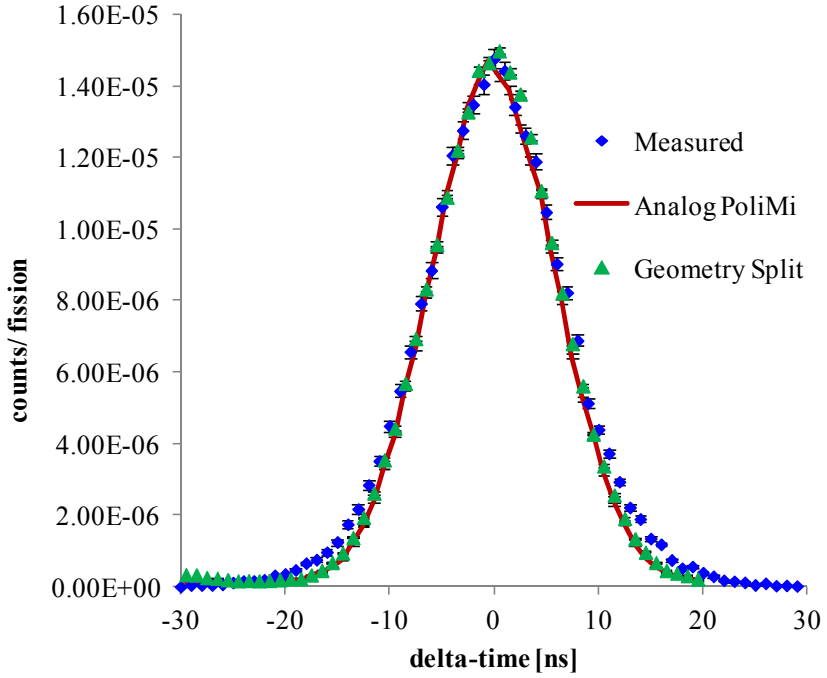


Figure 7– 20: Geometry split and analog MCNPX–PoliMi simulations compared with measurement data for the case of bare ^{252}Cf.

In Fig. 7–20, the bare cross–correlation comparison is shown. The simulations are within 17% of the measured data points (when averaged point–by–point from –15 ns to 15 ns range). There was an average speedup of a factor of *3.4* for the nonanalog case when compared to the analog case.

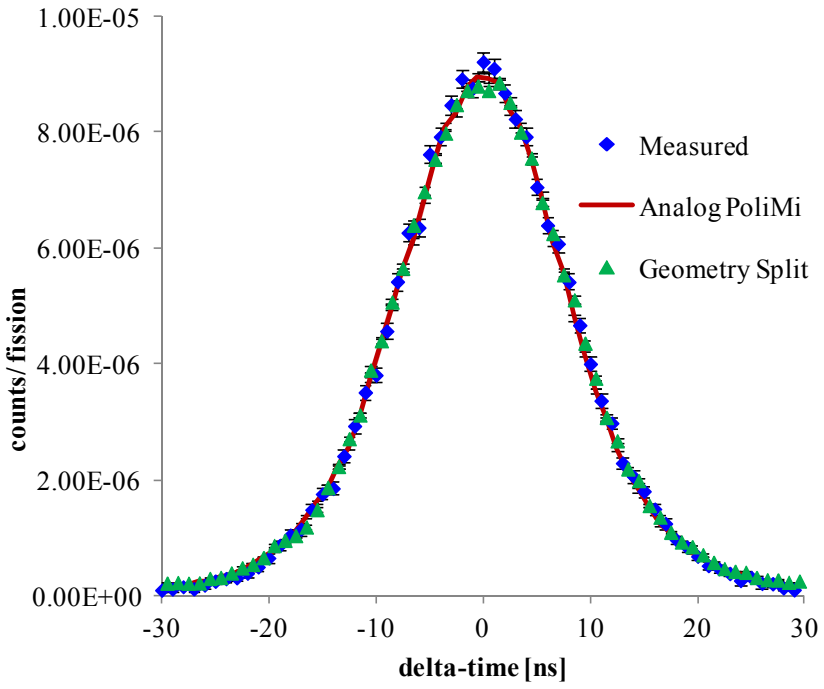


Figure 7– 21: Geometry split and analog MCNPX–PoliMi simulations compared with measurement data for the case of lead–shielded ^{252}Cf.

In Fig. 7–21, the lead–shielded cross–correlation comparison is shown. The simulations are within 7% of the measured data points. There was an average speedup of a factor of *16* for the nonanalog case when compared to the analog case.

In Fig. 7–22, the polyethylene–shielded cross–correlation comparison is shown. The simulations are within 11% of the measured data points. There was an average speedup of a factor of *2.3* for the nonanalog case when compared to the analog case. In the polyethylene case the speedup was limited due to the 1:2 splitting. The splitting ratio was reduced to decrease the misclassification of daughter particles as discussed previously in Subsection 7.3.2. With greater splitting it become more likely for neutrons to leave, scatter in polyethylene, and return to the detector leading to their misclassification.

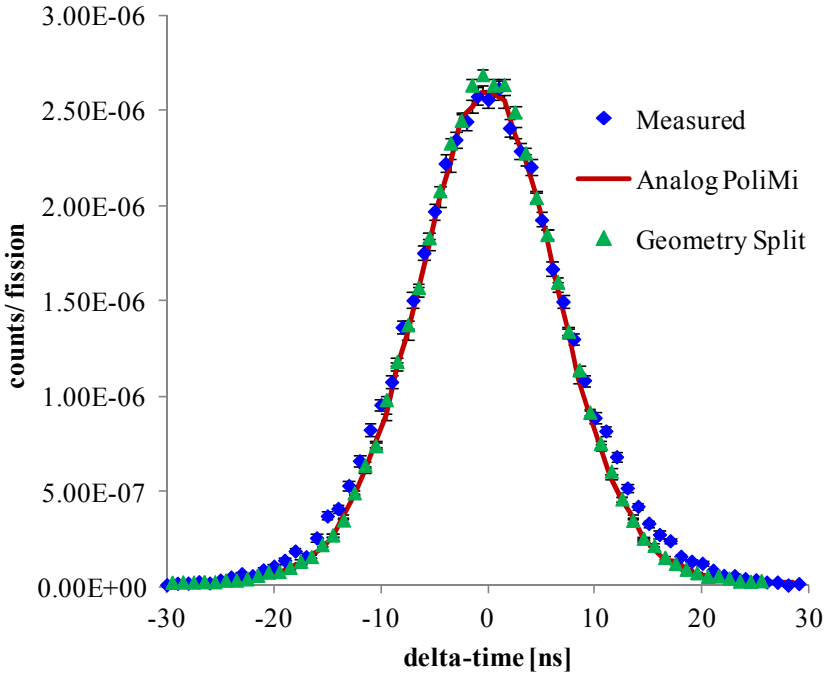


Figure 7– 22: Geometry split and analog MCNPX–PoliMi simulations compared with measurement data for the case of polyethylene–shielded ^{252}Cf.

7.5 Conclusion

In this chapter we have shown that MCNPX–PoliMi simulations can be performed in nonanalog mode using variance reduction techniques to produce neutron detector response for *correlated sources* with the condition that the particle weight does not change inside the detector. The nonanalog MCNPX–PoliMi postprocessor tallies the time–dependent cross–correlated counts by a combined weight product, $\hat{W}$. The improved nonanalog MCNPX–PoliMi tally mechanism also incorporated the multiple particle contribution to a pulse.

