5-2010

The Development of a Parameterized Scatter Removal Algorithm for Nuclear Materials Identification System Imaging

Brandon Robert Grogan
The University of Tennessee, bgrogan@utk.edu

To the Graduate Council:

I am submitting herewith a dissertation written by Brandon Robert Grogan entitled "The Development of a Parameterized Scatter Removal Algorithm for Nuclear Materials Identification System Imaging." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Nuclear Engineering.

John Mihalczo, Major Professor

We have read this dissertation and recommend its acceptance:

Thomas Handler, Laurence Miller, Lawrence Townsend

Accepted for the Council:
Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a dissertation written by Brandon Robert Grogan entitled "The Development of a Parameterized Scatter Removal Algorithm for Nuclear Materials Identification System Imaging." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Nuclear Engineering.

John Mihalczo, Major Professor

We have read this dissertation
and recommend its acceptance:

Thomas Handler

Laurence Miller

Lawrence Townsend

Accepted for the Council:

Carolyn R. Hodges
Vice Provost and Dean
of the Graduate School

(Original signatures are on file with official student records)

THE DEVELOPMENT OF A PARAMETERIZED SCATTER REMOVAL ALGORITHM FOR NUCLEAR MATERIALS IDENTIFICATION SYSTEM IMAGING

A Dissertation Presented for the
Doctor of Philosophy
Degree
The University of Tennessee, Knoxville

Brandon Robert Grogan
May 2010

Acknowledgements

Abstract

This dissertation presents a novel method for removing scattering effects from Nuclear Materials Identification System (NMIS) imaging. The NMIS uses fast neutron radiography to generate images of the internal structure of objects non-intrusively. If the correct attenuation through the object is measured, the positions and macroscopic cross-sections of features inside the object can be determined. The cross sections can then be used to identify the materials and a 3D map of the interior of the object can be reconstructed. Unfortunately, the measured attenuation values are always too low because scattered neutrons contribute to the unattenuated neutron signal. Previous efforts to remove the scatter from NMIS imaging have focused on minimizing the fraction of scattered neutrons which are misidentified as directly transmitted by electronically collimating and time tagging the source neutrons. The parameterized scatter removal algorithm (PSRA) approaches the problem from an entirely new direction by using Monte Carlo simulations to estimate the point scatter functions (PScFs) produced by neutrons scattering in the object. PScFs have been used to remove scattering successfully in other applications, but only with simple 2D detector models. This work represents the first time PScFs have ever been applied to an imaging detector geometry as complicated as the NMIS. By fitting the PScFs using a Gaussian function, they can be parameterized and the proper scatter for a given problem can be removed without the need for rerunning the simulations each time. In order to model the PScFs, an entirely new method for simulating NMIS measurements was developed for this work. The development of the new models and the codes required to simulate them are presented in detail. The PSRA was used on several simulated and experimental measurements and chi-squared goodness

of fit tests were used to compare the corrected values to the ideal values that would be expected with no scattering. Using the PSRA resulted in an improvement of the chi-squared test by a factor of 60 or more when applied to simple homogeneous objects.

Table of Contents

List of Tables

List of Figures

x

List of Abbreviations

Abbreviations

API	associated-particle imaging
APSTNG	associated-particle sealed-tube neutron generator
APT	associated-particle technique
CCD	charge-coupled device
CFD	constant fraction discriminator
COM	center of mass
CSV	comma separated variable
DD	deuterium-deuterium
DT	deuterium-tritium
DU	depleted uranium
FBP	filtered back projection
FBPGUI	Filtered Back Projection Graphical User Interface
FNR	fast neutron radiography
FWHM	full width at half maximum
HEU	highly enriched uranium
IDAS	Integrated Data Analysis Software
ISF	Inter-Array Scatter Function
LOK	Level of Knowledge
MCNP	Monte Carlo N-Particle
MFP	mean free path
NMIS	Nuclear Materials Identification System
PDF	probability distribution function
PMT	photomultiplier tube
PoliMi	Polytechnic of Milan
PSF	Point Spread Function
PScF	Point Scatter Function
PScFGE	Point Scatter Function Generating Equation
PSRA	Parameterized Scatter Removal Algorithm
RAM	Random Access Memory
RSICC	Radiation Safety Information Computational Center
SDEF	source definition
TNR	Thermal Neutron Radiography

1. Introduction

Neutron radiography is a powerful non-destructive technique that can be used to analyze the internal structure of an object. The technique consists of passing neutrons through an object and onto a position-sensitive detector or series of detectors. The relative number of neutrons reaching each position on the detector creates a "shadow" of the object on the detector which can then be used to generate a 2D image of the internal structure of the object. The basic layout of a neutron imaging measurement is shown in Figure 1-1.

If a reference measurement of the detector response without an object in place is available, a 2D attenuation map of the object can be generated using the relation

$$\tau(x,y) = -\ln\left(\frac{I(x,y)}{I_0(x,y)}\right), \tag{1.1}$$

where x and y represent the coordinates on the detector plane, τ is the attenuation of the radiation, I is the intensity with the object present, and I_0 is the reference intensity. I_0 is also referred to as the "void" intensity since the object present is nothing (i.e., a void). The attenuation is measured in mean free paths (MFP), where one MFP is the average distance traveled by a neutron before undergoing an interaction. [1] The MFP is dependent on the energy of the neutron and the material it is traveling through.

The exponential attenuation formula given in Equation 1.1 assumes that once a neutron suffers an interaction, it ceases to contribute to the measurement. This assumption is only valid if the neutron beam is thin and the object-to-detector distance is much greater than the size of the object. [2]
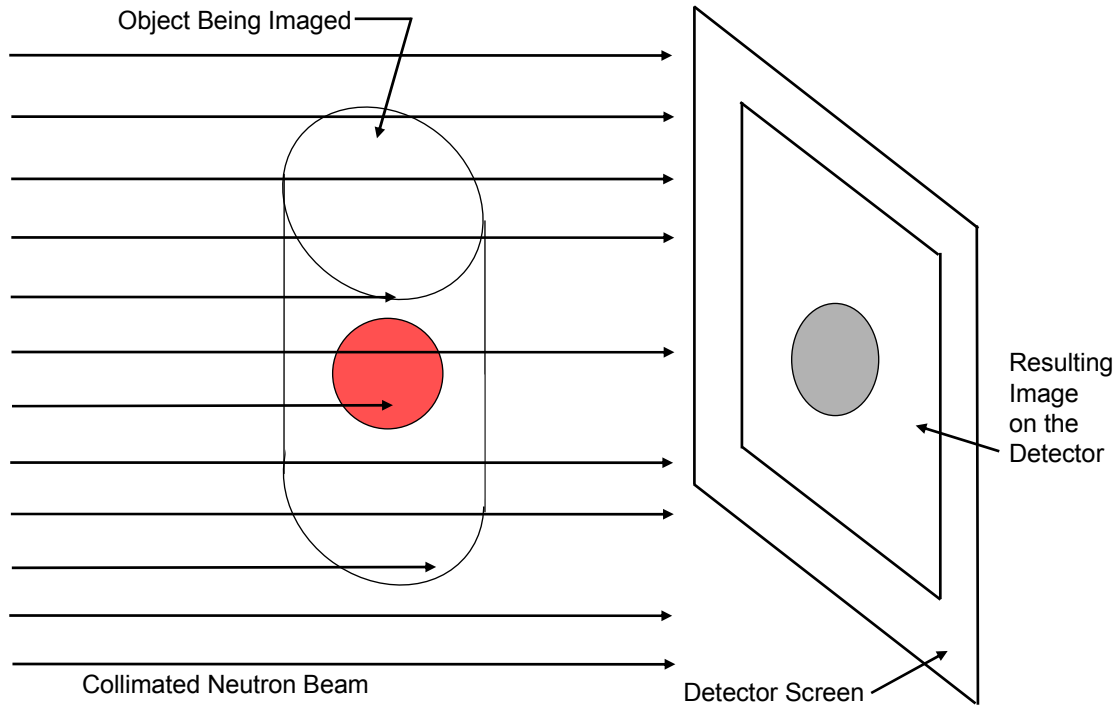
Figure 1-1. A basic neutron radiography layout. Neutrons passing through the object are attenuated, projecting a "shadow" of the object onto the detector screen.

If these conditions are not met, non-absorption events such as scattering can significantly alter the measured attenuation. In addition, neutron scattering from other objects in the room (room return) and background radiation can both contribute to the measured intensities. Taking these factors into account, the measured attenuation, τ_{meas}, can be expressed as

$$\tau_{meas}(x,y) = -\ln\left(\frac{I(x,y)+I_{SO}(x,y)+I_{SR}(x,y)+I_{BG}(x,y)}{I_0(x,y)+I_{SR}(x,y)+I_{BG}(x,y)}\right), \tag{1.2}$$

where I_{SO}, I_{SR}, and I_{BG} are the contributions to the intensity at the detector caused by scattering in the object, scattering in other objects in the room, and background radiation, respectively. These three contaminant terms cause the measured value of attenuation to be lower than the true attenuation in the object.

The Nuclear Materials Identification System (NMIS) uses fast neutron radiography to nonintrusively image the contents of a container. One possible application of NMIS imaging would be to verify the stated contents of a container for treaty verification purposes. Another possible application is the examination a suspicious container suspected of holding fissile material. By correctly measuring the attenuation map of the object, the macroscopic cross section values of materials inside can be determined. These cross section values can be used to help identify the materials in question.

NMIS uses an associated-particle sealed-tube neutron generator (APSTNG) as a neutron source for most imaging measurements. The neutron generator produces neutrons via the ^{2}H(^{3}H,n)^{4}He reaction. The resulting neutrons are monoenergetic and travel away from the reaction site back-to-back with the alpha particle (^{4}He ion) in the center of mass (COM) coordinate system. A pixelated alpha particle detector built into the neutron generator detects the alpha particles produced by the reaction. The alpha particle detector is used to time tag the neutrons and electronically collimate them into a series of small cones opposite the alpha particle detector. Because the neutrons are monoenergetic, the measurement of I and I_0 can be limited to a small time window consistent with the time-of-flight of the neutrons from the reaction site to the detectors.

A diagram of the major components of the NMIS imaging array is presented in Figure 1-2. The object being imaged is placed on a turntable between the deuterium-tritium (DT) generator and the imaging detector array. Three motors allow for the movement of the detector array, the DT generator, and the object in order to produce a full 3D tomographic image of the internal structure of the object if desired.
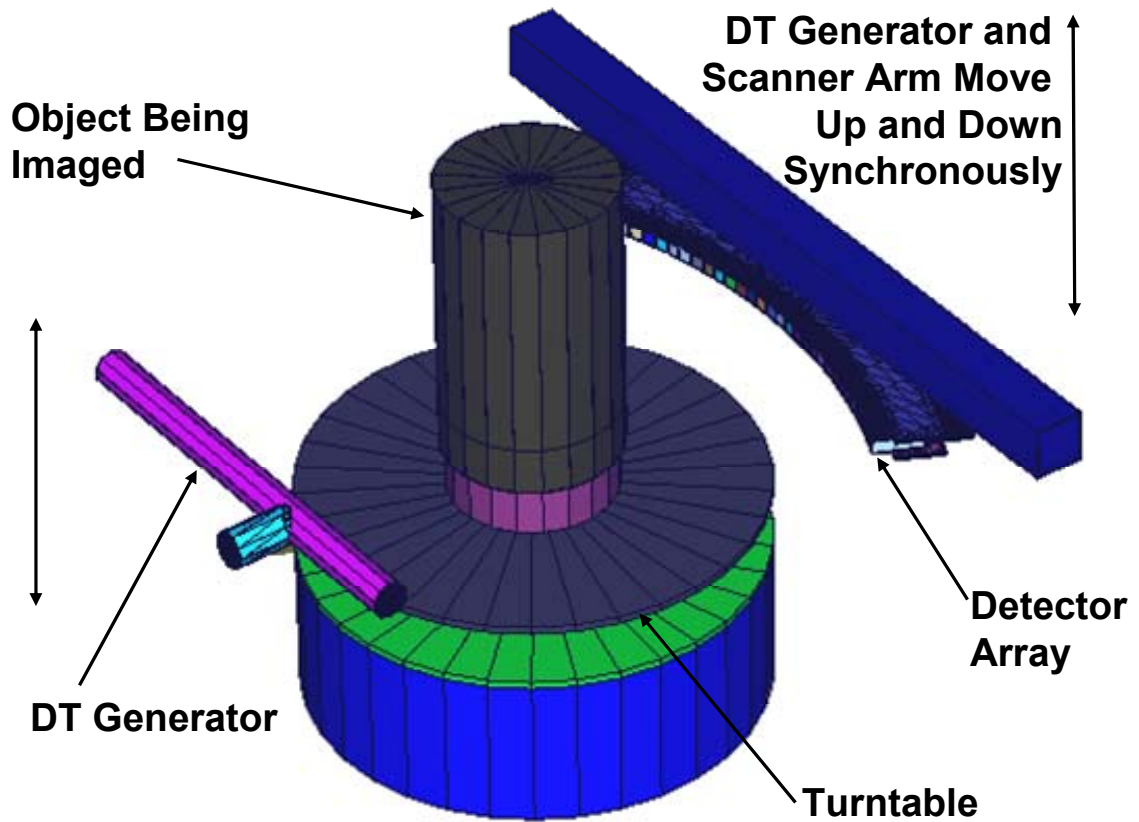
Figure 1-2. The major components of the NMIS imaging subsystem. Neutrons are produced by the DT generator, where the pixelated alpha particle detector time tags and electronically collimates them. The detector array then measures the transmission of neutrons through the object being imaged as a function of detector angle. The scanner can be moved up and down for 2D radiography, and the object can be rotated by the turntable for full 3D tomographic imaging. The detector array can be rotated laterally for subsampling.

The first motor rotates the turn table to allow for taking multiple projections through the object at different rotational angles. The second motor moves the entire imaging array and the DT generator up and down synchronously in order to produce images at multiple heights. The third motor rotates the entire detector array laterally. This motion is used to generate multiple images with the detectors in a slightly different position each time. All the images can be interlaced together to improve the horizontal angular resolution. For example, on the 110 cm radius arm, the angular separation between detectors is 1.67°. By making an image with the detectors in one position and then rotating the array by 0.835° and making another, the two images can be combined to halve the angular resolution. This procedure is known as subsampling the detectors. The detector positions are typically subsampled four to ten times at each height.

A typical time-of-flight curve for DT neutrons is shown in Figure 1-3. This curve is a time correlation between the alpha particle detector and an imaging detector 110 cm away. The fusion reaction occurs at $t = 0$ on the graph. The large peak (note the logarithmic scale) at approximately 22 ns is produced by the directly transmitted neutrons which travel at approximately 5.2 cm/ns. Because of the finite detector size and uncertainty of the electronic timing ($\sim \pm 1$ ns), a 5 ns wide correlation window is used to identify directly transmitted neutrons.

In order for scattered neutrons or background radiation to be mistaken for directly transmitted neutrons, they must arrive during one of the 5 ns correlation windows corresponding to an alpha particle.
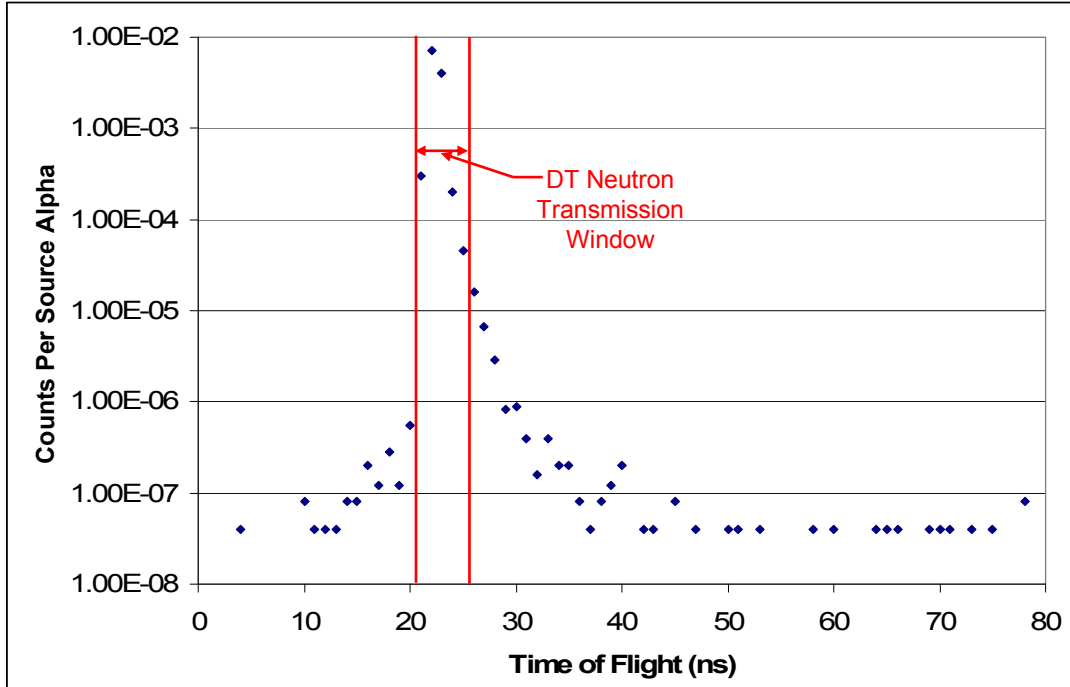
Figure 1-3. A time correlation curve between the alpha detector and a detector 110 cm away. The fusion reaction occurs at $t = 0$ ns on the graph. Note the logarithmic scale.

By keeping the rate of neutron production low enough that the average time between alpha counts is much greater than the width of the correlation window, the magnitude of the three contaminant terms in Equation 1.2 is greatly reduced. Because the background radiation is random, its effect on the measurement will be reduced by a factor of

$$f_{util} = \frac{N \Delta t_{window}}{T}, \tag{1.3}$$

where f_{util} is the utilization factor, N is the total alpha rate in all pixels, Δt_{window} is the width of the correlation window, and T is the total measurement time. Utilization factors of $<10^{-3}$ are typical with the neutron generator used for NMIS measurements. This reduction applies to both the passive background and the active background produced by uncorrelated neutrons falling outside the pixel cones. Correlated neutrons scattering from objects other than the one being imaged typically arrive at times larger than the maximum

6

of the correlation window with the exception of scattering from objects near the detector array. This contribution can be kept to a minimum by removing unnecessary objects from around the detector array.

Another technique used to prevent scattered neutrons from contributing to NMIS measurements is an anticoincidence technique designed to eliminate multiple detector counts in the same transmission time window. For imaging measurements, multiple counts in the same window are typically caused by cross-talk between detectors. In order to eliminate this cross talk, the anticoincidence correction keeps only the first count in an imaging detector for each neutron transmission window. Other counts are rejected as cross talk and ignored. An example of a single neutron generating multiple counts in the same directly transmitted neutron time window is shown in Figure 1-4.

The final contaminant is scattering in the object being imaged. The removal of this term is the greatest technological challenge to NMIS imaging today, and it will be the focus of this work. In a thick object, only a small portion of the DT neutrons are directly transmitted to the detector array. For example, if an object is 5 MFP thick, only 0.7% of source neutrons are directly transmitted to the detectors. If even a small fraction of the 99.3% of neutrons that interact scatter and reach the array during the correlation window, the measured attenuation will be significantly lower than the true value. This effect is accentuated when the object being imaged is close to the detector array or if it is composed of high atomic mass (high Z) materials. Neutrons scattering off high Z isotopes are preferentially forward scattered and lose only a small fraction of their energy in an elastic scattering event, so a large fraction of them will arrive in the 5 ns transmission window.

Count 1 – 21.2 ns
Count 2 – 22.4 ns
Count 3 – 23.1 ns

Incoming
Neutron
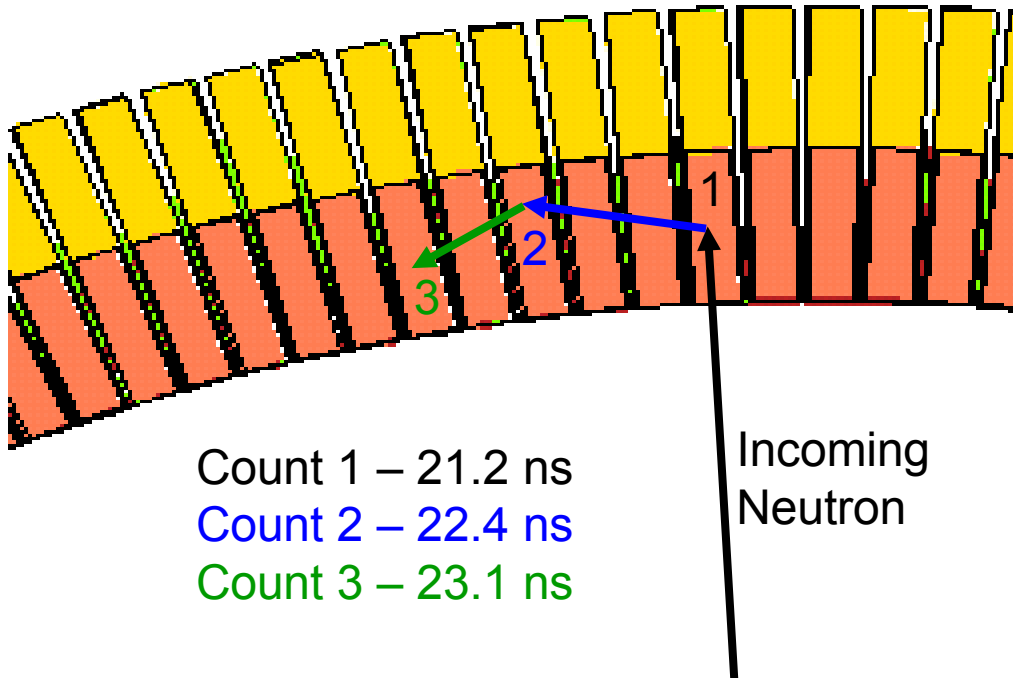
Figure 1-4. An example of a neutron producing multiple counts during the same neutron transmission window. Collision times are measured from the time of the associated alpha particle count. All three counts lie within the 5 ns time window. Only count 1 would be used for measuring attenuation in this example. Counts 2 and 3 would be treated as cross talk and ignored.

An object designed to produce a large object scattering effect is presented in Figure 1-5. A simulated 1D attenuation map resulting from a single horizontal slice through the object is shown in Figure 1-6. The "Ideal" curve was calculated using the known geometry, material composition, and 14 MeV cross sections of the object using the exponential attenuation formula given in Equation 1.1. The other curves show the measured values produced by various alpha detector geometries. From Figure 1-6, it is clear that subdividing the alpha detector signal reduces the effect of the object scattering, but even with 16 pixels, the measured attenuation is more than 1 MFP lower than the true value at the highest attenuation levels.

In this work, the scattering produced by objects composed of various types and thicknesses of materials at different object-to-detector distances from the NMIS detector array will be characterized. This will be accomplished by using Monte Carlo N-Particle (MCNP) simulations to measure the additional response per source neutron in each detector of the array as a function of the angle of scattering. This type of function is known as a point scatter function (PScF) and has been used successfully to reduce object scattering in other neutron radiography applications (See Chapter 2). Once the PScFs are measured, the resulting curves will be parameterized and the parameters will be fit using the scattering material, thickness, and object-to-detector distance as input variables.

These multivariate fit equations will be referred to as PScF Generating Equations (PScFGEs), and they will be used to determine the appropriate PScF based on inputs from the operator. The PScFGEs will be the primary component of the parameterized scatter removal algorithm (PSRA), which will remove the object scattering from the measured values and return the corrected attenuation values.
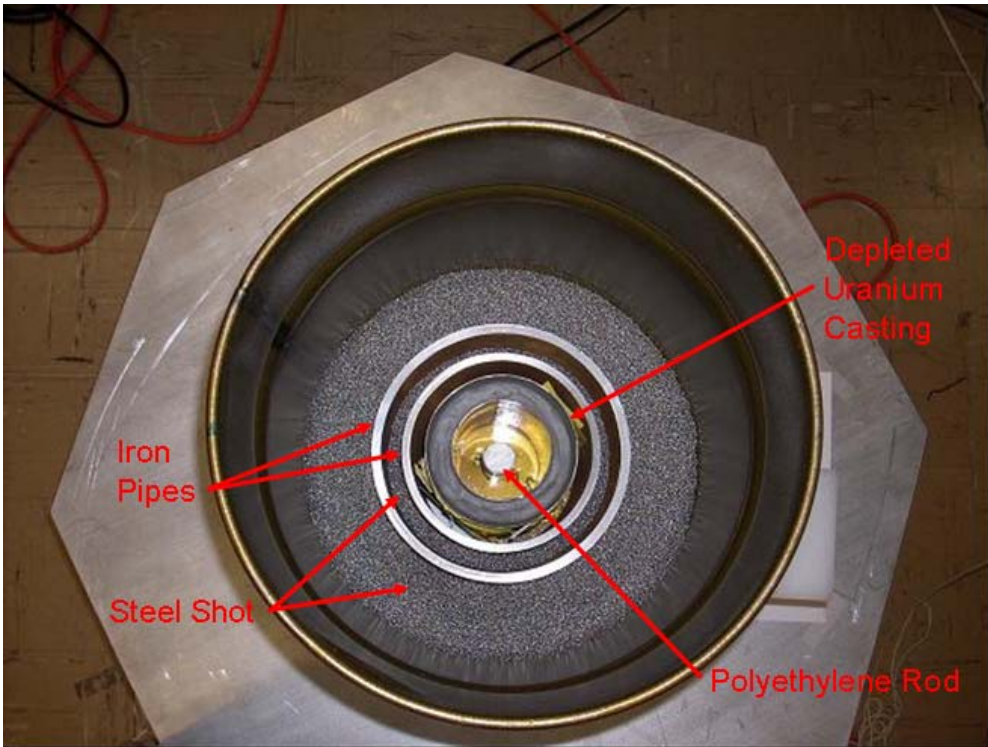
Figure 1-5. Overhead view of an object designed to accentuate the effects of forward scattering. The object is composed primarily of high Z materials.
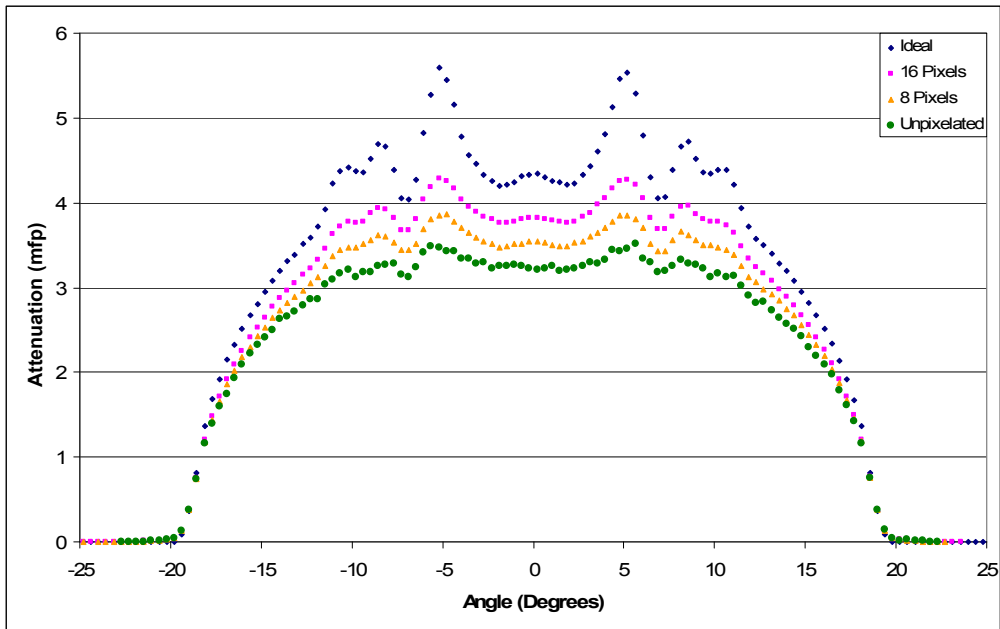
Figure 1-6. Attenuation curves of the object shown in Figure 1-5 using different levels of alpha detector pixelization. The "Ideal" curve assumes pure exponential attenuation using Equation 1.1.

This work will be significantly different from any application of PScFs to remove scattering that have been done before. In order to develop PScFs for NMIS imaging, the differences in source geometry, detector geometry must be accounted for. Previous work used a parallel beam source and a flat detector screen. By contrast, NMIS uses multiple pixelelated neutron cones and an array of scintillators. In addition, time-of-flight techniques are used to limit the number of scattered and background neutrons that contribute to the measured values. Therefore, the PScFs must consist only of scattering that arrives during the directly transmitted neutron windows. In order to simulate NMIS imaging and compute the value of these PScFs, new techniques had to be developed. The methodology used for simulating the measurements is presented in detail.

In the following chapters, this methodology will be explained in detail and the results will be presented. The history of NMIS imaging and previous attempts to remove scattering from radiography measurements are reviewed in Chapter 2. In Chapter 3, calculations designed to characterize the alpha detector pixels and provide a first order estimate of the object scattering effect on NMIS imaging is presented. The simulation methodology and each of the codes written in order to properly simulate an imaging measurement are discussed in Chapter 4. In Chapter 5, the simulations used to generate the PScFs are presented and the resulting scattering curves and their parameters are discussed. The resulting values of the PScF parameters are presented and used to develop the PScFGEs in Chapter 6. The development of the PSRA and its coding into a program to remove the scatter from measurements is also presented in Chapter 6. In Chapter 7, the PSRA is tested using both simulated and laboratory measurements.

Finally, the conclusions of the work are presented and future work that might improve

this technique is discussed in Chapter 8.

2. Review of Previous Work

<u>2.1 The Nuclear Materials Identification System</u>

The Nuclear Materials Identification System (NMIS) uses a fast processor to compute time correlations between detector signals. [3] The current processor acquires data from 10 input channels at a rate of 1 GHz. [4] When first developed and fielded in the mid- to late-90s, the NMIS was used to verify weapons components or determine the quantity of fissile material inside a sealed container. [5, 6] Although some measurements were conducted passively, [7] most used active interrogation to induce fissions in the fissile material. A ^{252}Cf spontaneous fission source mounted inside an ionization chamber allowed the processor to record the time of (time tag) each spontaneous fission. [3] One common measurements made was the time correlation between the ^{252}Cf source and a detector on the opposite side of the object. [5] Because of their different speeds, gamma rays and neutrons of various energies arrive at the detectors at different times after the spontaneous fission. If the source-detector separation is known, a specific window of time lags between the source and detector can limit the measurement to either gamma rays or a certain energy range of fission neutrons. [3] By comparing the number of correlations in the region of interest with and without a sample between the source and detector, the average attenuation for gammas and neutrons can be measured simultaneously.

In 2003, imaging capabilities were added to NMIS by adding multiple small-area detectors. [8] By measuring the transmission of radiation through the object as a function of position, a 2D attenuation map can be generated. [9] With a tagged ^{252}Cf source, both

gamma and neutron radiography can be conducted simultaneously by using multiple time windows of each source-detector time correlation. [8] These radiographs can subsequently be converted to 3D tomographs by taking multiple projections through the object at different angles or by assuming cylindrical symmetry. [8]

The next improvement to NMIS radiography capabilities came in the form of an improved source. A sealed-tube DT neutron generator produces monoenergetic 14.1 MeV neutrons. [9] An integral alpha particle detector electronically collimates and time tags each neutron produced within a desired solid angle. This is referred to as the associated-particle technique (APT). [10] Because the neutrons are approximately monoenergetic, most scattered neutrons and induced radiation can be eliminated by limiting transmission measurements to a small time window corresponding to the neutron time-of-flight from the generator to the detector.

The neutron generator currently used with NMIS is an API-120 produced by Thermo Scientific. [11] The target and alpha detector geometry define a cone approximately 45° wide. The imaging detectors consist of a horizontal array of 32 fast plastic scintillators. As many as 64 detector signals can be sampled simultaneously with the NMIS processor by combining signals with a fan-in module. Initially, a light opaque plastic mask was placed between the alpha detector and the photomultiplier tube. Various mask shapes were used to further collimate the neutrons into smaller cones or fan beams directed towards the detector array. [5, 12] More recently, a pixelated photomultiplier tube (PMT) has been coupled to the alpha detector to subdivide the alpha detector response into an 8×8 or a 16×16 array of pixels. [13] The NMIS uses a horizontal array of imaging detectors, so only a single row of 8 or 16 horizontal pixels are

used. This row of pixels is aligned so that the center of each is in line with the plane of the imaging detectors.

2.2 Previous Scattering Corrections in Thermal Neutron Radiography (TNR)

The field of radiography began in 1895 with the first X-ray radiogram of Anna Bertha Roentgen's hand. [14] Neutron radiography got off to a much later start. The first neutron radiographs were made by Kallmann and Kuhn in Berlin beginning in 1938. [15] Another facility in Berlin under O. Peter began making radiographs in 1944. [15] These facilities were destroyed by the Allies at the end of the Second World War and nuclear research in Germany did not restart until 1955. [15] Serious study of neutron radiography did not resume until about 1960 [16], and the technique was not standardized until the publication of the *Neutron Radiography Handbook* in 1981. [17]

Heller and Brenizer give a description of conventional neutron radiography. [16] Neutrons are first produced using an accelerator, radioisotope, or reactor source. In these previous studies, the neutrons are thermalized to the thermal/epithermal range (0.025 eV-10 keV) using a moderator and then collimated into a beam. The beam is passed through a sample and into a detector where the radiograph will be formed. Because of their low energy, these neutrons require a converter material to produce easily detectable radiation. The converter typically contains a material such as gadolinium which produces light through a capture reaction. This light is then recorded on film, a video camera, or a charge-coupled device (CCD) to produce the radiograph.

Segal et al. made one of the first non-analytical attempts to correct scatter in thermal neutron radiography in 1982. [18] They used a point spread function (PSF) to

15

quantify the contribution of scattered flux to each point in the imaging plane from each point on the surface of the sample. Although the PSF accounts for other factors such as imaging geometry and scatter from sources other than the sample, Segal et al. focused only on the scatter occurring in the sample. Traditionally, the PSF was solved directly using a Fourier transform, but this required the use of a general statistical operator, which the authors found unsatisfactory. A Monte Carlo code (Morse-CG) was used to simulate a monoenergetic, infinitely thin beam of neutrons impinging on infinite slabs of different materials and thicknesses. Two separate sample-to-detector separations were simulated. The scatter of neutrons away from the beam axis was tallied as a function of distance from the projected point on the detector. The PScF for each material and geometry was computed by fitting the fraction of scattered neutrons versus radial scattering distance using an exponential of the form $S(r) = Ae^{-\beta r}$. A and β were the fitting parameters measured by the simulation and they were presented in tabulated form for the various scenarios modeled.

In 1990, Hrdlicka and Peterka published a paper describing an experimental method for measuring and subtracting scatter from a thermal neutron radiograph. [19] By moving the sample far enough away from the imaging plane, they found that the scatter was spread uniformly across the detector. This scatter could then be determined by comparing the flux at a point on the detector outside the shadow of the sample with and without the sample in place. This difference could then be subtracted from both the transmitted (I) and direct (I_0) fluxes to remove the scatter. Hrdlicka and Peterka stated that while the minimum source-to-detector distance for using their method could be estimated mathematically, it was best verified experimentally.

In 1992, Kobayashi et al. suggested that the scatter could be estimated by using a strip of cadmium between the source and the sample. [20] Because cadmium is opaque to thermal neutrons, no directly transmitted neutrons would reach the umbra behind the cadmium strip and the scattering contribution could be measured directly there and then subtracted. Using this technique, they were able to estimate the effective total macroscopic cross section of homogenous samples to within ± 20% for most materials.

Two years later, Murata et al. used the technique of Kobayashi et al. for imaging. [21] A series of cadmium strips was laid across a concentric stack of disks. The neutron flux in the umbra shadows was measured and subtracted from the direct and uncollided fluxes to subtract the scatter from the radiograph. The areas lost to the cadmium strips were then reconstructed by interpolation. This was the first experimental scatter subtraction from a 2D thermal neutron radiographic image. [22]

In 1996, Tamaki et al. focused on the performance of several collimator designs for the qualitative removal of neutron scatter. [23] They designed a new gadolinium honeycomb collimator to obtain quantitative attenuation coefficients in slabs of sample material.

In 1999, Pleinert et al. used a Monte Carlo code to estimate the scatter in various samples. [24] They started by deriving a signal transfer function that mathematically transformed the detector response without a sample (I_0) into the response with a sample present (I). The signal transfer function used a discrete point attenuation function, point spread function, and an energy dependent detector response function to map the response at each detector pixel due to the attenuation and scatter of neutrons in each 2D pixel of the sample. The MCNP-4A code [25] was used to determine the PSFs and PScFs and

then a continuous signal transfer function was determined. The authors' motivation was the unknown moisture content in samples of known composition. Simulations showed that the signal transfer method measured the moisture content to within 3% of the true value, compared to 10% using a calibration curve and approximately 100% using uncorrected values.

Kardjilov et al. extended on the method of Pleinert et al. by determining the scattered response at each discretized point on the detector due to each discrete point on the source. [26] They described the portion of the detector response due to scattering in the sample as a point scatter function (PScF). The group used the MCNP-4B code [27] to simulate the imaging of objects at the Neutron Transmission Radiography (NEUTRA) facility [28] at the Paul Scherrer Institute (PSI) in Villigen, Switzerland. The simulated objects included a slab and a step-wedge made of Plexiglas. The PScF was computed for each simulation and then an identical experimental measurement was conducted. The PScFs were applied to the measurements and the resulting attenuation curves were within a few percent of the true values.

Hassanein et al. expanded on the work of Kardjilov et al. to account for the differing path lengths of scattered neutrons as a function of angle. [29] The angle affects both the intensity of scattered neutrons and the detector response due to an increased angle of incidence. The MCNPX 2.4.0 code [30] was used to derive the PScF for different sample configurations. The PScFs were then mathematically weighted to account for the angular dependence of the detector response. The authors noted that the PScF depended on the sample material and geometry, energy spectrum, detector, and

sample-to-detector distance. They recommended that a library for the PScFs of various materials should be constructed for use in future experiments.