Comparisons of time–dependent cross–correlation distributions were made for nonanalog and analog MCNPX–PoliMi simulations for different configurations.

Simulations were also done to validate the laboratory measurements. For the nonanalog cases, geometry splitting with Russian roulette was performed. It was found that the nonanalog results agree well with the analog results. The average speedup in the computation time for the case of bare ^{252}Cf was a factor of 3.4, and for the lead–shielded case it was a factor of 16, whereas for the polyethylene–shielded it was only a factor a 2.6. The polyethylene case speedup was limited due to the 1:2 splitting ratio; a greater ratio causes misclassification of particles, which is a limitation of the MCNPX–PoliMi data file and not the method. The bare case simulation results were within 17% of the measurement, the lead–shielded comparisons were within 7% of the measurement, and the polyethylene–shielded were within 11% of the measurement.

Thus, direct nonanalog MCNPX–PoliMi can be used to speedup the simulation of results, produce results that can be validated by measurements, and simulate different types of scintillation detector response (including time–dependent detector responses).

7.6 References

[1] S. A. Pozzi, E. Padovani, and M. Marseguerra, "MCNP–PoliMi: A Monte Carlo Code for Correlation Measurements," *Nuclear Instruments and Methods to Physics research A,* **513**, 550 (2003).

[2] S. A. Pozzi, E. Padovani, M. Flaska and S. D. Clarke, "MCNP–PoliMi Post–Processing Code Version 1.9," ORNL/TM–2007/33 (2007).

[3] S. Prasad, S. D. Clarke, S. A. Pozzi and E. W. Larsen, "Time–Dependent Neutron Detector Response Simulation for Shielded 252Cf," *Nuclear Science Symposium Conference Record (NSS/MIC),* Knoxville, Tennessee, 30 October – 6 November 2010, 10.1109/ NSSMIC.2010.5873894, 911 (2010).

[4] S. Prasad, S. D. Clarke, S. A. Pozzi, E. W. Larsen, "Organic Scintillation Detector Response Simulation Using Non–analog MCNPX–PoliMi*," PHYSOR 2012 – Advances in Reactor Physics,* Knoxville, Tennessee,15 – 20 April*,* 2012 (2012).

[5] E. Padovani, S. A. Pozzi, S. D. Clarke, E. C. Miller, "MCNPX–PoliMi User's Manual," (2012).

[6] G. F. Knoll, *Radiation Detection and Measurements*, John Wiley & Sons, Inc., Hoboken (2010).

[7] L. L. Carter and E. D. Cashwell, "Particle–Transport Simulation with the Monte Carlo Method," ERDA Critical Review Series TID 26607, U.S. Department of Energy, Washington D.C. (1975).

[8] "MCNP—A General N–Particle Transport Code, Version 5, Volume I: Overview and Theory," X–5 MONTE CARLO TEAM, LA–UR–03–1987, Los Alamos National Laboratory (2003).

[9] S. Prasad, S. D. Clarke, S. A. Pozzi, E. W. Larsen, "Time–Dependent Simulation of Neutron Detector Response using MCNP," *Transactions of the 2010 American Nuclear Society*, **102**, 119 (2010).

[10] S. Prasad, S. D. Clarke, S. A. Pozzi, E. W. Larsen, "Simulation of Neutron Pulse Height Distribution with a Response Matrix Method," *Nuclear Science and Engineering*, **172**, 78 (2012).

[11]] S. A. Pozzi, S. D. Clarke, W. Walsh, E. Miller, J. Dolan, M. Flaska, B. Wieger1, A. Enqvist, E. Padovani, J. K. Mattingly, D. Chichester, and P. Peerani, "MCNPX–PoliMi for Nuclear Nonproliferation Applications," accepted by *Nuclear Instruments and Methods to Physics Research A* (2012).

Chapter 8
Summary and Future Work

8.1 Summary

In this thesis, we have developed methods to simulate neutron scintillation detector response using the Monte Carlo code MCNPX–PoliMi. We have shown that MCNPX–PoliMi is compatible with the traditional variance reduction techniques, if appropriate methods are used. First, a response matrix method (RMM) was formulated in Ch. 5 to compute neutron pulse height distributions for scintillation detectors. Next, in Ch. 6 and Ch. 7, tally mechanisms were formulated for direct nonanalog MCNPX–PoliMi simulations of pulse height distributions, time–of–flight curves, and cross–correlation tallies.

As discussed in Ch. 5, the RMM utilizes a single detector response matrix, which is combined with the incident neutron energy to calculate the detector pulse height distribution (PHD). It is seen that the RMM acts like a variance reduction tool in itself; however, it is not exact because the elements of the matrix are obtained by a finite number of tallies and contain statistical errors. Traditional variance reduction methods exactly preserve the mean. Nonetheless, the PHDs computed with the RMM for a given number of source particles are better converged than the analog MCNPX–PoliMi PHDs for the same number of source particles. The RMM also allows the use of variance reduction techniques to estimate the neutron current incident on the detector face. In fact, one could make use of deterministic methods to calculate the incident neutron current. As

noted in Ch. 5, the user must invest time to pre–compute the response matrix. However, once the response matrix has been computed for a given detector type, it can be used for any simulation involving the same detector type.

To correctly compute the PHDs for sources not incident normally on the detector face, RMM includes a radial leakage correction factor. This factor is the ratio of the volume averaged flux for the given setup to the volume averaged flux of the setup in which source particles are normally incident on the detector face. The radial leakage correction factor is problem–specific, but it can be determined using Monte Carlo calculations already employed in the simulation of the incident neutron current.

In Ch. 5, a comparison of the analog method, the RMM, and the RMM with source biasing is made. It is shown that the RMM significantly reduces the variance throughout the PHD. For RMM with source biasing, an average *FOM* improvement (savings in computation time) of a factor of 600 is achieved in the case of polyethylene shields, and a factor of 300 is achieved in the case of lead shields over the entire PHD. Using the RMM with source biasing decreases the variance and keeps it nearly constant throughout the distribution.

Another advantage of the RMM is that it does not require MCNPX–PoliMi data files to be stored and post–processed. The analog simulation requires storage of large data files and requires time consuming post–processing of these files. If the user chooses to improve the convergence using an analog MCNPX–PoliMi simulation, not only does the Monte Carlo simulation take longer, but the size of the data file produced also increases such that the post–processing of the data file also takes longer. The RMM takes the same amount of time to calculate a PHD, regardless of the number of source particles used to

calculate the incident current. The floating point operations for the RMM is fixed and determined by the size of the response matrix.

In Ch. 5, the RMM PHDs for a polyethylene– and lead–shielded ^{252}Cf source are validated by measurements. Good agreement is seen between the simulated and measured PHDs. As discussed, the RMM provides good speedup in the simulation of these shielded cases. However, it is challenging to extend the RMM to speedup simulation of time–dependent response. Thus, a method to directly run MCNPX–PoliMi with variance reduction techniques to simulate PHDs and time–dependent response is developed in Ch. 6 and Ch. 7.

In Ch. 6, MCNPX–PoliMi simulations are performed in nonanalog mode to calculate PHDs and time–of–flight curves for uncorrelated sources (one neutron per source event). It is possible to simulate scintillation detector response using nonanalog MCNPX–PoliMi if the Monte Carlo problem is separated into nonanalog and analog components. The nonanalog component of the problem is the exterior of the detector, where the particle is allowed to acquire and change weights based on importances specified by the user. The interior of the detector is the analog part of the Monte Carlo problem. The particle weight is kept constant inside the detector, where it produces light pulses. A pulse formed by the particle is given the weight of the particle contributing to it. All pulses are checked against the detector threshold to determine if they contribute to the detector response.

For the nonanalog cases in Ch. 6, source biasing, geometry splitting with Russian roulette, and a combination of both are simulated. These techniques are described in detail in Ch. 4. It is found that nonanalog results agree well with the analog cases.

Furthermore, improvement factors are calculated to analyze the speedup in the simulation time. It is found that the best results are obtained with source biasing and geometry splitting combined. Source biasing helps in modifying particle distributions for higher energies to obtain more scores, whereas geometry splitting is implemented to maintain the same population of neutrons throughout the shield (as neutrons move in one direction from the source towards the detector). Neutron weights are adjusted as the neutron population is modified.

In Ch.7, the method introduced in Ch. 6 is extended to *correlated* sources. For correlated sources in MCNPX–PoliMi, multiple neutrons are emitted from the same source (fission) event. These neutrons are independently sampled in energy and in direction of flight. The energy distribution from which neutrons of a fission event are sampled depends on the multiplicity of the fission event. The flight direction of the neutrons depends on the flight direction of the lighter fission fragment. It is important to understand the above physics models and their roles, such as in cases in which a given detector can receive contributions by the split daughters of different neutrons from the same fission event.

In Ch. 7, the nonanalog MCNPX–PoliMi postprocessor tallies the time–dependent cross–correlated counts by a combined weight product, $\hat{W}$. This combined weight is given by the product of the individual weights of the pulses in each of the detectors. Comparisons of time–dependent cross–correlation distributions are made for nonanalog and analog MCNPX–PoliMi simulations for different configurations. Good agreement is seen for the symmetric cases with and without polyethylene shielding, and with 1:4 splitting ratio. However, for the case in which the source is placed asymmetrically

between two detectors, the nonanalog case with a 1:8 splitting ratio does not agree well with analog results. As mentioned earlier, this is because with increased splitting, a single pulse may receive contributions by split daughters of different neutrons. Therefore, the algorithm is modified to incorporate contributions of multiple particles to the same pulse within the same PGT. The number of pulses formed when the split daughters from different neutrons enter the same detector within the same PGT is given by the number of combinations of the daughter particles, as detailed in Ch. 7. Better agreement is seen between the analog and the nonanalog results after making the above improvements.

Simulations were also done to validate laboratory measurements. To validate measurements with nonanalog simulations, geometry splitting with Russian roulette was used. It was found that nonanalog results agree well with the analog cases. The average speedup in the computation time for the case of bare ^{252}Cf was a factor of 3.4, for the lead–shielded case it was a factor of 16, and for the polyethylene–shielded it was a factor of 2.6. The polyethylene case speedup was limited due to the 1:2 splitting ratio. Greater ratios cause misclassification of particles, which is a limitation of the MCNPX–PoliMi data file and not the method. The bare case simulation results were within 17% of the measurement, the lead–shielded comparisons were within 7% of the measurement, and the polyethylene–shielded were within 11% of the measurement. Thus, it is seen that the nonanalog MCNPX–PoliMi simulations with the proper tally mechanisms help preserve the physics of detector response, and are capable of providing accurate answers in shorter simulation times. The direct nonanalog MCNPX–PoliMi approach is also general: it can simulate PHDs, TOF, and cross–correlation responses.

In the next section some thoughts on future work are provided.

8.2 Future Work

For the work done on RMM formulation and simulations, the following suggestions for future work are made:

1) Generalization of the RMM by creating a response matrix solely to calculate the radial leakage correction factor would be more efficient. This would eliminate the need of performing an additional simulation to calculate the volume averaged flux for the case where neutrons are not normally incident on the detector face.

2) The neutron current in the RMM problems is calculated using separate Monte Carlo simulations from the ones that are used to pre–compute the response matrix itself. Thus, it may be feasible to use spatial decomposition such that certain nodes of a cluster can compute the response matrix, while the others can be used to calculate the neutron incident on the detector face. Such a method would easily allow changing detector characteristics or performing sensitivity analysis on scintillation detector response.

The next few recommendations are regarding the nonanalog MCNPX–PoliMi simulations.

1) It is recommended to add a column in MCNPX–PoliMi data file that indicates the daughter number of a particle. This would allow the user to increase the splitting ratio even in the presence of highly scattering material (such as polyethylene) just outside the detector.

2) Presently, source biasing does not correctly bias particles if the built–in MCNPX–PoliMi correlated sources are invoked. This is because the developers of MCNPX–PoliMi had not originally intended the use of variance reduction techniques with the code. But source biasing could be easily included in MCNPX–PoliMi so that users can take advantage of source biasing even for correlated sources. The tally mechanism should remain the same as shown with the geometry split simulations in Ch 7.

3) To optimize the use of nonanalog problems, it is recommended to make use of adjoint methods (such as those discussed in Ch. 4). This will be most useful to reduce simulation times, and will not require trial and error from the user.

A different approach to speedup MCNPX–PoliMi may be to investigate ways of convolving time–dependent neutron current to yield time–dependent cross–correlation scintillation detector response.

Appendices

Appendix A

This appendix provides the code that was written in Matlab to compute PHDs using the RMM as detailed in Ch. 5. Error propagation has also been included.

```
clear all;
tic

% INITIALIZATION
load ('CtoolsMatrix.mat'); % Andreas coeff
R = CtoolsMatrix;
load('ErrorMatrix.mat');% Contains relative error
E = Err;
NPS = 1e7; % *****NPS****
rows = 741;
cols = 1007;
start = 0.01;

%  THIS CHANGES
load ('pPb4a.mat');
F = pPb4a;
load ('r30pb4.mat');
L = r30pb4;

RF = zeros(rows,cols);%Multiple of R and F
Var = zeros(rows,cols);% Standard deviation
N = zeros(cols,2);
F(:,1)= F(:,1)*NPS;

% Set threshold
s = 1;
for t = 0.01:0.01:0.06 % Change this to change THRESH-HOLD, last no. is 1 set
less than thresh-hold
    R(:,s) = 0;
    s = s+1;
end

for i = 1:rows
    RF (i,:) = F(i,1)*R(i,:);%creates i_th row in matrix RF
    % Pnl(i) = ((1-L(i,1))*1e8)/(F(i,1)*NPS);
    RF (i,:) = (L(i,1))*RF(i,:);
    for j = 1:cols
        if ((E(i,j)==Inf))
            E(i,j)=0;
        end
        % 1st term is var in current, 2nd in P_nonleakage, 3rd in response
        % matrix element (in any term first var is the relative errors)
        Var(i,j) = ((F(i,2)^2*F(i,1)^2*L(i,1)^2) +
(L(i,2)^2*F(i,1)^2*L(i,1)^2) + (E(i,j)^2*F(i,1)^2*L(i,1)^2))*R(i,j)^2;
    end
end
```

```
N(:,1) = start:0.01:10.07;
for i = 1:cols
    N(i,2) = sum(RF(:,i));
    N(i,3) = sqrt(sum(Var(:,i)));
end
toc

% scale the pulse height to the big source strength
load ('BIGsource.mat');
Spol = 329902;
Spb = 328249;
N(:,2) = N(:,2)*Spb/(2*NPS);
N(:,3) = N(:,3)*Spb/(2*NPS);
% plot(N(:,1), N(:,2))
```