De Beer et al. used the methods of Kardjilov et al. and Hassanein et al. to parameterize the PScFs due to the thickness of a water sample and sample to detector distances. [22] They used the MCNP-4C code [31] to determine the PScF for each configuration and then used a Gaussian function to fit it. The full width at half maximum (FWHM) and maximum of the Gaussians were fit versus thickness and object-to-detector distances. These curve fits could then be used to generate the PScF and subtract it from measurements for a given sample thickness and object-to-detector distance without the need to rerun Monte Carlo simulations.

2.3 Previous Scattering Corrections in Fast Neutron Radiography (FNR)

While thermal neutron radiography has achieved a degree of relative maturity, fast neutron radiography (FNR) is still not a commonly used procedure. [32] This is largely because of the initial difficulty in capturing the fast neutron response using thermal radiography techniques. E. Tochilin tested fast neutron radiography with traditional film techniques in 1965 and wrote that the poor response of photographic films to neutrons with energies >10 keV presented serious technical difficulties and the buildup of scattered neutrons and induced gammas limit the technique to objects less than a few inches thick. [33] Tochilin suggested that organic scintillators offered a better response by acting as image intensifiers. [33] In 1970, H. Berger examined several techniques including direct film, film coupled with image intensifying screens, track-etch detectors, and scintillator screens coupled with film. [34] Berger concluded that the

19

scintillator-film method produced the best contrast and was less sensitive to secondary radiation than other techniques. [34]

J. S. Brzosko et al. wrote in 1992 that the previous problems of fast neutron source strength and detection systems "have now been overcome." [35] The authors used a Monte Carlo code (3D-MCSC-RWR [36]) to simulate neutron radiographs on a scintillator fiber detector with fast and thermal neutrons and compared the results. They concluded that FNR was the only viable option for slabs >5 cm thick and that FNR offered an advantage in portable source strength because neutrons are not lost in the thermalization process[1]. Brzosko et al. reduced the problem of neutron scatter and gamma ray contamination in their FNR simulations by using a 7 MeV threshold for their scintillators.

In 1994, Rhodes et al. described a rudimentary 1D radiograph that they constructed using an associated-particle DT neutron generator and a single imaging detector. [37] Two hollow lead spheres were placed in sand between the imaging detector and the APSTNG. The counts in the detector corresponding to the DT neutron time-of-flight were recorded as a function of the horizontal position of the detector to generate an attenuation map. The authors reported that because the imaging detector was sufficiently far from the sample, scattering was negligible.

One of the first attempts to quantify the scattering in FNR was made by Yoshii and Kobayashi in 1996. [38] Their work sought to apply the method of Hrdlicka and Peterka [19] to FNR. The authors tested the detector response in the center of the neutron

---

[1] Fixed, thermal reactor-based radiography systems do not suffer from this limitation because they have a ready supply of high flux thermalized neutrons. [16]

20

shadow produced by cylindrical iron samples of various thicknesses and sample-to-detector distances. They concluded that the scattered neutron intensity at the center of the shadow produced by a 3 cm thick sample dropped to a negligible level if it were located a distance of twice the radius plus 1 cm from the detector screen.

Two papers published in 2002 by Rahmanian et al. [39] and Ambrosi and Watterson [40] in 2004 used ray tracing and the MCNP-4A code [25] to study the effects of source geometry and neutron scatter on FNR. These papers focused primarily on the effects of source and detector geometry, but they concluded that while scattering lowered the overall contrast of a feature in the sample, it did not affect the resolution.

In 2009, Hassan used Monte Carlo simulations to estimate the PScF of various samples using ^{252}Cf, deuterium-deuterium (DD), and DT sources. [41] His work was based off the methodology developed for thermal neutron radiography by Kardjilov et al., [26] Hassanein et. al, [29] and de Beer et al. [22] Unlike the previous TNR work which focused mainly on hydrogenous materials, Hassan used primarily metallic objects. He showed that various parameters, such as the shape of the sample, material, source type, sample-to-detector distance, and beam divergence could be used as fitting parameters for the FWHM and maximum of the Gaussian PScF. Hassan examined each of these effects separately and suggested that future work could be used to determine the correct PScF using sample parameters without the need for a simulation.

One fast neutron imaging application that bears a strong resemblance to the NMIS detector layout is the fast neutron / gamma-ray radiography scanner developed by The Commonwealth Scientific & Industrial Research Organisation [*sic*] (CSIRO). [42-44] The CSIRO scanner uses a high intensity pulsed DT neutron source and ^{60}Co gamma ray

21

source to image cargo containers at the Brisbane International Airport. The neutron and gamma images are formed separately, and material compositions are determined by measuring the ratio of the measured neutron and gamma attenuations. [42] The neutron detector array consists of 192 2 cm × 2 cm plastic scintillators in a 2D array. [42] Liu et al. reported that they observed a deviation of approximately 10% from narrow-beam geometry due to scattering. [44] They further stated the scattering was "highly local, between neighbouring [*sic*] pixels and can be corrected rather accurately." [44] No details of the methodology used were given. Unlike NMIS the associated particle technique is not used to time tag the neutrons, but the neutrons are physically collimated by a steel collimator. [42]

3. Initial Calculations

Before proceeding to the development of the parameterized scattering removal algorithm (PSRA), it is necessary to perform some initial calculations. The results of these calculations will prove invaluable for modeling NMIS imaging measurements and understanding the physics behind the scattering. In Section 3.1, the DT neutron pixels will be examined in detail in order to develop a high fidelity MCNP model. In Section 3.2, a first order approximation of the PScFs for simple cases will be derived using basic physics principles. These calculations will serve to validate the more robust models that will be developed using MCNP simulations and to help provide insights that may be useful for developing those simulations.

3.1 Determining the Profile of the API-120 Pixels

In order to accurately simulate NMIS imaging, the distribution of the DT neutron pixels needs to be accurately characterized. This will be accomplished by combining the kinematics of the ^{3}H(^{2}H,n)^{4}He reaction with the geometry and operational characteristics of the API-120 neutron generator and the H8500 PMT. This information will be used to develop a Monte Carlo code to simulate the neutron cones corresponding to each alpha pixel. Once these pixel profiles are calculated, they will be validated against experimental results and then coded into *MCNP-PoliMi* decks to simulate imaging measurements.

3.1.1 The Kinematics of the DT Reaction

The Thermo Scientific API-120 neutron generator produces neutrons via the reaction

$$D + T \rightarrow \alpha + n + Q, \tag{3.1}$$

where D is a deuterium (^{2}H) atom, T is a tritium (^{3}H) atom, α is an alpha (^{4}He) particle, n is a neutron, and Q is the mass difference between the reactants and the products. The API-120 generator produces this reaction by accelerating ions from a DT plasma source into a fixed target with embedded deuterium and tritium atoms. For this reaction, the Q-value is 17.5893 MeV. [45] The masses of the alpha particle and neutron are 4.00260 and 1.00866 AMU, respectively. [46] In the COM coordinate system, the kinetic energies of the products are

$$T_\alpha = \frac{m_n}{m_n + m_\alpha} Q = 0.201279 Q = 3.54035 \, \text{MeV} \tag{3.2}$$

$$T_n = \frac{m_\alpha}{m_n + m_\alpha} Q = 0.798721 Q = 14.0489 \, \text{MeV} \,. \tag{3.3}$$

Because there are only two products, the neutron and alpha particle leave the reaction site back-to-back in the COM coordinate system. The momentum of the incoming deuteron or triton is transferred to the products causing the angle between them to be folded forward so that it is less than 180° in the LAB coordinate system. This effect also causes the kinetic energy of the outgoing particles to have an angular dependence with particles in the forward direction (<90° from the direction of the incoming deuteron or triton) having greater energy than shown in Equations 3.2 and 3.3 and particles in the backward direction having less.

A diagram of the DT reaction is presented in **Figure 3-1**. A deuteron or triton comes in traveling in the –y direction and fuses with a deuterium or tritium atom. In the COM system, the alpha particle and neutron move away from the fusion site back-to-back at an angle θ with respect to the x-axis in the x-y plane. The COM velocity vectors are converted to the LAB frame by adding the velocity of the center of mass of the deuteron and triton to the velocity vectors of the alpha particle and neutron. This folds the resulting velocity vectors forward and the alpha particle and neutron now form angles of φ and ψ, respectively, with the x-axis.

For the scenario shown in **Figure 3-1**, the COM velocity can be computed using the equation [47]

$$V_{CM} = \frac{m_1 v_1}{m_1 + m_2},$$
(3.4)

where m_1 is the mass of the incoming particle moving at a velocity of v_1, and m_2 is the mass of the stationary particle. Using classical dynamics, the speed of the incoming particle can be calculated using the equation

$$v = \sqrt{\frac{2E}{m}},$$
(3.5)

where E is the kinetic energy of the particle and m is its rest mass. For example, a deuteron with a kinetic energy of 90 keV has a speed of 0.009795c, where c is the speed of light in a vacuum. Using the same equation, the speeds of the neutron and alpha particle are 0.173015c and 0.043564c, respectively. The neutron speed calculated using classical dynamics differs from the relativistic result by only 1%. The deuteron and alpha particle results are even closer because of their lower speed.

25
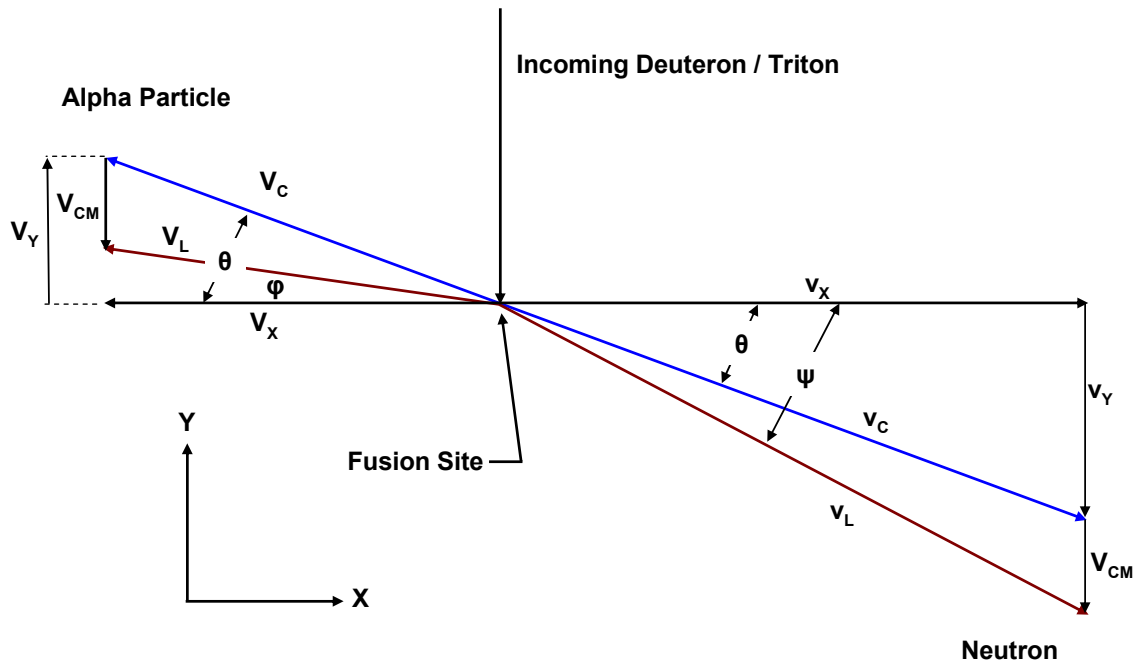
Figure 3-1. A diagram of the DT reaction and its products. A particle traveling in the –y direction initiates a DT fusion resulting in an outgoing alpha and neutron. In the COM system, these particles move away from each other back-to-back at an angle of θ with respect to the x-axis. The momentum of the deuteron or triton which triggered the DT reaction causes the alpha and neutron angles to be folded forward to angles φ and ψ in the LAB coordinate system.

These results indicate that classical dynamics can produce sufficiently accurate results for the problem here. For the 90 keV deuteron incident on a triton, the velocity of the COM would then be

$$V_{CM} = \frac{2.014}{2.014 + 3.016} 0.009795c = 0.003922c .$$

(3.6)

From **Figure 3-1**, it can be seen that the angle of the alpha particle in the LAB coordinate system can be related to its angle in the COM system via the relation [47]

$$\tan \phi = \frac{V_y + V_{CM}}{V_x} = \frac{V \sin \theta + V_{CM}}{V \cos \theta}.$$

(3.7)

Because V and V_{CM} are known, Equation 3.7 can easily be solved for φ if θ is known. This relation will be used in Section 3.1.4 to calculate the LAB angles of the neutron and alpha particle given the COM angle.

3.1.2 Determining the Mean Center of Mass Velocity

The API-120 generator produces neutrons by accelerating a plasma consisting of deuterium and tritium ions onto a titanium target that has adsorbed a DT gas. Because the mass of the incoming particle will differ depending on whether an incoming triton or deuteron initiates fusion, the COM velocity will also be different. Because both the incoming plasma and the gas in the target are mixed, the DT reaction competes with the DD and TT reactions. However, at low energies, the DT cross section is approximately two orders of magnitude greater than these competing reactions. For this reason, the API-120 operates with a maximum voltage of 90 kV, which accelerates deuterons and

tritons to energies near the peak of the DT cross section. The data in Figure 3-2 shows the relative cross sections of the three reactions. [45]

According to the manufacturer, the majority of excited plasma consists of molecular (DD^+, DT^+, TT^+) rather than atomic ions. [10] These singly ionized molecules have a greater mass and thus a lower velocity than the atomic ones. In addition, when one of the atoms in the molecule fuses, the second one will continue on and not contribute its momentum to the resulting neutron and alpha particle. This also results in a lowering of the cross section because the energy of the fusing particle is only its mass fraction of the total molecular energy. If the titanium atoms are much more numerous than the deuterium and tritium atoms, the incoming ions are equally likely to have any energy between 0 and the initial beam energy when they encounter a deuterium or tritium atom. The average energy of the ion when it initiates a DT fusion can be calculated using the equation

$$\overline{E} = \frac{\int_0^{E_{max}} E\sigma(E)dE}{\int_0^{E_{max}} \sigma(E)dE}, \tag{3.8}$$

where E_{max} is the incident energy of the ion and $\sigma(E)$ is the DT reaction cross section. Using this equation, a deuteron with an incident energy of 90 keV would have an average energy of 70.5 keV when initiating DT fusion. Using Equations 3.5 and 3.6, the resulting center of mass velocity is 0.003471c, which is only 89% that of the incoming deuteron is at its initial beam velocity.
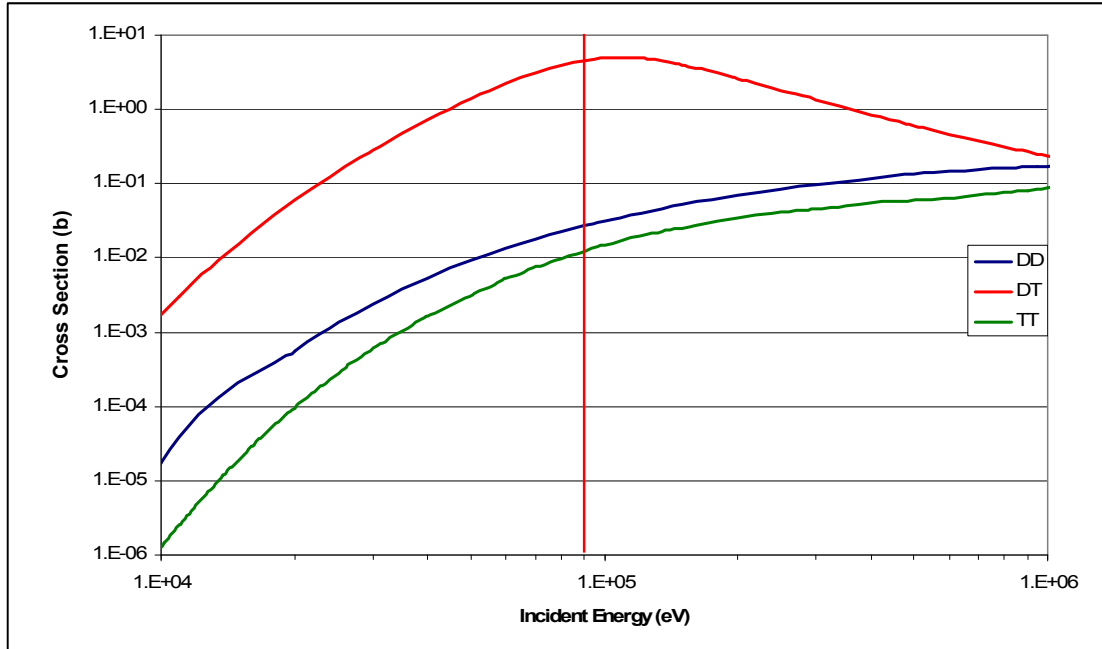
Figure 3-2. The DD, DT, and TT fusion cross sections from 10 keV to 1 MeV. The DT reaction dominates over most of this region. The vertical line shows the maximum accelerator voltage (90 keV) produced by the API-120 generator.

In order to calculate an average COM velocity, the following assumptions were made. First, the accelerating voltage was assumed to be 90 keV, which is the typical operating voltage of the API-120. All ions in the accelerated beam were assumed to have a kinetic energy exactly equal to the accelerating voltage. The composition of ions in the beam was assumed to be 10% atomic and 90% molecular. The atomic portion consists of an equal quantity of deuterium and tritium ions. The molecular ions consist of 25% DD^+, 25% TT^+, and 50% DT^+. Another assumption was that the slowing of the incoming ions did not change their direction vector significantly. Using the SRIM-2008.04 code [48], the average range of 90 keV deuterons in titanium is 680 nm and the range of 60 keV deuterons is 500 nm. The penetration depth of a beam of 90 keV deuterons incident on a 180 nm (680 nm – 100 nm) titanium slab is depicted in Figure 3-3.

29
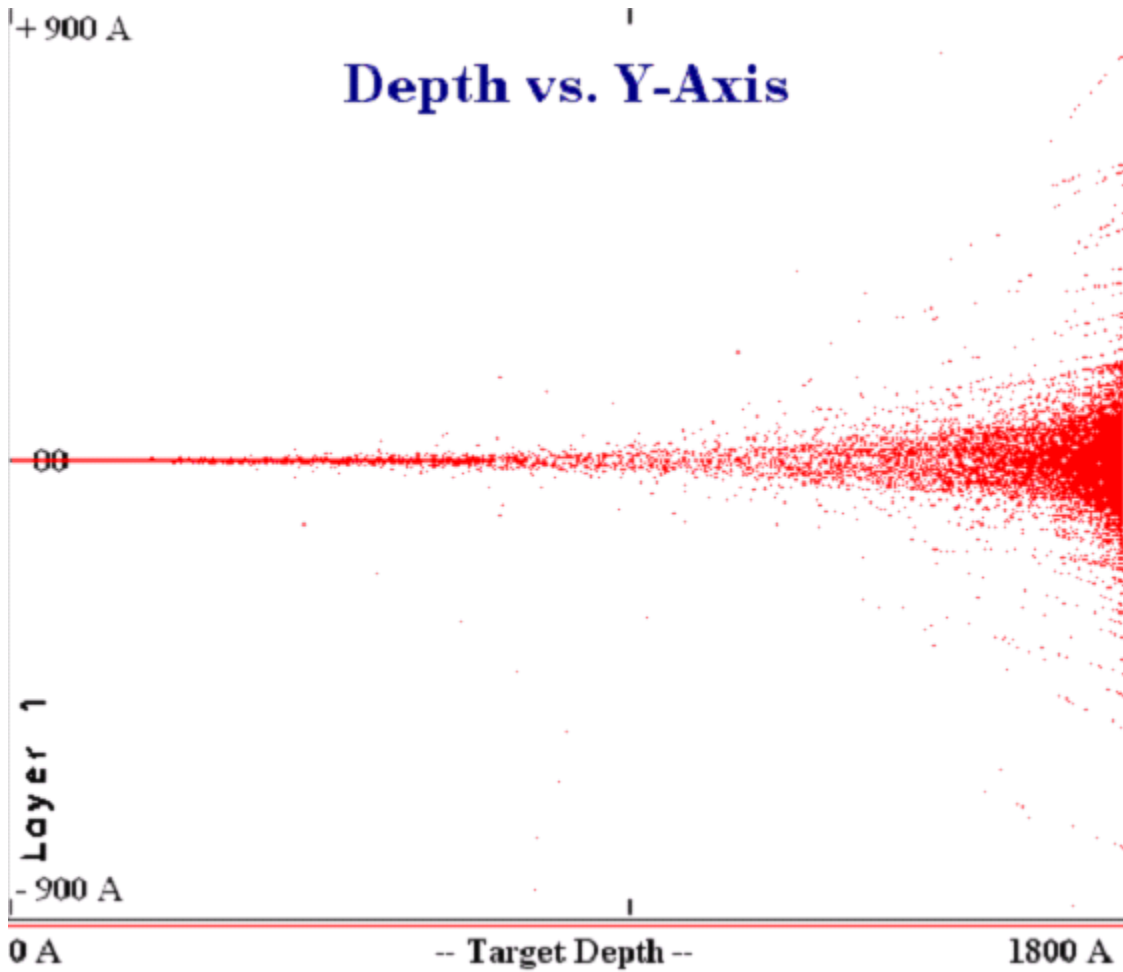
Figure 3-3. Results from a SRIM simulation of 90 keV deuterons incident on a 180 nm thick titanium target. This thickness approximates the depth required to slow a deuteron from 90 keV to 60 keV. The plot shows only a relatively small divergence of the deuterons as they travel through this distance.

The deuteron beam shows only a small divergence from the initial direction, even at the back side where the average energy has been slowed to approximately 60 keV.

One final assumption is that the magnitude and direction of the outgoing alpha particle velocity will not be significantly changed exiting the titanium target. Using the SRIM code, the average depth of interaction for a 70.5 keV deuteron (average energy at interaction for an incoming 90 keV deuteron) is 110 nm. The average energy loss of a 3.54 MeV neutron through 110 nm of titanium is approximately 30 keV, less than 1% of the initial energy. The plot in Figure 3-4 shows that these alpha particles suffer almost no angular divergence crossing this thickness of material. Thus, the assumption that the alpha particles velocities are not significantly altered exiting the target is quite reasonable.

In order to determine the average COM velocity for all DT fusions initiated by the beam described in the preceding paragraphs, a weighted average of COM velocities from all of the possible reactions was used. All of the possible reactions are given in Table 3-1. The beam fraction is the fraction of that ion in the beam. The DT$^+$ ion is divided into two equal rows because it can interact with either a deuteron or a triton. E_{max} is the mass fraction of the interacting atom in the molecular ion multiplied by 90 keV. $\overline{E}$ is the average interaction energy calculated using E_{max} in Equation 3.8. The COM velocity and cross section at that energy were then multiplied together and normalized to give the fraction of reactions due to each interaction. Finally, a weighted sum was calculated by multiplying V_{CM} by the reaction fraction. The sum of the last column yields the weighted average for the COM velocity of 0.003306c. This value will be used in Section 3.1.4 to calculate the resulting neutron angle distribution.
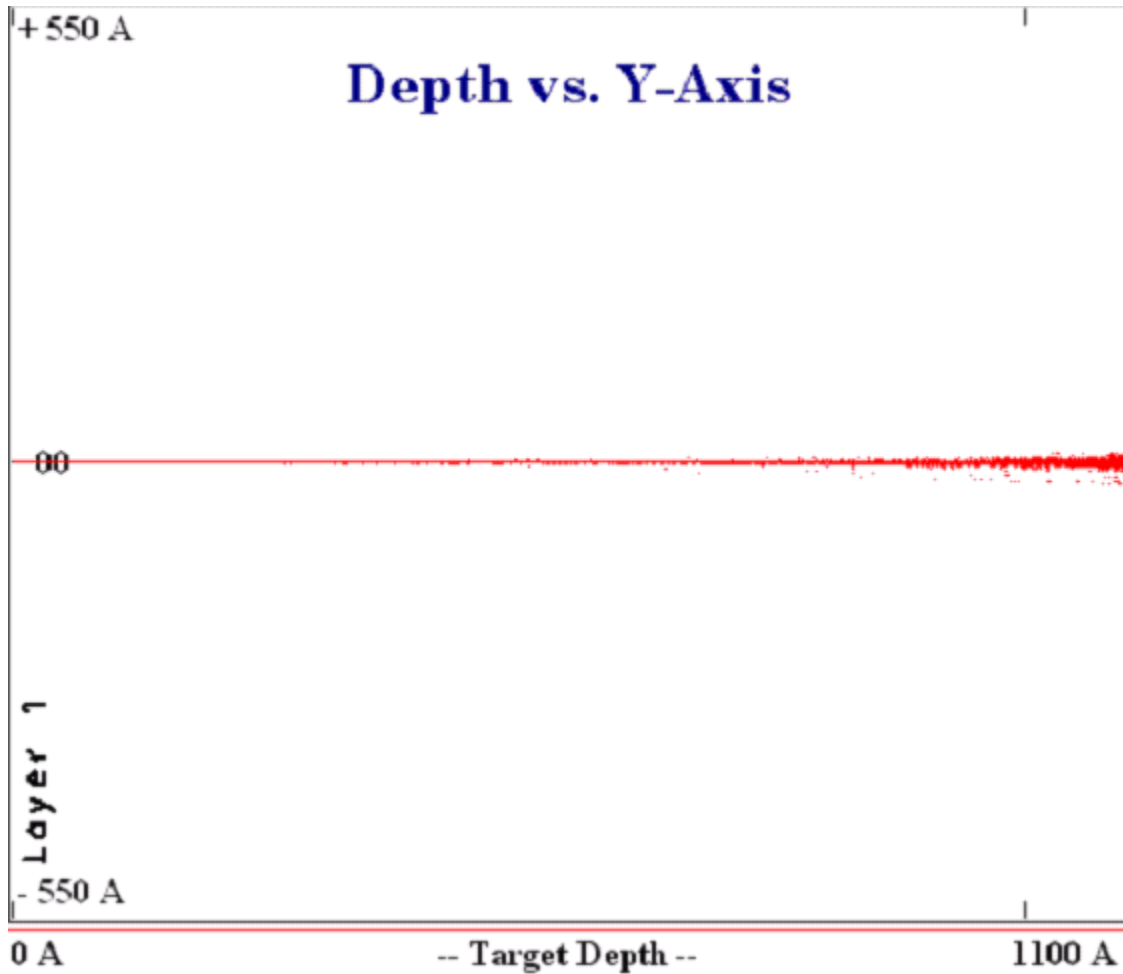
Figure 3-4. Results from a SRIM simulation of 3.54 MeV alpha particles incident on a 110 nm thick titanium slab. This thickness is the average interaction depth of a 90 keV deuteron. The beam of alpha particles shows almost no divergence traveling through this thickness of material.

Table 3-1. The possible DT reactions. The average DT cross section of each possible reaction was calculated and weighted by the fraction of that ion in the beam to calculate the weighted average COM velocity.

Projectile	Target	Beam Frac	E_{max} (keV)	$\overline{E}$ (keV)	$V_{CM}(\overline{E})$ (fraction of c)	$\sigma(\overline{E})$ (b)	Rx Frac	$V_{CM} \times$ Rx. (fraction of c)
D^+	T	0.05	90	70.5	0.003471	3.1	0.1897	0.000659
T^+	D	0.05	90	70.5	0.004248	3.1	0.1897	0.000806
DD^+	T	0.225	45	36.3	0.002491	0.50	0.1373	0.000342
DT^+	D	0.225	54	44	0.003356	1.0	0.2746	0.000921
DT^+	T	0.225	36	29.1	0.002230	0.26	0.0714	0.000159
TT^+	D	0.225	45	36.3	0.003048	0.50	0.1373	0.000418
					Weighted average COM velocity:			**0.003306**

3.1.3 The API-120 Target and PMT Geometry

The geometry of the DT target spot, alpha particle detector, and photomultiplier tube in the API-120 DT neutron generator is pictured in Figure 3-5. The target consists of a titanium metal which has adsorbed a DT gas. The target is mounted at a 45° angle with respect to both the incoming ion beam and the alpha detector. In order to create smaller cones, the target spot was limited to a 5 mm diameter circle on the target by placing a mask over the target to stop incoming ions outside of that range. [49] The alpha particle scintillator crystal is located 57 mm from the center of the target spot perpendicular to the direction of the incoming ion beam. A thin layer of aluminum in front of the scintillator stops light and any deuterium or tritium ions that are scattered from the target from reaching the scintillator crystal. The scintillator is an inorganic YAP:Ce crystal. Alpha particles interacting in the scintillator generate photons which are transmitted through a fiber optic faceplate.
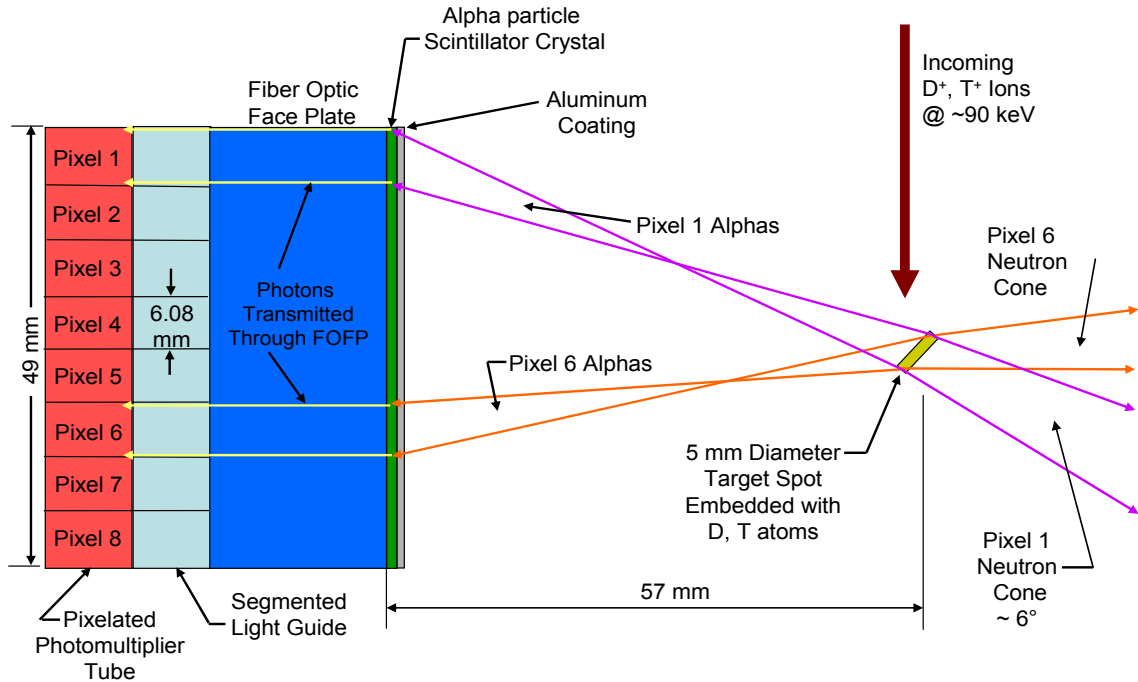
Figure 3-5. The geometry of the API-120 target spot, alpha particle detector, and pixelated Hamamatsu H8500 photomultiplier tube.

A segmented light guide then spreads the light evenly onto the face of one of the PMT pixels. This light guide makes the response of the pixels less sensitive to the relative position of the incoming photons on the pixel. The PMT shown is a Hamamatsu H8500 PMT [13] with eight rows of eight pixels. Each pixel is a 6.08×6.08 mm square except for pixels 1 and 8 which are 6.26 mm wide. For NMIS imaging, only a single row of pixels centered vertically on the target spot is used.

3.1.4 A Monte Carlo Simulation to Determine Neutron Cones

In order to calculate the effective neutron cones, a Monte Carlo code named *MCPixelGeom* was written using the Fortran-90 programming language. [50] The code uses the API-120 geometry and the kinematics derived in the previous sections to

simulate DT reactions. The program begins by randomly sampling a point on the DT target spot and the outgoing alpha particle (and neutron) angle in the COM coordinate system. The alpha angle is then converted to the LAB frame using Equation 3.7 and the location where the alpha particle intersects the plane of the scintillator is calculated. No light transfer from the scintillator to the PMT pixels is simulated. Instead, each (x,z) point on the scintillator plane is assumed to correspond directly to the same (x,z) point at the pixel plane. If this location corresponds to one of the alpha pixels, the LAB angle of the outgoing neutron is calculated using Equation 3.7. For each neutron, the horizontal (x-y) and vertical (y-z) components of the LAB angle are calculated to the nearest 0.1°.

This process is repeated over and over again. For each pixel, a two-dimensional array is used to tally the direction of the outgoing neutrons correlated to it by adding 1 to a bin corresponding to the (x-y) and (x-z) angles of the outgoing neutron. After all of the iterations are complete, the array is normalized by dividing each bin by the average number of alpha particles in each pixel. An additional array stores the total number of correlations occurring in a 2.54 × 2.54 cm detector centered vertically ($\theta_{xz} = 0$) at a given neutron angle (θ_{xy}). These values are calculated using the equation

$$R_{det}\left(\theta_{xy}\right) = \int_{-\theta_{det}}^{\theta_{det}} \int_{\theta_{xy}-\theta_{det}}^{\theta_{xy}+\theta_{det}} R\left(\theta_{xy}, \theta_{xz}\right) d\theta_{xy} d\theta_{xz}, \qquad (3.9)$$

where $R(\theta_{xy}, \theta_{xz})$ is the number of neutrons per source alpha in a given horizontal and angular bin; θ_{det} is the half angle subtended by the detector face; and $R_{det}(\theta_{xy})$ is the neutrons per source alpha entering the detector face. For a 2.54 × 2.54 cm detector located 110 cm from the DT target spot, $\theta_{det} \approx 0.6°$. The resulting arrays are output to a

text file for further analysis and processing. The source code for the *MCPixelGeom*

program is shown in Appendix A.

3.1.5 Results of the Monte Carlo Pixel Simulation

A 3-D view of the simulated pixel profiles is presented in Figure 3-6. Note that

the pixel on the left (pixel 1) shows a rounded top and a square base. This effect is

caused by the convolution of shape of the pixel (square) with the target spot (round).

Progressing from left to right, each successive pixel becomes more and more rectangular.

This effect occurs because the angle of incidence to the target is increasing which causes

the apparent size of the target spot to be smaller. The variation in neutron angle is then

dominated by the position of the alpha particle on the square pixel face. If the target spot

was infinitesimally small, the neutron profiles would be perfect rectangles.

The total neutron correlation profile is shown in Figure 3-7. This figure was

generated by summing the correlations for all pixels in each angle bin. Although the

individual pixel shapes varied somewhat, the shape of the total profile is uniform with the

exception of the ends. The right end shows a steeper drop off than the left because of the

angle of the target spot.

The detector response versus horizontal angle at a vertical angle of 0° generated

using Equation 3.9 and assuming 100% detector efficiency is given in Figure 3-8.

Integrating the angular bins over the angles incident on the detector face tends to soften

the corners of the square pixels a bit, but the flat tops are still visible on the higher

numbered pixels. The center of the pixels is shifted approximately -5.5° due to the

momentum of the incoming deuterons and tritons.