Appendix B

The C++ code used to generate results in Ch. 6 is provided here.

```
#ifndef _POSTPROCESS_
#define _POSTPROCESS_

#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <vector>
#include <time.h>

using namespace std;

class PostProcess {

   // Filename
   char fileName[256];
   FILE * pfile;
   FILE * xfile;
   FILE * pulsefile;
   FILE * tallyfile;
   FILE * readweightsfile;

   vector<double> read_weights;
   //History items
   vector<int> history_no;
   vector<int> particle_no;
   vector<int> particle_type;
   vector<int> interaction;
   vector<int> nuclei;
   vector<int> cell;
   vector<double> energy_dep;
   vector<double> time;
   vector<double> x, y, z;
   vector<double> weight;
   vector<int> gen_no;
   vector<int> scatter_no;
   vector<int> un;
   vector<double> energy_inc;

   //Pulse item

   //Tally item
   double phd_hi, phd_lo, phd_binsz, tof_hi, tof_lo, tof_binsz;
   double xcorrel_hi, xcorrel_lo, xcorrel_binsz;
   double phd[1000][5];
   double tof[100][5];
   double xcorrel[201][3];
   int det1, det2;
   int pmax, tmax, xmax, xmin, count;

   public:
```

```
    PostProcess(char *fileName);
    ~PostProcess();
    bool read_history();
    void clear_history();
    void form_pulse();
    void write_tally();
    int psum;
    int isum;
};
#endif //_POSTPROCESS_


#include "PostProcess.h"

//Constructor
PostProcess::PostProcess(char *name)
{
  pfile = fopen("/home/shikhap/nonanalog/ES/Po4sb.d", "r");

  if(pfile == NULL)
  {
   printf("Error\n");
   exit(0);
  }

  // filling arrays for tallying
  phd_hi = 10;
  phd_lo = 0;
  phd_binsz= 0.01;

  tof_hi = 100;
  tof_lo = 0;
  tof_binsz= 1;

  pmax = (int) ceil((phd_hi - phd_lo)/phd_binsz);
  tmax = (int) ceil((tof_hi - tof_lo)/tof_binsz);
  count = 0;

  for (int n=0; n<pmax ;n++)
  {
    phd[n][0]= phd_lo + n*phd_binsz;
    phd[n][1]= 0;
    phd[n][2]= 0;
    phd[n][3]= 0;
    phd[n][4]= 0;
  }

  for (int n=0; n<tmax ;n++)
  {
    tof[n][0]= tof_lo + n*tof_binsz;
    tof[n][1]= 0;
    tof[n][2]= 0;
    tof[n][3]= 0;
    tof[n][4]= 0;
  }

  pulsefile = fopen("pulses.txt", "w");
```

```
}

//Destructor
PostProcess::~PostProcess()
{
fclose(pfile);
fclose(pulsefile);
}

bool PostProcess::read_history()
{
  bool ret=true;
  long size;
  int line=0;
  /* line keeps track of vector row */
  do{
       fscanf(pfile, " ");

        int int_var;
        double float_var;

       fscanf(pfile, "%d", &int_var);
        history_no.push_back(int_var);

        if (line > 0)
       {
                if ((history_no[line])!= (history_no[(line-1)]))
              {
                    form_pulse();
                    clear_history();

                    line = 0;
                    history_no.push_back(int_var);
                }
       }

        fscanf(pfile, " ");
        fscanf (pfile, "%d", &int_var);
        particle_no.push_back(int_var);

        fscanf(pfile, " ");
        fscanf(pfile, "%d", &int_var);
        particle_type.push_back(int_var);

        fscanf(pfile, " ");
        fscanf(pfile, "%d", &int_var);
        interaction.push_back(int_var);
        int interact = int_var;

       fscanf(pfile, " ");
       fscanf(pfile, "%d", &int_var);
       nuclei.push_back(int_var);

       fscanf(pfile, " ");
       fscanf(pfile, "%d", &int_var);
       cell.push_back(int_var);
```

```
fscanf(pfile, " ");
fscanf(pfile, "%lf", &float_var);
energy_dep.push_back(float_var);

fscanf(pfile, " ");
fscanf(pfile, "%lf", &float_var);
time.push_back(float_var);

fscanf(pfile, " ");
fscanf(pfile, "%lf", &float_var);
x.push_back(float_var);

fscanf(pfile, " ");
fscanf(pfile, "%lf", &float_var);
y.push_back(float_var);

fscanf(pfile, " ");
fscanf(pfile, "%lf", &float_var);
z.push_back(float_var);

fscanf(pfile, " ");
fscanf(pfile, "%lf", &float_var);
weight.push_back(float_var);

fscanf(pfile, " ");
fscanf(pfile, "%d", &int_var);
gen_no.push_back(int_var);

fscanf(pfile, " ");
fscanf (pfile, "%d", &int_var);
scatter_no.push_back(int_var);

fscanf(pfile, " ");
fscanf(pfile, "%d", &int_var);
un.push_back(int_var);

fscanf(pfile, " ");
fscanf(pfile, "%lf", &float_var);
energy_inc.push_back(float_var);

if (feof(pfile))
{
       form_pulse();
       clear_history();
         cout<<"File ended\n";
       return(false);
}
 else if ((interact != -99)&&(interact != -1))
 {
       form_pulse();
       clear_history();
       line = 0;
}
else
{
        line++;
}
} while(ret);
```

```
  return (ret);
}

void PostProcess::clear_history()
{
   history_no.clear();
   particle_no.clear();
   particle_type.clear();
   interaction.clear();
   nuclei.clear();
   cell.clear();
   energy_dep.clear();
   time.clear();
   x.clear(); y.clear(); z.clear();
   weight.clear();
   gen_no.clear();
   scatter_no.clear();
   un.clear();
   energy_inc.clear();
}

void PostProcess::form_pulse()
{
       double pgt = 10.0 ; /* pulse generation time in [ns] */
       const double thresh_hold = 0.07;// MeVee set thresh-hold seen in
measurements

       const double coef_C = 0.02;     //light conversion for carbon
       const double coef_aH = 0.03495; //light conversion for hydorgen, quadratic
       const double coef_bH = 0.1424;  //light conversion for hydrogen, linear
       const double coef_cH = -0.0362; //light conversion for hgydrogen, intercept

        double temp_time, temp_weight, temp_height; // of a pulse
        double temp_energy, l;
        int daughter=1;
        int diff=0;
        int diff_particle;

        int sz = (history_no.size() - 1);//for a new history or a capture event
length of vectors are bigger by 1
       for (int i=0; i<sz; i++)
       {
                double t_stop = (10 * time[i] + pgt);
                double t = 10*time[i];
                double light = 0;
                temp_time = 10*time[i];
                temp_weight= weight[i];
                int k;
                do{
                           temp_energy = energy_dep[i];
                     if (nuclei[i]==1001)
                             l = coef_aH*temp_energy*temp_energy +
coef_bH*temp_energy + coef_cH;
                         else if(nuclei[i]==6000)
                             l = coef_C*temp_energy;

                     light = light+l;
```

```
                        i++;
                        t = 10*time[i]; //this is grabbing the time for next line
                        diff = scatter_no[i]-scatter_no[i-1];
                        diff_particle = particle_no[i]-particle_no[i-1];
                } while((t<t_stop) && (i<sz) && ( ((interaction[i-1]==-
1)&&(diff==0)) || ((interaction[i-1]==-99)&&(diff==1))));

                i--;
                temp_height=light;

                if (history_no[i]==33973)
                cout<<light<<" temp_time "<<temp_time<<" scatter
"<<scatter_no[i]<<endl;

             int bin_phd = (int) ceil((temp_height-phd_lo)/phd_binsz);
             int bin_tof = (int) ceil((temp_time-tof_lo)/tof_binsz);

             if ((temp_height > thresh_hold) && (bin_phd < pmax) && (bin_tof
<tmax))
             {
                    int bin_phd = (int) ceil((temp_height-phd_lo)/phd_binsz);
                    int bin_tof = (int) ceil((temp_time-tof_lo)/tof_binsz);

                       phd[bin_phd][1] = phd[bin_phd][1] + temp_weight;
                    tof[bin_tof][1] = tof[bin_tof][1] + temp_weight;
                       phd[bin_phd][2] = phd[bin_phd][2] + 1;
                    tof[bin_tof][2] = tof[bin_tof][2] + 1;
                    fprintf(pulsefile, "%d %lf %lf %lf %d\n",history_no[i],
temp_height, temp_time, temp_weight, daughter);

                }

                 if (((interaction[i]==-1)&&(scatter_no[i+1]!=scatter_no[i])) ||
((interaction[i]==-99)&&(scatter_no[i+1]<=scatter_no[i])))
                {     daughter++;
                      if (history_no[i]==33973)
                      cout<<daughter<<" temp_time "<<temp_time<<" scatter
"<<scatter_no[i]<<endl;
                }

       }
}

void PostProcess::write_tally()
{
       FILE * wpfile;
       FILE * wtfile;


        cout<<"making files\n";
        wpfile = fopen("phd.txt", "w");
        wtfile = fopen("tof.txt", "w");

        //fclose (pulsefile);
        tallyfile = fopen ("pulses.txt","r"); // rem to remove fromconstructor
also
        int hist;
```

```
         double hyt, tym, wt;
         bool ret=true;
        do
         {
               fscanf (tallyfile, "%d", &hist);
               fscanf (tallyfile, " ");
               fscanf (tallyfile, "%lf", &hyt);
               fscanf (tallyfile, " ");
               fscanf (tallyfile, "%lf", &tym);
               fscanf (tallyfile, " ");
               fscanf (tallyfile, "%lf", &wt);
               fscanf (tallyfile, " ");
               fscanf (tallyfile, "%d");

                 int phd_bin = (int) ceil((hyt-phd_lo)/phd_binsz);
                 phd[phd_bin][3] = phd[phd_bin][3] + wt;
                 phd[phd_bin][4] = phd[phd_bin][4] + 1;

                 int tof_bin = (int) ceil((tym-tof_lo)/tof_binsz);
                 tof[tof_bin][3] = tof[tof_bin][3] + wt; //also remove the extra
column here & in header
                 tof[tof_bin][4] = tof[tof_bin][4] + 1;

               if (feof(tallyfile))
                     ret = false;

        } while (ret);

         fclose(tallyfile);

        int p = 1000;
        int t = 100;

         for (int i =0; i<=p; i++)
               fprintf(wpfile, "%lf %lf %lf\n", phd[i][0], phd[i][1], phd[i][2]);

        for (int j = 0; j<=t; j++)
               fprintf(wtfile, "%lf %lf %lf\n", tof[j][0], tof[j][1], tof[j][2]);

        fclose(wpfile);
        fclose(wtfile);
         cout<<count<<endl;
}
```