36
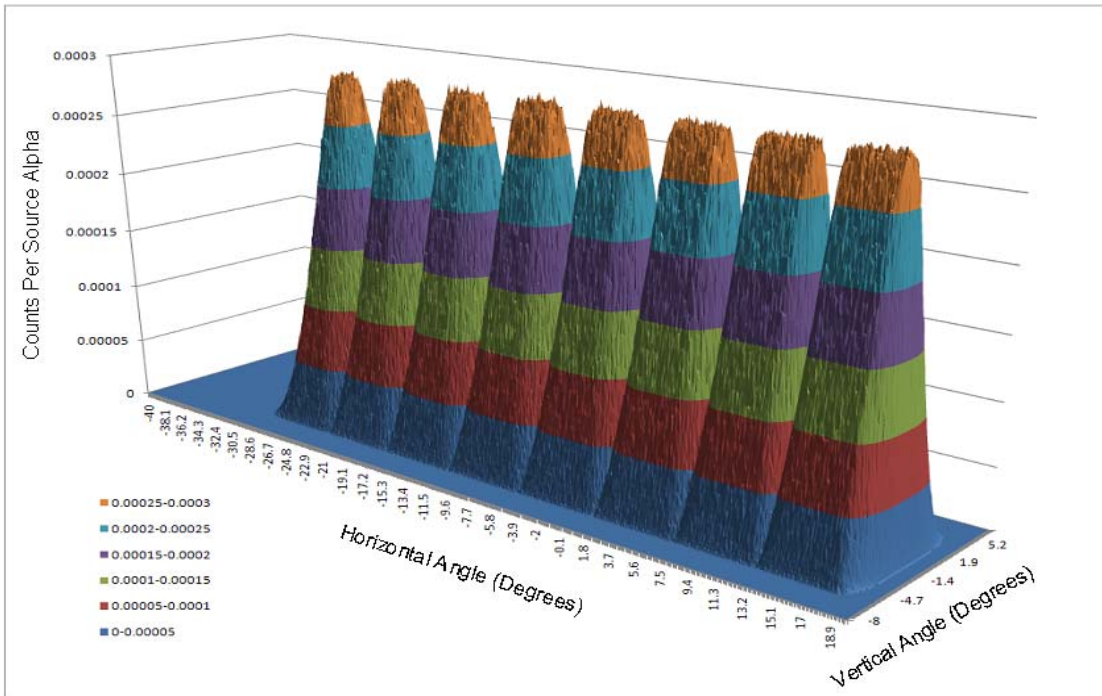
Figure 3-6. Profile of the alpha pixels produced by the *MCPixelGeom* simulation for an eight pixel row of the H8500 PMT.
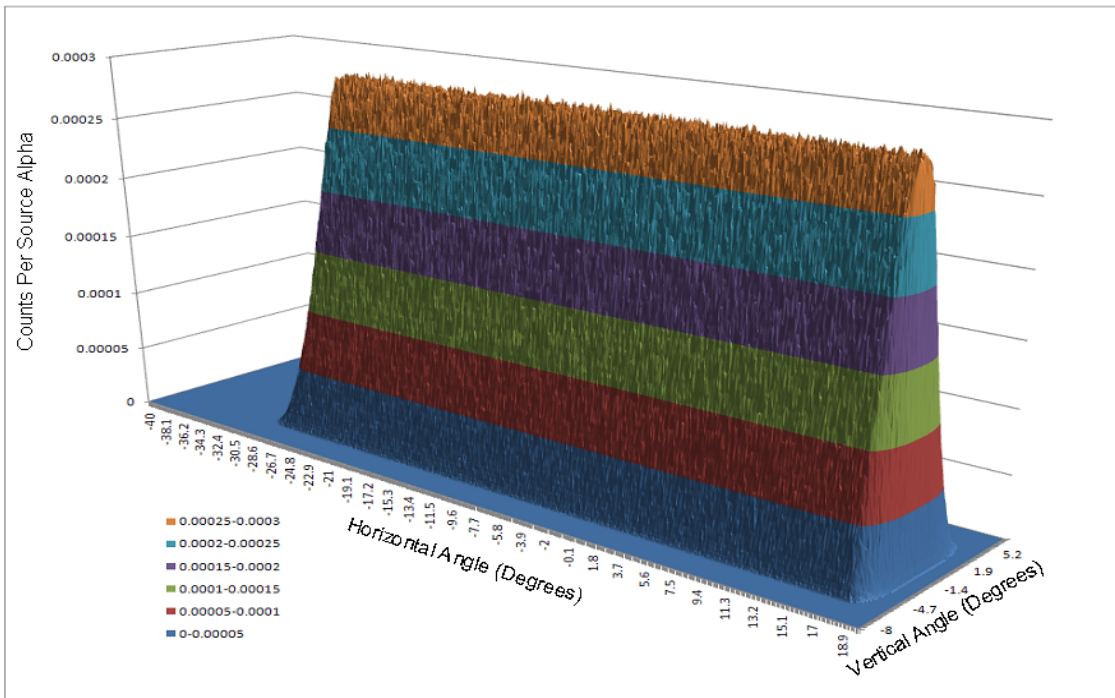
Figure 3-7. The total correlation profile produced by summing all eight pixels. Although the individual pixel shapes varied considerably, the overall shape is nearly uniform with the exception of the ends.
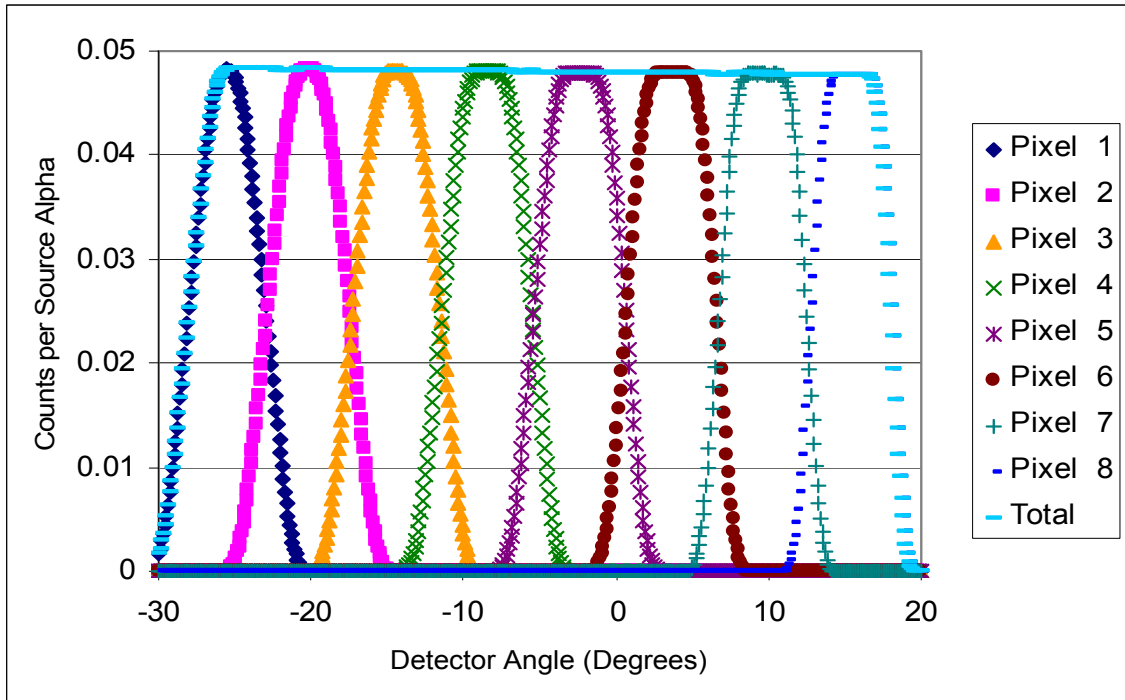
Figure 3-8. The detector profiles produced by each pixel and the total detector response.

The data plotted in Figure 3-8 represents an idealized version of the alpha-neutron correlation curves assuming a perfect, uniform response across the entire range of pixels. In the laboratory, the light transmission from the scintillator crystal, through the fiber optic face plate and light guide, and the response across the face of the detectors all serve to alter the shape and size of the pixel profiles. In particular, light near the edge of a pixel boundary tends to produce a smaller number of photons and/or bleed across pixels and generate cross talk. This has the effect of rounding out the tops of the pixels and widening the tails at the base. In addition, the *MCPixelGeom* simulations were conducted using the average COM velocity for incoming ions. In practice, there will be a distribution of COM velocities which will tend to spread the pixel distributions.

The change in the pixel shapes alters the overall neutron profile as well. This results in an overall profile which is not flat. This can have a deleterious effect on imaging measurements. Because the exact form of the alterations is a function of many factors such as PMT voltage and constant fraction discriminator settings, they can change from measurement to measurement and would be difficult to model. In addition, the absolute position of the peaks (and thus the fluctuations) can change slightly each time the generator is removed from and replaced on its mount. In order to produce the most widely useful model, a combination of the idealized pixels of Figure 3-8 and those seen in the laboratory will be used.

A cosine squared (COS^2) function has a rounded top similar to that seen in the laboratory pixels and a short tail like the pixels generated in the model. This function also has the benefit that two curves separated by exactly one FWHM sum to a horizontal line between the peaks. This creates a smooth, flat profile like that seen in Figure 3-8. Because the API-120 is mounted at an angle to correct for the forward shift in the neutrons, the approximate peak locations were determined from experimental data. Although these may change slightly from measurement to measurement, the simulations should still follow the data well except at the edges of the total pixel profile.

3.1.6 Validation of the Pixel Model with Experimental Data

The modeled pixels and an experimental pixel profile are compared in Figure 3-9. In general, the modeled pixels match the peak locations and the shape of the top of the experimental pixels well. The experimental pixel profiles widen at the base as a result of detector and pixel cross talk.
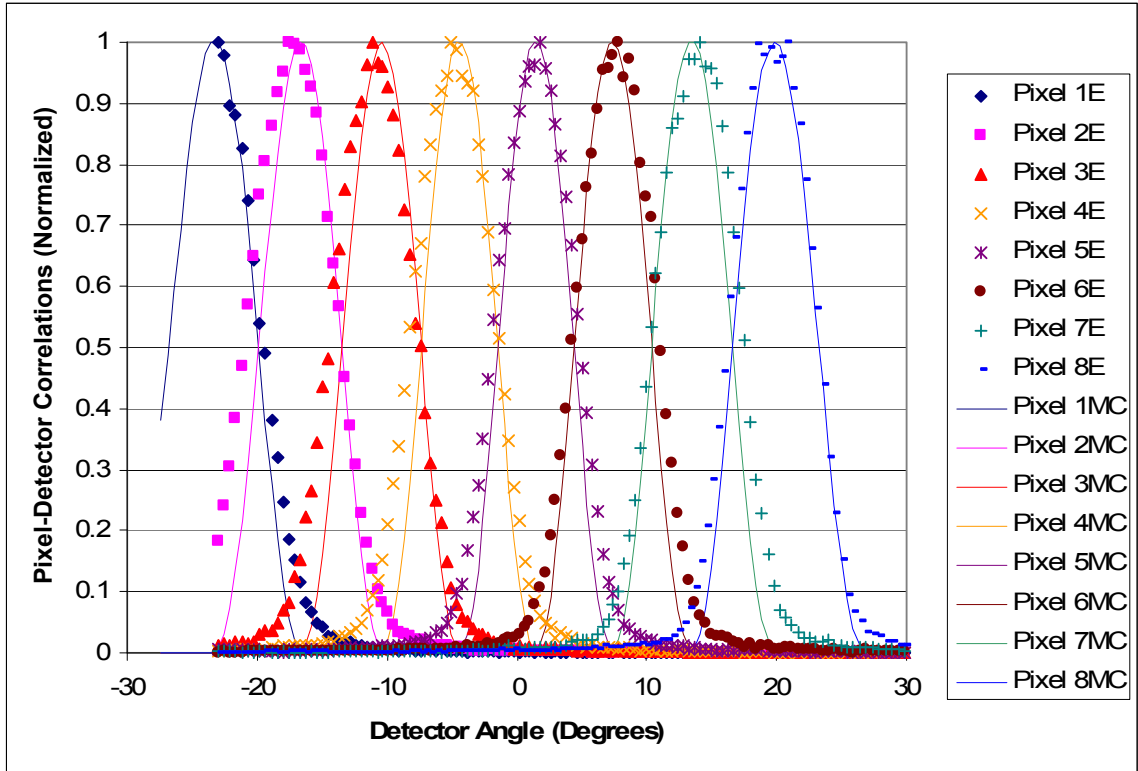
Figure 3-9. A comparison of simulated and experimental pixel profiles. The simulated pixels generally match the experimental ones with the exception of the tails of the pixels. An *E* stands for experimental and an *MC* stands for Monte Carlo in the legend.

The center of the modeled pixels has been matched to the center of the experimental pixels at -1.5°. This offset is considerably smaller than the calculated value of -5.5° because the generator was mounted at an angle in an attempt to correct for the forward momentum of the neutrons and alpha particles.

In order to validate the pixel model, a test object was constructed and a 1D attenuation profile was measured in both the laboratory and simulations. The object used is pictured in Figure 1-5. In the laboratory measurement, the detectors were subsampled six times in order to improve angular resolution. Each subsample was measured for 15 minutes, resulting in a total measurement time of 90 minutes. For the simulations, each subsample was modeled separately. For each pixel, a total of 2.5×10^7 neutrons was used which is approximately the number of alpha counts per pixel detected during the laboratory measurements. The simulation procedure will be explained in greater detail in Chapter 4.

A comparison of the simulated and measured attenuation curves is presented in Figure 3-10. The center of the simulated curve was shifted two detector positions (~0.4°). This discrepancy was likely due to the object being slightly off center in the laboratory measurement. Otherwise, the experimental and simulation measurements show excellent agreement indicating that the modeled pixel curves are suitable for simulating NMIS imaging measurements.

3.2 A First Order Approximation of Elastic Scattering

Before conducting simulations to measure scattering, it is desirable to perform a simple calculation of the scattering effect.
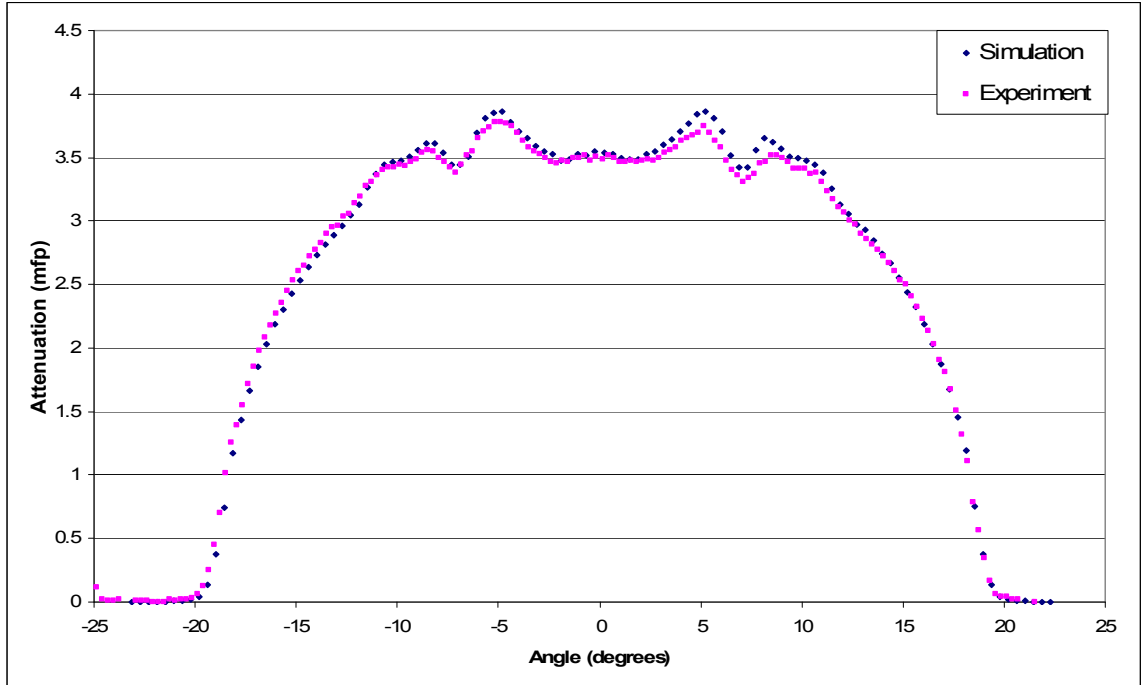
Figure 3-10. A comparison of simulated and experimental attenuation curves of the object shown in Figure 1-5. The simulated curve was shifted two detector positions to align the two curves.

This will be a first order calculation that only takes into account single elastic scattering in a thin object. As the thickness of the intervening target increases, multiple scatterings will contribute a greater portion of the scattering component. The scattering calculation here will take into account the fact that the neutrons are time tagged, so scattered neutrons add to the signal only if they arrive during the time window used to define directly transmitted neutrons.

3.2.1 Scattering Geometry

The geometry used in these scattering calculations is shown in Figure 3-11. A thin beam of DT neutrons is incident on a horizontal detector array. The front faces of the detectors are aligned so that each is a distance, R, from the neutron source. The depth of the detector crystals from the front to the back faces is L. A thin slab of material is placed in the beam at a distance, D, from the point where the neutron beam crosses the front face of the detector array. Neutrons scattering in the slab scatter through an angle, φ, before arriving at the detector array. The path length of the neutrons from the scattering site in the slab to the front of the array is D'. The detector angle, θ, defines the angle between the initial neutron beam direction and the location where the scattered neutron arrives at the front of the array relative to the DT target spot.

Using the law of cosines, the relationship between the length of D' and the detector angle can be written

$$D' = R^2 + (R-D)^2 - 2(R-D)(R)\cos\theta. \tag{3.10}$$

43
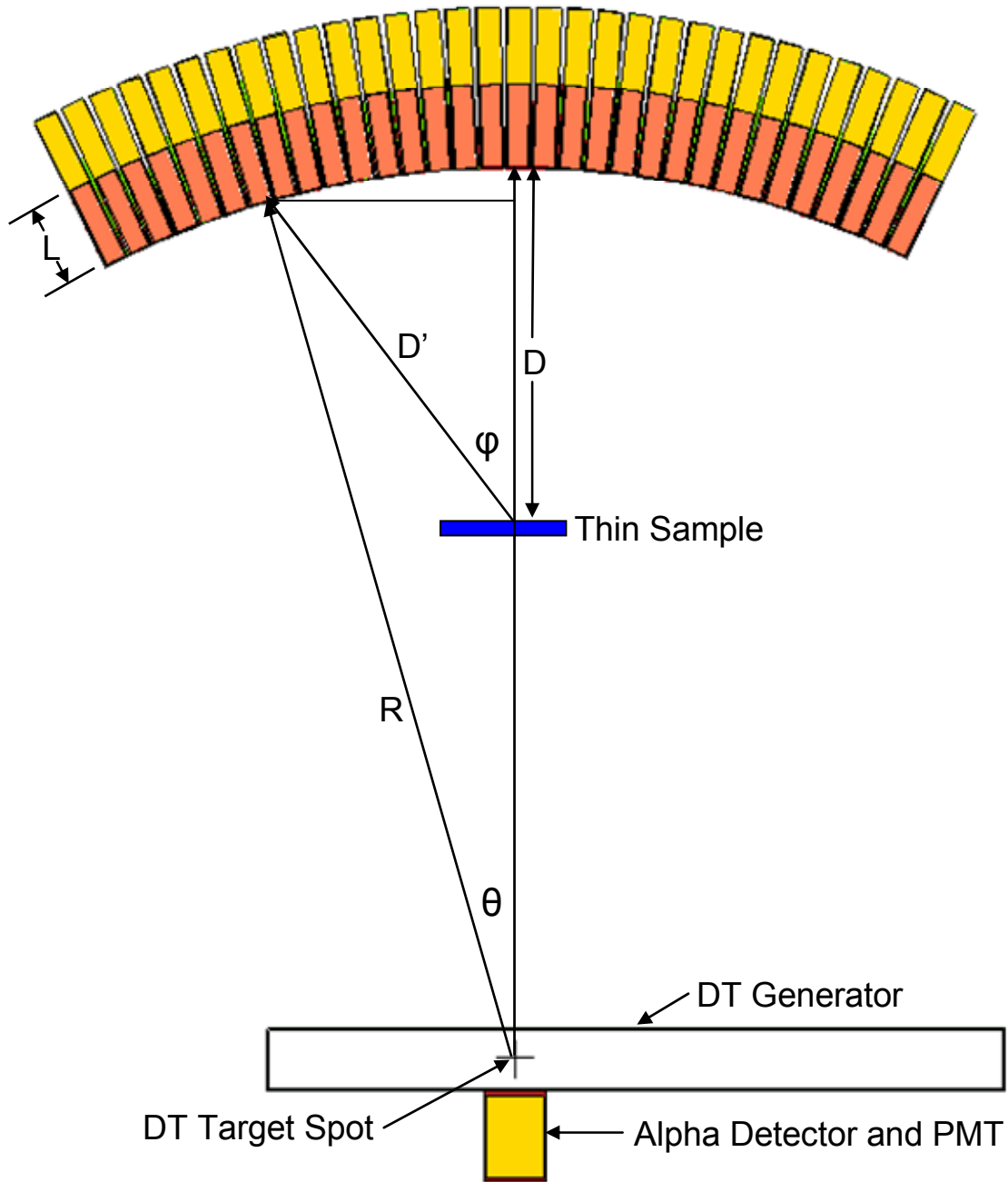
Figure 3-11. The geometry used for the elastic scattering calculations. Neutrons interacting in the sample scatter through an angle φ. When measured in reference to the DT target spot, the neutrons are scattered through a detector angle θ.

The scattering angle, φ, can then be related to the detector angle, θ, by the relation

$$D'\sin\phi = R\sin\theta. \tag{3.11}$$

3.2.2 The Derivation of the Scattering Function

In order for a scattered neutron to be mistaken for a directly transmitted one, it must reach the detector array during the time window assigned for DT neutrons and generate a pulse in a detector. The total scattered response in a detector at angle θ is the product of the incoming neutron flux, Φ; the probability of scattering towards the detector; and the probability that the incoming neutron generates a pulse that is mistaken for a directly transmitted neutron. Mathematically, this can be written

$$R(\theta) = \Phi_0 P_{scatter}(\theta) Eff(\theta). \tag{3.12}$$

The next several paragraphs will detail how this response function is calculated.

For the purpose of these calculations, the electronic timing resolution of events will be assumed to be perfect. This limits the fast neutron window to the time between when a directly transmitted neutron arrives at the front of the detector crystal and when it leaves the back face. For an unscattered neutron travelling at speed v, the time window is limited to the range

$$\frac{R}{v} \leq t \leq \frac{R+L}{v}. \tag{3.13}$$

The fraction on the left side of Equation 3.13 will be referred to t_{min} and the fraction on the right as t_{max}.

When a neutron scatters, its energy decreases and its path length to the detector array increases. Each of these two factors increases the arrival time at the front of the detector. The new arrival time is now

$$t' = \frac{R-D}{v} + \frac{D'}{v'},$$

(3.14)

where v' is the neutron speed after scattering. Because $t' - t_{min} < t_{max} - t_{min}$, the maximum depth the scattered neutron can penetrate into the detector decreases.

In order to calculate v', the scattered neutron energy must be known. Fortunately, there is an exact relationship between the angle of scattering and the outgoing energy. From Duderstadt and Hamilton [47], the outgoing energy is

$$E_f = \left[\frac{(1+\alpha)+(1-\alpha)\cos\phi_C}{2} \right] E_i,$$

(3.15)

where E_i and E_f are the neutron energy before and after scattering, φ_c is the scattering angle in the COM coordinate system, and α is the minimum possible neutron energy after the collision. α is calculated using the equation

$$\alpha = \left(\frac{A-1}{A+1} \right)^2,$$

(3.16)

where A is the atomic mass number of the scattering nuclide. The COM scattering angle, φ_c, is related to the LAB scattering angle φ via the equation [47]

$$\tan\phi = \frac{\sin\phi_C}{\frac{1}{A} + \cos\phi_C}.$$

(3.17)

46

Solving for φ_c, this equation is transcendental; however, it can be solved easily by using an iterative method. Once the outgoing neutron energy is known, its speed can be calculated using Equation 3.5.

With the velocity of the scattered neutron calculated, the new effective detector depth can be calculated using the equation

$$L' = (t_{max} - t')v'. \tag{3.18}$$

For the purposes of this calculation, all neutrons will be assumed to enter perpendicular to the detector face. Although this assumption is rather poor for large scattering angles, the complex detector geometry makes it extremely difficult to define the actual detector interacted with as a function of scattering angle at large angles. The PScF simulations in Chapter 5 will model the detector cells explicitly, thereby eliminating this problem.

A scintillator detects neutrons primarily through elastic scattering of neutrons on the hydrogen in the detector. The intrinsic efficiency of a scintillator to fast neutrons (ε) can be written [51]

$$\varepsilon = \frac{N_H \sigma_H}{N_H \sigma_H + N_C \sigma_C} \left\{ 1 - \exp\left[-\left(N_H \sigma_H + N_C \sigma_C \right)L' \right] \right\}, \tag{3.19}$$

where N_H and N_C are the atom densities of hydrogen and carbon and σ_H and σ_C are the microscopic scattering cross sections for hydrogen and carbon. The cross sections are energy dependent, so they will also be affected by the resulting energy of the scattered neutron. In order to use these cross sections analytically, their tabulated values were fit using the JMP 7 statistical package. [52] The resulting fits are

$$\sigma_H(E) = \exp\left(1.6431845 - 0.324906E + 0.021717E^2 - 0.000632E^3 \right)$$
$$\sigma_C(E) = 4.4929537 - 0.906192E + 0.0718101E^2 - 0.001809E^3, \tag{3.20}$$

where the cross sections are in barns. The hydrogen cross section fit gives good results for energies 1 MeV < E < 14.1 MeV. Because of resonances, the carbon cross section fit is only accurate in the range 9 MeV < E < 14.1 MeV. As will be shown in Section 3.2.3, neutrons at energies below 9 MeV have little impact on the PScF.

The efficiency calculation in Equation 3.19 assumes that any neutron elastically scattering off a hydrogen atom will generate a pulse in the detector. In practice, scintillators are operated with a pulse height threshold. The NMIS imaging detectors typically use a detector threshold of approximately 1 MeV for neutrons. When a neutron scatters off a hydrogen atom, it can transfer any amount of energy between 0 and its total kinetic energy to the proton. The distribution of these energies is uniform. [47] Neglecting the small contribution from neutrons that scatter more than once, the probability a neutron with an energy of E' scattering off a hydrogen atom generates a pulse is

$$PH(E') = \frac{E' - E_{thresh}}{E'}. \tag{3.21}$$

Because the neutron energy and effective detector depth can be expressed as explicit functions of energy, the total probability a scattered neutron generates a pulse in a detector can be written

$$Eff(\theta) = \varepsilon(\theta)PH(\theta). \tag{3.22}$$

The probability that a neutron first scatters while passing through an infinitesimal layer, dx, of a slab of material composed of a single isotope can be written

$$P(scatter\ in\ dx) = N\sigma_s \exp(-N\sigma_s(E)x), \tag{3.23}$$

where N is the atomic number density, σ_s is the microscopic scattering cross section, and x is the distance the neutron has penetrated into the slab before the first scattering. In order for the scattered neutron to reach the detector array, it must then escape the back side of the slab. The total probability of scattering in a layer at a distance x into the slab and then escaping can be written

$$P(scatter\ and\ escape) = N\sigma_s \exp(-N\sigma_s(E)x)\exp(-N\sigma_t(E)(T-x)), \qquad (3.24)$$

where T is the total thickness of the slab and σ_t is the total microscopic scattering cross section. Equation 3.24 assumes that all further interactions will prevent the neutron from reaching the back side of the slab, i.e., it is a first order scattering approximation. The equation also neglects the increased distance between the initial scattering location and the back of the slab due to the scattering angle; however, for small scattering angles, this difference should be relatively small. Integrating Equation 3.24 from x = 0 to x = T yields

$$P(scatter\ and\ escape) = \frac{\sigma_s}{\sigma_t - \sigma_s}[\exp(-N\sigma_s(E)T) - \exp(-N\sigma_t(E)T)]. \qquad (3.25)$$

Rather than describing the slab thickness in units of length that will change for each material, it is desirable to use attenuation lengths. One attenuation length (MFP) is the thickness which attenuates the beam by a factor of 1/e. The thickness of the slab in attenuation lengths, τ, can be written

$$N\sigma_t T = \tau. \qquad (3.26)$$

Using Equation 3.26, Equation 3.25 can be rewritten

$$P(scatter\ and\ escape) = \frac{\sigma_s}{\sigma_t - \sigma_s}\left[\exp\left(-\frac{\sigma_s(E)}{\sigma_t(E)}\tau\right) - \exp(-\tau)\right]. \qquad (3.27)$$

49

The probability of scattering to a detector at angle θ can then be written

$$P_{Scatter}(\theta) = \frac{\frac{d\sigma_s}{d\Omega}(\theta)\Omega_{det}(\theta)}{\sigma_t - \frac{d\sigma_s}{d\Omega}(\theta)\Omega_{det}(\theta)}\left[\exp\left(-\frac{\tau}{\sigma_t}\frac{d\sigma_s}{d\Omega}(\theta)\Omega_{det}(\theta)\right) - \exp(-\tau)\right], \quad (3.28)$$

where $\frac{d\sigma_s}{d\Omega}(\theta)$ is the differential scattering cross section and $\Omega_{det}(\theta)$ is the solid angle

subtended by the detector at angle, θ.

The differential scattering cross sections are available online in tabulated form

from the National Nuclear Data Center. [53] The cross sections are given in terms of

COM scattering angle, which can then be converted to the LAB frame using Equation

3.17. As with the intrinsic efficiency calculation, calculating the Ω_{det} explicitly as a

function of angle would be extremely difficult at large scattering angles because of the

complex detector geometry. Instead, the solid angle subtended by the detector will be

approximated using the relation

$$\Omega_{det} = \frac{l^2}{D'^2}, \quad (3.29)$$

where the solid angle Ω_{det} is in units of steradians and l is the dimension of square

detector face.

Combining all of the terms from the previous paragraphs, Equation 3.12 can be

rewritten

$$R(\theta) = \Phi_o\varepsilon(\theta)PH(\theta)\left\{\frac{\frac{d\sigma_s}{d\Omega}(\theta)\Omega_{det}(\theta)}{\sigma_t - \frac{d\sigma_s}{d\Omega}(\theta)\Omega_{det}(\theta)}\left[\exp\left(-\frac{\tau}{\sigma_t}\frac{d\sigma_s}{d\Omega}(\theta)\Omega_{det}(\theta)\right) - \exp(-\tau)\right]\right\}.$$

$$(3.30)$$

The PScF into a detector at angle, θ, is the ratio of the scattering response divided by the response of the detector at $\theta = 0$ to the beam with no slab in place. The equation for the PScF is then

$$PScF(\theta) = \frac{\varepsilon(\theta)PH(\theta)\left\{\dfrac{\dfrac{d\sigma_s}{d\Omega}(\theta)\Omega_{det}(\theta)}{\sigma_t - \dfrac{d\sigma_s}{d\Omega}(\theta)\Omega_{det}(\theta)}\left[\exp\left(-\dfrac{\tau}{\sigma_t}\dfrac{d\sigma_s}{d\Omega}(\theta)\Omega_{det}(\theta)\right)-\exp(-\tau)\right]\right\}}{\varepsilon(0)PH(0)}.$$

(3.31)

3.2.3 Elastic Scattering Calculation Results

The scattering calculations were performed using a Fortran-90 program named *Elastic* written for this purpose. This program first reads a text file containing the atomic mass number and cross sections into memory and then performs the calculations shown in the previous sections. The parameters of the detector and target geometry were $R = 110$ cm, $D = 40$ or 70 cm, $L = 10.16$ cm, $l = 2.54$ cm, $\tau = 3$ MFP, and $E_{thresh} = 1$ MeV. Four different scattering isotopes were modeled to sample across a wide range of atomic masses. The isotopes were ^{1}H, ^{12}C, ^{56}Fe, and ^{208}Pb. The results of each calculation were output to a text file for analysis. The source code used for the *Elastic* program is shown in Appendix B.

The calculated results for some of the important components of the PScF for the four scattering nuclei at 40 cm from the detector array are shown in Figure 3-12 through Figure 3-15. The x-axis of all of these figures is the detector angle, θ, through which the neutron was scattered in degrees.

51
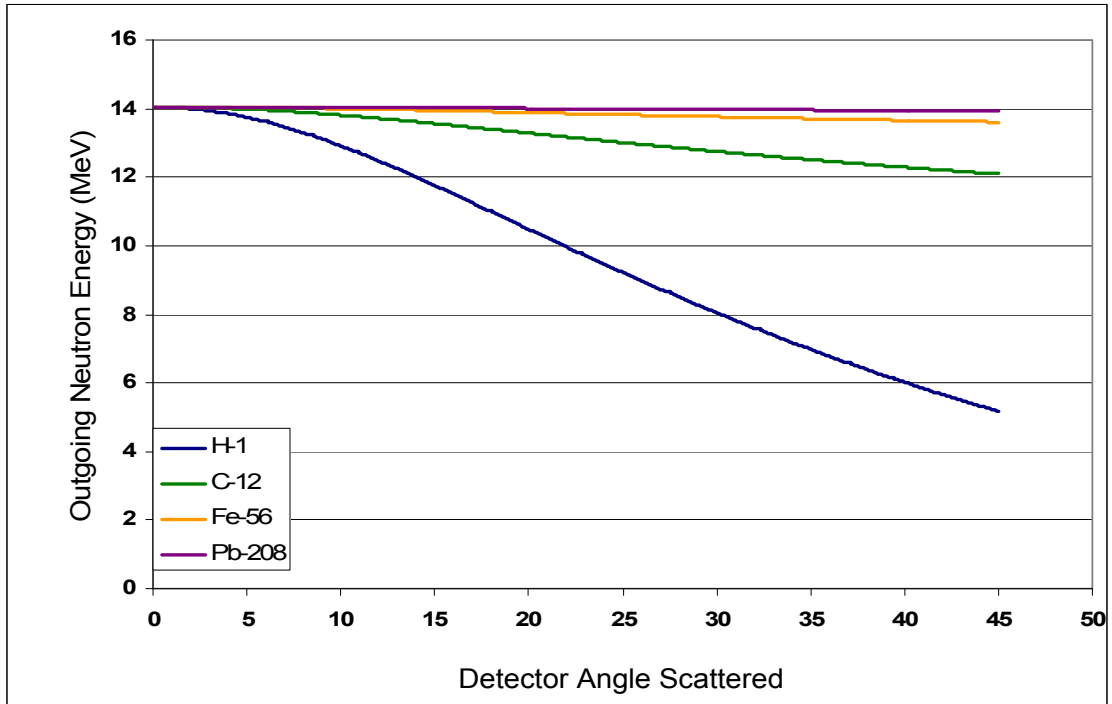
Figure 3-12. A plot of the outgoing energy of neutrons scattered off four different isotopes as a function of the detector angle the neutron scatters through in degrees.
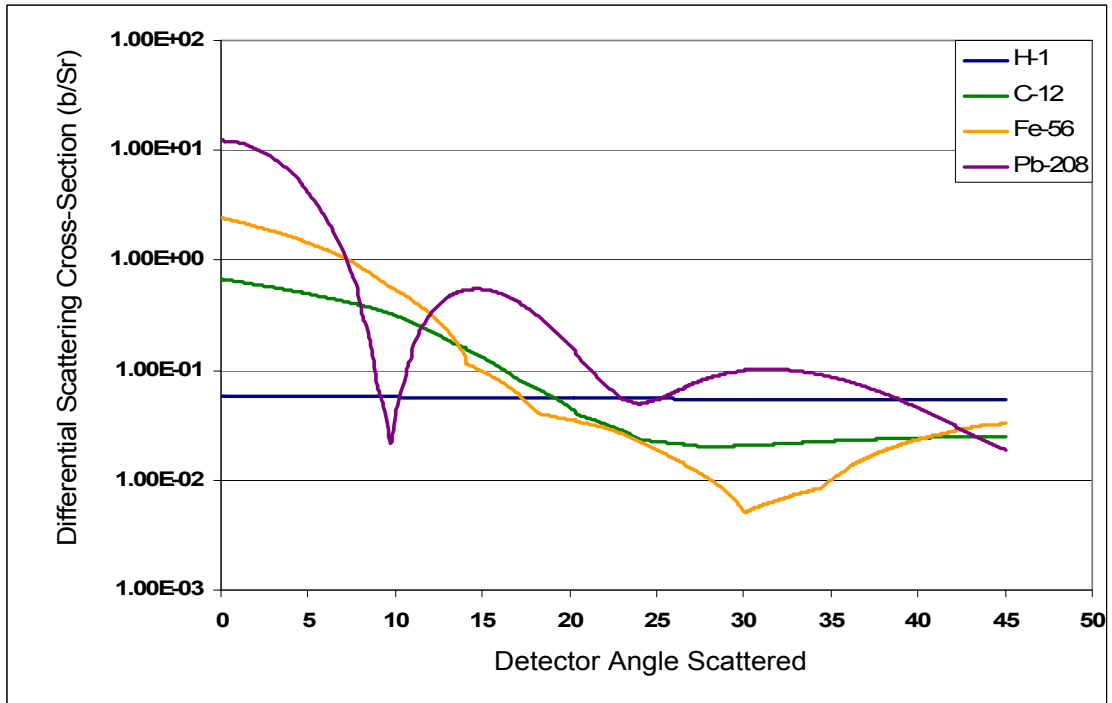
Figure 3-13. The differential scattering cross sections of four nuclides as a function of the detector angle the neutron scatters through in degrees.
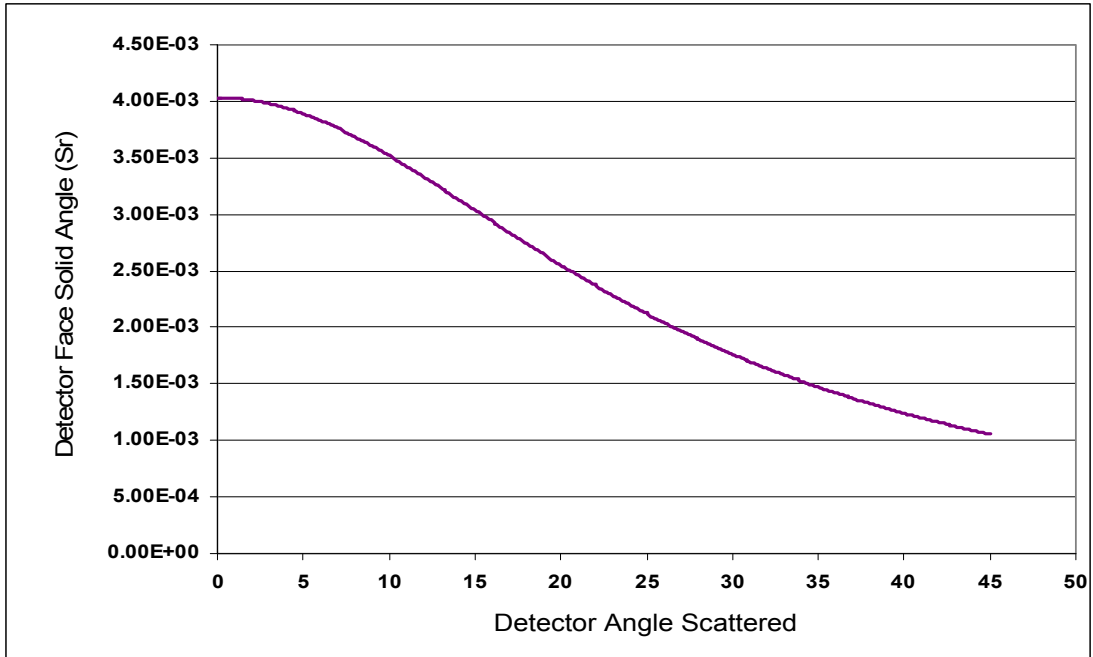
Figure 3-14. The solid angle subtended by the detector face as a function of the detector scattering angle in degrees.
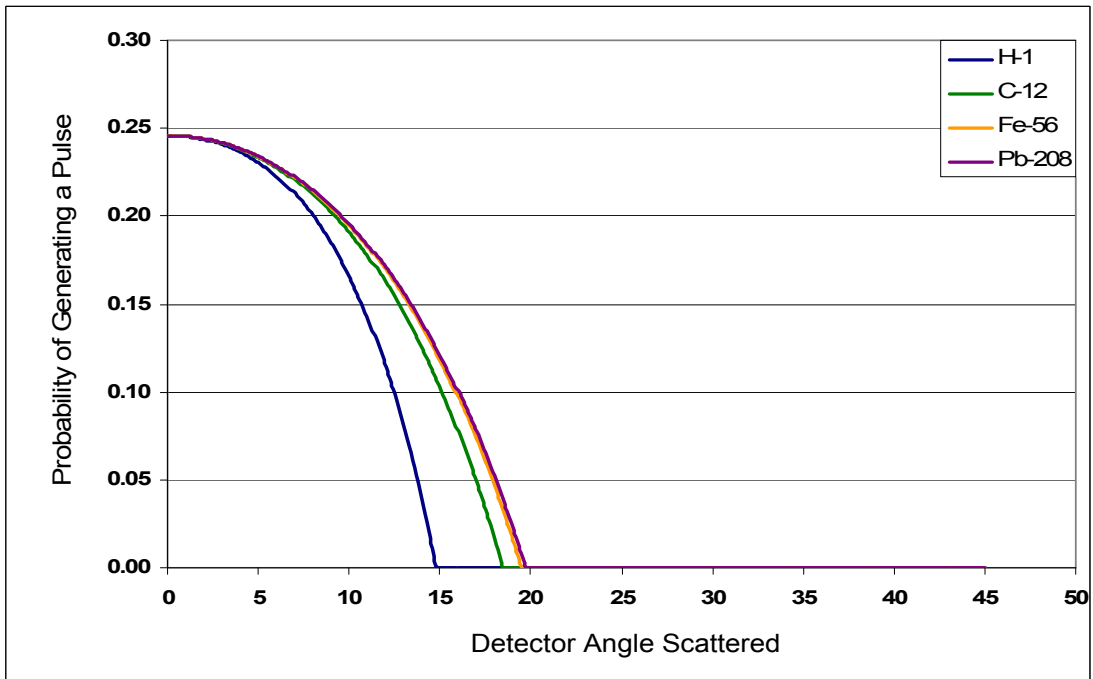
Figure 3-15. The probability of each of the five nuclides generating a pulse as a function of detector scattering angle in degrees.