Appendix C

The C++ code used to generate the results in Ch. 7 is provided below.

```
#ifndef _POSTPROCESS_
#define _POSTPROCESS_

#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <vector>
#include <time.h>

using namespace std;

// Detector cell no.s
int first_detector = 1;
int second_detector = 2;

struct Daughter_info
{
   int num_daughters;
   vector<int> start_index;
   vector<int> stop_index;
};
typedef Daughter_info Daughter_info_t;

class PostProcess {

   // Filename
   char fileName[256];
   FILE * pfile;
   FILE * xfile;
   FILE * pulsefile;
   FILE * tallyfile;
   FILE * readweightsfile;

   vector<double> read_weights;

  //History items
   vector<int> history_no;
   vector<int> particle_no;
   vector<int> particle_type;
   vector<int> interaction;
   vector<int> nuclei;
   vector<int> cell;
   vector<double> energy_dep;
   vector<double> time;
   vector<double> x, y, z;
   vector<double> weight;
   vector<int> gen_no;
   vector<int> scatter_no;
   vector<int> un;
```

```
   vector<double> energy_inc;

   //Particle items
   vector<Daughter_info_t> det1_particle_daughters;
   vector<Daughter_info_t> det2_particle_daughters;
   vector<int> particle_id;

   //Tally item
   double phd_hi, phd_lo, phd_binsz, tof_hi, tof_lo, tof_binsz;
   double xcorrel_hi, xcorrel_lo, xcorrel_binsz;
   double phd[1000][5];
   double tof[100][5];
   double xcorrel[201][3];
   int det1, det2;
   int pmax, tmax, xmax, xmin, count;

   // For tracking time to calculate time-x-correlation
   vector<double> time_det1, time_det2, weight_det1, weight_det2, height_det1,
height_det2;

   // case items
   vector <int> case_particle;
   vector <int> case_interaction;
   vector <int> case_nuclei;
   vector <double> case_energy;
   vector <double> case_time;
   vector <double> case_weight;

   public:
     PostProcess(char *fileName);
     ~PostProcess();
     bool read_history();
     void clear_history();
     void form_pulse(int detector);
     void write_tally();

     // new for correlated VR
     void calculate_XCorrelation();
     void find_particles();
     void form_numParticle1(int detector_no);
     void form_numParticle2(int detector_no);
     void form_numParticle3(int detector_no);
};
#endif //_POSTPROCESS_

// March 5th 2012
// Changing PostProcess.cpp to include nonanalog X-correlation, PHD and TOF
#include "PostProcessVR.h"


//Constructor
PostProcess::PostProcess(char *name)
{
  //pfile = fopen("/home/shikhap/nonanalog/correl/10pol30-1b.d", "r");
  pfile = fopen("/nobackup/shikhap/xcorrel/gs8asbare/asGSBare80.d", "r");
  if(pfile == NULL)
  {
   printf("Error opening file.\n");
```

```
   exit(0);
  }

  // filling arrays for tallying
  phd_hi = 10;
  phd_lo = 0;
  phd_binsz= 0.01;

  tof_hi = 100;
  tof_lo = 0;
  tof_binsz= 1;

  xcorrel_hi = 100;
  xcorrel_lo = -100;
  xcorrel_binsz = 1;

  pmax = (int) ceil((phd_hi - phd_lo)/phd_binsz);
  tmax = (int) ceil((tof_hi - tof_lo)/tof_binsz);
  xmax = (int) ceil((xcorrel_hi - xcorrel_lo)/xcorrel_binsz);
  count = 0;

  for (int n=0; n<pmax ;n++)
  {
    phd[n][0]= phd_lo + n*phd_binsz;
    phd[n][1]= 0;
    phd[n][2]= 0;
    phd[n][3]= 0;
    phd[n][4]= 0;
  }

  for (int n=0; n<tmax ;n++)
  {
    tof[n][0]= tof_lo + n*tof_binsz;
    tof[n][1]= 0;
    tof[n][2]= 0;
    tof[n][3]= 0;
    tof[n][4]= 0;
  }

  for (int n=0; n<xmax; n++)
  {
    xcorrel[n][0] = xcorrel_lo + n*xcorrel_binsz;
    xcorrel[n][1] = 0;
    xcorrel[n][2] = 0;
  }

  det1 = 1;
  det2 = 2;
  pulsefile = fopen("pulses.txt", "w");
}

//Destructor
PostProcess::~PostProcess()
{
fclose(pfile);
fclose(pulsefile);
}
```

```
bool PostProcess::read_history()
{
  bool ret=true;
  int line=0;
  /* line keeps track of vector row */
  do{
       fscanf(pfile, " ");

        int int_var;
        double float_var;

       fscanf(pfile, "%d", &int_var);
        history_no.push_back(int_var);

        if (line > 0)
       {
                if ((history_no[line])!= (history_no[(line-1)]))
              {
                     cout<<"history"<<history_no[line-1]<<endl;
                    find_particles();
                    clear_history();

                    line = 0;
                    history_no.push_back(int_var);
                }
       }

        fscanf(pfile, " ");
        fscanf (pfile, "%d", &int_var);
       particle_no.push_back(int_var);

        fscanf(pfile, " ");
        fscanf(pfile, "%d", &int_var);
       particle_type.push_back(int_var);

        fscanf(pfile, " ");
        fscanf(pfile, "%d", &int_var);
       interaction.push_back(int_var);

        fscanf(pfile, " ");
        fscanf(pfile, "%d", &int_var);
       nuclei.push_back(int_var);

        fscanf(pfile, " ");
        fscanf(pfile, "%d", &int_var);
       cell.push_back(int_var);

       fscanf(pfile, " ");
        fscanf(pfile, "%lf", &float_var);
       energy_dep.push_back(float_var);

        fscanf(pfile, " ");
        fscanf(pfile, "%lf", &float_var);
       time.push_back(float_var);

       fscanf(pfile, " ");
        fscanf(pfile, "%lf", &float_var);
       x.push_back(float_var);
```

```
        fscanf(pfile, " ");
         fscanf(pfile, "%lf", &float_var);
        y.push_back(float_var);

        fscanf(pfile, " ");
         fscanf(pfile, "%lf", &float_var);
        z.push_back(float_var);

        fscanf(pfile, " ");
         fscanf(pfile, "%lf", &float_var);
        weight.push_back(float_var);

        fscanf(pfile, " ");
         fscanf(pfile, "%d", &int_var);
        gen_no.push_back(int_var);

         fscanf(pfile, " ");
         fscanf (pfile, "%d", &int_var);
        scatter_no.push_back(int_var);

        fscanf(pfile, " ");
         fscanf(pfile, "%d", &int_var);
        un.push_back(int_var);

        fscanf(pfile, " ");
         fscanf(pfile, "%lf", &float_var);
        energy_inc.push_back(float_var);

        if (feof(pfile))
        {
               find_particles();
               clear_history();
                 cout<<"File ended\n";
               return(false);
        }
        else
        {
                line++;
        }
       } while(ret);

  return (ret);
}

void PostProcess::clear_history()
{
   history_no.clear();
   particle_no.clear();
   particle_type.clear();
   interaction.clear();
   nuclei.clear();
   cell.clear();
   energy_dep.clear();
   time.clear();
   x.clear(); y.clear(); z.clear();
   weight.clear();
   gen_no.clear();
```

```
   scatter_no.clear();
   un.clear();
   energy_inc.clear();
}

void PostProcess::find_particles()
{
       cout<<"in find_particles"<<endl;
        Daughter_info_t particle_temp; // this is only a temp struct as defined in
header file

       int sz = (history_no.size()-1);
       int diff, diff_particle;

       int daughter = 1;
       particle_id.push_back(particle_no[0]);
       particle_temp.num_daughters = daughter; // why was this pushback when it
isn't a vector
       particle_temp.start_index.push_back(0);

       for (int i=1; i<sz; i++)
        {
             //if (interaction[i] ==0) // do this to skip the capture interaction
             //     i++;
             diff = scatter_no[i]-scatter_no[i-1];
               diff_particle = abs(particle_no[i]-particle_no[i-1]);
             int len = particle_id.size();

             if (i==(sz-1))
               {

                  if ((diff_particle >= 1) || (cell[i]!=cell[i-1]))
                  {
                        particle_temp.stop_index.push_back(i-1);

                        // Check to see which detector the particle belongs to
                        if (cell[i-1]==first_detector)
                            det1_particle_daughters.push_back(particle_temp);
                        else if (cell[i-1]==second_detector)
                            det2_particle_daughters.push_back(particle_temp);

                              particle_temp.start_index.clear();
                        particle_temp.stop_index.clear();

                        // Start new temp struct
                        daughter = 1;
                        particle_id.push_back(particle_no[i]);
                        particle_temp.num_daughters = daughter;
                        particle_temp.start_index.push_back(i);

                  }

                  else if ( ((interaction[i]==-1) && (diff!=0)) ||
((interaction[i]==-99) && (diff!=1)) )
                  {
                         particle_temp.stop_index.push_back(i-1);
                         daughter++;
                         particle_temp.num_daughters=daughter;
```

```
                        particle_temp.start_index.push_back(i);

                }

                    particle_temp.stop_index.push_back(i);

                    // Check to see which detector the particle belongs to
                    if (cell[i]==first_detector)
                            det1_particle_daughters.push_back(particle_temp);
                    else if (cell[i]==second_detector)
                            det2_particle_daughters.push_back(particle_temp);

                    particle_temp.start_index.clear();
                    particle_temp.stop_index.clear();

              }

              // IF THE LINE CORRESPONDS TO A DIFFERENT PARTICLE
            else if ((diff_particle >= 1) || (cell[i]!=cell[i-1]))
            {
                  // Record the stop index of previous
                  // Push the temporary struct in appropriate detector struct,
                  // and clear the old one
                  particle_temp.stop_index.push_back(i-1);

                  // Check to see which detector the particle belongs to
                     if (cell[i-1]==first_detector)
                         det1_particle_daughters.push_back(particle_temp);
                  else if (cell[i-1]==second_detector)
                         det2_particle_daughters.push_back(particle_temp);

                  particle_temp.start_index.clear();
                     particle_temp.stop_index.clear();

                  cout<<"num of daughters"<<particle_temp.num_daughters<<endl;

                     // Start new temp struct
                  daughter = 1;
                     particle_id.push_back(particle_no[i]);
                  particle_temp.num_daughters = daughter;
                  particle_temp.start_index.push_back(i);

                  cout<<"different particle at "<<i<<endl;
                  cout<<"critical cell no is "<< cell[i-1]<<endl;
            }

            // IF LINE CORREPONDS TO A DIFFERENT DAUGHTER
            else if  ( ((interaction[i]==-1) && (diff!=0)) ||
((interaction[i]==-99) && (diff!=1)) )
            {
                  daughter++;
                  particle_temp.num_daughters=daughter;
                     particle_temp.stop_index.push_back(i-1);
                  particle_temp.start_index.push_back(i);

                  cout<<"different daughter at "<<i<<endl;
            }
```

```
	}


	int len1 = det1_particle_daughters.size();
	 int len2 = det2_particle_daughters.size();
	cout<<"len2"<<len2<<endl;
	//Check for all the particles and daughters
	for (int p = 0 ; p < det1_particle_daughters.size(); p++)
		for (int d = 0 ; d < det1_particle_daughters[p].num_daughters ; d++)
			cout<<"Particle "<<p<<"and daughter "<<d<<"starts at
"<<det1_particle_daughters[p].start_index[d]<<" and ends at
"<<det1_particle_daughters[p].stop_index[d]<<endl;

	if (len1 >1)
		form_numParticle2(first_detector);
	else
		 form_numParticle1(first_detector);
	 //else if (len1==3)
	//	form_numParticle3(first_detector);

	if (len2 >1)
		 form_numParticle2(second_detector);

	 else
		form_numParticle1(second_detector);


	 //else if (len2==3)
	 //	form_numParticle3(second_detector);

	 // Call X-correlation function here
	calculate_XCorrelation();
	det1_particle_daughters.clear();
	det2_particle_daughters.clear();

	 // HAVE ALREADY CLEARED THIS IN FOR-LOOP, REDUNDANT?
	particle_temp.num_daughters=0;
	particle_temp.start_index.clear();
	particle_temp.stop_index.clear();
}

void PostProcess::form_pulse(int detector)
{
	double pgt = 10.0 ; /* pulse generation time in [ns] */
	const double thresh_hold = 0.07;// MeVee set thresh-hold seen in
measurements

	const double coef_C = 0.02;	//light conversion for carbon
	const double coef_aH = 0.03495; //light conversion for hydorgen, quadratic
	const double coef_bH = 0.1424;  //light conversion for hydrogen, linear
	const double coef_cH = -0.0362; //light conversion for hgydrogen, intercept


	double temp_time, temp_weight, temp_height, temp_energy, l; // of a pulse

	 // THIS SECTION WILL READ ALL LINES OF A GIVEN CASE AND CONVERT TO PULSES
	 //
```

```
        int size = (case_particle.size());
        vector<int> case_check(size,0);
        int case_sum = 0;  //case_check.sum();
        double t_start, t_stop;

// This case_check is a vector contains elements that will turn from 0 to 1 once
in a pulse
        while (case_sum<size){

              // FIND t_start AND t_stop
                // find the smallest time that hasn't been included  yet
              t_start = 1e6; // times should not be bigger than 1e6
              cout<<case_check.size()<<endl;
              for( int q=0; q<size; q++)
              {
                     if ((case_check[q]==0) && (case_time[q]*10 < t_start))
                            t_start = case_time[q]*10;
              }
              t_stop = t_start+10;
              temp_height = 0;

 // FOR ALL LINES, CHECK FOR THE ONES THAT FALL B/W t_start AND t_stop, MAKE PULSE
                temp_weight = 1;
                int current_particle = -1;
                int pulse_len = 0;
                for (int r=0; r<size; r++)
                 {

                     double t = 10*case_time[r];
//IF THE NEW COLLISION FALLS WITHIN THE TIME OF THE PULSE
                     if ((t<t_stop) && (t>=t_start) && (case_check[r]==0))
                     {
                            pulse_len++;

                            //CALCULATE WEIGHT CHANGE AFTER EACH PARTICLE
                                if ( (pulse_len == 1) || ((pulse_len > 1) &&
(case_particle[r]!=current_particle)) )
                                  {
                                  temp_weight *= case_weight[r];
                                  current_particle = case_particle[r];
                                  }

                                  temp_energy = case_energy[r];
                            if (case_nuclei[r]==1001)
                                   l = coef_aH*temp_energy*temp_energy +
coef_bH*temp_energy + coef_cH;
                            else if(case_nuclei[r]==6000)
                                   l = coef_C*temp_energy;

                            temp_height = temp_height+l;
                            case_check[r] = 1;
                                case_sum++;

                     }

              }

                int bin_phd = (int) ceil((temp_height-phd_lo)/phd_binsz);
```

```
            int bin_tof = (int) ceil((temp_time-tof_lo)/tof_binsz);

             if ((temp_height > thresh_hold) && (bin_phd < pmax) && (bin_tof
<tmax))
             {
                    cout<<"inside threshold loop"<<temp_weight<<endl;


                    phd[bin_phd][1] = phd[bin_phd][1] + temp_weight;
                    tof[bin_tof][1] = tof[bin_tof][1] + temp_weight;
                    phd[bin_phd][2] = phd[bin_phd][2] + 1;
                    tof[bin_tof][2] = tof[bin_tof][2] + 1;
                    fprintf(pulsefile, "%d %lf %lf %lf\n", history_no[0],
temp_height, temp_time, temp_weight);
                       if (detector == first_detector)
                    {
                           cout<<"first detector"<<endl;
                                  time_det1.push_back(temp_time);
                           cout<<"time"<<endl;
                                weight_det1.push_back(temp_weight);
                           cout<<"weight"<<endl;
                                  height_det1.push_back(temp_height);
                           cout<<"height"<<endl;
                    }
                    else if (detector == second_detector)
                           {

                                time_det2.push_back(temp_time);
                                weight_det2.push_back(temp_weight);
                                height_det2.push_back(temp_height);