The outgoing energy of the scattered neutron, E_f, computed using Equation 3.15 is shown in Figure 3-12. Neutrons scattering from heavy nuclides such as iron and lead lose almost no energy even at large scattering angles, while lighter nuclei lose a considerable fraction of their energy. The differential scattering cross sections are plotted in Figure 3-13. The hydrogen cross section is almost completely flat and featureless, indicating nearly isotropic scattering. As the atomic mass increases, the cross sections become increasingly forward peaked. In addition, heavy nuclei, particularly lead, show a pronounced diffraction pattern. The detector solid angle computed using Equation 3.29 is presented in Figure 3-14. Because the solid angle is independent of the scattering isotope, only a single curve is shown. The probability a scattered neutron generates a pulse, *Eff(θ)*, calculated using Equation 3.22, is shown in Figure 3-15. Because the energy of the scattered neutron is the most important component of *Eff(θ)*, the efficiency of neutrons scattered from heavy nuclei drops more slowly than those scattering off light nuclei. The efficiency rapidly drops to zero at the point when a scattering results in the neutron arriving after the end of the correlation window. In the laboratory, where there is some uncertainty in the timing, the final drops would be more gradual and extend out farther.

The calculated values of the point scatter functions for the four scattering nuclei are shown in Figure 3-16. The curves follow the general shape of the differential scattering cross sections superimposed with the drop off in detector efficiency at larger scattering angles. With the exception of lead, the PScFs are generally well behaved and monotonically decreasing.
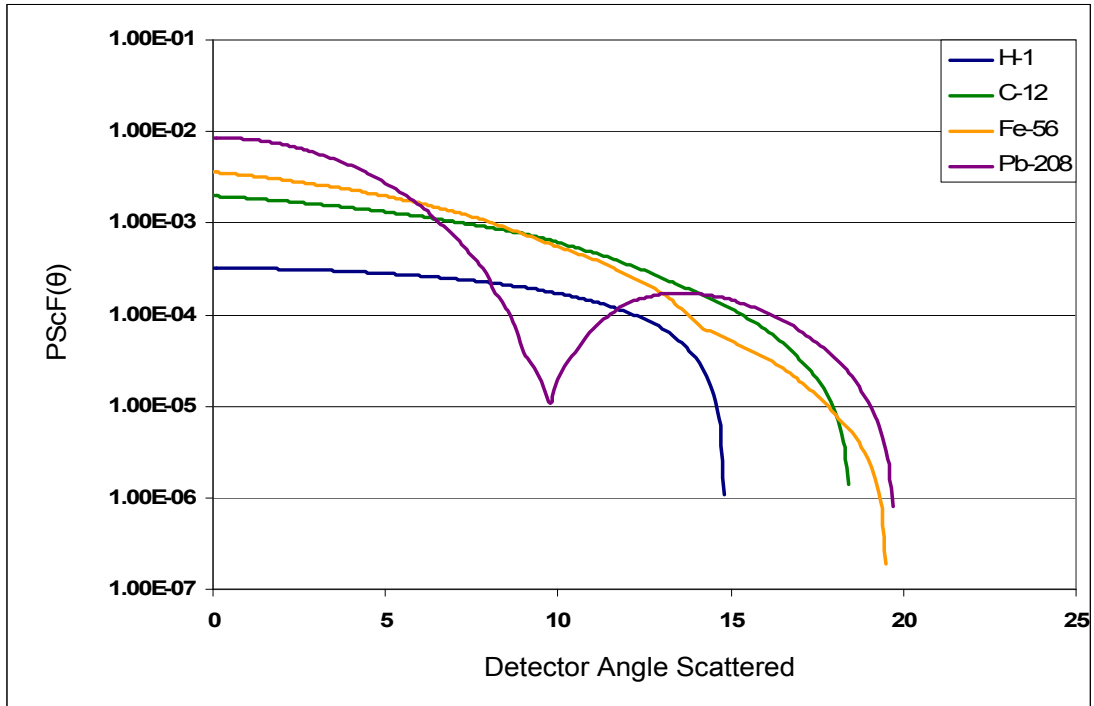
Figure 3-16. The point scatter functions for each of the four nuclides at a 40 cm object-to-detector distance.

The diffraction pattern of the lead PScF will result in some inaccuracy in the tail region of the Gaussian functions used for fitting the scattering in Chapter 5. In a real slab of scattering material with a finite thickness, scattering will occur along the entire thickness, which should blur the features of the diffraction pattern to some extent.

The PScFs of lead and carbon, respectively, at 40 and 70 cm from the detector array are compared in Figure 3-17 and Figure 3-18. As the slab moves farther from the array, the point scatter functions widen and the maximums decrease. The shapes of the functions remain nearly constant. As the slab moves farther from the array, the PScFs widen because the scattering angle, φ, corresponding to a given detector angle θ decreases. With a slab of finite thickness, some of the scattering will occur farther from the array and the PScF curve will widen.
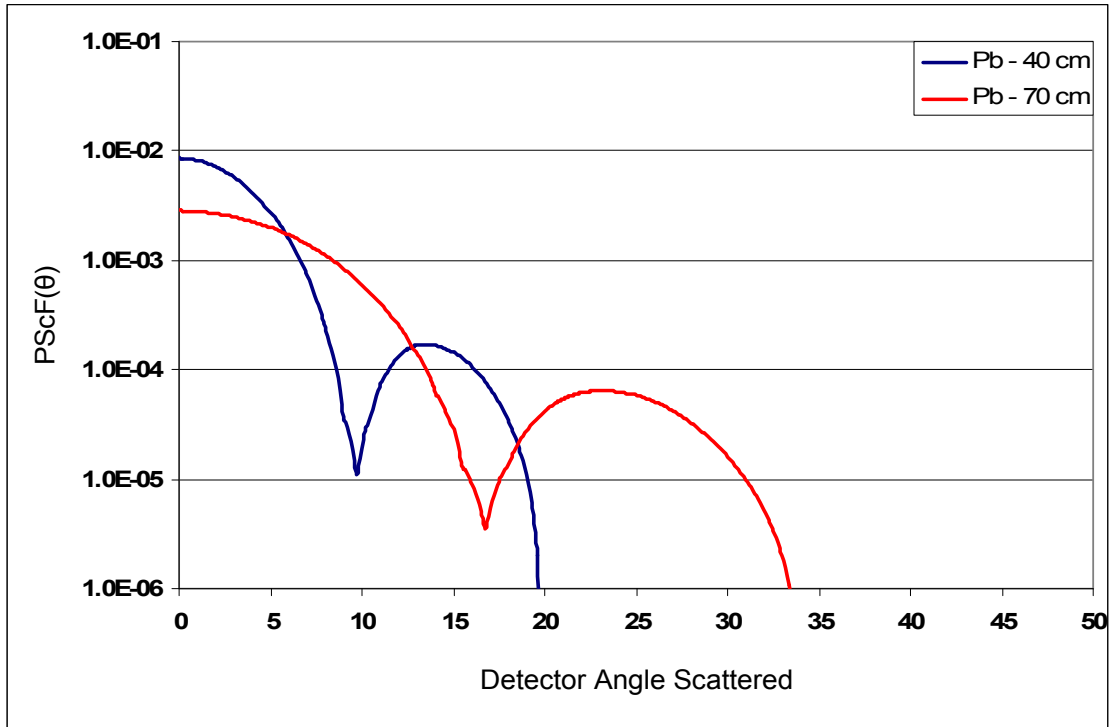
Figure 3-17. A comparison of the PScFs for ^{208}Pb at 40 and 70 cm from the array.
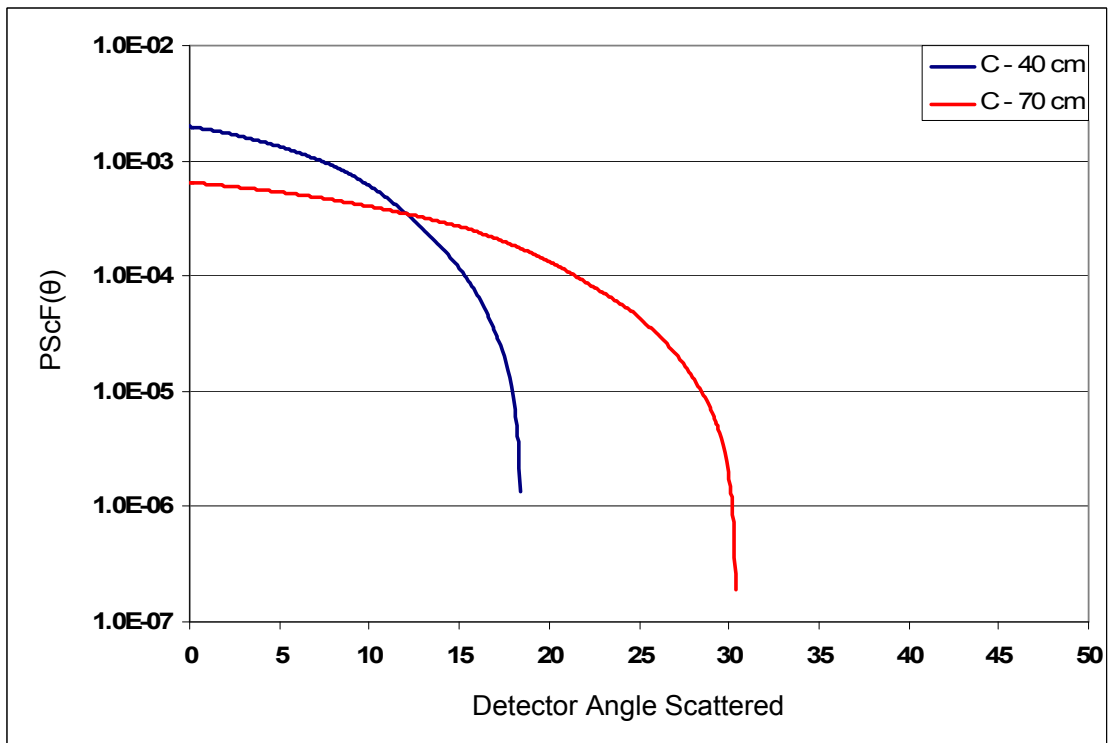
Figure 3-18. A comparison of the PScFs for ^{12}C at 40 and 70 cm from the array.

56

The PScF curves developed in this chapter will be compared to the ones measured using

MCNP simulations after the PScFs are developed in Chapter 5.

4. Simulation Code Development

4.1 The *MCNP-PoliMi* Code

For this work, the *MCNP-PoliMi* code [54] was used for simulating all NMIS imaging measurements. *MCNP-PoliMi* is a modification of the MCNP-4C code [31] designed to model each neutron-nucleus interaction as accurately as possible. Unlike newer versions of MCNP, *MCNP-PoliMi* can only utilize a single processor for running simulations. The version of *MCNP-PoliMi* (1.22) used for these simulations is a slightly modified version of the standard version (1.0) available from the Radiation Safety Information Computational Center (RSICC). [55] The memory dynamically allocated storage variable in all modules was changed to "mdas=500000000" to allow for more complex models. In addition, the format of the output files was altered slightly in order to more easily identify events corresponding to directly transmitted neutrons or cross talk between detectors. The modules were compiled for the Linux operating system with the Portland Group [56] compiler. The modules were compiled with the "–Mlfs" flag, which allows the program to produce output files in excess of 2 gigabytes.

The latest versions of MCNP, *MCNP-5* [57] and *MCNPX* [58], make some approximations that generally produce accurate results when averaged over a large number of particles. However, when the quantity of interest is the behavior of a single particle, these approximations can generate unsatisfactory results. [59] One example is the production of neutrons and gamma rays from fission events. *MCNP-PoliMi* samples the number of neutrons and gammas produced by each fission event from a probability

58

distribution function, whereas the other codes use average values[2]. When calculating source-triggered multiplicities or detector-detector correlations, these approximations can yield incorrect results because the relatively rare fission events which produce a large number of particles simultaneously are not accounted for.

Another area where standard MCNP codes do not accurately model each collision is the production of gamma rays. These codes sample the gamma ray production of each interaction from a single table without regard to the type of neutron interaction. This table gives the average number and energy of gamma rays produced by all possible interactions, which speeds simulation times at the cost of the fidelity of individual interactions. By sampling gamma ray production separately for each different type of reaction, *MCNP-PoliMi* produces accurate results for each history.

MCNP-PoliMi was selected for NMIS modeling because the more accurate modeling of interactions produces more accurate results for multiplicities and detector-detector correlations. Although these features are not generally important for the simulation of imaging measurements, another important feature of *MCNP-PoliMi* is that it produces an ASCII text file containing information about all interactions in detector cells specified by the user. This information includes the type of projectile, nucleus interacted with, type of interaction, energy imparted by the interaction, time of the interaction (relative to the start of the history), position of the interaction, and the number of previous interactions. A small section of one of these output files with headers identifying each column is shown in Figure 4-1.

---

[2] The latest version of *MCNPX* (2.6.0) now samples the fission neutron multiplicity distribution correctly, but still uses average values for the gamma ray production. [60]

History Number	Particle Number	Particle Type	Nucleus	Cell Number		Energy Imparted	Interaction Time	x	y	z	Particle Weight	Generation Number	Number of Scatterings	Code	Initial Energy
8301	1	1	-99	1001	471	13.89887	4.312	86.26	10.03	-1.07	1.000	0	0	0	1.410E+01
8301	1	1	-99	1001	471	0.04337	4.314	86.26	10.03	-1.07	1.000	0	1	0	2.004E-01
8301	1	1	-99	1001	475	0.06031	5.081	86.45	13.86	0.67	1.000	0	2	0	1.570E-01
8301	1	1	-99	6000	475	0.02516	5.120	86.35	13.99	0.71	1.000	0	3	0	9.669E-02
8301	1	1	-99	1001	471	0.04911	6.430	86.15	9.52	-1.14	1.000	0	4	0	7.154E-02
8301	1	1	-99	1001	471	0.02158	6.437	86.14	9.51	-1.15	1.000	0	5	0	2.242E-02
8301	1	1	-99	1001	471	0.00060	7.729	85.93	9.15	-0.84	1.000	0	6	0	8.424E-04
8301	1	1	-99	1001	471	0.00019	13.410	84.78	8.94	-0.67	1.000	0	8	0	2.264E-04
8382	10	2	1	6	439	0.37189	3.757	94.38	-38.73	0.52	1.000	1	0	19	5.760E-01
8677	5	1	-99	1001	427	0.61476	10.184	77.32	-51.43	0.64	1.000	1	2	0	6.912E-01
8933	1	1	-99	1001	443	1.20065	17.883	85.48	-31.44	-0.87	1.000	0	5	0	1.446E+00
8949	1	1	-1	6000	463	0.07997	4.299	85.78	-2.64	0.93	1.000	0	0	0	1.410E+01
8949	2	2	1	6	463	2.44359	4.316	90.72	-2.97	-0.08	1.000	0	0	51	4.439E+00
8949	2	2	1	1	463	0.55382	4.317	90.96	-3.05	-0.28	1.000	0	1	51	1.995E+00
9115	5	2	1	6	419	0.01793	4.008	76.34	-64.33	0.62	1.000	0	5	4	2.006E-01
9115	5	2	1	6	419	0.02549	4.016	78.56	-64.08	-0.35	1.000	0	6	4	1.827E-01
10106	2	2	1	6	463	2.26776	3.373	85.07	-1.19	-0.89	1.000	0	1	4	4.149E+00
10138	1	1	-99	6000	447	0.01759	12.868	89.25	-26.27	-0.68	1.000	0	17	10	1.304E+00
10138	1	1	-99	1001	447	0.80011	12.902	89.77	-26.22	-0.81	1.000	0	18	10	1.286E+00
10138	1	1	-99	6000	447	0.05036	12.927	89.93	-26.08	-0.70	1.000	0	19	10	4.864E-01
10138	1	1	-99	1001	447	0.24389	13.146	89.12	-24.42	0.06	1.000	0	20	10	4.362E-01
10207	2	1	-99	1001	515	2.75086	6.537	75.14	69.48	0.48	1.000	1	0	10	5.745E+00
10502	13	2	1	6	519	0.96229	17.458	78.67	76.81	-1.04	0.999	1	0	19	2.004E+00
11406	1	1	-99	6000	503	0.11277	10.650	79.82	52.94	1.18	1.000	0	4	40	1.265E+00
11852	1	1	-99	6000	503	0.80053	5.699	86.13	54.91	0.39	1.000	0	3	10	6.668E+00
11870	1	1	-99	6000	435	0.11337	19.287	86.10	-41.60	0.74	1.000	0	20	10	5.381E-01
11870	1	1	-99	6000	435	0.04806	19.450	86.02	-41.15	-0.66	1.000	0	21	10	4.247E-01
11870	1	1	-99	6000	435	0.08184	19.462	85.92	-41.15	-0.69	1.000	0	22	10	3.767E-01
11870	1	1	-99	1001	439	0.27344	19.944	87.28	-37.92	0.19	1.000	0	23	10	2.946E-01
11870	1	1	-99	1001	439	0.00145	20.500	88.39	-38.06	0.24	1.000	0	24	10	2.118E-02
11870	1	1	-99	1001	439	0.01407	20.766	88.87	-38.16	0.39	1.000	0	25	10	1.969E-02

Figure 4-1. A section of a .dat file with column headings. The *MCNP-PoliMi User's Guide* [54] provides a detailed explanation of each column.

This text file, typically referred to as a .dat file because of its standard extension, is then analyzed with a post-processing code. The post-processor explicitly simulates the response function, energy threshold, and dead time of a specific detector type to calculate time correlations and multiplicities. For simulated imaging measurements, the post-processor also calculates the number of correlations falling in the relevant time windows and sums them up for each detector. Specific details of the post-processor, *PoliMiPP*, used for this work are provided in Section 4.3.2.

4.2 The Methodology of Simulating an NMIS Imaging Measurement

The process of running a single *MCNP-PoliMi* simulation is fairly simple. First, an input deck is created with the desired source, detector cells, and other geometry such

as the object being interrogated or imaged. The *MCNP-PoliMi* program is then executed

and produces a .dat file with all of the collision information in the detector cells. The .dat

file is post-processed to calculate correlations, multiplicities, and a file with the extension

.peaks which contains the number of correlations falling in the desired time window for

each detector.

Simulating an imaging measurement requires considerably more work. Even a

single 1D slice through an object requires many simulations. Each of the pixels has a

different angular distribution of source neutrons. A separate simulation is required for

each. In addition an NMIS imaging measurement typically subsamples the detector

locations in order to increase the angular resolution of the image. This is done by

repeating the measurement multiple times, and rotating the entire detector array slightly

with each subsample. Because *MCNP-PoliMi* explicitly records the interactions in each

detector volume, each subsample requires a separate simulation with the appropriate

angular transform of the detector array. If there are m pixels and n subsamples, a total of

$m \times n$ *MCNP-PoliMi* runs are required to simulate the measurement.

In order to minimize the tedious work (and the likely errors) involved with

building, executing, post-processing, and recombining each of the $m \times n$ *MCNP-PoliMi*

runs manually, a series of codes were written to make the process largely automated. A

flowchart of the programs used to simulate an NMIS imaging measurement is presented

in Figure 4-2. The *MakeInp* program reads a single PoliMi input deck containing the

geometry of the object being imaged and (if desired) other objects in the room. Using

information about the detector array and source distribution contained in another file, it

produces *MCNP-PoliMi* input decks for each pixel-subsample combination and places

61
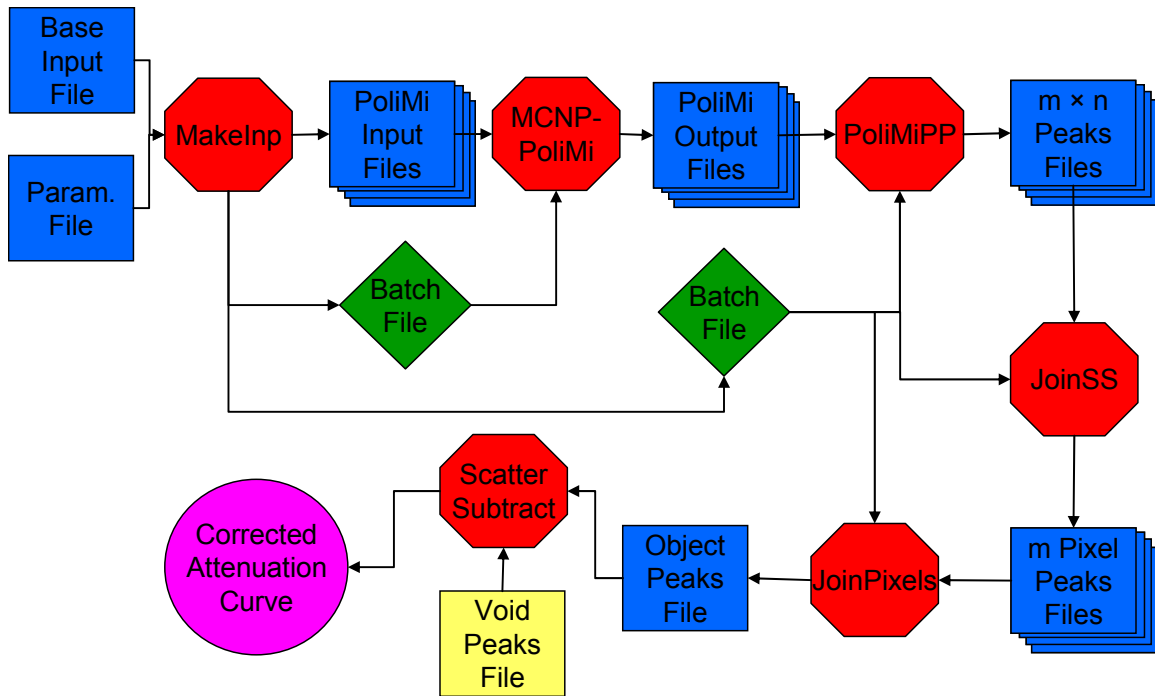
Figure 4-2. A flow chart showing the procedure used to simulate an NMIS imaging measurement with *MCNP-PoliMi.*

them in appropriate directories. The *MakeInp* program also produces two batch files.

The first batch file starts all of the *MCNP-PoliMi* runs (two different versions are created

- one for Windows systems and one for Linux servers). Once all of the runs are

complete, the second batch file post-processes the resulting .dat files and then combines

all of the resulting .peaks files into a single file. The *ScatterSubtract* code will then read

the final output and, using the output from a void measurement as well, apply the scatter

subtraction methodology developed in this work to produce the corrected attenuation

values. With the exception of *MCNP-PoliMi*, all of the programs shown in Figure 4-2

were written specifically for the purpose of simulating NMIS measurements. A detailed

description of each of these codes is presented in Section 4.3.

4.3 Fortran Codes For Simulating Imaging Measurements

4.3.1 The *MakeInp* Code

The *MakeInp* program takes a single base input deck containing information about the object being imaged and produces a series of *MCNP-PoliMi* input decks with the proper detector array configuration and pixel source distribution. The program is launched from an MS-DOS command prompt or batch file using the syntax

MakeInp <Parameter File>.

The parameter file contains information specified by the user about the detector array and the DT pixels as well as the name of the base input file. The information about the detector array includes the number and size of detectors, their separation, the source-to-detector distance and the detector cell numbers. The information about the pixels includes the number of pixels, FWHM of the pixels, angular offset, and the number of angular bins to be used for simulating the neutron distribution of the pixels. The parameter file also specifies the number of times each detector is to be subsampled, the number of source neutrons to be used for each simulation, and the file name of the base input deck. If the parameter file name is not specified at the command prompt, the program prompts the user to input the name. An example of a parameter input file is presented in Appendix C.

Once the parameter file name is input, the program reads the value of each parameter from the file and then closes it. Each value is specified with a keyword, just as in an MCNP deck. Using this information, *MakeInp* calculates the values needed to generate the surfaces for each of the detectors in the array. These values are computed for an array with no angular rotation, i.e., the center of the detector array lies on the x-z

plane. These detector surface values are computed using the source-to-detector distance, number of detectors, and detector-to-detector separation values read in from the parameter file and saved in an array for later use.

Once the detector positions are computed, the base input deck is opened and read into memory. The base deck is a standard *MCNP-PoliMi* deck containing information about the object being imaged and other objects in the room, such as the turntable used for rotating the object. The base input deck does not contain an "NPS" card since the number of source particles is specified in the parameter file. It also contains three special comments of the form "c 111111," "c 222222," or "c 333333" to indicate the location where the *MakeInp* program should add the detector cell cards, detector surface cards, and data cards in each of the resulting output decks. An example of a base input deck is given in Appendix C.

The program then creates a directory structure that will be used to hold the resulting PoliMi decks it creates. The directory name is the first four characters of the base input file. Once the directory is created, the program loops over the pixel numbers. For each pixel, a subdirectory is created to hold all of the input decks relating to that pixel. The program then loops over the subsample number and creates a blank file which will hold the input deck for the specific pixel-subsample combination. The filename given to this new input deck is the first two characters of the base file name plus the pixel number plus the subsample number followed by the extension ".i." For example, if the base input file is named "void.inp," the input deck for pixel 5, subsample 1 would be "vo51.i." This naming scheme is necessitated by the fact that *MCNP-PoliMi* file names must contain eight or fewer total characters. Up to 99 pixels and 99 subsamples can be

utilized with this naming scheme while maintaining a file name of eight characters or less.

Next, the program writes the *MCNP-PoliMi* cards to this new input file. This is accomplished by looping over each line read in from the base input deck. The output depends on the contents of the line from the input file.

- If the input line = "c 111111," the detector cell cards are written. Each cell card contains the appropriate detector cell number, material number, density, and surfaces for the detector. Cell numbers are assigned so that they increase sequentially with angle and each detector position is given a unique cell number. If there are four subsamples and the starting detector number is 401, subsample one would have detector cells 401, 405, 409, …; subsample two would have detector cells 402, 406, 410, …; etc.

- If the input line = "c 222222," the detector surface cards are written. The detector surfaces are specified using a BOX macrobody card. The BOX card specifies a starting point and three perpendicular vectors for the detector crystal. It also specifies an angular transform specific to the subsample. Each detector surface card is given the same number as its corresponding cell card.

- If the input line = "c 333333," the detector transform (TR), source definition (SDEF), number of source particles (NPS), random number (DBCN), physics (PHYS), problem cutoff (CUT), and PoliMi specific (IDUM, RDUM) cards are written.

  o The detector transform cards rotate the entire detector array by up to $\pm\frac{1}{2}$ the angle between detector centers for subsampling and translates the

position of the array if the source location is not at the origin of the MCNP coordinate system.

o The SDEF card specifies the location of the DT neutron source, the direction of the center of the current pixel, and a probability distribution function which defines the shape of the pixel. The source particles are monoenergetic neutrons. No deuterons or alphas are simulated and all neutrons emerge from a point source. The FWHM parameter from the parameter file determines both the width of the pixel PDF and the distance from the center of one pixel to the next. The absolute angle of each pixel includes an offset which specifies how many degrees the center of the pixels lies from the y-axis.

o The number of source particles is taken directly from the parameter file. All simulations created by *MakeInp* use the same number of source particles.

o For each simulation, the same random number seed and stride is specified with the DBCN card. Within a given pixel, the starting history for each subsequent subsample is incremented by the total number of histories in previous subsamples. This prevents any pattern in the random numbers from repeating across subsamples.

o The PHYS:n and PHYS:p cards specify how the program is to handle the physics parameters. *MCNP-PoliMi* requires analog operation, so these cards specify the proper values.

o The CUT:n and CUT:p cards specify when a history should be ended. A maximum time window is specified in the parameter file, and this value is written into the cut cards. The time window represents the maximum time difference the post-processor will use when computing source-detector and detector-detector correlations.

o The IDUM and RDUM cards provide *MCNP-PoliMi*-specific information. The IDUM card can be used to specify one of the built in PoliMi sources such as ^{252}Cf spontaneous fission. This card is also where the number of detectors and cell number of each is specified. Only detectors listed on the IDUM card have their collision information written into the .dat file. The RDUM card specifies the minimum energy that must be transferred by an interaction in a detector cell for it to be recorded in the .dat file.

- All other lines from the base input deck are written directly into the output deck with no changes.

Once the output deck is written, it is copied into the appropriate directory form where it will be executed. The command for executing that deck is written into the execution batch file, and the command for post-processing it is written into the post-processor batch file. At the end of each subsample, a command calling the *JoinSS* program is written into the post-processor batch file. A call for the *JoinPixels* is added to the very end. Once the batch files are completed, they are moved into the correct directory and the *MakeInp* program terminates. An example of an input deck produced by *MakeInp* is shown in Appendix C.

4.3.2 The *PoliMiPP* Post-Processor

After the *MCNP-PoliMi* simulations are complete, the *PoliMiPP* code is used to post-process the resulting .dat files. Although the RSICC release of the *MCNP-PoliMi* code includes a post-processor, it was found to be unsatisfactory for some aspects of NMIS simulations. The RSICC post-processor is written in the MATLAB [61] programming language. This code can only process small .dat files of less than ~100 MB. Larger files have to be separated into multiple pieces smaller than this limit before post-processing. The MATLAB code processes the files at a considerably slower rate than a comparable Fortran code. The MATLAB code also lacks some features, such as an adjustable detector dead time. The *PoliMiPP* code was written to provide all of the necessary features and the ability to process large .dat files of up to 4 GB without the need for splitting. The source code for *PoliMiPP* is shown in Appendix D.

PoliMiPP is executed from an MS-DOS command line with the syntax

PoliMiPP <.dat file> <start detector> <NPS> <Correlation Window> <dead time> <detector threshold>.

All of the input values are optional with the exception of the .dat file name. The start detector specifies which detector should be used as the start detector when computing cross-correlations. For imaging measurements, this value is set to "0" which tells the post-processor not to calculate cross-correlations. The NPS value should be the same one specified in the *MCNP-PoliMi* input deck for proper normalization. The correlation window specifies the maximum time difference that should be used when computing source-detector correlations and detector-detector cross-correlations. The correlation window time is in nanoseconds. By default, the *PoliMiPP* output will be binned in 1 ns

intervals, but if a window size of greater than 2048 ns is entered, bins of 1 μs will be used

instead. The detetector dead time is also specified in ns. Once a pulse is generated in a

detector, any other events in that detector will be ignored until the dead time expires. The

detector threshold specifies the minimum neutron energy that can generate a pulse if it

transfers its full energy in a collision. The energy threshold is in MeV. If only the .dat

file name is entered at the command line, the other values will assume default values.

These values are the equivalent of launching *PoliMiPP* with the values

PoliMiPP <.dat file> 0 1 256 35 1.0.

The *PoliMiPP* code begins by opening the .dat file and reading through it to

determine the number of source histories it contains. In order to minimize the total

amount of random access memory (RAM) used by the program, the entire .dat file is not

read into memory. Instead, each history is read into memory individually. For a given

history, each event is sorted by the detector number and then by interaction time. For

each event, the light output is computed using an algorithm based on the detector type.

For a plastic scintillator, the light output (pulse height) produced by an energy transfer,

E_{in}, is [62]

- Photon on hydrogen or carbon: $PH = E_{in}$.

- Neutron scattering off hydrogen: $PH = 0.125 * E_{in} + 0.0364 * E_{in}^2$.

- Neutron scattering off carbon: $PH = 0.02 * E_{in}$.

Once the light output for each event in the history is computed, the detector threshold is

applied to see which events produce an electronic pulse. Multiple events that occur

within the pulse generation time can combine to produce an electronic pulse if their sum

is greater than the threshold. Once a pulse is triggered, any events occurring in that

detector during the specified dead time after the pulse are ignored. Each pulse produced is written into an array for further processing. While recording the pulses, each pulse is checked to determine if it was produced by a directly transmitted neutron.

Once the list of pulses is calculated, the *PoliMiPP* code calculates the multiplet for each history. The multiplet is the total number of pulses (from all detectors) produced in a given history within the total time window specified in the program. In addition to the total multiplet, the total number of pulses produced only by neutrons is computed. The total number of times each multiplet (single, double, triple, etc) occurs is recorded and then output into a multiplicity file with a ".multip" extension.

After the multiplicities are computed, the post-processor calculates the source-detector correlation for each detector. This is accomplished by recording the number of times a pulse occurs in a given 1 ns time bin from the beginning of each history in a given detector. The correlations are divided into total, neutron, and gamma pulses. For an event in which more than one pulse occurs, cross-correlations are also computed if one of the pulses occurs in the start detector. These cross-correlations are computed by recording the time difference between each pulse in the start detector and all other pulses for that history in 1 ns bins. The cross-correlations are broken down into total, neutron-neutron (nn), photon-photon (pp), neutron-photon (np), and photon-neutron (pn) subsets. Each subset of correlations and cross-correlations is then output to text files.

Once the correlations are calculated, the location of the neutron peak is calculated by determining the time bin of the total correlation that contains the largest number of pulses. The time limits of the peak time window are then set at ± 2 ns from the peak. For each detector, the total number of counts within the peak window is recorded. In addition

70

to the total number of counts in the peak, the counts produced by directly transmitted neutrons are recorded.

Once the peak correlation window is determined, *PoliMiPP* applies an algorithm to eliminate counts in the total peak values produced by cross talk between detectors. This method follows the one applied in the NMIS data acquisition software. If a single history produces more than one count in the peak correlation time window, only the one with the smallest interaction time is counted. All others are considered to be produced by cross talk between detectors. In the unlikely event that two events occur simultaneously (because of rounding of the interaction time in the .dat file) the one in the higher numbered detector cell is discarded. Once these computations are completed, the number of total, direct, and no cross talk counts in the fast time correlation window are written out into a .peaks file. The total NPS value is also written into the .peaks file for normalization purposes.

For neutron radiography, the .peaks file is the final product of the post-processor program. The .peaks file consists of seven columns. The first column lists the detector cell numbers in ascending order. The next two columns give the number of total correlations in the peak correlation window and the mean correlation time of the window. The fourth and fifth columns present the same information for directly transmitted neutrons, and the sixth and seventh give the no cross talk peak correlations and means. The first row of the output is a header row with labels for all columns except for the detector cell number column. Instead of a label, the number of source histories is written above the detector column. This value will be used for normalization in the *ScatterSubtract* code. The .peaks file is used for reconstructing the attenuation maps

71

using the *JoinSS*, *JoinPixels*, and *ScatterSubtract* programs. The data in the .peaks file

will be used for extracting the PScFs in Chapter 5. An example of a .peaks file is shown

in Appendix D.

4.3.3 The *JoinSS* and *JoinPixels* Codes

Once all of the .dat files for a given pixel have been post-processed, the *JoinSS*

program combines them into a single .peaks file that contains the entire transmission

profile for that pixel across all subsamples. The *JoinSS* code is launched from the

command line with the syntax

JoinSS <File Base> <# SS> <# Detectors> <First Detector #> <Source to Detector

Distance> <Detector to Detector Distance>.

 The File Base consists of the first two characters of the base input file name (from the

MakeInp code) and the pixel number, e.g., vo4. The next two values specify the number

of subsamples and the number of detectors in the array. The first detector number is the

cell number of the lowest numbered detector cell in subsample 1, which will be the

lowest for all subsamples. This detector will have the most negative detector angle. The

source-to-detector distance and detector-to-detector distance are used to calculate the

detector angles. Detector-to-detector distance is the distance between the centers of

adjacent detectors in the array.

JoinSS opens the .peaks files for each subsample and reads the detector call

numbers, total correlations, direct correlations, and no cross talk correlations from each

into an array. Simultaneously, the angle of each detector position is computed. The

subsamples are interleaved in such a way that the detector positions are sorted by detector

angle. Then, this interleaved array is written into a single .peaks file which contains all of the values for the pixel. This .peaks file contains five columns. The first is the detector angle in degrees. The second is the detector cell number. The third, fourth, and fifth columns contain the number of total, direct, and no cross talk correlations in the fast neutron peak. The first row is a header row and, as with the original .peaks files, the NPS value is included in the first column of the header row.

Once the *JoinSS* program has rejoined all of the subsamples for each pixel, the *JoinPixels* program combines the output into a single file. The *JoinPixels* program requires that all of the *JoinSS* files are in a single directory. The post-processing batch file produced by the *MakeInp* code copies those files into the root directory of the simulation before launching the *JoinPixels* program. The *JoinPixels* program is launched from a command line with the syntax

JoinPixels <base file name> <number of pixels>.

The base file name is the first two characters of the original *MakeInp* base input deck.

When the *JoinPixels* program is launched, it opens each of the pixel .peaks files and reads them into an array in memory. It then writes the output into a single .peaks file for the simulation. The output in the final .peaks file is divided into three groups of columns representing the total, direct, and no cross talk values. Each group consists of a column with the detector angles, a column with the detector cell numbers, a column for each pixel, and a column with a total value which is the sum of all pixels. One additional value contains the uncertainty of the total column. The uncertainty is calculated assuming standard counting statistics by taking the square root of the total correlations. The final output file contains a header row with labels for each column except for the

73

first, which gives the NPS value for normalization. This final .peaks file will be used in

combination with one from a void measurement to calculate the attenuation profile using

the *ScatterSubtract* code. The source code for the *JoinSS* and *JoinPixels* programs is

shown in Appendix E.

4.3.4 The *ScatterSubtract* Code

Once the final .peaks file for the simulation has been finished, the *ScatterSubtrac*t

program implements the point scatter removal algorithm in order to remove the scattered

component from the measured values. The function of this program will be discussed in

detail in Section 6.2 where the implementation of the PSRA is presented.

5. Modeling and Point Scatter Function Extraction

The PScFs will be discussed in detail in this chapter. In the first section the definition and mathematical description of the PScFs as used in this work will be discussed. A definition from earlier work was provided in Chapter 2, but the function will be modified slightly to account for the geometry of the NMIS detector array. In the second section, the models used to measure the PScFs and the scenarios which were modeled are described. In the third section, the procedure for extracting the PScF parameters for each simulation is presented. The parameterized PScFs will be compared to the elastic scattering calculations from Chapter 3 in the final section of this chapter.

5.1 Description of the Point Scatter Functions

Some previous work with PScFs was described in Chapter 2. Those authors described the point scatter function in terms of the probability a neutron would arrive a given distance from its projected point on a 2D detector screen. This work requires a somewhat modified definition of the PScF in order to account for the more complicated detector geometry and the finite detector volumes. The definition of the PScF used here will be **the number of additional fast neutron correlations due to scattering in the object being imaged recorded in a detector whose center is an angle, θ, away from the original detector the neutron was directed towards per uncollided neutron detected by the original detector in a void measurement.** Mathematically, this relation can be represented

$$PScF(\theta) = \frac{I_{SO}(\theta)}{I_0},$$

(5.1)

75

where $I_{SO}(\theta)$ is the number of additional counts in the detector an angle θ away from the original detector as measured from the DT target spot and I_0 is the number of DT neutrons detected in the original detector in the void measurement. The PScF accounts for neutrons scattered at any position in the object along the original neutron path. The PScF geometry is depicted in Figure 5-1.

Each detector will have its own PScF which describes the scattering contribution to all detectors (including itself) in the array produced by neutrons directed towards it. The total scattering contribution *into* each detector in the array would then be superposition of the PScFs from all detectors in the array. The total object scattering contribution into a given detector can be written as a discrete summation of the contribution from all detectors. This summation can be written

$$I_{SO}(i) = \sum_{j=1}^{n} I_0(j) PScF_j[(j-i)\omega],\qquad(5.2)$$

where i represents the position of the detector being scattered into, j represents the detector whose neutrons are contributing additional scattering, $I_0(j)$ is the directly transmitted neutrons directed towards detector j, n is the number of detectors in the array, and ω is the separation angle between adjacent detectors. If the contributions from background and room return are negligible, the true attenuation measured by detector i can be written

$$\tau(i) = \frac{I_{meas}(i) - \sum_{j=1}^{n} I_0(j) PScF[(j-i)\omega]}{I_0(i)}.\qquad(5.3)$$
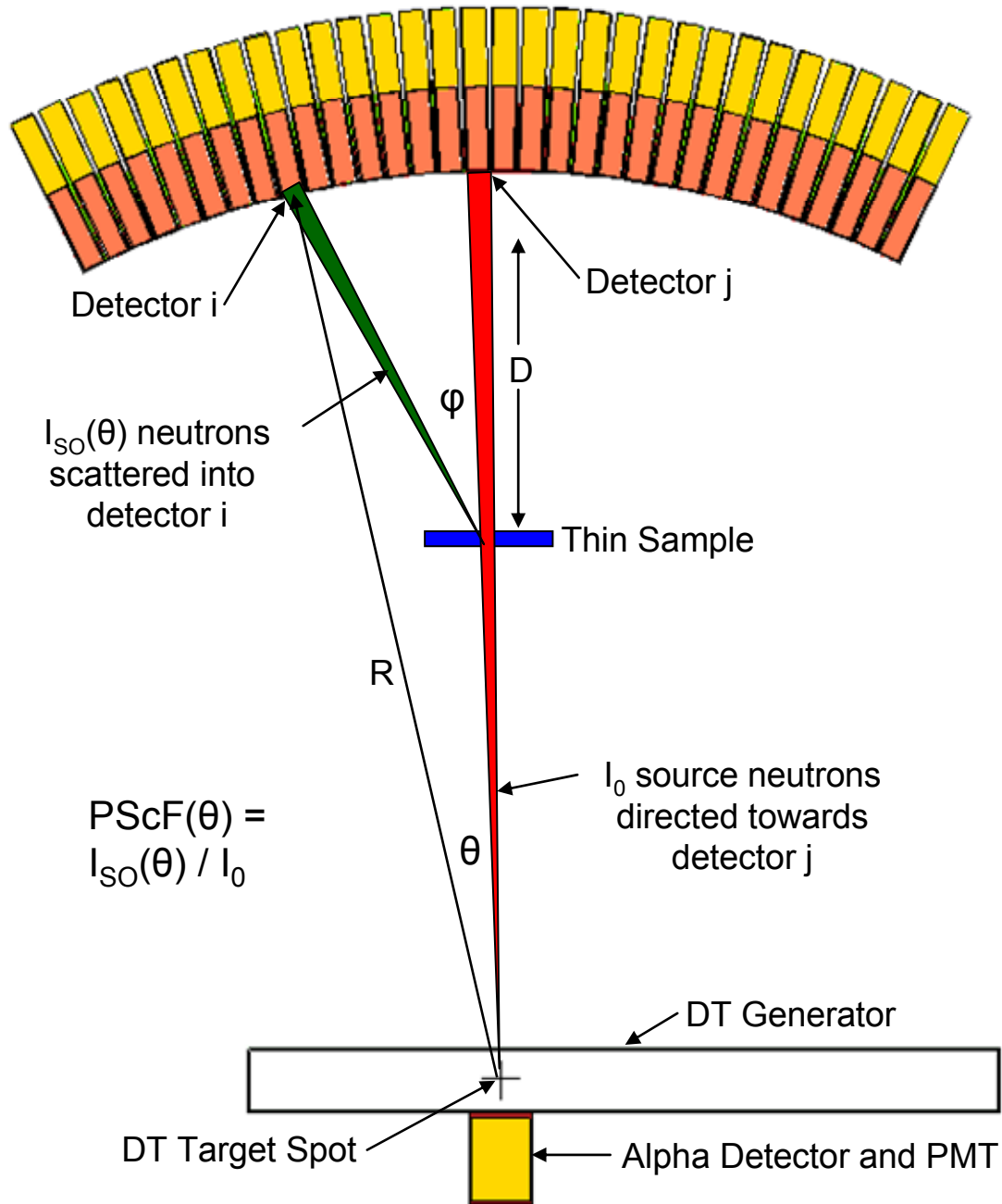
Figure 5-1. The geometry of the NMIS PScF. A total of I_0 neutrons directed towards detector j would be detected in a void measurement. Of those neutrons, $I_{SO}(\theta)$ scatter in the object being imaged and are detected in detector i whose center is an angle, θ, away from the original detector. Although a thin sample is shown for clarity, the PScF includes scattering from any position along the neutron track through the object, even if it is several MFP thick.

Because the I_0 and I_{SO} values cover an entire detector, the PScFs must be applied to each subsample individually. Otherwise, the attenuation values will be overcorrected and the resulting attenuation values will be too high.

If all of the PScFs are known, the object scattering can be removed from the object measurement and the true attenuation profile of the object can be calculated. Unfortunately, the PScFs depend on the geometry and materials of the object so they will change each time a new object is imaged. Calculating the exact PScFs each time would require an MCNP calculation using the correct object geometry and materials. Since the goal of radiography is to measure those values, this approach is impractical.

A more reasonable approach is to develop a library of generalized PScFs and apply the one that is the most applicable to a particular measurement. The remainder of this work will focus on this technique. The scattering calculations in Chapter 3 showed that the neutron scattering function changed with the scattering material and also with the distance between the object and the detector array. In addition, the thickness of the object will play a large role in the size and shape of the PScFs. In the remaining sections of this chapter, the focus will be on developing a library of PScFs that cover a broad range of object characteristics, and parameterizing them. In Chapter 6, a method for selecting the best available PScFs for each detector position in a given object will be developed.

5.2 Modeling and Simulation of the Point Scatter Functions

MCNP-PoliMi simulations will be used to estimate the PScFs for a wide range of scenarios in order to build a library. Because the post-processor reports the number of

correlations in the DT neutron peak for both directly transmitted and no cross talk values (which approximate measurements in the laboratory) the values of both the I_0 and I_{SO} can be calculated directly using the values in the .peaks files. In order to measure these values, the source term needs to approximate the geometry shown in Figure 5-1. In order to accomplish this, a pixel was centered on one of the detectors at the center of the array and then collimated horizontally through the use of a zero importance cell surrounding the neutron source. The collimating cell had a slit in it that had the same horizontal angular width as the detector. No vertical collimation was performed so that the section of the pixel had its full vertical extent. This allows the simulations to account for neutrons directed directly above or below the scattering detector when computing the scattering. The resulting neutron profile is a vertical fan which falls off as a COS^2 (cosine squared) shape toward the top and bottom edges. Because the fan is at the center of the pixel, it matches the vertical profile of the overall fan (the sum of all pixels – see Section 3.1.5) very well. This feature means that the PScFs derived here can translated to any detector in the array as the scattering detector.

Although the possible object configurations and materials are infinite, the NMIS operator will have only a limited amount of information available to be used for computing the correct PScFs to use for a given image. The exterior geometry of the object will be known. This is typically the dimensions of the drum or canister whose contents are being imaged. The location of the exterior with respect to the imaging array and source can easily be measured when setting up the measurement.

To a certain degree, the composition of the outer layer of material in the object is also available to the NMIS operator. Its composition can be estimated by looking at a

source-detector correlation curve for a detector located perpendicular to a pixel's neutron cone passing tangentially through the outside of the drum. Some specific nuclides, most notably hydrogen and low Z isotopes with widely spaced energy levels, can be identified by looking for peaks due to elastic and inelastic scattering. The operator might also have knowledge of what the shielding is supposed to be based on the declared contents of the object being imaged. For the purposes of this work, it will be assumed that only the composition of the outer layer of shielding can be determined and the PScF used will be based on this information. In order to be as widely applicable as possible, the PScFs will be calculated using a homogenous layer of shielding. During the validation phase, some work will be devoted to checking how well PScFs based on only the outer layer of material work with objects composed of more than one material.

A final piece of information that is easily available from an NMIS measurement in the laboratory is the measured attenuation curve. For each detector position, the best PScF can be selected based on the measured attenuation at that location and inputs from the operator about the shielding material and the object-to-detector distance. The resulting PScFs can then be subtracted to produce the corrected curve and new PScF values can be selected based on the corrected attenuation values. This iteration will be continued until the attenuation values converge at, ideally, the true attenuation curve.

Taking the available information about the object being imaged into consideration, the three variables selected for generating the PScFs are the object-to-detector distance, object material, and material thickness in attenuation lengths. The object used is a homogenous cylindrical shell of material oriented vertically with its radius centered on the DT target spot. This configuration gives the object a consistent

object-to-detector distance across the entire array removing possible variations in the profile of the resulting PScF created by the shape of the object. The configuration of the source and target that will be used for these simulations is shown in Figure 5-2.

Four different materials will be used for measuring the PScFs. These are polyethylene, carbon (graphite), iron, and lead. These materials cover a wide range of atomic masses and they are materials frequently used in shielding and structural applications. Fourteen thicknesses of materials will be used ranging from 0.5 to 7.0 MFP by half-integer values. Seven object-to-detector distances (measured from the outer surface of the object) ranging from 30 to 90 cm in 10 cm increments will be used for each material. Because the source-to-detector distance is 110 cm and the DT generator tube has a radius of approximately 4 cm, some combinations of material, object-to-detector distance, and thickness were physically impossible and were omitted from these simulations. In total, there are 300 allowable material, object-to-detector distance, and thickness combinations. All of the configurations used in these simulations are presented in Table 5-1.

Each of these simulations was given a unique identifier that consists of the first four letters of the material with a number appended to the end. The number is the sum of the object-to-detector distance and the nominal thickness in MFP. For example, 3 MFP of polyethylene at 50 cm from the detector is given the identifier *Poly53*.

The total number of source particles used for each scenario was determined by making short runs and extrapolating the number needed to generate a sufficiently large .dat file to get good statistics on the number of scattered counts in each detector.
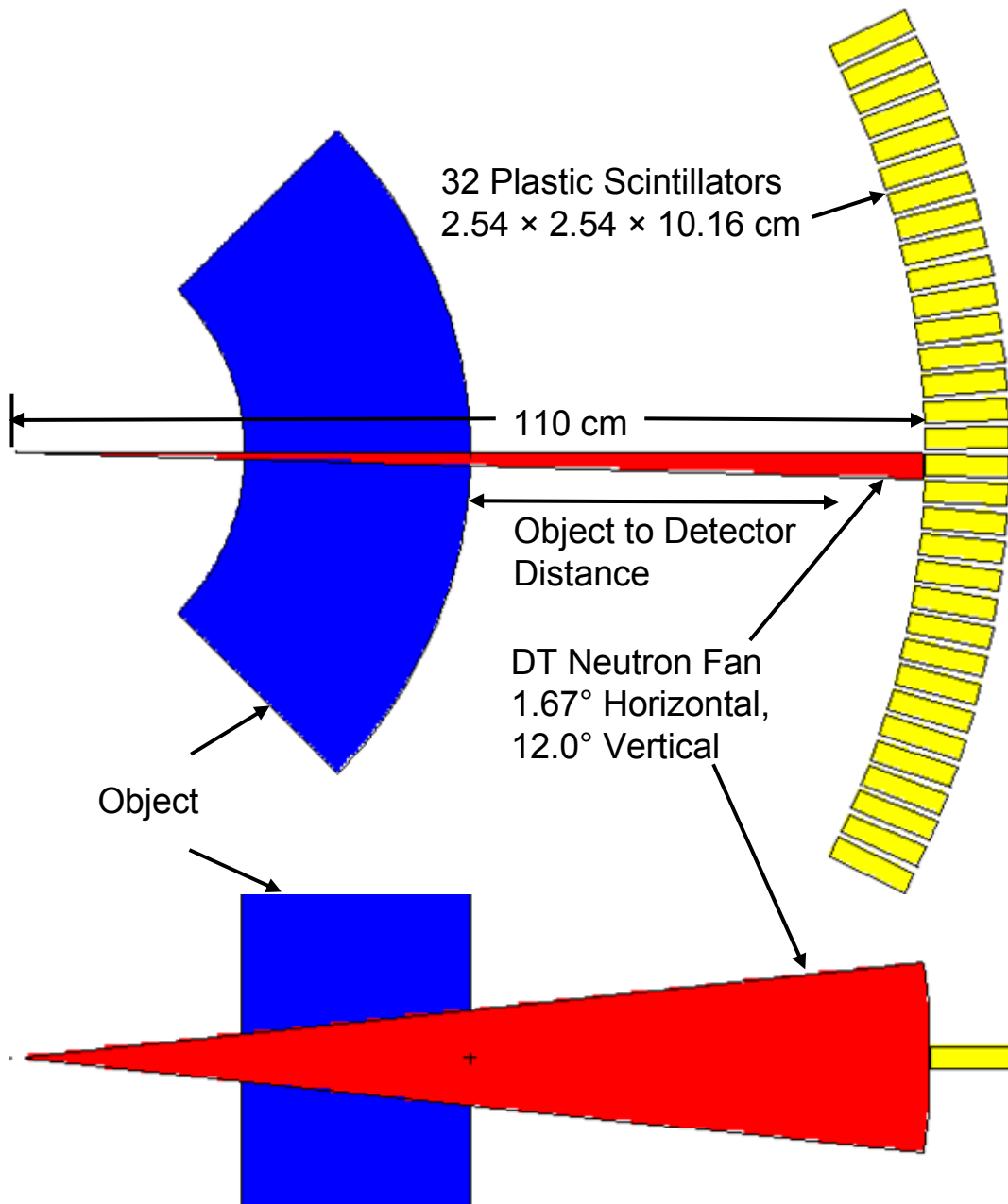
Figure 5-2. The configuration used for calculating the NMIS PScFs using _MCNP-PoliMi_ simulations.

Table 5-1. The configurations simulated for the purpose of developing an NMIS PScF library.

Object-to-Detector Distance	Allowable Thicknesses by Material			
	Polyethylene	Carbon	Iron	Lead
30 cm	0.5-7.0 MFP	0.5-7.0 MFP	0.5-7.0 MFP	0.5-7.0 MFP
40 cm	0.5-7.0 MFP	0.5-7.0 MFP	0.5-7.0 MFP	0.5-7.0 MFP
50 cm	0.5-6.0 MFP	0.5-6.0 MFP	0.5-7.0 MFP	0.5-7.0 MFP
60 cm	0.5-5.0 MFP	0.5-5.0 MFP	0.5-7.0 MFP	0.5-7.0 MFP
70 cm	0.5-3.5 MFP	0.5-4.0 MFP	0.5-7.0 MFP	0.5-6.0 MFP
80 cm	0.5-2.5 MFP	0.5-2.5 MFP	0.5-5.5 MFP	0.5-4.5 MFP
90 cm	0.5-1.5 MFP	0.5-1.5 MFP	0.5-3.0 MFP	0.5-2.5 MFP

The smallest .dat file size was just under 100 MB, and many .dat files exceeded 1 GB.

The number of source neutrons used for each combination of material and thickness are

given in Table 5-2. The same values were used for all object-to-detector distances for a

given material. Note that the neutron source strengh is for the entire pixel, including the

parts that were cut off by the zero importance collimating sphere. The actual number of

neutrons projected towards the object comprises only the portion of the pixel which

passes through the slit in the collimator, but because this value scales linearly with the

total source term, knowledge of its exact value is not required.

In addition to these simulations, a void simulation was run. The void simulation

had a source strength of 4×10^7 neutrons. It used the same geometry shown in Figure 5-2

except that no object was present. The void simulation will provide the I_0 value that will

be used to compute the PScF for each of the other simulations shown above.

Table 5-2. The source strength used for each of the PScF simulations.

Thickness (MFP)	Number of Source Neutrons ($\times 10^7$)			
	Polyethylene	Carbon	Iron	Lead
1	4	4	4	4
2	4	4	4	4
3	10	4	4	4
4	10	10	4	4
5	10	10	4	4
6	20	10	4	4
7	20	10	4	4

5.3 Parameterization of the NMIS Point Scatter Functions

5.3.1 The Void Simulation

The first task involved in parameterizing the PScFs was to run the void simulation and extract the results. The .peaks file generated from the void simulation is shown in Figure 5-3. The neutron fan was directed onto detector cell 461 in the center of the array. The *Direct* column of the .peaks files confirms that the only detector cell which recorded directly transmitted neutron correlations was detector 461. The *Total* and *No XTalk* results for detector 461 are almost identical to the *Direct* value because the only possible source of additional counts would be a neutron that scattered off detector 461 into another detector cell and back into 461.

Unlike the *Direct* column, the *Total* and *No XTalk* columns show a sizable number of correlations in the detectors surrounding 461. Although this is expected for the *Total* column, it is somewhat surprising for the *No XTalk* one, because the only way for a neutron to reach a detector other than 461 in the void simulation is for it to scatter from 461.

84

```
400000000  Total      Mean(T)      Direct   Mean(D)     No XTalk   Mean(N)
       401       92  2.3402E+01         0  0.0000E+00        72  2.3375E+01
       405       89  2.3382E+01         0  0.0000E+00        63  2.3413E+01
       409      110  2.3255E+01         0  0.0000E+00        75  2.3280E+01
       413      139  2.3108E+01         0  0.0000E+00        97  2.3093E+01
       417      186  2.3134E+01         0  0.0000E+00       127  2.3126E+01
       421      218  2.2977E+01         0  0.0000E+00       148  2.2973E+01
       425      284  2.2863E+01         0  0.0000E+00       187  2.2834E+01
       429      380  2.2755E+01         0  0.0000E+00       271  2.2738E+01
       433      569  2.2626E+01         0  0.0000E+00       385  2.2590E+01
       437      771  2.2578E+01         0  0.0000E+00       515  2.2540E+01
       441     1275  2.2547E+01         0  0.0000E+00       906  2.2539E+01
       445     2612  2.2660E+01         0  0.0000E+00      1938  2.2708E+01
       449     6394  2.2909E+01         0  0.0000E+00      4834  2.2920E+01
       453    27689  2.3115E+01         0  0.0000E+00     16009  2.2848E+01
       457   163536  2.2674E+01         0  0.0000E+00     88416  2.2525E+01
       461  5115751  2.1772E+01   5115435  2.1772E+01   5115540  2.1772E+01
       465   163357  2.2679E+01         0  0.0000E+00     87863  2.2532E+01
       469    27803  2.3114E+01         0  0.0000E+00     16073  2.2855E+01
       473     6549  2.2890E+01         0  0.0000E+00      4946  2.2907E+01
       477     2493  2.2677E+01         0  0.0000E+00      1840  2.2746E+01
       481     1268  2.2559E+01         0  0.0000E+00       876  2.2567E+01
       485      778  2.2557E+01         0  0.0000E+00       575  2.2558E+01
       489      543  2.2637E+01         0  0.0000E+00       389  2.2614E+01
       493      386  2.2767E+01         0  0.0000E+00       257  2.2786E+01
       497      318  2.2840E+01         0  0.0000E+00       222  2.2829E+01
       501      211  2.2938E+01         0  0.0000E+00       146  2.2897E+01
       505      179  2.3061E+01         0  0.0000E+00       129  2.2977E+01
       509      147  2.3143E+01         0  0.0000E+00       107  2.3159E+01
       513      122  2.3262E+01         0  0.0000E+00        84  2.3202E+01
       517       97  2.3330E+01         0  0.0000E+00        71  2.3352E+01
       521       79  2.3380E+01         0  0.0000E+00        56  2.3339E+01
       525       63  2.3429E+01         0  0.0000E+00        49  2.3408E+01
```

Figure 5-3. The contents of the Void.peaks file.

If a neutron created a pulse in 461 and then scattered into another detector cell and created a pulse, the second pulse would be discarded as cross talk.

The source of the scattering can be found by examining the equations for detector efficiency (Equation 3.19) and the Pulse Height Factor (Equation 3.21) in Section 3.2.2. In order to generate a detector pulse, an incoming neutron needs to generate enough light in the detector to overcome the threshold. The results in Figure 5-3 were generated using the default neutron threshold of 1 MeV. With that setting, only neutrons imparting more than 1 MeV to a hydrogen atom in the detector cell generated a pulse. Neutrons scattering off a carbon atom or imparting less than 1 MeV to a hydrogen atom will not generate a pulse. Since no pulse is generated in detector 461 by these types of scattering, if the neutron is subsequently detected in another cell, that pulse is not considered cross talk.

In order to verify that the threshold is the cause of the scattering in the *No XTalk* column of Figure 5-3, the Void.peaks file was post-processed again using a detector threshold of 0.001 MeV. The.peaks file generated using the 0.001 MeV detector threshold is shown in Figure 5-4. At this low threshold value, the only counts in the *No XTalk* column outside of cell 461 are a small number of counts in cell 457. These counts are a result of the logic the *PoliMiPP* code uses to determine which pulse is cross talk. If two pulses for a given history have the same interaction time, the one in the higher numbered cell is discarded. Because the time resolution in the .dat file is limited to 100 ps, some small number of pulses will be misidentified in this manner.

```
400000000   Total      Mean(T)    Direct      Mean(D)    No XTalk    Mean(N)
      401          94   2.3404E+01        0   0.0000E+00         0   0.0000E+00
      405          94   2.3372E+01        0   0.0000E+00         0   0.0000E+00
      409         117   2.3248E+01        0   0.0000E+00         0   0.0000E+00
      413         145   2.3110E+01        0   0.0000E+00         0   0.0000E+00
      417         199   2.3121E+01        0   0.0000E+00         0   0.0000E+00
      421         233   2.2974E+01        0   0.0000E+00         0   0.0000E+00
      425         302   2.2858E+01        0   0.0000E+00         0   0.0000E+00
      429         409   2.2746E+01        0   0.0000E+00         0   0.0000E+00
      433         605   2.2621E+01        0   0.0000E+00         0   0.0000E+00
      437         844   2.2582E+01        0   0.0000E+00         0   0.0000E+00
      441        1447   2.2584E+01        0   0.0000E+00         0   0.0000E+00
      445        3553   2.2924E+01        0   0.0000E+00         0   0.0000E+00
      449       10323   2.3186E+01        0   0.0000E+00         0   0.0000E+00
      453       51209   2.3286E+01        0   0.0000E+00         0   0.0000E+00
      457      362006   2.2812E+01        0   0.0000E+00      1765   2.1732E+01
      461    11573820   2.1782E+01 11573117   2.1782E+01  11572055   2.1782E+01
      465      360881   2.2815E+01        0   0.0000E+00         0   0.0000E+00
      469       51286   2.3288E+01        0   0.0000E+00         0   0.0000E+00
      473       10483   2.3171E+01        0   0.0000E+00         0   0.0000E+00
      477        3353   2.2933E+01        0   0.0000E+00         0   0.0000E+00
      481        1408   2.2599E+01        0   0.0000E+00         0   0.0000E+00
      485         851   2.2561E+01        0   0.0000E+00         0   0.0000E+00
      489         594   2.2652E+01        0   0.0000E+00         0   0.0000E+00
      493         414   2.2756E+01        0   0.0000E+00         0   0.0000E+00
      497         336   2.2836E+01        0   0.0000E+00         0   0.0000E+00
      501         223   2.2937E+01        0   0.0000E+00         0   0.0000E+00
      505         189   2.3048E+01        0   0.0000E+00         0   0.0000E+00
      509         159   2.3157E+01        0   0.0000E+00         0   0.0000E+00
      513         131   2.3244E+01        0   0.0000E+00         0   0.0000E+00
      517         103   2.3311E+01        0   0.0000E+00         0   0.0000E+00
      521          85   2.3376E+01        0   0.0000E+00         0   0.0000E+00
      525          67   2.3433E+01        0   0.0000E+00         0   0.0000E+00
```

Figure 5-4. The Void.peaks file post-processed using a detector threshold of 0.001 MeV.

This problem cannot be corrected without modifying the *MCNP-PoliMi* source code; however, because the time resolution means the two events large enough to generate a pulse must occur very close together, the effect will be exceedingly small at more realistic detector thresholds.

The *No XTalk* values in Figure 5-3 indicate that even after the cross talk correction is made, there is a good deal of inter-array scattering that is not accounted for. Before the object scatter can be calculated, this scatter needs to be addressed. If not, its presence will reduce the measured attenuation values. This phenomenon will also present itself in the object simulations, creating a scattering effect caused by the directly transmitted neutrons in addition to that caused by scattering in the object. Unless corrected, the inter-array scattering of directly transmitted neutrons will change the shape of the fitted PScFs.

In order to address this problem, the scatter in the *No XTalk* column of Figure 5-3 was used to define an inter-array scatter function (ISF). The ISF is defined exactly the same way as the PScF except that the object scatter (I_{SO}) in Equation 5.1 is replaced with the inter-array scatter (I_{SA}). The mathematical expression for the ISF is

$$ISF(\theta) = \frac{I_{SA}(\theta)}{I_0}. \tag{5.4}$$

The first attempt to fit the ISF was made using a Gaussian of the form

$$ISF_{fit}(\theta) = \frac{A \exp\left(-\theta^2 / 2S^2\right)}{I_0}, \tag{5.5}$$

where $ISF_{fit}(\theta)$ is the number of scattered counts at angle, θ, per directly transmitted neutron, A is the maximum of the Gaussian fit, S is the standard deviation of the fit, and

I_0 is the number of directly transmitted neutrons detected in cell 461. An iterative

algorithm was used to find the best values of A and S_{SA} which minimized the χ^2 value

between the fit and the data. The χ^2 value is computed using the formula

$$\chi^2 = \sum_{i=1}^{n} \left(\frac{I_{SA,data}(i) - I_{SA,fit}(i)}{\sigma_{data}(i)} \right)^2 . \tag{5.6}$$

The method used to find the fit is nearly identical to the one that will be used to fit the

PScFs which will be described in the next section. Because self scatter is almost

nonexistent in the inter-array scatter, the detector at which the source neutrons were

directed (cell 461) was not used to generate the fit. That point will also be ignored when

subtracting the inter-array scatter from measured values of I_0. Unfortunately, the actual

ISF function proved to be too tail-heavy (high kurtosis) to be accurately represented by a

Gaussian fit. Although the Gaussian fit data within a few degrees of the detector very

well, it fell far below the actual ISF values farther away. Because the ISF contributions

from each detector will be superimposed, this effect will be greatly magnified and will

result in an undercorrection of the inter-array scattering. A plot of the Gaussian fit of the

inter-array scatter function along with the no cross talk values is given in Figure 5-5. The

total scattering is also shown for reference.

Using an exponential to fit the ISF resulted in the opposite problem – fit values in

the tail region were too large. A good fit in both the peak and the tails regions was

developed using a combination of a Gaussian and an Exponential function. This fit takes

the form

$$ISF_{fit}(\theta) = \frac{A\exp\left(-\theta^2/2S^2\right)}{I_0} + \frac{B\exp\left(-|\theta|/T\right)}{I_0}, \tag{5.7}$$
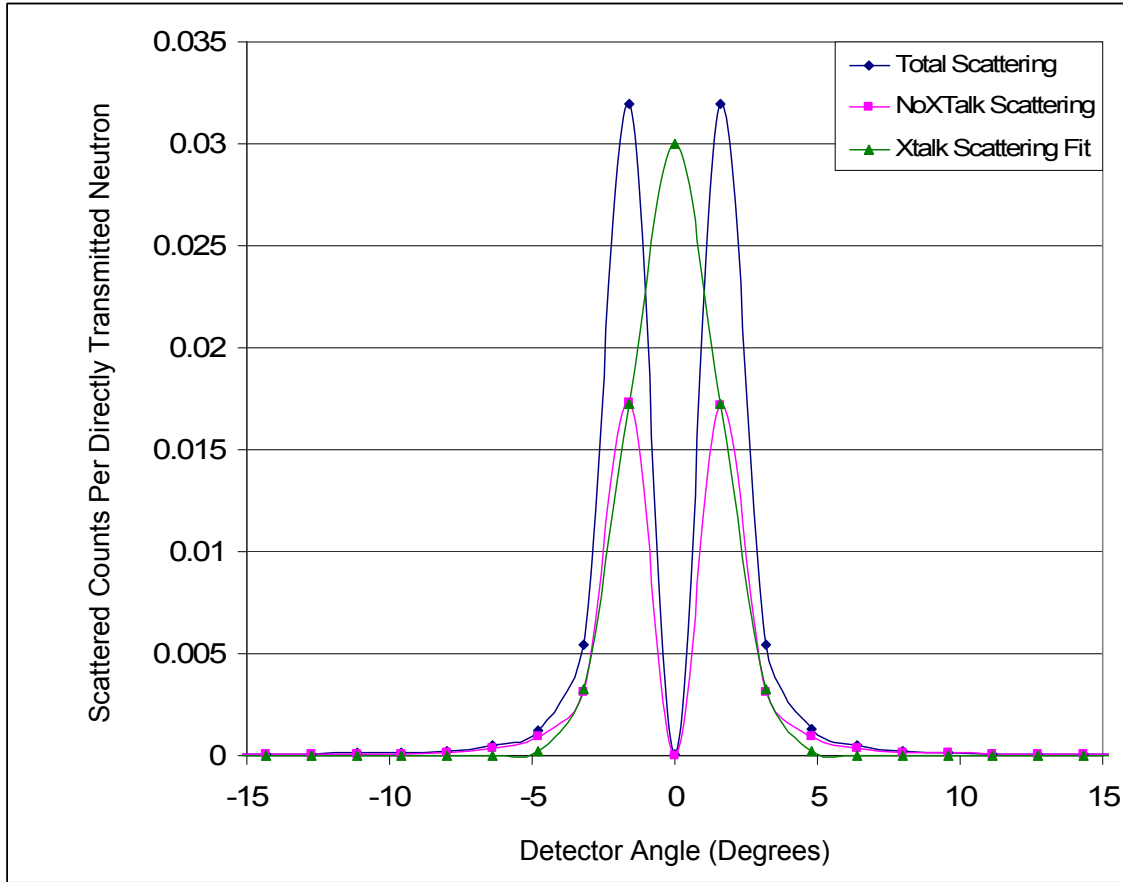
Figure 5-5. A comparison of the no cross talk inter-array scattering and the fitted ISF generated using a single Gaussian. The single Gaussian fits the data poorly at scattering angles of greater than ~4°. The *Total* scattering is shown for reference. The center point of the fit curve is not used for removing the inter-array scattering and is shown only for reference.

where *A* and *B* are the magnitude of the Gaussian and the Exponential respectively, and *S* and *T* are the standard deviations of the Gaussian and the Exponential. This dual fit lowered the χ^2 value of the fit by more than an order of magnitude. A plot of the new fit as well as the *No XTalk* and *Total* scattering values is presented in Figure 5-6. The best fit parameters for the ISF are

A = 0.029698; B = 0.004047; S = 1.420884°; and T = 2.911001°.

5.3.2 Extracting the Point Scatter Functions

The point scatter function for each of the scenarios described in Section 5.2 was extracted using a program called *GaussFit* which was written for this purpose. The source code for the program is shown in Appendix F. The procedure this program uses to extract the PScFs will be described in this section.

The *GaussFit* is launched from an MS-DOS command prompt using the syntax

GaussFit <Object .peaks file> <Void .peaks file> <Object-to-detector Distance> <Attenuation>.

The attenuation entered on the command line is the nominal attenuation of the object used when creating the PScF simulation. The value is only used for assigning a name to the scenario. The program then opens each of the .peaks files and reads them into an array. The source term (NPS value) for each is read in so that it can be used for normalization purposes. The angle of each detector position is calculated in terms of the angle between its center and the center of the detector (cell 461) into which the source neutrons were directed. For the simulation geometry, the angle between adjacent detector centers was approximately 1.67°.

91
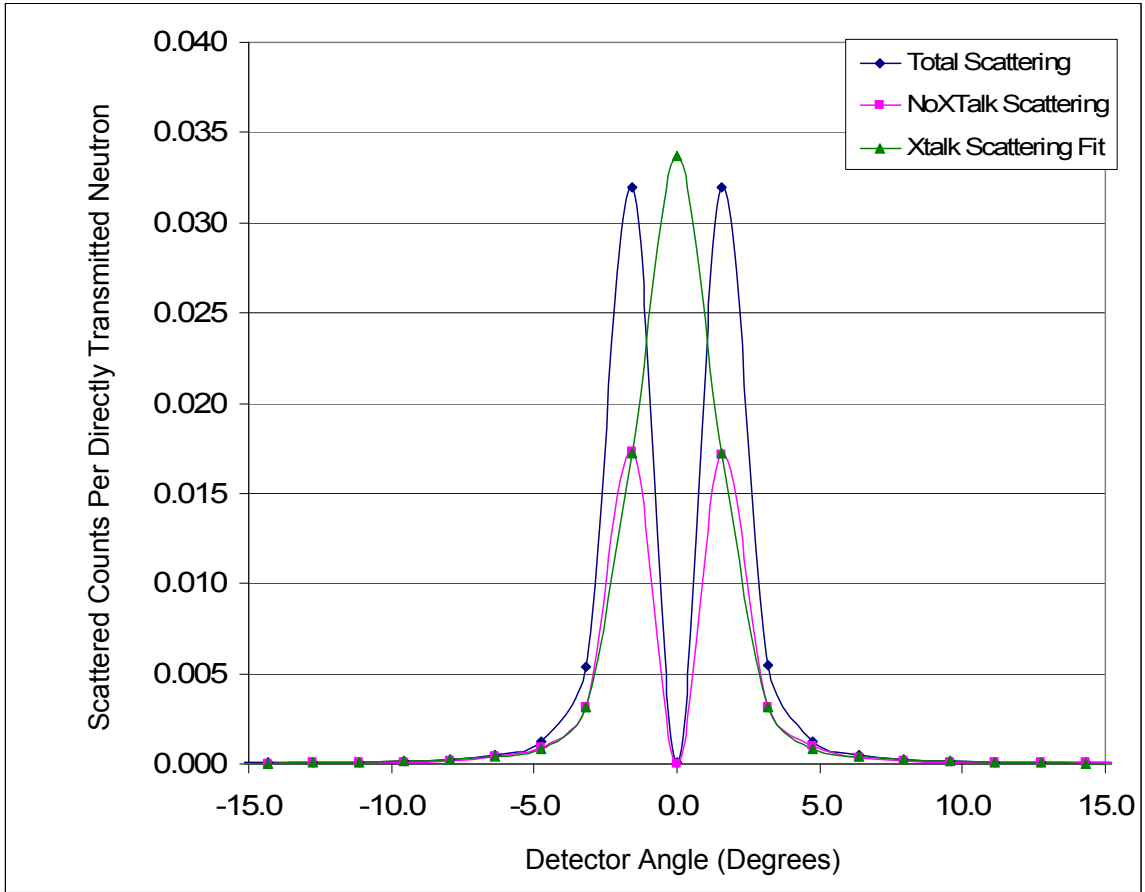
Figure 5-6. A comparison of the inter-array scatter values to the combination Gaussian-Exponential Fit. The *Total* scattering is shown for reference. The center point of the fit curve is not used for removing the inter-array scattering and is shown only for reference.

While reading the correlation peak values in, the total number of scattered counts in each detector is determined by subtracting the number of correlations in the *Direct* column from the number in the *No XTalk* column.