                       }
                    }

       }
      cout<<"outside while loop"<<endl;
}

void PostProcess::calculate_XCorrelation()
{
       cout<<"inside Xcorrelation"<<endl;
        //THIS SECTION WILL CALCULATE TIME-CROSS-CORRELATIONS, after reading all
of history
        //
_____________________________________________________________________________
        double time_delt, weight_delt;
        for (int i=0; i<time_det1.size(); i++)
        {
              for (int j=0; j<time_det2.size(); j++)
              {

                    time_delt = time_det1[i]-time_det2[j];
                       weight_delt = weight_det1[i]*weight_det2[j];

                    int bin_xcorrel = (int) ceil((time_delt-
xcorrel_lo)/xcorrel_binsz);
                    xcorrel[bin_xcorrel][1] = xcorrel[bin_xcorrel][1] +
weight_delt;
```

```
                        xcorrel[bin_xcorrel][2] =  xcorrel[bin_xcorrel][2] + 1;

                }
        }

         time_det1.clear(); time_det2.clear(); weight_det1.clear();
weight_det2.clear();
         height_det1.clear(); height_det2.clear();

}

void PostProcess::write_tally()
{
        FILE * wpfile;
        FILE * wtfile;
        FILE * wxfile;

         cout<<"making files\n";
         wpfile = fopen("phd.txt", "w");
         wtfile = fopen("tof.txt", "w");
         wxfile = fopen("xcorrel.txt", "w");

        int p = 1000;
        int t = 100;
        int x = 201;
        for (int i =0; i<=p; i++)
               fprintf(wpfile, "%lf %lf %lf\n", phd[i][0], phd[i][1], phd[i][2]);

        for (int j = 0; j<=t; j++)
               fprintf(wtfile, "%lf %lf %lf\n", tof[j][0], tof[j][1], tof[j][2]);

        for (int k = 0; k<=x; k++)
                 fprintf(wxfile, "%lf %lf %lf\n", xcorrel[k][0], xcorrel[k][1],
xcorrel[k][2]);

        fclose(wpfile);
        fclose(wtfile);
        fclose(wxfile);
}

void PostProcess::form_numParticle1 ( int detector_no )
{
        cout<<"form_num1 for det no"<<detector_no<<endl;
        vector<Daughter_info_t> temp_particle_daughters;

         // COPY PARTICLE INTO A TEMP VARIABLE FOR PROCESSING
         if (detector_no == first_detector)
                 temp_particle_daughters = det1_particle_daughters;
         else if (detector_no == second_detector)
                 temp_particle_daughters = det2_particle_daughters;

        int sz = (history_no.size() - 1);//for a new history length of vectors are
bigger by 1
         // EACH OF THE THREE NESTED LOOPS IS FOR EACH DAUGHHTER OF A PARTICLE
         // TOGETHER, THEY GIVE ALL POSSIBLE COMBINATION OF DAUGHTERS FOR THREE
PARTICLE
         //
```

```
	daughter_info_t temp_struct;
	temp_struct = temp_particle_daughters[0];
	int ch = temp_struct.num_daughters;
	cout<<ch<<"num of daughters....next start index"<<endl;

	 //for (int a = 0; a<temp_particle_daughters[0].num_daughters; a++)
	 for (int a = 0; a<ch; a++)
	 {
				int case_len = 0;
				    for (int s=0; s<sz; s++)
				    {
						//FOR EACH LINE OF HISTORY CHECK TO SEE IF
IT BELONGS TO DAUGHTER GIVEN BY "a"
						if ((temp_struct.start_index[a] <= s) &&
(s <= temp_struct.stop_index[a]))
						{
							// ALREADY DECLARED IN HEADER

					    case_particle.push_back(particle_no[s]);

					    case_interaction.push_back(interaction[s]);
					     case_nuclei.push_back(nuclei[s]);

					     case_energy.push_back(energy_dep[s]);
					    case_time.push_back(time[s]);
					    case_weight.push_back(weight[s]);
					    case_len++;

						}

				    }
				form_pulse( detector_no);
				 case_particle.clear(); case_interaction.clear();
				case_nuclei.clear();
				 case_energy.clear(); case_time.clear();
				case_weight.clear();
	}
	temp_particle_daughters.clear();
}

void PostProcess::form_numParticle2(int detector_no)
{
	 cout<<"form_num2"<<endl;
	vector<Daughter_info_t> temp_particle_daughters;

	 // COPY PARTICLE INTO A TEMP VARIABLE FOR PROCESSING
	 if (detector_no == first_detector)
		temp_particle_daughters = det1_particle_daughters;
	 else if (detector_no == second_detector)
		temp_particle_daughters = det2_particle_daughters;

	 int sz = (history_no.size() - 1);//for a new history length of vectors are
bigger by 1

	 // EACH OF THE TWO NESTED LOOPS IS FOR EACH DAUGHTER OF A PARTICLE "a" AND
"b"
```

```
        // TOGETHER, THEY GIVE ALL POSSIBLE COMBINATION OF DAUGHTERS FOR TWO
PARTICLE
        //
_____________________________________________________________________________
        for (int a = 0; a<temp_particle_daughters[0].num_daughters; a++)
        {
                for (int b=0; b<temp_particle_daughters[1].num_daughters; b++)
                {
                            int case_len = 0;
                                for (int s=0; s<sz; s++)
                                {
                                        //FOR EACH LINE OF HISTORY CHECK TO SEE IF
IT FITS IN THIS COMBO

                                        if
(((temp_particle_daughters[0].start_index[a] <= s) && (s <=
temp_particle_daughters[0].stop_index[a]))  ||
((temp_particle_daughters[1].start_index[b] <= s) && (s <=
temp_particle_daughters[1].stop_index[b])))
                                        {
                                                // ALREADY DECLARED IN HEADER

                                    case_particle.push_back(particle_no[s]);

                                    case_interaction.push_back(interaction[s]);
                                    case_nuclei.push_back(nuclei[s]);
                                    case_energy.push_back(energy_dep[s]);
                                    case_time.push_back(time[s]);
                                    case_weight.push_back(weight[s]);
                                    case_len++;

                                        }

                                }
                            form_pulse( detector_no);
                            case_particle.clear(); case_interaction.clear();
                            case_nuclei.clear();
                            case_energy.clear(); case_time.clear();
                            case_weight.clear();

                }
        }
       temp_particle_daughters.clear();
}

void PostProcess::form_numParticle3( int detector_no)
{
       vector<Daughter_info_t> temp_particle_daughters;

        // COPY PARTICLE INTO A TEMP VARIABLE FOR PROCESSING
        if (detector_no == first_detector)
              temp_particle_daughters = det1_particle_daughters;
       else if (detector_no == second_detector)
                temp_particle_daughters = det2_particle_daughters;

       int sz = (history_no.size() - 1);//for a new history length of vectors are
bigger by 1
```

```
	// EACH OF THE THREE NESTED LOOPS IS FOR EACH DAUGHTER OF PARTICLES "a","b"
AND "c"
	// TOGETHER, THEY GIVE ALL POSSIBLE COMBINATION OF DAUGHTERS FOR THREE
PARTICLE
	//
_____________________________________________________________________________
	for (int a = 0; a<temp_particle_daughters[0].num_daughters; a++)
	{
		for (int b=0; b<temp_particle_daughters[1].num_daughters; b++)
		{
			for (int c=0; c<temp_particle_daughters[2].num_daughters; c++)
			{
				int case_len = 0;
				for (int s=0; s<sz; s++)
				{
					//FOR EACH LINE OF HISTORY CHECK TO SEE IF IT
FITS IN THIS COMBO

					if (((temp_particle_daughters[0].start_index[a]
<= s) && (s <= temp_particle_daughters[0].stop_index[a]))  ||
((temp_particle_daughters[1].start_index[b] <= s) && (s <=
temp_particle_daughters[1].stop_index[b]))  ||
((temp_particle_daughters[2].start_index[c] <= s) && (s <=
temp_particle_daughters[2].stop_index[c])))
					{
						// ALREADY DECLARED IN HEADER
						case_particle.push_back(particle_no[s]);

					case_interaction.push_back(interaction[s]);
						case_nuclei.push_back(nuclei[s]);
						case_energy.push_back(energy_dep[s]);
						case_time.push_back(time[s]);
						case_weight.push_back(weight[s]);
						case_len++;

					}

				}
				  form_pulse( detector_no);
				 case_particle.clear(); case_interaction.clear();
				 case_nuclei.clear();
				 case_energy.clear(); case_time.clear();
				 case_weight.clear();
			}
		}
	}
	temp_particle_daughters.clear();


}
```