Once the .peaks files are read in, the program calculates the true attenuation of the directly transmitted neutrons using the formula

$$\tau = -\ln\left(\frac{C_{direct}}{C_{0,diewct}} \times \frac{NPSV}{NPSO} \right),$$ (5.8)

where NPSV and NPSO are the source values for the void and object simulations. Because the neutrons were only directed at a single detector, there is only a single attenuation value for each object simulation. Initial guesses for the maximum and standard deviation of the PScF fits are then taken from the object data. The initial guess for the maximum is the largest number of scattered counts in the array. The initial guess for the standard deviation is the standard deviation of the scattering angle of the data, which is computed via the formula

$$\sigma_{scatter} = \sqrt{\frac{\sum_{i=1}^{n} C(\theta)\theta^2}{\sum_{i=1}^{n} C(\theta)}}.$$ (5.9)

These initial guesses are then passed into a subroutine that uses them along with the NPS values, the attenuation, and the data from the .peaks files to find the best fit for the PScF.

The fitting routine begins by subtracting the portion of the scattering attributable to inter-array scattering from the total scattering to determine the scattering that is due only to scattering in the object. The form (Equation 5.7) and best fit values for the ISF were shown in Section 5.3.1. Because by definition, the ISF is generated by unattenuated

neutrons reaching the detector array, its magnitude must, by definition, follow the exponential attenuation shown in Equation 5.8. Thus, for the object measurement, the number of counts due to scattering in the array, C_{SA}, at angle, θ, is

$$C_{SA}(\theta) = C_{direct}(0) ISF(\theta) \frac{NPSV}{NPSO} \exp(-\tau).$$ (5.10)

After subtracting the inter-array scattering values, the object scattering values are ready to be fit. The fitting is accomplished by means of an iterative, mesh-based algorithm. The procedure begins by assigning a range of possible maximum values between 0 and twice the initial guess; and a range or standard deviations between 0 and five times the initial standard deviation guess. Each of these ranges is divided into 201 equally spaced values. The algorithm then computes a fit of the form

$$C_{SO,fit}(\theta) = M \exp\left(-\frac{\theta^2}{2U^2}\right),$$ (5.10)

where C_{SO} is the object scattering, M is the maximum value of the Gaussian fit function, and U is the standard deviation. The χ^2 goodness of fit is computed for each combination of M and U values. The M and U values which result in the smallest χ^2 are selected. The range of possible maximums and standard deviations is then divided by 5 and the procedure is repeated. The procedure is iterated 10 times to determine the best fitting parameters very accurately. In order to prevent physically unrealistic scenarios, negative values of the maximum or standard deviation are rejected automatically.

At the conclusion of the fitting algorithm, the resulting maximum is normalized to a per source neutron basis by dividing it by the number of direct correlations in the void measurement. The normalized parameters are then output to a file along with the χ^2 value

of the fit. The fit values of the object scattering and inter-array scattering are also

normalized and output to a file as is the total (inter-array + object) scattering and the

actual object scattering data. The data written to the output files is appended each time

the *GaussFit* program is run, so that after running the code with each of the simulations

shown in Section 5.2, all of the PScF parameters are in a single file. This data will be

used in Chapter 6 to develop fits for the PScF parameters which will become the point

scatter function generating equations (PScFGEs). The fit parameters for each simulation

are listed in Appendix G.

5.3.3 The Point Scatter Function Fits

In this section the goodness of the Gaussian PScF fits will be examined and

compared with the actual scattering values. The distribution of the χ^2 values for the PScF

fits is shown in Figure 5-7. This is an extremely skewed distribution with the majority of

the values lying close to zero and a few vary large values. The distribution has a mean χ^2

of 3771 and a median value of 564.

A plot the χ^2 values organized by material, then by the object-to-detector distance,

and then by the attenuation is presented in Figure 5-8. Organized in this manner, a clear

cyclical pattern can be seen. The worst fits (largest χ^2 values) occur at the lowest

attenuation value at each object-to-detector distance. The fits are also worst at the

smallest object-to-detector distances. This is not unexpected because the χ^2 tests weights

each point by the inverse of the variance at each point, which is simply the number of

scattered correlations at that location.

95

Distributions

Chi Squared

	Quantiles		
100.0%	maximum		54547
99.5%			54399
97.5%			36661
90.0%			10077
75.0%	quartile		2361
50.0%	median		564
25.0%	quartile		90
10.0%			53
2.5%			38
0.5%			27
0.0%	minimum		26

Moments	
Mean	3771.5238
Std Dev	8735.7166
Std Err Mean	504.35683
upper 95% Mean	4764.0625
lower 95% Mean	2778.985
N	300
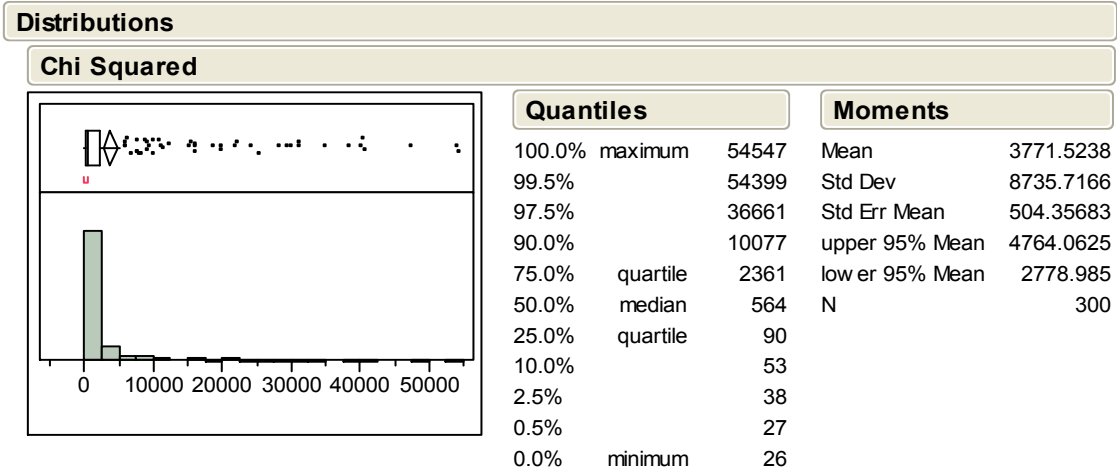
Figure 5-7. The distribution of the χ^2 goodness of fit tests resulting from the Gaussian fits of the PScF functions.
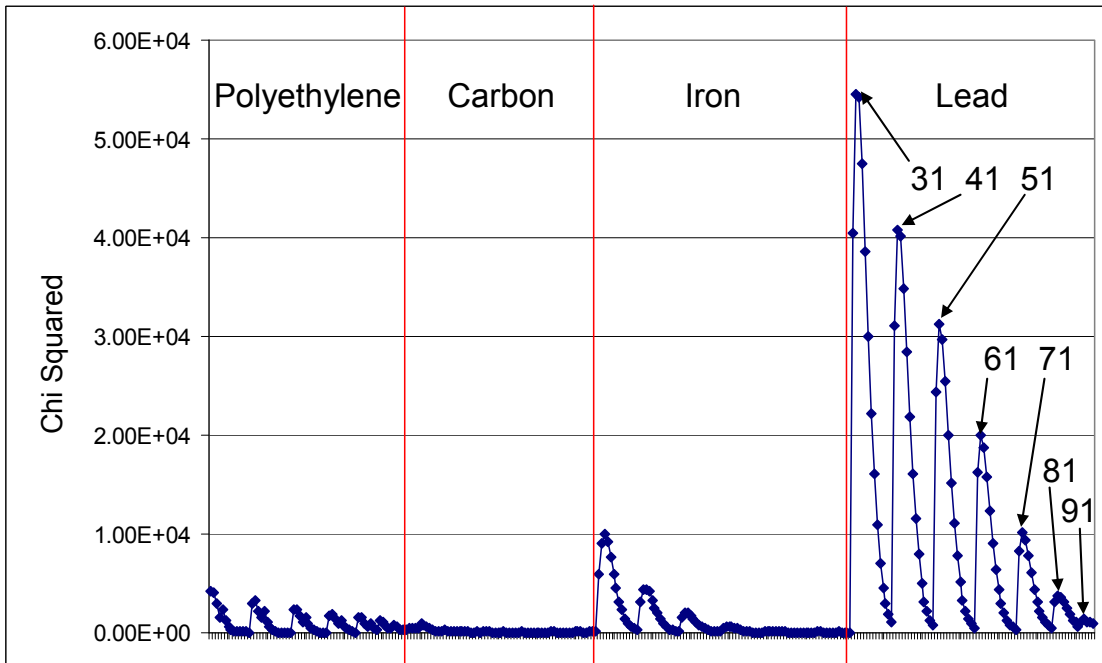
Figure 5-8. A line graph of the results of the χ^2 goodness of test results for the PScF Gaussian functions. The values are sorted by material, then by the object-to-detector distance, and finally by the object thickness.

Scenarios with lower attenuation values and those closer to the detector array tend to have more scattered counts and thus a lower variance. This causes even very small deviations of the fit from the data to contribute a very large value to the goodness of fit test.

The data in Figure 5-8 also shows a large isotopic variation of magnitude of the goodness of fit results. Lead produced the largest χ^2 values by far. All of the fits with χ^2 values above 10,000 were from scenarios with a lead object. Iron showed the next worst results, followed by polyethylene. All carbon scenarios produced a very good fit. The remainder of this section will focus on examining individual PScF fits.

The fit of the Carb61 scenario is presented in Figure 5-9. The original scattering data is shown as well as the PScF, ISF, and total (PScF+ISF) fit for the scenario. This scenario represents one of the best fits with a χ^2 value of only 70. The total fit follows the data extremely well and almost no divergence between the fitted scattering and the data is visible. This behavior can most likely be attributed to the fact that the carbon is composed of a single element whose scattering cross section is not heavily forward peaked and shows no significant diffraction pattern. Thus, the scattering in the object follows a Gaussian distribution.

The fit for one of the worst fitting scenarios, Lead32, is shown in Figure 5-10. The χ^2 value of the fit is 47,600. The locations of the largest relative errors are at the center of the scattering function and the tails. The source of this discrepancy is the diffraction pattern visible in the tails of the scattering data.
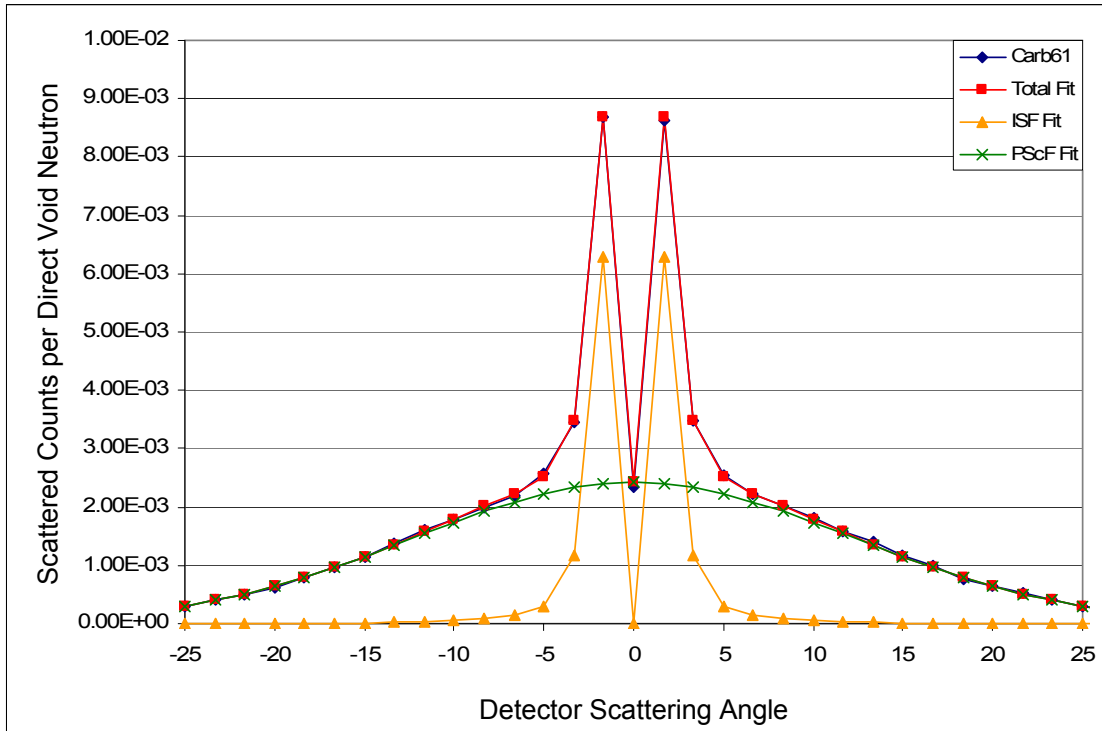
Figure 5-9. A comparison of the fit scattering functions to the scattering data for the Carb61 scenario.
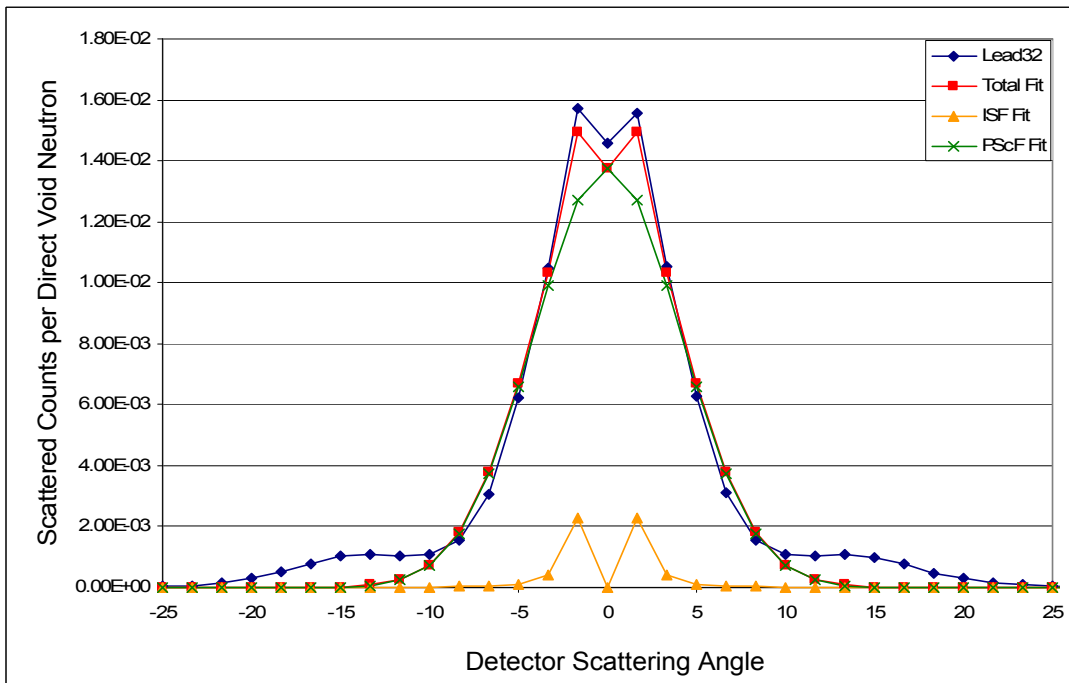
Figure 5-10. A comparison of the fit scattering functions to the scattering data for the Lead32 scenario.

98

These tails cause the fit function to be wider and, as a result, to underestimate the scattering in the center of the data. The diffraction patterns are non-Gaussian, so even the wider Gaussian function falls below them.

Another scenario with a lead object, Lead45, is shown in Figure 5-11. This scenario has a much lower χ^2 value, 5050, than the previous one; however, this can be attributed to the smaller number of scattered counts. The relative errors of the fit scattering function are larger. Although the diffraction pattern peaks are not visible, the non-Gaussian tails can be seen clearly. These tails pull the Gaussian fit wider, resulting in an underestimate of the scattering in the center of the function.

The fit for a polyethylene scenario, Poly63, is shown in Figure 5-12. The fit of this scenario has a χ^2 value of 692. The fit generally follows the data very well. There is some evidence of non-Gaussian behavior visible in two humps in the data near $\pm 15°$. The source of these humps is likely the presence of both hydrogen and carbon in the polyethylene. As shown in Section 3.2, these elements have different scattering functions and the resulting superposition is likely responsible for the humps. The magnitude of the discrepancy is very small compared to the data, so it should have very little impact when applying the PSRA.

The fit of the Iron31 scenario is illustrated in Figure 5-13. The χ^2 value of the fit is 9020, which is one of the largest values for any iron scenario. Despite this large χ^2 value, the fit follows the data very well. The iron cross section is somewhat forward peaked, which slightly widens and lowers the center of the fit. The effect is very small compared to that caused by lead, so the fit remains very good.
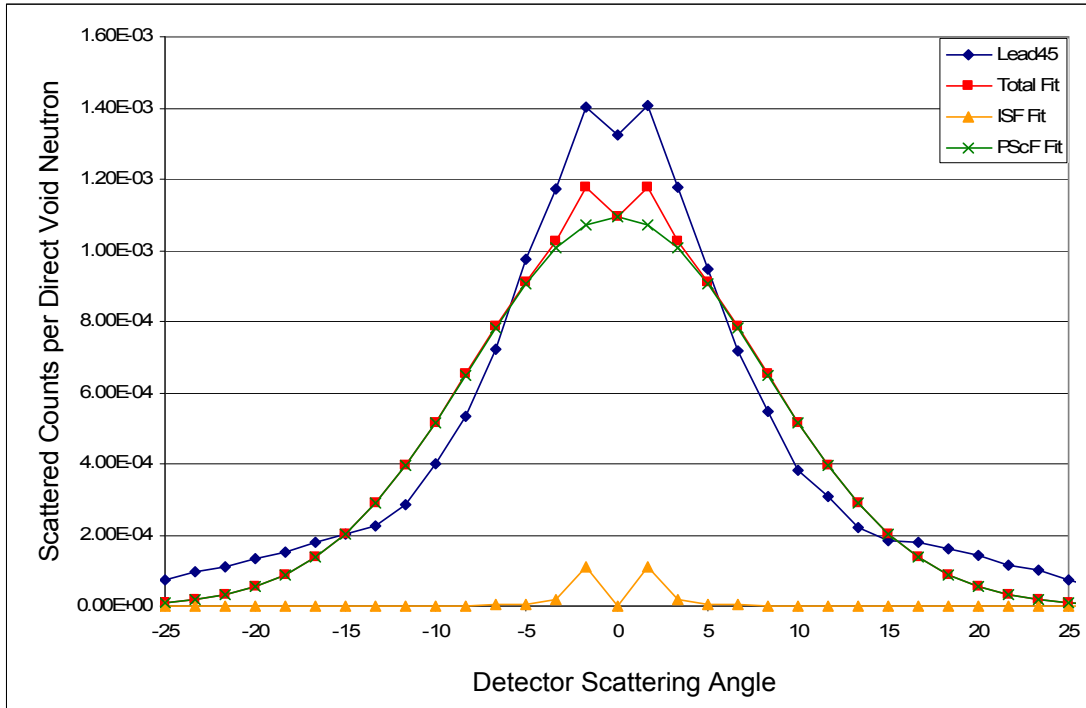
Figure 5-11. A comparison of the fit scattering functions to the scattering data for the Lead45 scenario.
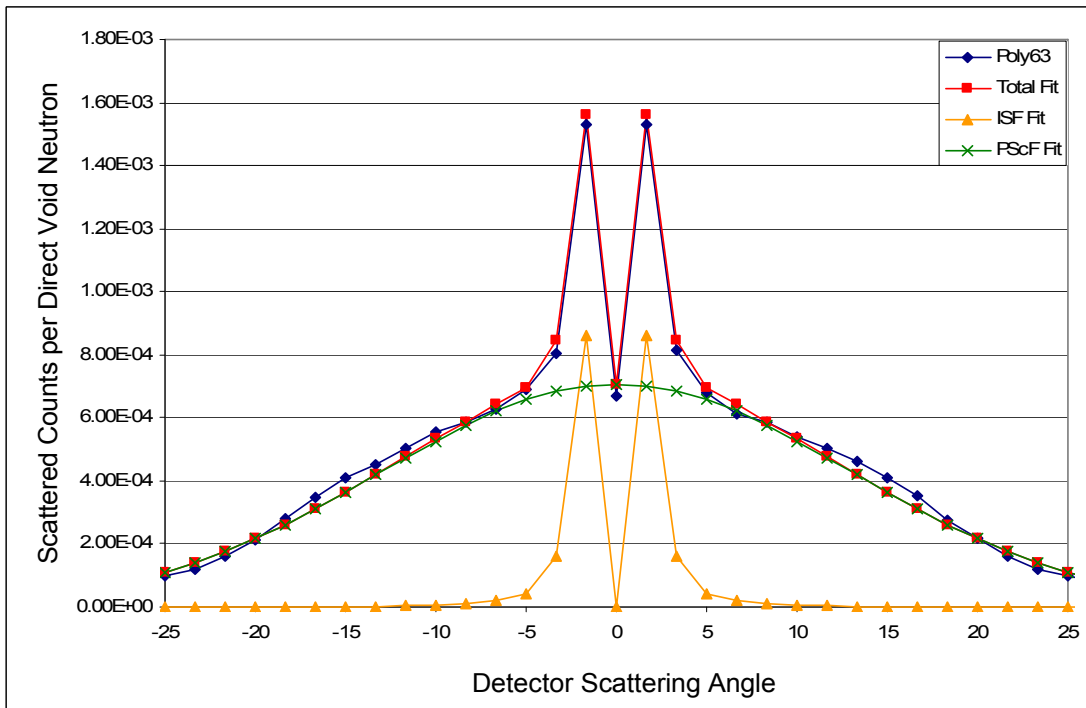
Figure 5-12. A comparison of the fit scattering functions to the scattering data for the Poly63 scenario.

100
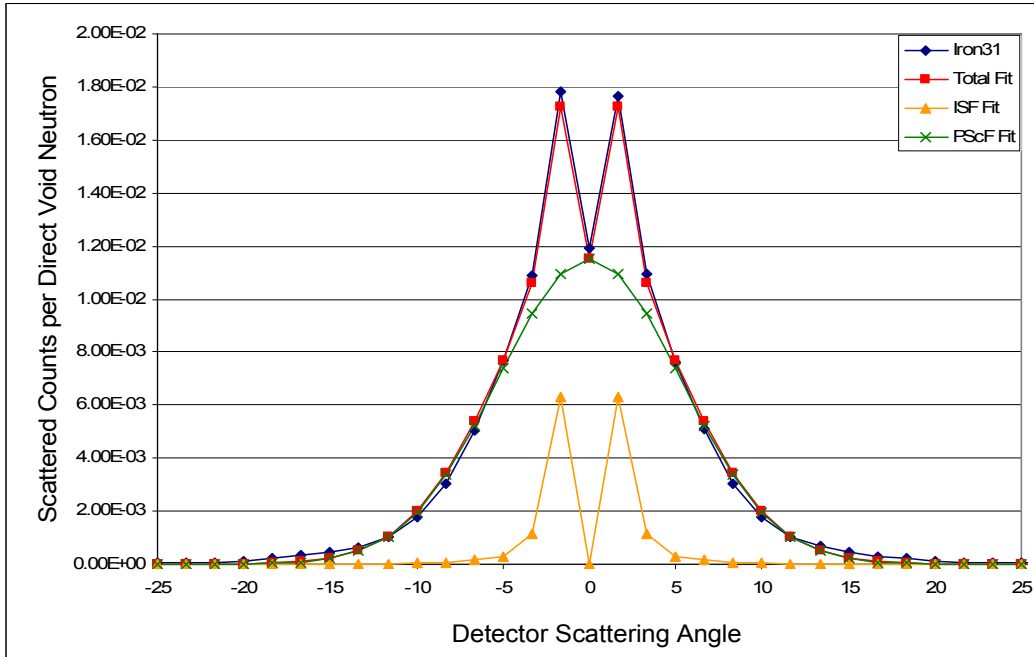
Figure 5-13. A comparison of the fit scattering functions to the scattering data for the Iron31 scenario.

These scenarios show a representative sample of the fits across the range of materials, object thicknesses, and object-to-detector distances modeled. The carbon scenarios in particular are extremely well represented by a Gaussian fit. The polyethylene and iron scenarios show a noticeable divergence from a Gaussian, but are still well characterized by the fit. The lead data shows a fairly strong degree of non-Gaussian behavior which results in a fairly large underestimation of the scattering in some scenarios.

The behavior of the lead scattering would be better represented by a combination of functions (as with the ISF) or even Legendre polynomials. However, the use of such functions might result in an overfitting of the data. If the shape of the scattering function is altered markedly by the presence of geometric features in the object, the resulting fits might produce larger errors than the Gaussian one. The shape would also be isotope

dependent, which could again create a larger error than the Gaussian if the lead data values were used for another heavy material, such as tungsten or depleted uranium. In addition, one parameter that will be considered while developing the PScFGEs is the level of knowledge of the operator. In cases where the operator is unsure of the exact shielding material, an approximation of the PScF using a combination of either the lead and iron or lead, iron, and carbon results will be used.

This type of approximation is possible only if all of these materials use the same parameterization. Therefore, this work will employ the Gaussian fits for lead, because even though they are not the ideal solution for that particular material they offer the best solution across the entire range of materials used here.

5.4 Comparison of Point Scatter Functions to Elastic Scattering Calculations

This section will compare the results of the elastic scattering calculations from Section 3.2.3 to the results of the PScF simulations. The elastic scattering calculations were performed assuming a thin slab of material 3 MFP thick located at either 40 or 70 cm from the detector array. Those combinations of values were chosen because they are also used in the PScF calculations for the ease of comparison. The materials used in the PScF simulations were also identical to the ones from the elastic scattering calculations with the exception hydrogen. Polyethylene (CH_2) was used instead of pure hydrogen which is a gas at room temperature.

While the PScF simulations explicitly modeled the random walk of particles through the shielding material and into the detector cells using the Monte Carlo method, the elastic scattering calculations focused on the differences between scattering isotopes

by removing the physical geometry of the slab. Thus, scatterings in the elastic model were all treated as if they occurred at the back edge (closest to the detector array) of the object while in the PScF simulations they were distributed throughout the material. The elastic scattering calculations also assumed that all scattered neutrons approached the detectors perpendicular to their front face regardless of the scattering angle. The elastic scattering calculations considered only one channel – single elastic scattering – contributing to the additional counts in the array. In reality, many other possible channels exist including multiple elastic scattering, inelastic scattering, (n, 2n) reactions, and possibly even (n, γ) capture. The relative size of these other contributions to the "scattered" signal, which the PScF simulations modeled explicitly, are nuclide specific. Finally, the elastic scattering calculations ignored the uncertainty in the arrival time of the neutrons to the detectors. Accounting for this uncertainty requires the use of a wider correlation time bin to define directly transmitted neutrons, which allows a larger fraction of the scattered counts to be misidentified as directly transmitted. Thus, while the elastic scattering calculations should be fairly similar to the MCNP PScFs, these factors will cause some divergence between the results.

The elastic scattering calculation results for hydrogen and carbon are plotted with the simulation results for polyethylene at a 40 cm object-to-detector distance in Figure 5-14. The object scattering data was produced by subtracting the ISF values from the measured data. The PScF fit is shown as well. Because polyethylene is composed of two parts hydrogen and one part carbon, its elastic scattering value would be expected to lie somewhere between the values of those two pure isotopes.
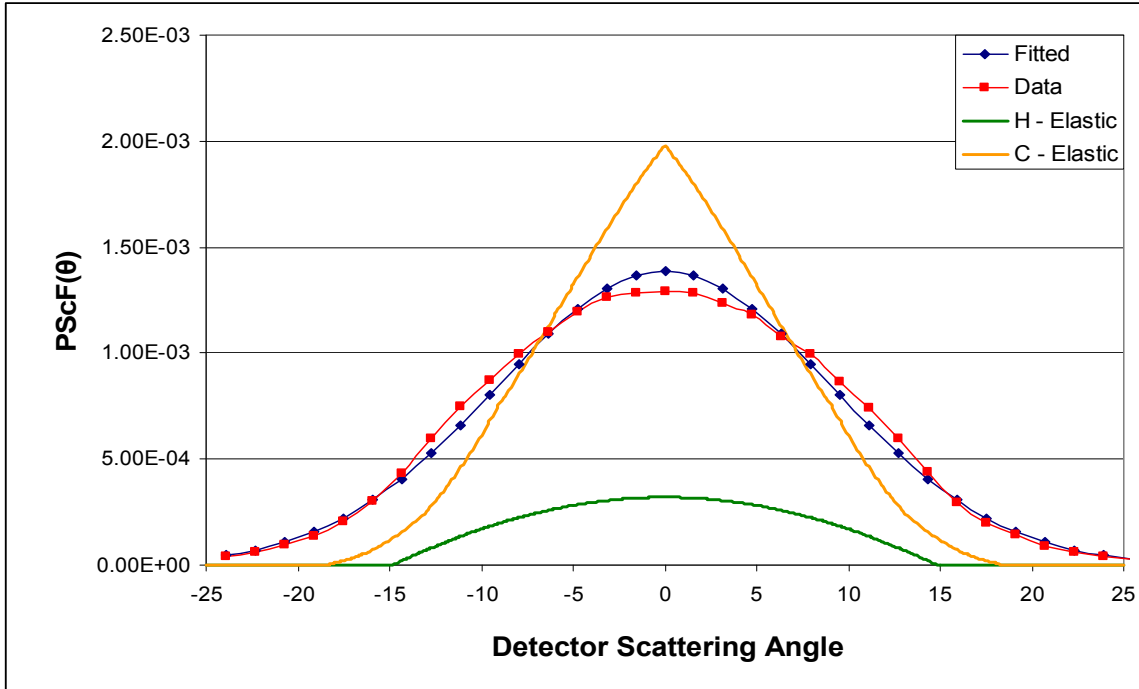
Figure 5-14. A comparison of the PScF simulation results for polyethylene to the elastic scattering calculations for hydrogen and carbon. The object-to-detector distance is 40 cm. The fit of the object scattering data is also shown for reference. Note that the sharp peak in the center of the carbon elastic curve is due to interpolation of the differential scattering cross section file.

The simulation curves are wider and less peaked because of the finite shield thickness, multiple scatterings, a larger correlation window, and other factors discussed in the previous paragraph.

The comparison of the carbon, iron, and lead simulations to their respective elastic scattering calculations are presented in Figure 5-15 through Figure 5-17. The maximum scattering value of the carbon simulation matches the elastic scattering result almost perfectly. With iron and particularly with lead, the elastic calculations resulted in a larger peak than the simulation values. In the MCNP data, multiple scatterings in the shield material tended to spread out the peaks and reduce the forward scattering effect to some degree.

104
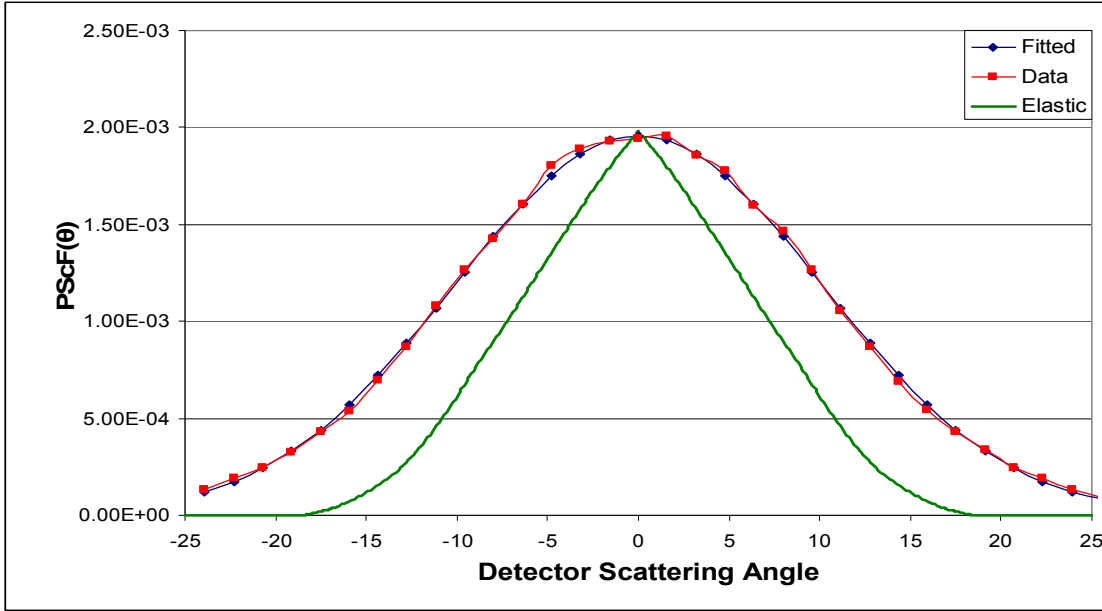
Figure 5-15. A comparison of the PScF simulation results for carbon to the elastic scattering calculations. The object-to-detector distance is 40 cm. The fit of the object scattering data is also shown for reference. Note that the sharp peak in the center of the elastic curve is due to interpolation of the differential scattering cross section file.
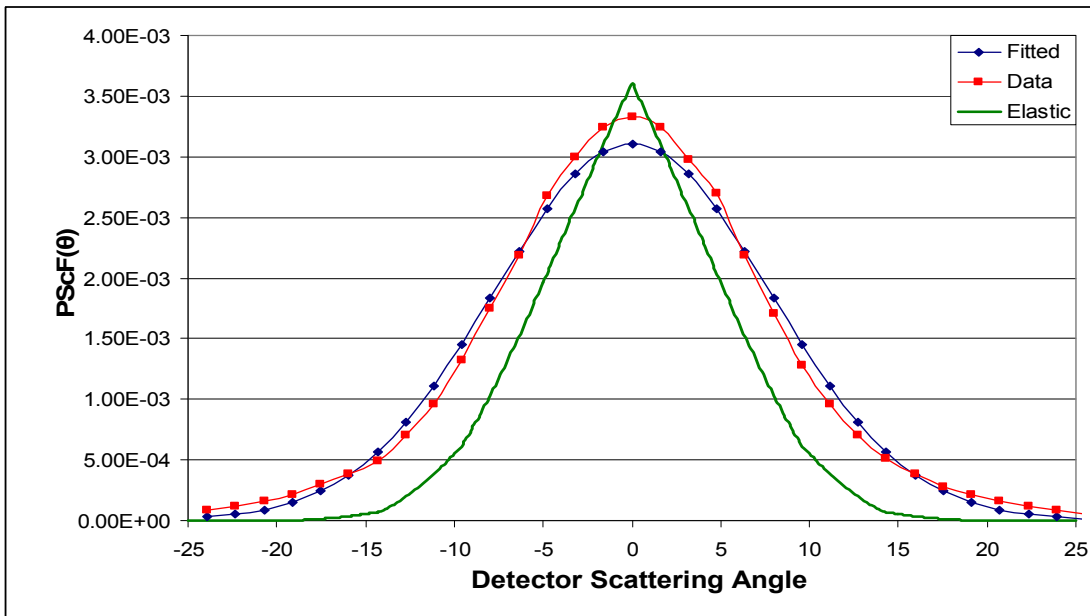
Figure 5-16. A comparison of the PScF simulation results for iron to the elastic scattering calculations. The object-to-detector distance is 40 cm. The fit of the object scattering data is also shown for reference. Note that the sharp peak in the center of the elastic curve is due to interpolation of the differential scattering cross section file.
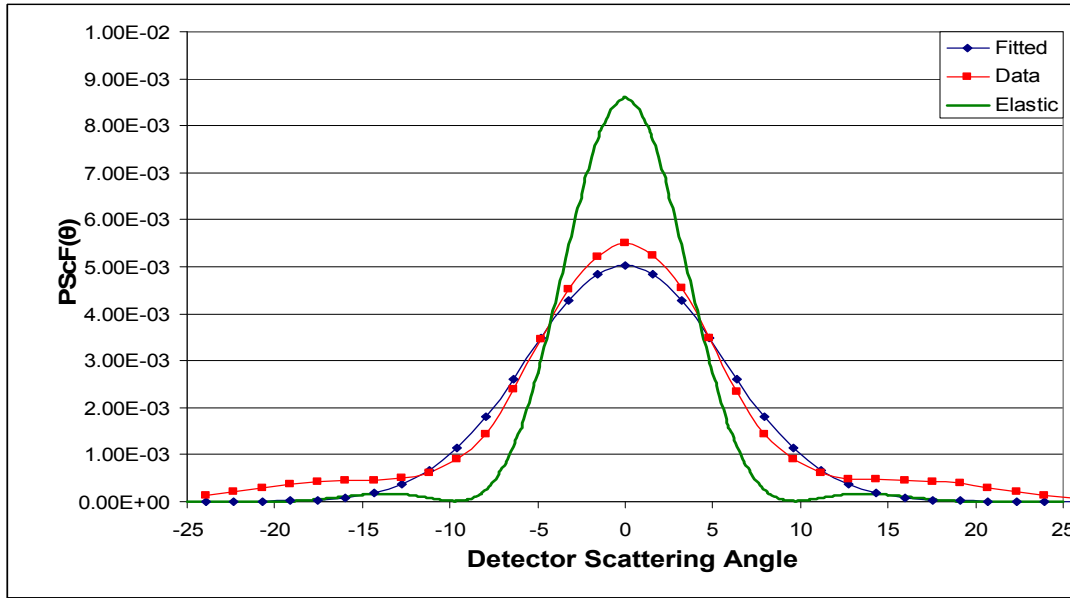
Figure 5-17. A comparison of the PScF simulation results for lead to the elastic scattering calculations. The object-to-detector distance is 40 cm. The fit of the object scattering data is also shown for reference.

The diffraction pattern maxima visible in the tail region of the elastic scattering calculation are visible in the simulation data, but they are more widely spread due to the superposition of scatterings from all parts of the object.

At a 70 cm object-to-detector distance, the width of the elastic scattering calculations more closely matches that of the simulation results because the widening due to the object thickness is not as pronounced at this larger distance. As a result, the maximum values of the elastic calculations are also lower relative to the corresponding simulation results. The comparisons of the elastic scattering calculations and the simulation results are shown in Figure 5-18 through Figure 5-21.

The comparisons in this section show relatively good agreement between the simulation results and the elastic scattering calculations. This provides validation that the elastic scattering calculations were performed correctly.
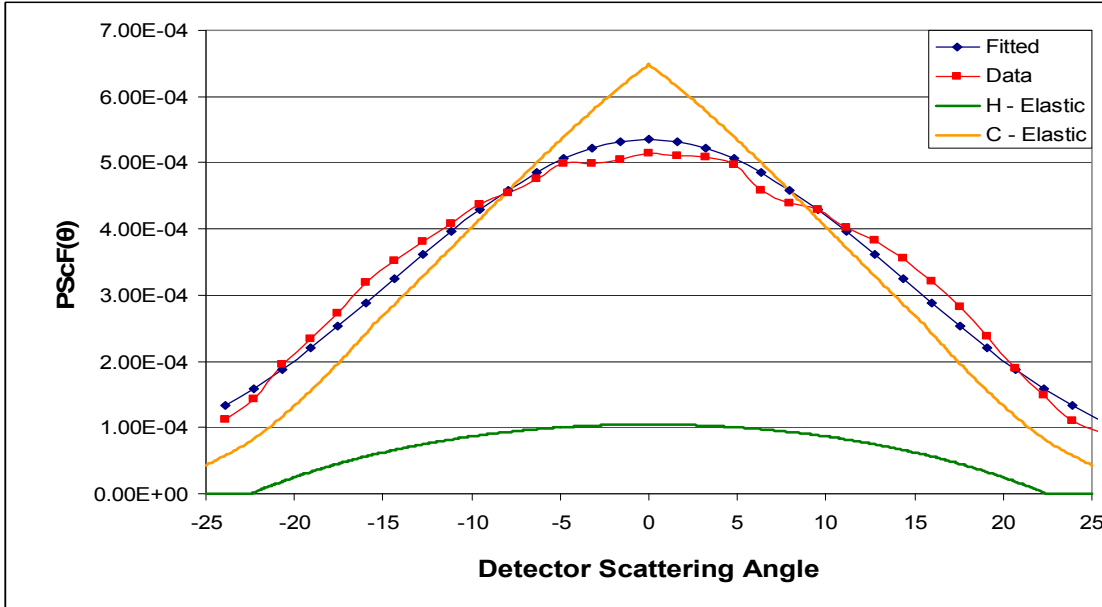
Figure 5-18. A comparison of the PScF simulation results for polyethylene to the elastic scattering calculations for hydrogen and carbon. The object-to-detector distance is 70 cm. The fit of the object scattering data is also shown for reference. Note that the sharp peak in the center of the carbon elastic curve is due to interpolation of the differential scattering cross section file.
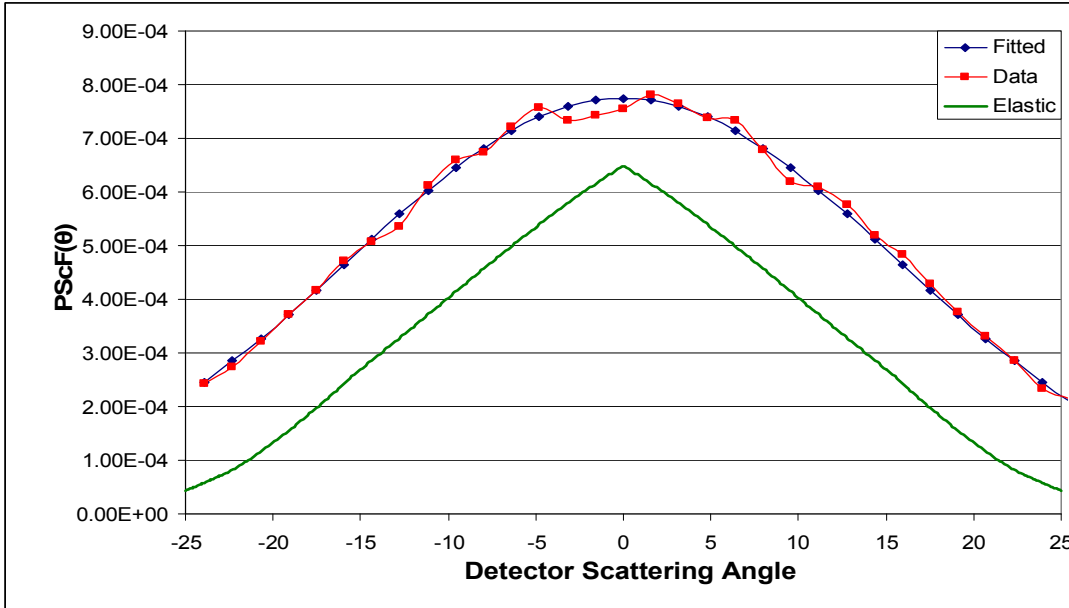
Figure 5-19. A comparison of the PScF simulation results for carbon to the elastic scattering calculations. The object-to-detector distance is 70 cm. The fit of the object scattering data is also shown for reference. Note that the sharp peak in the center of the elastic curve is due to interpolation of the differential scattering cross section file.

107
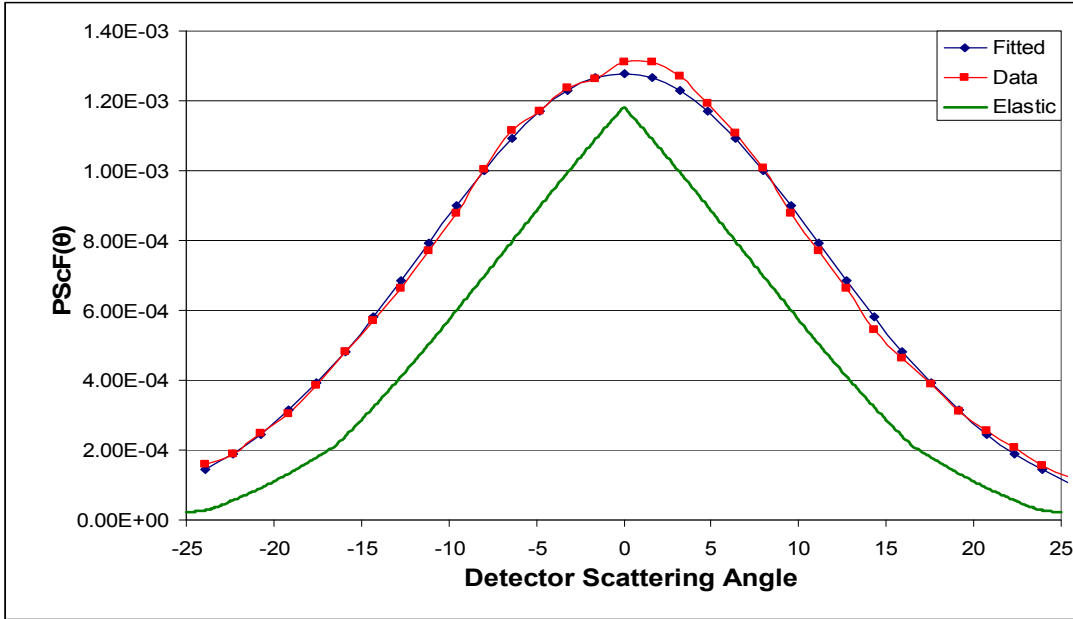
Figure 5-20. A comparison of the PScF simulation results for iron to the elastic scattering calculations. The object-to-detector distance is 70 cm. The fit of the object scattering data is also shown for reference. Note that the sharp peak in the center of the elastic curve is due to interpolation of the differential scattering cross section file.
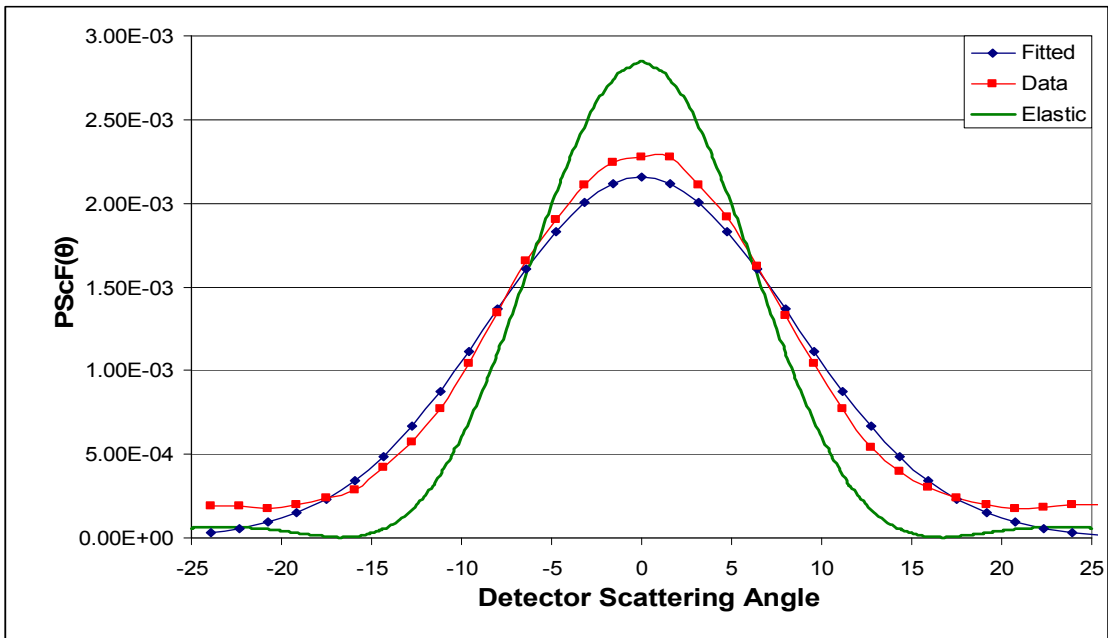
Figure 5-21. A comparison of the PScF simulation results for lead to the elastic scattering calculations. The object-to-detector distance is 70 cm. The fit of the object scattering data is also shown for reference.

They also show the same general trends as the simulation data indicating that the underlying physics of the scattering process is well understood. Despite this, the divergence of the elastic scattering results from the simulation is still large enough that their use as PScFs would produce unsatisfactory results. The sources of this divergence were discussed at the beginning of this section. While some improvements could be made in the elastic scattering calculations to produce more accurate results, they would largely require knowledge of the internal geometry of the scattering object, which is generally unknown. Thus, the choice to measure the PScFs with simulations rather than calculate them analytically is justified.

6. Final Form of the Parameterized Scatter Removal Algorithm

With the PScFs extracted and fit with Gaussian functions, the next goal is to develop an algorithm which will automatically remove the scattering from a measurement using operator inputs and the measured attenuation values. Because the library of materials, thicknesses, and distances from the detector array are necessarily limited, most real measurements will not correspond exactly to one of the cases modeled in the previous chapter. This situation requires either choosing the PScF values which are most similar to the measurement from a library or fitting the parameters using multivariate methods. The latter option is likely to produce better results if good fits can be developed. In this chapter, these fits will be developed using the JMP 7 statistical software package [52] and then incorporated into the *ScatterSubtract* code which will subtract the scattering from measured attenuation values. The methodology used to remove the scatter from the measurements will be discussed in detail.

6.1 Fitting the Point Scatter Function Parameters

6.1.1 Univariate PScF Parameter Fits

As discussed in the previous chapter, the information available to the NMIS operator during a measurement typically consists of the distance between the object and the array, the measured attenuation, and to a certain extent, the material composition of the object. The cases selected for developing the library of PScF fit parameters modeled a large sample of possible values. Thus, the fits for the Gaussian parameters will be based on these three input variables.

Before attempting to generate a multivariate fit of the Gaussian PScF parameters using these variables, an analysis of the PScF parameter response to each individual variable is warranted. This analysis will help determine the maximum order of the input variables to be used in the PScFGEs. The individual fits will also help determine if any linearizing transforms of either the input variables or the parameter response are required.

A simple line graph of the Maximum of the PScF function (M in Equation 5.10) for each of the scenarios modeled in Section 5.2 is shown in Figure 6-1. The regions corresponding to each of the materials are indicated on the graph. Moving from left to right within each material, the first scenarios have the smallest object-to-detector separation and within each separation value, the first objects have the smallest attenuation. Thus, the scenarios are ordered Poly 30.5, Poly31, Poly31.5, …, Poly40.5, Poly41, etc. The data in Figure 6-1 shows the clear cyclical nature of the Maximum. Scenarios with the object closer to the detectors produce a larger maximum and within each object-to-detector distance for a given material, the maximum peaks at an attenuation value of 1 MFP and then falls off. Heavier isotopes also have larger maximum values than lighter ones.

A line graph of the Gaussian Standard Deviation for each scenario is shown in Figure 6-2. The cyclical pattern is evident on this plot as well. In general, higher attenuations and larger object-to-detector distances produce larger fit Standard Deviations. Polyethylene and carbon produce approximately the same standard deviations. Iron and lead have smaller standard deviations due to their increasingly forward peaked scattering cross sections.
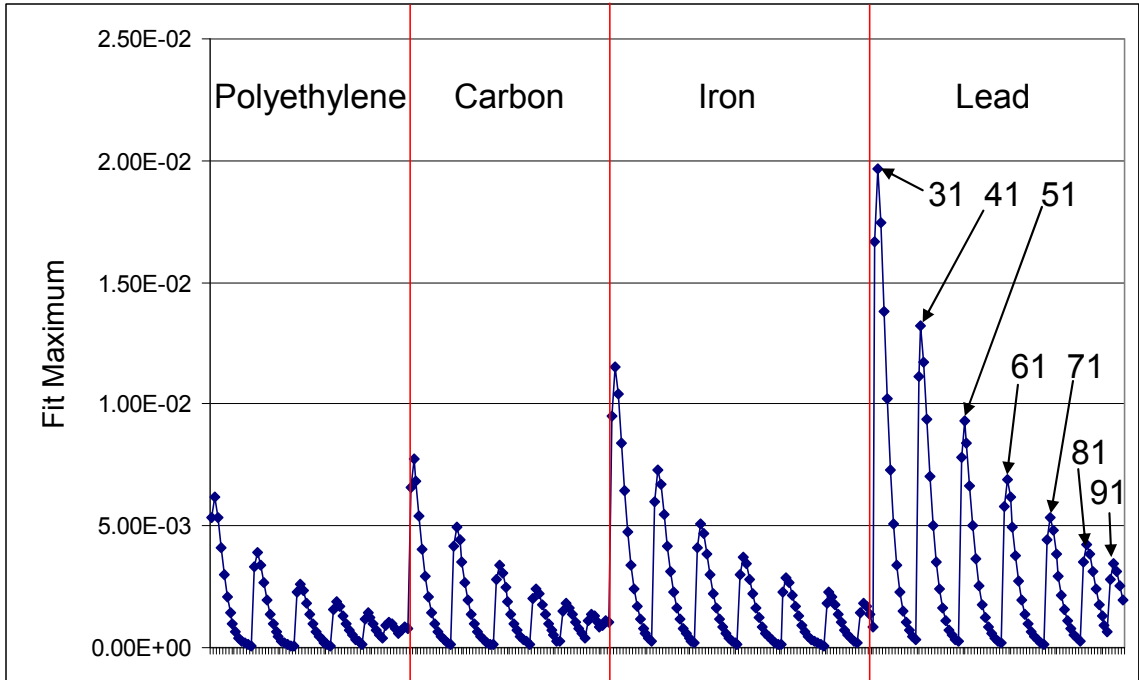
Figure 6-1. A line graph of the Maximum of the PScF Gaussian fit for each of the modeled scenarios. The scenarios are ordered first by material, then by object-to-detector distance (D) from lowest to highest, and then by material thickness (τ).
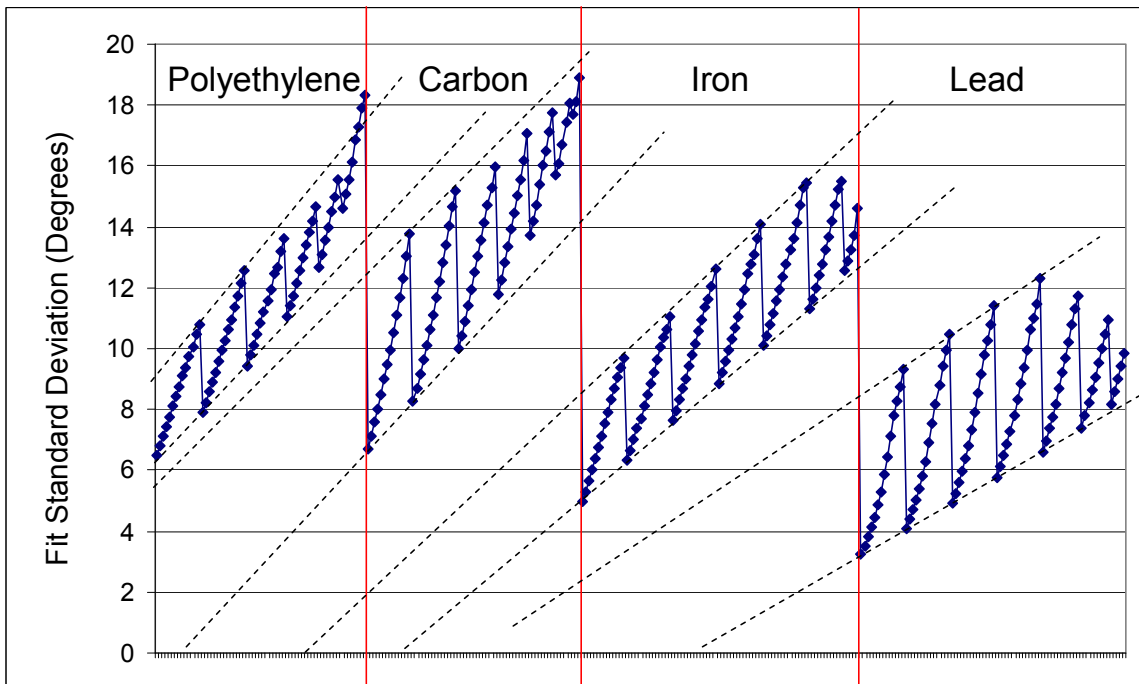
Figure 6-2. A line graph of the Standard Deviation (in degrees) of the PScF Gaussian fit for each of the modeled scenarios. The scenarios are ordered first by material, then by object-to-detector distance (D) from lowest to highest, and then by material thickness (τ).

The dashed lines surrounding the values for each material are an attempt to line up similar object thicknesses. The divergence of the line from the data points indicates some degree of non-linearity in the standard deviation values with respect to the object-to-detector distances. Within each object-to-detector distance, some degree of non-linearity is also visible, particularly in lead. This will be examined in more detail later in this section.

Oneway plots of the Maximum and Standard Deviation parameters for each material are presented in Figure 6-3. The primary information that can be gleaned from this figure is that each material has a different distribution of Gaussian fitting parameters, confirming the need to account for each material separately in the multivariate fits.

The Maximum versus object-to-detector distance (D) for 1.5 MFP of carbon is plotted in Figure 6-4. The shape of the data suggests exponential decay. Other combinations of material and thickness show a similar shape. Direct polynomial fits of these data points produce poor results and diverge wildly from the exponential decay trend outside of the data region even at orders as high as 6. In order to address this problem, a logarithmic transform is warranted. A plot of ln(Max) versus D is presented in Figure 6-5. Even after the transform, there is some degree of nonlinearity. A third order polynomial fits the data almost perfectly ($R^2 = 0.999994$) and maintains the same trend outside of the data region. Higher order polynomial terms have p-values of greater than 0.1 when used to fit the data and are not useful for predicting the variation of ln(Max). Thus, D will be allowed to enter the multivariate fit of ln(Max) up to an order of 3.
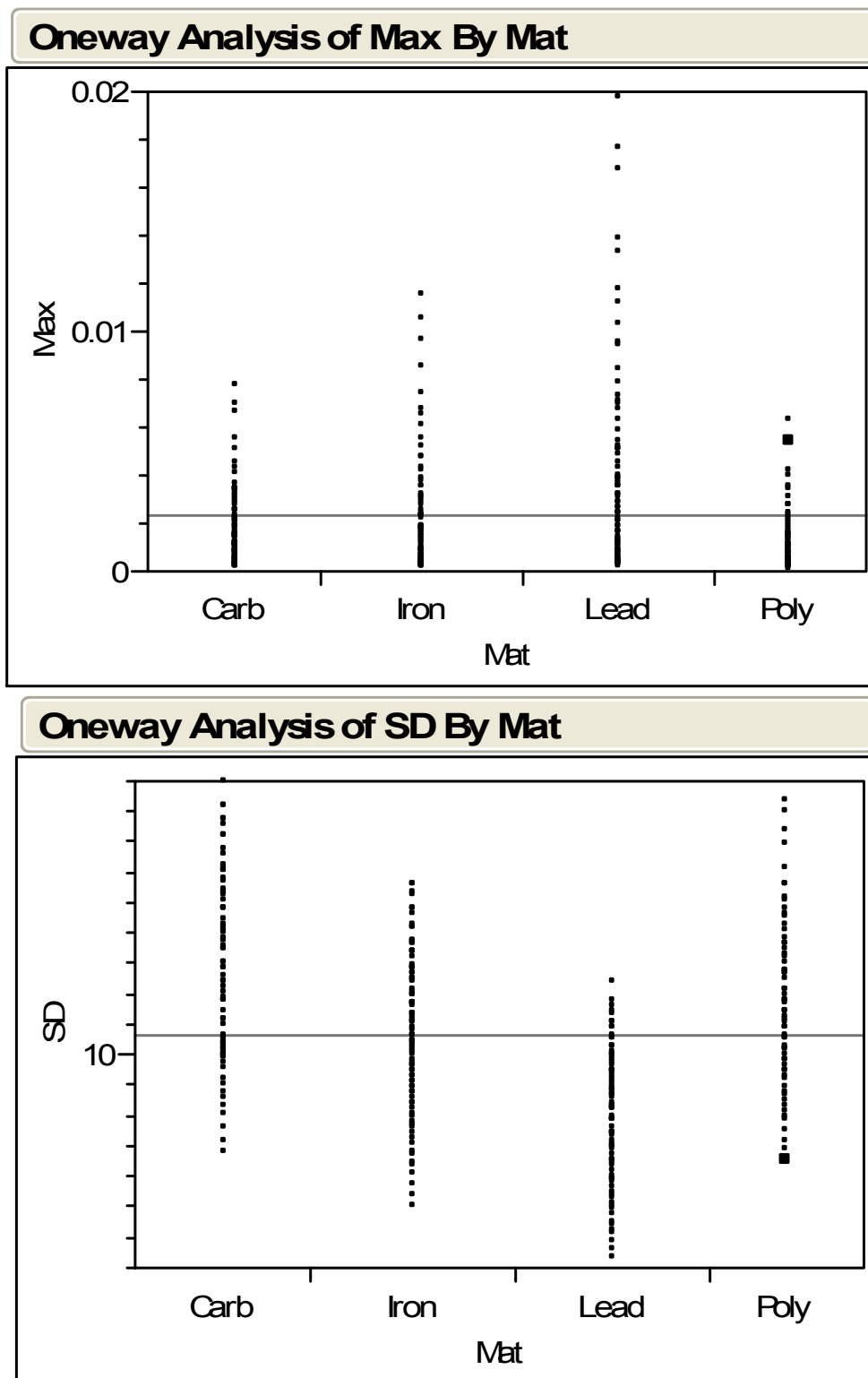
Figure 6-3. Oneway plots of the Maximum (top) and Standard Deviation (bottom) of the PScF Gaussian fits grouped by material.
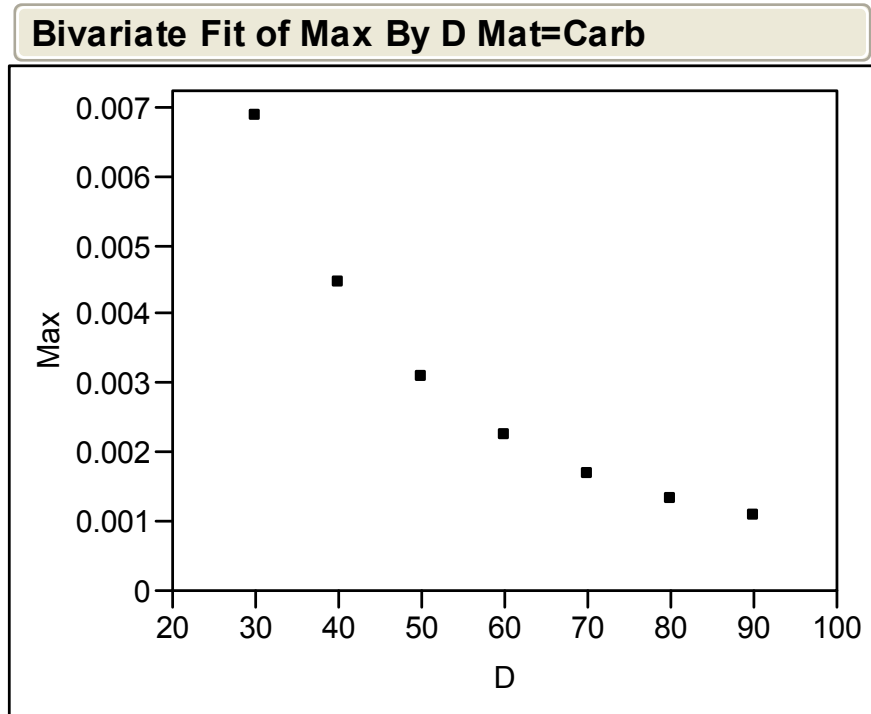
Figure 6-4. A plot of the Maximum of the PScF Gaussian fit by object-to-detector distance (D) in cm for 1.5 MFP of carbon.
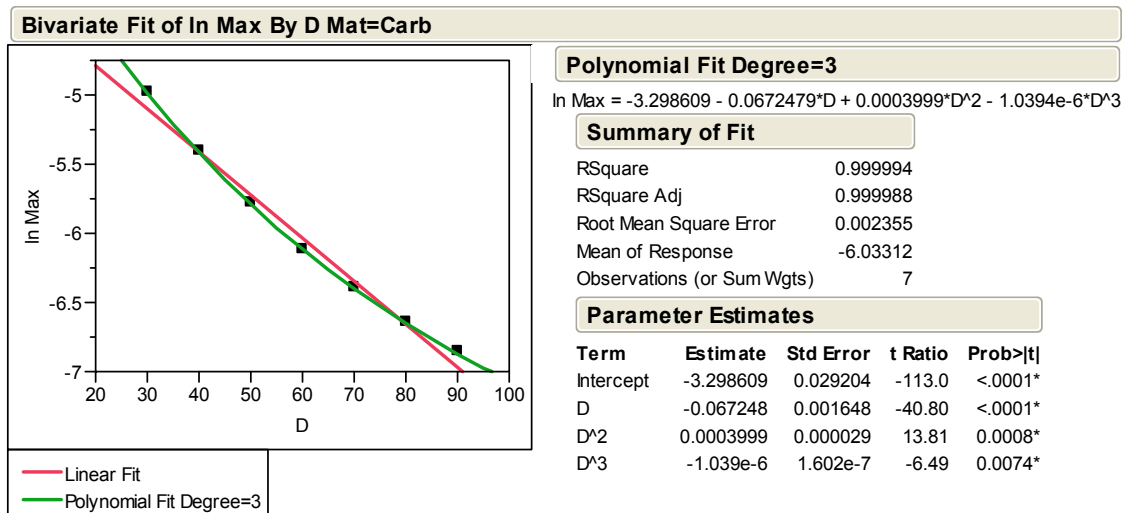
Figure 6-5. A graph of ln(Max) of the PScF fit versus object-to-detector distance (D) in cm for 1.5 MFP of carbon. A linear and 3rd order polynomial fit of the data is shown. Asterisks in the "Prob>|t|" column indicate statistical significance at the p=0.05 level. Other asterisks indicate multiplication. The "^" symbol indicates exponentiation.

115

The Maximum versus thickness for carbon at D = 50 cm is plotted in Figure 6-6. Other combinations of material and object-to-detector distance generate a similar shape. In order to show the trend at low material thicknesses, additional PScF simulations were performed in increments of 0.1 between 0.1 and 1.0 MFP for carbon at D=50. The Maximum increases with the material thickness up to 1 MFP and then begins to drop off in an exponential fashion beyond that. This behavior is expected since the average interaction depth in a material is 1 MFP. At smaller thicknesses, fewer neutrons scatter in the object and at larger thicknesses, many of the neutrons that do scatter are absorbed or scattered away from the detectors before exiting.

The shape of the data in Figure 6-6 corresponds with that of the function τEXP(-τ). A fit of the Maximum versus τEXP(-τ) is shown in Figure 6-7. A linear fit shows an excellent correlation with an R^2 value of 0.999925. Thus, the use of τEXP(-τ) rather than τ is called for in the multivariate model. However, since D required a logarithmic transform of the Maximum, the same transform needs to be applied to the attenuation. In order to accomplish this, the variable β is defined, where

$$\beta = \ln[\tau \exp(-\tau)] = \ln(\tau) - \tau. \tag{6.1}$$

Two fits of ln(Max) versus β are shown in Figure 6-8. A linear fit has an R^2 value of 0.999499. There is also a slight nonlinearity, and a second order fit (not shown) produces a slightly higher R^2 of 0.999859. Third and higher order terms have p-values greater than 0.1 and will not be considered in the multivariate fits.

The Standard Deviation of the PScF fit versus thickness for carbon at D=50 is plotted in Figure 6-9. Except in the very low attenuation range, a second order polynomial fit follows the data points extremely well (R^2 = 0.999082).

116
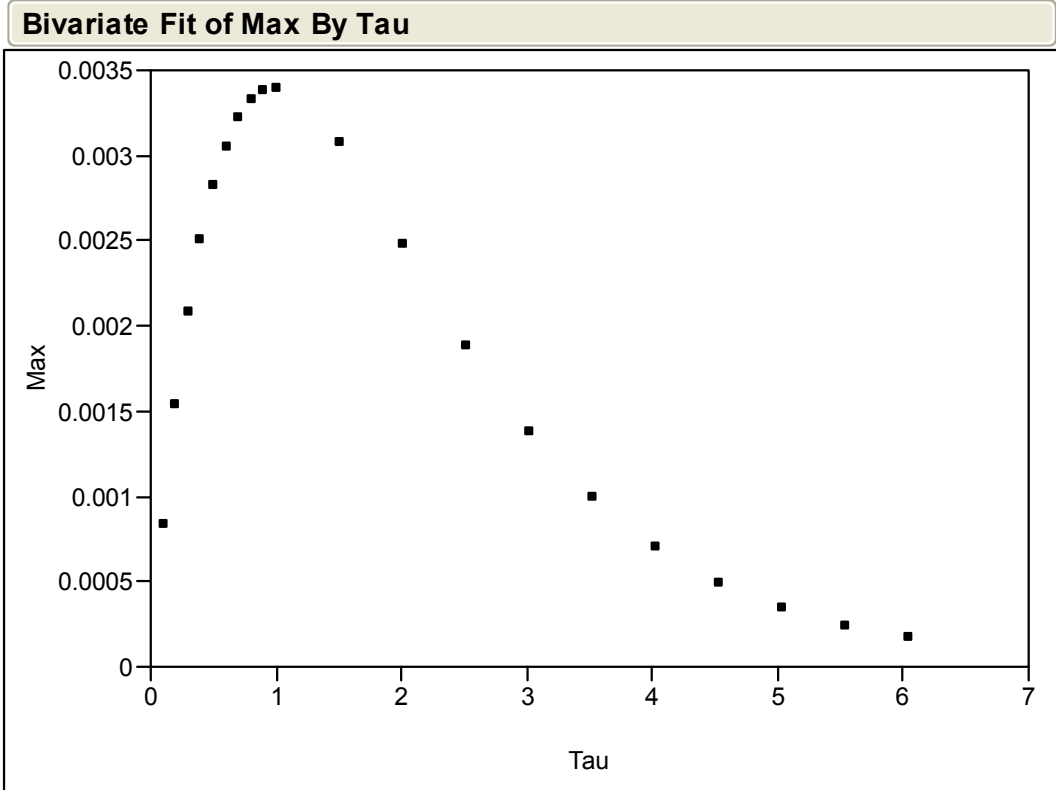
Figure 6-6. A plot of the Maximum of the PScF Gaussian fit versus thickness (τ) for carbon at D = 50 cm.
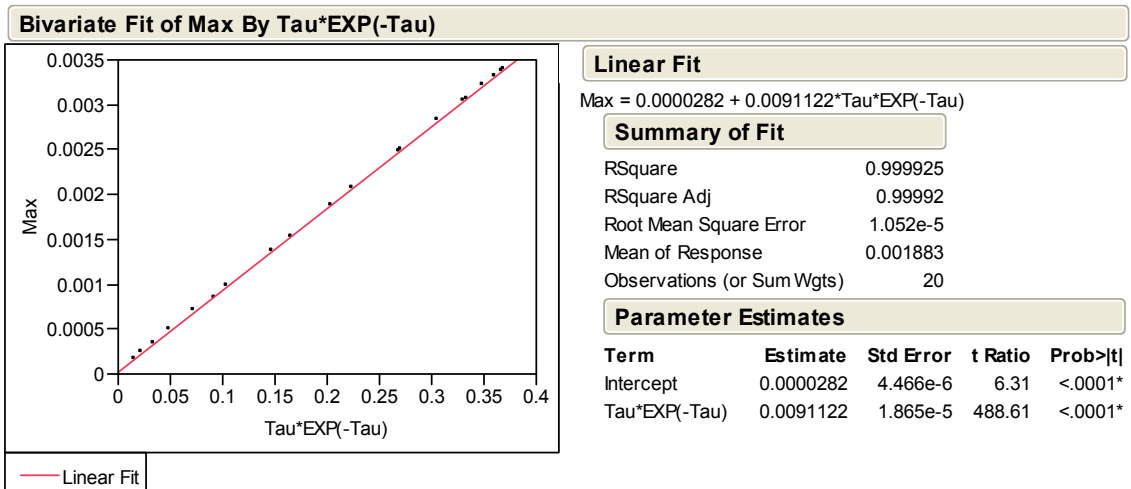
Figure 6-7. A plot of the Maximum of the Gaussian PScF fit versus τEXP(-τ) for carbon at D = 50 cm. A linear fit of the data is shown. Asterisks in the "Prob>|t|" column indicate statistical significance at the p=0.05 level. Other asterisks indicate multiplication. The "^" symbol indicates exponentiation.

Bivariate Fit of ln Max By Beta

ln Max = -4.753816 + 0.9511783*Beta

Linear Fit

Summary of Fit

RSquare	0.999499
RSquare Adj	0.999471
Root Mean Square Error	0.022209
Mean of Response	-6.59646
Observations (or Sum Wgts)	20

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	-4.753816	0.010917	-435.4	<.0001*
Beta	0.9511783	0.005019	189.53	<.0001*
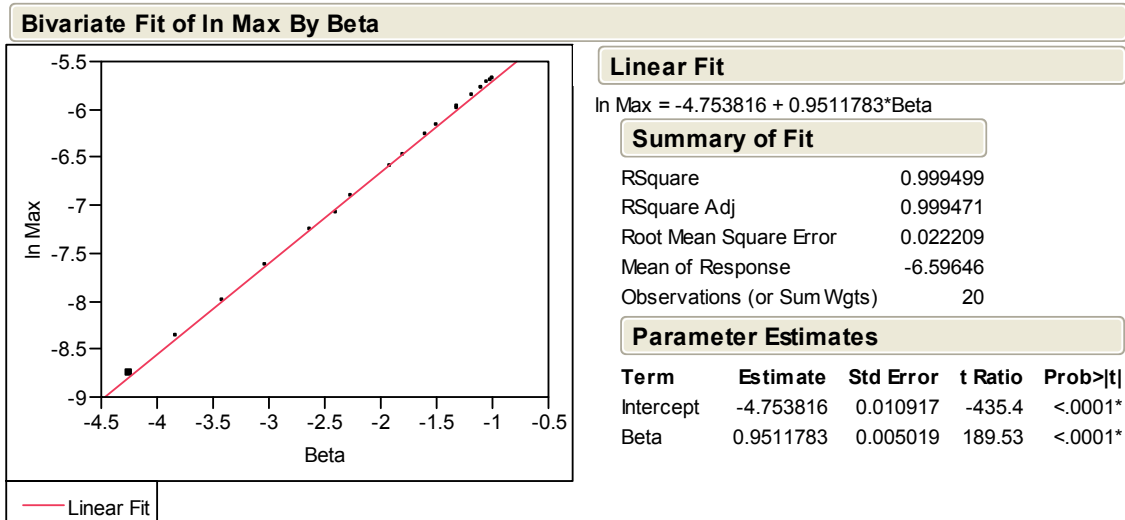
Figure 6-8. A plot of ln(Max) of the PScF Gaussian fit versus β for carbon at D = 50 cm. A linear fit of the data is shown. Equation 6.1 defines the variable β. Asterisks in the "Prob>|t|" column indicate statistical significance at the p=0.05 level. Other asterisks indicate multiplication.

Bivariate Fit of Std. Dev. By Tau

Polynomial Fit Degree=2

Std. Dev. = 9.6288109 + 0.8003238*Tau + 0.0412273*Tau^2

Summary of Fit

RSquare	0.999082
RSquare Adj	0.998974
Root Mean Square Error	0.06542
Mean of Response	11.71
Observations (or Sum Wgts)	20

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	9.6288109	0.030084	320.06	<.0001*
Tau	0.8003238	0.030861	25.93	<.0001*
Tau^2	0.0412273	0.005287	7.80	<.0001*
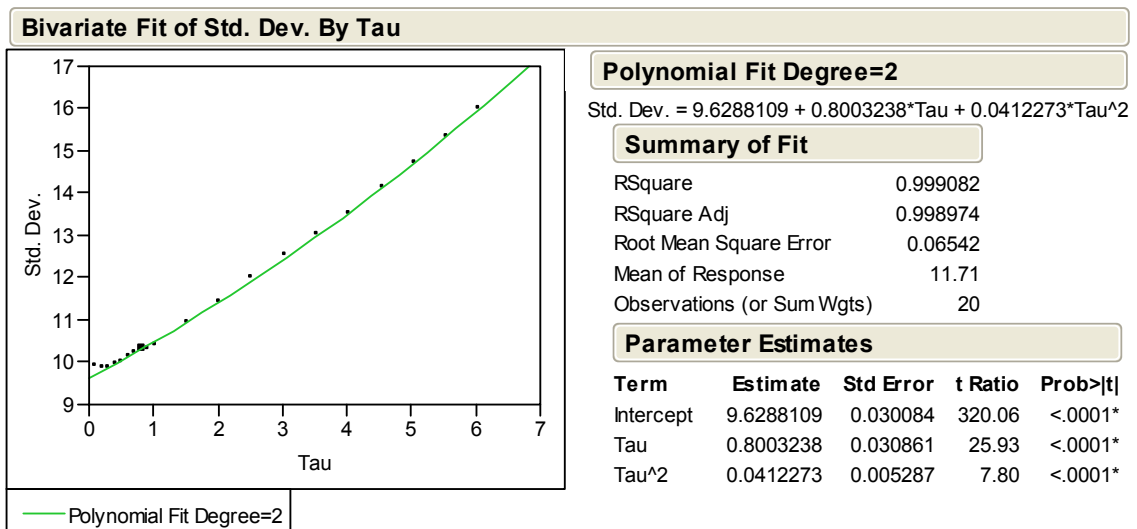
Figure 6-9. A plot of the Standard Deviation of the PScF Gaussian fit versus thickness for carbon at D = 50 cm. A second order polynomial fit is shown. Asterisks in the "Prob>|t|" column indicate statistical significance at the p=0.05 level. Other asterisks indicate multiplication. The "^" symbol indicates exponentiation.

118

Even at the lowest attenuation value ($\tau = 0.1$ MFP), the divergence of the data from the fit is less than 0.25°. Thus, the second order polynomial explains the behavior of the model adequately, and the variable, τ, will be allowed to enter the multivariate model for the Standard Deviation fit up to a polynomial order of two.

A plot of the PScF Standard Deviation versus D for 1.5 MFP of carbon is presented in Figure 6-10. Other combinations of material and thickness produce a similar shape. As with the fit in Figure 6-9, a second order polynomial fit follows the data points extremely well ($R^2 = 0.999951$). Thus, D will be allowed to enter the multivariate model for the standard deviation fit up to a polynomial order of two.

6.1.2 Multivariate PScF Fits

Now that the univariate relationships between the input variables and the PScF parameters are determined, a multivariate model can be constructed. Before this can proceed, one final issue needs to be addressed. As discussed in the previous section, the distribution of Gaussian standard deviations and maximums are different for each material. Because the material is a categorical variable, it is more difficult to use in a model than the continuous variables D and τ.

In order to use the material variable in the multivariate model, a series of Boolean variables (e.g., Iron = 0 or 1) would have to be used separately and crossed with each of the continuous terms. With n materials, this would multiply the number of (potential) terms in the model by n-1, resulting in a much more complicated fit equation. With a large material library, the fit equations could become unmanageably large and statistical software like JMP 7 would have difficulty converging the solution.

Bivariate Fit of Std. Dev. By D

Polynomial Fit Degree=2

Std. Dev. = 3.2892857 + 0.1266786*D + 0.0005179*D^2

Summary of Fit

RSquare	0.999951
RSquare Adj	0.999926
Root Mean Square Error	0.035102
Mean of Response	12.96143
Observations (or Sum Wgts)	7

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	3.2892857	0.12954	25.39	<.0001*
D	0.1266786	0.004644	27.28	<.0001*
D^2	0.0005179	3.83e-5	13.52	0.0002*

Polynomial Fit Degree=2

Figure 6-10. A plot of the Standard Deviation of the PScF Gaussian fit versus object-to-detector distance (D) for 1.5 MFP of carbon. A second order polynomial fit is shown. Asterisks in the "Prob>|t|" column indicate statistical significance at the p=0.05 level. Other asterisks indicate multiplication. The "^" symbol indicates exponentiation.

Instead, in this work each material will be fit separately. This will give each material a unique set of coefficients. The total number of coefficients will be approximately the same using this method, but the resulting fit equations (the PScFGEs) will be much simpler.

Using this technique also allows for the averaging of materials (such as iron and lead) to provide for circumstances where the exact material composition of the object is unknown. In addition to the four materials modeled in Section 5.2 (polyethylene, carbon, iron, and lead), two averaged fits will be developed to reflect a less than perfect level of operator knowledge. In the first case, the PScF parameters of iron and lead will be averaged. This average will be referred to as "material 5." In the second case, the PScF parameters of carbon, iron, and lead will be averaged and designated "material 6."

The allowable input variables for each model were determined based on the univariate models in the previous section. In addition, each of the terms was allowed to

120

cross with the other terms up to one order less than the highest significant polynomial term in the univariate models. For example, in the univariate fits of the PScF maximum, β was significant up to the second order and D was significant up to third order. The allowable cross terms are βD and βD^2. The allowable terms for the fits of the PScF Maximums and Standard Deviations are given in Table 6-1.

The multivariate fitting of PScFGEs for each of the materials was performed using a mixed stepwise model. This method steps through the allowable terms and adds the one with the highest significance (lowest p-value) to the model as long as its p-value is below a preset threshold. This process is continued and one term is added to the model with each iteration until there are no more with p-values below the threshold. The process of adding terms to the model sequentially is known as a forward stepwise method. In addition to this, the mixed stepwise method checks for the significance of the terms entered in the model after each iteration. If the significance of one or more of them rises above a second preset threshold, the one with the highest p-value is removed. For all models considered here, the thresholds for entry and removal from the model were set at $p = 0.10$.

Table 6-1. A list of the terms allowed in the multivariate models for the Maximum and Standard Deviation of the PScF fits.

Gaussian Parameter	LN(Maximum)	Standard Deviation
Allowable Terms	D	D
	D^2	D^2
	D^3	τ
	β	τ^2
	β^2	τD
	βD	
	βD^2	

Once all of the significant terms are entered into the model, the PScF parameter was fit using a standard least squares method.

The results of the ln(Max) fit for carbon are given in Figure 6-11. All of the possible terms in Table 6-1 were significant for this model except for βD^2. Carbon will have a coefficient of 0 for that term in the PScFGEs. The resulting model has an extremely high R^2 value of 0.999599. This R^2 value is inflated because of the logarithmic transform of the Gaussian Maximum. Nevertheless, it indicates that the model can predict the value of the Maximum with a high degree of accuracy. The predicted values of the Maximum versus the actual values for each of the carbon scenarios are plotted in Figure 6-12. A point lying on a line with a slope of 1 and an intercept of 0 indicates that the model has predicted its value exactly. All of the points in Figure 6-12 lie very close to the line, confirming that the model predicts the behavior of the Maximum very well. Fits of the other three materials produced similar results.

The results of a fit of the Gaussian Standard Deviation for carbon are presented in Figure 6-13. All of the allowable terms shown in Table 6-1 are significant in this model. The R^2 value of 0.998776 shows that the model predicts the behavior of the PScF Standard Deviation very well. Unlike the fit of ln(Max), this R^2 value is not inflated. A plot of the predicted Standard Deviation versus the actual one is shown in Figure 6-14. All of the data points lie very close to the line, confirming the model's predictive ability.

The predicted Maximum values and predicted Standard Deviations versus the actual values for all four of the object materials are plotted in Figure 6-15. Each of the predicted values is using the material-specific coefficients and terms determined by the stepwise fitting procedure.

Whole Model

Summary of Fit

RSquare	0.999599
RSquare Adj	0.999558
Root Mean Square Error	0.024161
Mean of Response	-7.18223
Observations (or Sum Wgts)	65

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	6	84.474043	14.0790	24117.24
Error	58	0.033859	0.000584	**Prob > F**
C. Total	64	84.507902		<.0001*

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	-2.015225	0.113321	-17.78	<.0001*
D	-0.076794	0.005963	-12.88	<.0001*
D*D	0.0005113	0.000105	4.87	<.0001*
D*D*D	-1.711e-6	5.979e-7	-2.86	0.0058*
Beta	1.2881986	0.0241	53.45	<.0001*
Beta*Beta	0.0205344	0.002806	7.32	<.0001*
D*Beta	-0.002419	0.000307	-7.87	<.0001*

Figure 6-11. The resulting multivariate model of ln(Max) for carbon. Asterisks in the "Prob>|t|" and "Prob > F" columns indicate statistical significance at the p=0.05 level. All other asterisks indicate multiplication.

Bivariate Fit of Pred. Max By Max

Linear Fit

Pred. Max = 0 + 1*Max
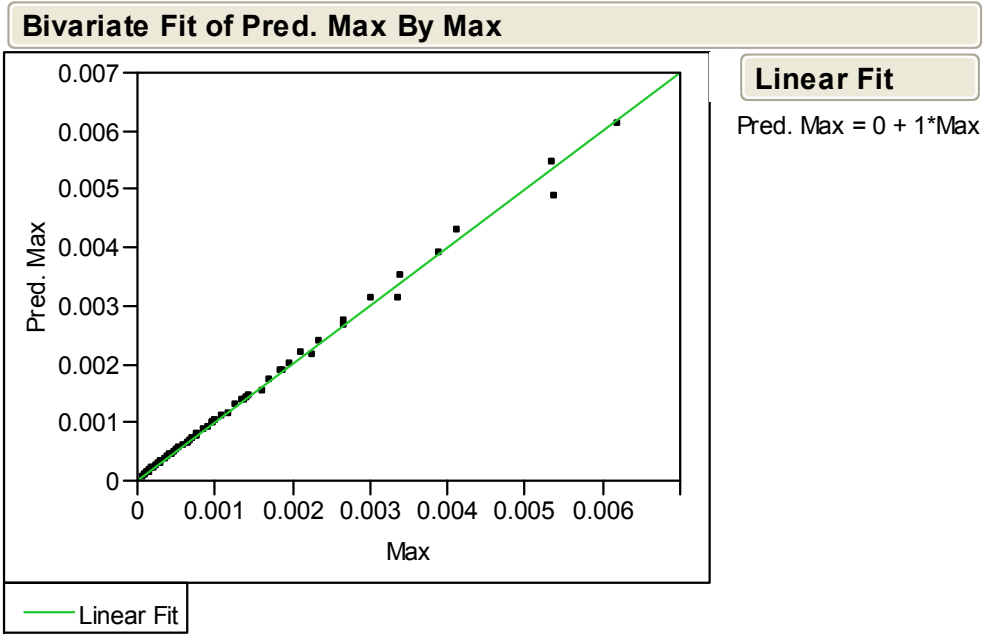
Figure 6-12. A graph of the predicted PScF Maximum values versus the actual values for carbon. Points lying on the line correspond to predicted values which match the actual ones.

Whole Model

Summary of Fit

RSquare	0.998776
RSquare Adj	0.998672
Root Mean Square Error	0.102652
Mean of Response	11.65769
Observations (or Sum Wgts)	65

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	5	507.17665	101.435	9626.248
Error	59	0.62170	0.011	Prob > F
C. Total	64	507.79835		<.0001*

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	3.9549397	0.211871	18.67	<.0001*
D	0.0491041	0.006815	7.21	<.0001*
D*D	0.0010183	5.218e-5	19.51	<.0001*
Tau	0.3312332	0.054078	6.13	<.0001*
Tau*Tau	0.0136081	0.004527	3.01	0.0039*
D*Tau	0.0072002	0.000715	10.07	<.0001*

Figure 6-13. The resulting multivariate model for the PScF Standard Deviation for carbon. Asterisks in the "Prob>|t|" and "Prob > F" columns indicate statistical significance at the p=0.05 level. Other asterisks indicate multiplication.

Bivariate Fit of Predicted Std. Dev. By Std. Dev.
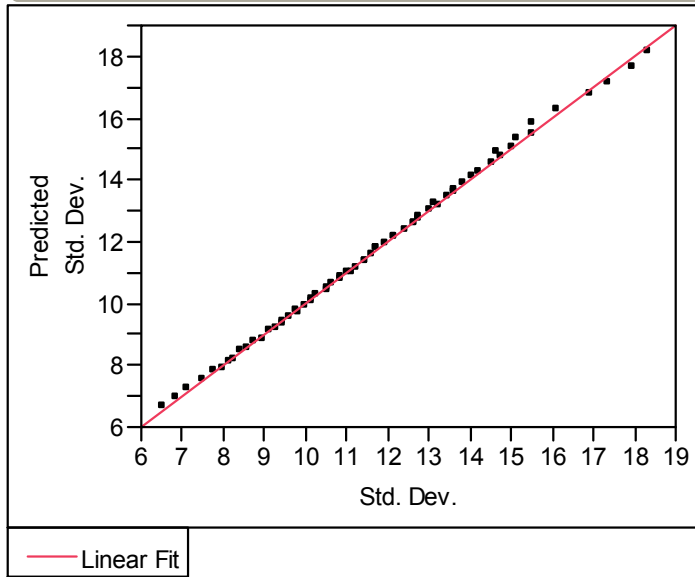
Linear Fit

Predicted Std. Dev. = 0 + 1*Std. Dev.

Linear Fit

Figure 6-14. A graph of the predicted PScF Standard Deviation values versus the actual values for carbon. Points lying on the line correspond to predicted values which match the actual ones.
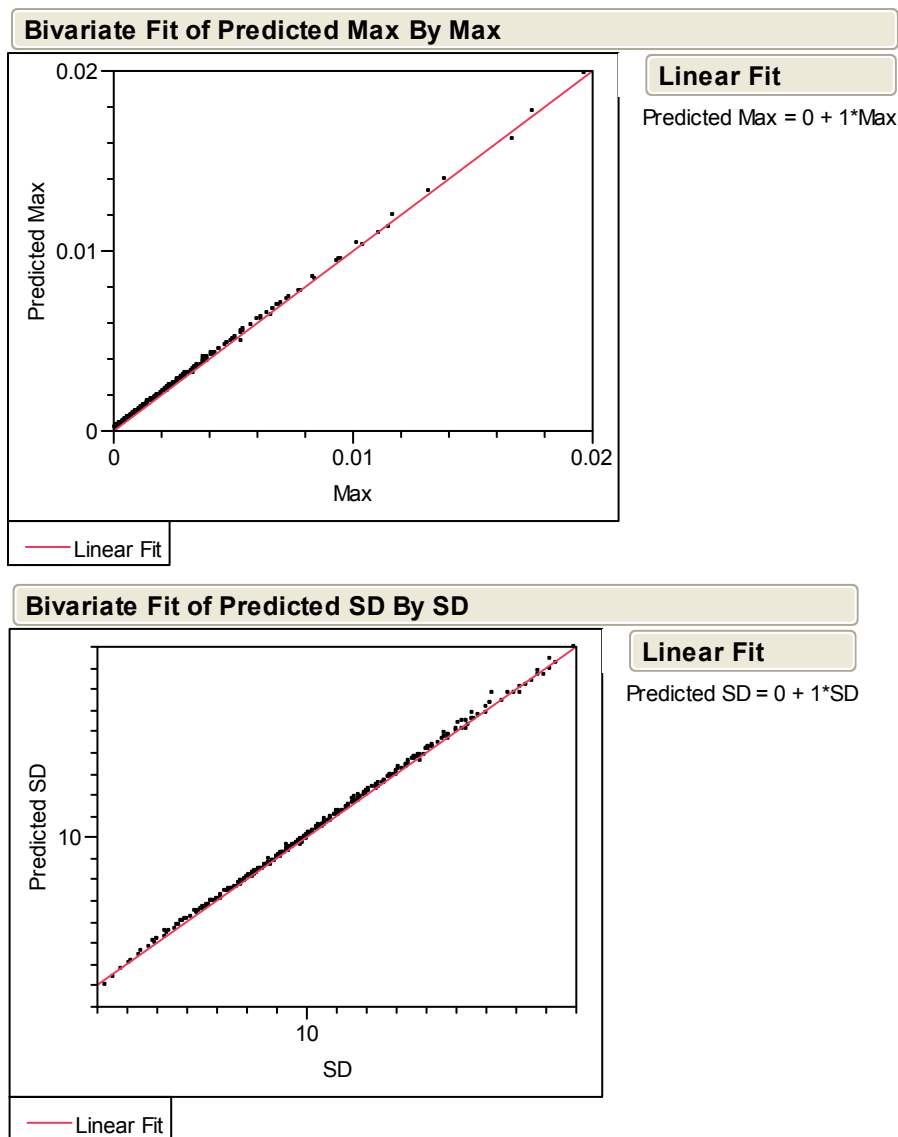
Figure 6-15. Plots of the predicted values of the PScF Maximum versus the actual values (top) and the predicted values of the PScF Standard Deviation versus the actual values (bottom). These plots show the values for polyethylene, carbon, iron, and lead using their individual fits.

All of the data points lie very close to the slope 1 line for both the Maximum and Standard Deviation of the Gaussian PScF fits, indicating the model is predicting the parameters very well for all nuclides.

The results of the ln(Max) fits for materials 5 (average of iron and lead) and 6 (average of carbon, iron, and lead) are presented in Figure 6-16. The R^2 values for these fits (0.960758 for 5 and 0.893104 for 6) are quite a bit lower than they were for the fits of individual materials. The predicted values of the Gaussian Maximum versus the actual ones are plotted in Figure 6-17. A clear grouping of the values by material can be seen in both plots. The averaging tends to under predict the Maximum for the heavier materials being averaged and over predict it for lighter materials. The divergence tends to increase with increasing values of the PScF Maximum. The only material that follows the slope 1 line closely is iron in material 6.

The results for the model of the PScF Standard Deviation for materials 5 and 6 are shown in Figure 6-18. As with the fits of ln(Max), these fits show much lower R^2 values than the individual materials. The predicted values versus the actual ones are plotted in Figure 6-19. As with the Maximum, the averaged materials tend to group together and there is a significant deviation from the slope 1 line in most cases. These results indicate that the averaged values may not produce acceptable results when subtracting the scattering from measured values. These fits will be tested in the next chapter to determine their usefulness.

Whole Model

Summary of Fit

RSquare	0.960758
RSquare Adj	0.96004
Root Mean Square Error	0.250233
Mean of Response	-6.48104
Observations (or Sum Wgts)	168

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	3	251.41568	83.8052	1338.389
Error	164	10.26910	0.0626	Prob > F
C. Total	167	261.68479		<.0001*

Lack Of Fit

Source	DF	Sum of Squares	Mean Square	F Ratio 0.3098
Lack Of Fit	157	8.977259	0.057180	Prob > F
Pure Error	7	1.291845	0.184549	0.9968
Total Error	164	10.269104		Max RSq
				0.9951

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	-2.11251	0.191464	-11.03	<.0001*
D	-0.046441	0.007149	-6.50	<.0001*
D*D	0.0001538	0.000062	2.48	0.0142*
Beta	0.9314129	0.015397	60.49	<.0001*

Whole Model

Summary of Fit

RSquare	0.893104
RSquare Adj	0.8913
Root Mean Square Error	0.393108
Mean of Response	-6.56493
Observations (or Sum Wgts)	242

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	4	305.99501	76.4988	495.0279
Error	237	36.62461	0.1545	Prob > F
C. Total	241	342.61962		<.0001*

Lack Of Fit

Source	DF	Sum of Squares	Mean Square	F Ratio 0.9967
Lack Of Fit	175	27.020090	0.154401	Prob > F
Pure Error	62	9.604518	0.154912	0.5196
Total Error	237	36.624608		Max RSq
				0.9720

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	-1.647906	0.356588	-4.62	<.0001*
D	-0.07204	0.011636	-6.19	<.0001*
D*D	0.0003299	0.000089	3.71	0.0003*
Beta	1.0771524	0.070575	15.26	<.0001*
D*Beta	-0.003809	0.001359	-2.80	0.0055*

Figure 6-16. The resulting multivariate models for the natural logarithm of the Maximum of the PScF Gaussian fits for material 5 (left) and material 6 (right). Asterisks in the "Prob>|t|" and "Prob > F" columns indicate statistical significance at the p=0.05 level. Other asterisks indicate multiplication.
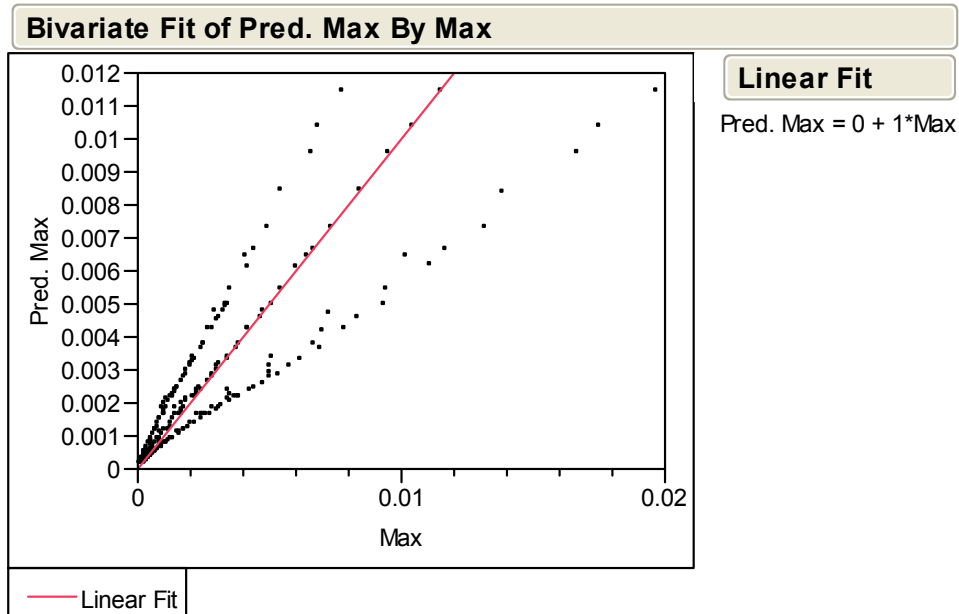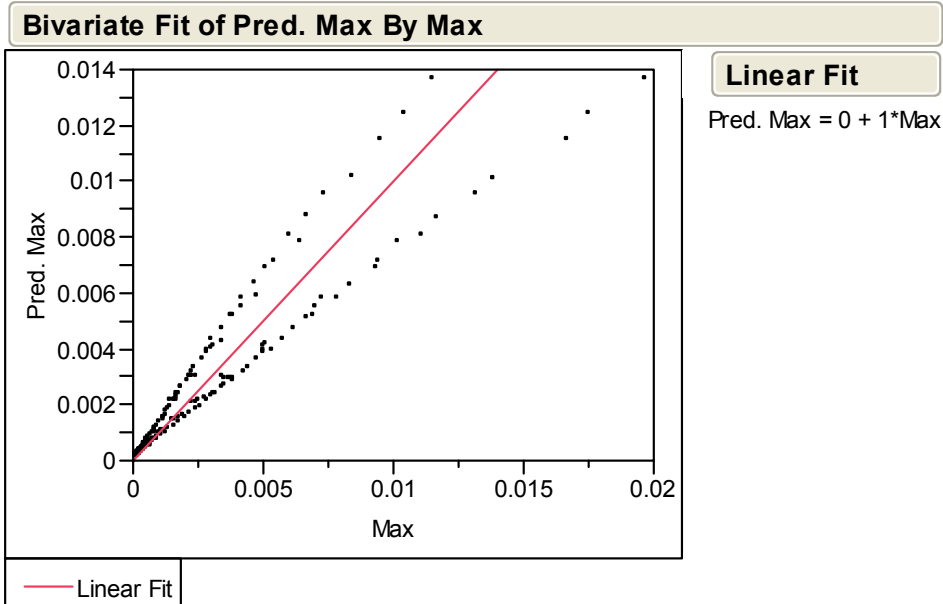
Figure 6-17. Graphs of the predicted PScF Maximum values versus the actual values for material 5 (top) and material 6 (bottom). Points lying on the line correspond to predicted values which match the actual ones. Note the grouping of points that correspond to the individual materials which are being averaged in the definition of these two materials.

Whole Model

Summary of Fit

RSquare	0.758914
RSquare Adj	0.755992
Root Mean Square Error	1.378507
Mean of Response	9.300298
Observations (or Sum Wgts)	168

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	2	987.0140	493.507	259.7019
Error	165	313.5466	1.900	Prob > F
C. Total	167	1300.5607		<.0001*

Lack Of Fit

Source	DF	Sum of Squares	Mean Square	F Ratio
Lack Of Fit	158	278.71535	1.76402	0.3545
Pure Error	7	34.83130	4.97590	Prob > F
Total Error	165	313.54665		0.9915
				Max RSq
				0.9732

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	-0.159086	0.432319	-0.37	0.7134
D	0.1125601	0.005984	18.81	<.0001*
Tau	0.9099116	0.054631	16.66	<.0001*

Whole Model

Summary of Fit

RSquare	0.440757
RSquare Adj	0.436077
Root Mean Square Error	2.454873
Mean of Response	10.34533
Observations (or Sum Wgts)	242

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	2	1135.1526	567.576	94.1816
Error	239	1440.3101	6.026	Prob > F
C. Total	241	2575.4626		<.0001*

Lack Of Fit

Source	DF	Sum of Squares	Mean Square	F Ratio
Lack Of Fit	177	981.9692	5.54785	0.7505
Pure Error	62	458.3408	7.39259	Prob > F
Total Error	239	1440.3101		0.9245
				Max RSq
				0.8220

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	1.8894153	0.636153	2.97	0.0033*
D	0.1123858	0.009114	12.33	<.0001*
Tau	0.7124436	0.080944	8.80	<.0001*

Figure 6-18. The resulting multivariate models for the Standard Deviation of the PScF Gaussian fits for material 5 (left) and material 6 (right). Asterisks in the "Prob>|t|" and "Prob > F" columns indicate statistical significance at the p=0.05 level.

Bivariate Fit of Predicted Std. Dev. By Std. Dev.
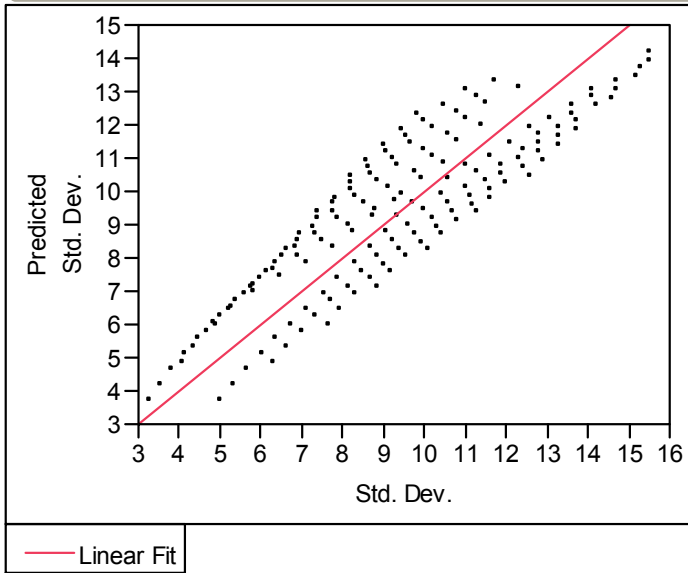
Linear Fit

Predicted Std. Dev. = 0 + 1*Std. Dev.

Bivariate Fit of Predicted Std. Dev. By Std. Dev.
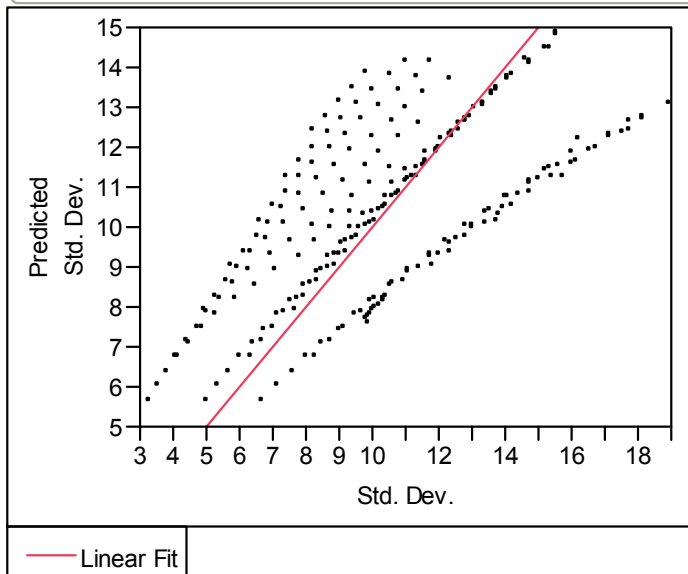
Linear Fit

Predicted Std. Dev. = 0 + 1*Std. Dev.

Figure 6-19. Graphs of the predicted PScF Standard Deviation values versus the actual values for material 5 (top) and material 6 (bottom). Points lying on the line correspond to predicted values which match the actual ones. Note the grouping of points that correspond to the individual materials which are being averaged in the definition of these two materials.

6.1.3 The Point Scatter Function Generating Equations

With the multivariate fits completed, the final form of the PScFGEs can be developed. The PScFGEs will allow the *ScatterSubtract* program (detailed in the next section) to determine the Gaussian Maximum (Max) and Standard Deviation (SD) for the PScF fits

$$PScF(\theta) = Max \times EXP\left(-\frac{\theta^2}{2SD^2}\right), \qquad (6.2)$$

for any combination of material, object-to-detector distance (D), and attenuation (τ). The forms of the PScFGEs are based on the terms used in the multivariate fits which are listed in Table 6-1. At least one material used all of the possible terms in the model. Therefore, the PScFGEs are

$$Max = EXP\left(a_0 + a_1 D + a_2 D^2 + a_3 D^3 + a_4 \beta + a_5 \beta^2 + a_6 \beta D + a_7 \beta D^2\right)$$
$$SD = b_0 + b_1 D + b_2 D^2 + b_3 \tau + b_4 \tau^2 + b_5 \tau D \qquad (6.3)$$

The coefficients of the two PScFGEs are dependent on the material of the object. The values of the Maximum and Standard Deviation coefficients for each of the six materials (and averaged materials) are listed in Table 6-2 and Table 6-3.

6.2 Implementation of the Parameterized Scatter Removal Algorithm

The PSRA is implemented in the *ScatterSubtract* code. As discussed briefly in Section 4.3.4, the *ScatterSubtract* program calculates the corrected attenuation values for the correlation data in the void and object .peaks files. The program first uses an iterative routine to remove the inter-array scatter from the void measurement.

Table 6-2. Material specific coefficients for the Maximum PScF generating equation (Equation 6.3, top)

Material	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
Poly	-2.015225E+00	-7.679420E-02	5.112759E-04	-1.711049E-06	1.288199E+00	2.053442E-02	-2.418549E-03	0
Carb	-2.028606E+00	-7.184288E-02	4.461472E-04	-1.348499E-06	1.131994E+00	1.638951E-02	-1.976645E-03	0
Iron	-1.582257E+00	-8.241122E-02	6.540168E-04	-2.353720E-06	1.037106E+00	8.299626E-03	-2.858238E-03	1.541697E-05
Lead	-1.128685E+00	-6.752032E-02	4.280224E-04	-1.315156E-06	1.214529E+00	2.257471E-02	-2.402139E-03	0
Mat. 5	-2.112510E+00	-4.644079E-02	1.537551E-04	0	9.314129E-01	0	0	0
Mat. 6	-1.647906E+00	-7.204015E-02	3.299292E-04	0	1.077152E+00	0	-3.809334E-03	0

Table 6-3. Material specific coefficients for the Standard Deviation PScF generating equation (Equation 6.3, bottom)

Material	b_0	b1	b_2	b_3	b_4	b_5
Poly	3.954940E+00	4.910408E-02	1.018272E-03	3.312332E-01	1.360807E-02	7.200204E-03
Carb	2.199167E+00	1.227376E-01	4.967885E-04	6.212162E-01	3.649710E-02	4.528034E-03
Iron	1.120331E+00	1.216453E-01	0	5.337523E-01	1.092541E-02	3.126423E-03
Lead	-5.296592E-01	1.124436E-01	-2.060663E-04	6.195354E-01	4.911718E-02	0
Mat. 5	-1.590860E-01	1.125601E-01	0	9.099116E-01	0	0
Mat. 6	1.889415E+00	1.123858E-01	0	7.124436E-01	0	0

It then uses the PScFGEs to remove the object scatter from the object measurement and find the corrected attenuation values using a second iterative routine. The source code for the *ScatterSubtract* program can be found in Appendix H. The remainder of this section will present the details of how the PSRA is implemented by *ScatterSubtract*.

ScatterSubtract is launched from an MS-DOS command line using the syntax

ScatterSubtract <Object .peaks file> <Void .peaks file> <Number of subsamples> <Number of detectors in the array> <Object-to-detector distance> <Material> <Level of Knowledge>.

Acceptable entries for the material are *Polyethylene*, *CH2*, *Carbon*, *C12, Iron*, *Fe*, *Lead*, or *Pb*. Only the first two letters are used by the program, so abbreviations such as *Poly* or even *Po* are also acceptable. The level of knowledge (LOK) represents the operator's knowledge about the shielding material. This information is entered into the program using the number 1, 2, or 3. The LOK value has no effect if the material is polyethylene. If the material is iron or lead and the LOK value is 2, the PScFGE coefficients for material 5 (average iron/lead values) are used. Similarly, if the LOK value is 3 and the material is carbon, iron, or lead, the PScFGE coefficients for material 6 (average carbon/iron/lead values) are used.

Once the input from the command line is read, the program opens the two .peaks files and begins reading them into memory. First, the NPS values are read and stored for normalization purposes. Then, the correlation peak values are read into the *ObjPeaks* and *VoidPeaks* arrays. The dimensions of these two arrays are *NumDets*+1 rows by 6 columns by *NumSS* panes where *NumDets* is the number of detectors in the array and *NumSS* is the number of subsamples. These arrays are divided into subsamples because

135

the ISF and PScF is applied to each subsample separately. The detector angle, total

correlations, direct correlations, and no cross talk correlations are read into the first four

columns of these arrays. The no cross talk values are also written into the fifth column as

the initial guess for the corrected values. The sixth column is initially left empty, but it

will be used later in the program. The major arrays used in the *ScatterSubtract* program

along with their dimensions and purposes are listed in Table 6-4. All arrays are of the

REAL (floating point) data type.

Once the two peaks arrays have been filled, the program finds the corrected value

of C_0 by subtracting the ISFs from the measured values. The purpose is to solve the

equation

$$C_{0,corr}(i) = C_{0,meas}(i) - \sum_{\substack{j=1 \\ i \neq j}}^{n} C_{0,direct}(j) ISF(j \rightarrow i), \qquad (6.4)$$

for each detector position where $C_{0,corr}$ is the corrected peak correlation value, $C_{0,meas}$ is

the measured correlation value, $C_{0,\,direct}$ is the direct (true) neutron correlation value, and

$ISF(j \rightarrow i)$ is the number of additional correlations in detector i per directly transmitted

neutron correlation in detector j. This equation is applied separately to each subsample

and the index n is the number of detectors in the array. The ISF values are computed

using Equation 5.7 and the best fit ISF parameters are given in Section 5.3.1. The angle,

θ, is the difference between the detector angles of i and j. These values are stored in the

ISF array. The rows of the array represent the detector being scattered into (i) and the

columns represent the detector, j, which is the source of the scatter.

Table 6-4. The main data arrays used in the *ScatterSubtract* program. All arrays use the REAL data type.

Array Name	Array Dimensions	Array Contents	Column Contents
ObjPeaks	NumDets[*]+1 Rows × 6 Columns × NumSS[†] Panes	Fast neutron peak correlation values for the object measurement.	1 – Detector Angle 2 – Total 3 – Direct 4 – No Cross Talk 5 – Corrected 6[‡] – Fractional Error of Corrected Value
VoidPeaks	NumDets+1 Rows × 6 Columns × NumSS Panes	Fast neutron peak correlation values for the void measurement.	1 – Detector Angle 2 – Total 3 – Direct 4 – No Cross Talk 5 – Corrected 6[‡] – Fractional Error of Corrected Value
Attenuation	NumDets+1 Rows × 6 Columns × NumSS Panes	Neutron attenuation values for each measurement.	1 – Detector Angle 2 – Total 3 – Direct 4 – No Cross Talk 5 – Corrected 6[‡] – Fractional Error of Corrected Value
ISF	NumDets Rows × NumDets+1 Columns × NumSS Panes	The number of additional counts in detector i (rows) per directly transmitted counts in detector j (columns) due to inter-array scattering.	ISF of detector j
PScF	NumDets Rows × NumDets+1 Columns × NumSS Panes	The number of additional counts in detector i (rows) per directly transmitted counts in detector j (columns) due to object scattering.	PScF of detector j

[*]NumDets is the number of detectors in the array.
[†]NumSS is the number of subsamples in the measurement.
[‡]This is the final value stored in this array. It is used for other purposes during the course of the program.

Table 6-4 (continued)

Array Name	Array Dimensions	Array Contents	Column Contents
Uncertainty	NumDets* Rows × 5 Columns × NumSS† Panes	Holds the uncertainty (1 σ) of the calculated attenuation values.	1 – Angle 2 – Total 3 – Direct 4 – No Cross Talk 5 – Corrected
Scatter	NumDets Rows × 5 Columns × NumSS Panes	Holds the fraction of each correlation value which was due to scattering rather than directly transmitted neutrons.	1 – Angle 2 – Total 3 – Direct 4 – No Cross Talk 5 – Corrected
ChiSq	3 Columns	Hold the result of the χ^2 goodness of fit test comparing the attenuation curve to the Direct (true) attenuation.	1 – Total 2 – No Cross Talk 3 – Corrected

*NumDets is the number of detectors in the array.
†NumSS is the number of subsamples in the measurement.

Although the value of $C_{0,direct}$ is known for simulations, the PSRA is intended to be applied to laboratory measurements. Therefore, the direct values will be assumed to be unknown. *ScatterSubtract* will use those values only for checking the accuracy of the corrected values. Without the direct values, the corrected void correlation values must be solved iteratively. For each iteration, the equation

$$C_{0,corr}^{(l)}(i) = C_{0,meas}(i) - \sum_{\substack{j=1 \\ i \neq j}}^{n} C_{0,corr}^{(l-1)}(j) ISF(j \rightarrow i), \tag{6.5}$$

is solved, where the superscript l represents the iteration number. When $l = 1$, the no cross talk correlation values are used as the initial guess. With each iteration, the value of the summation in Equation 6.5 is calculated for each detector position and stored in the last column of the *ISF* array. That value is then subtracted from the no cross talk value to generate the new corrected value, which is stored in the sixth column of the *VoidPeaks* array. In the event that a correction results in a value less than 0, the corrected value is set to 0. After each iteration, the sum of the corrected values is stored in the last row/pane of the array. That value is compared to the sum of the previous iteration to check for convergence. The convergence equation is

$$\varepsilon = \frac{\sum C_{0,corr}^{(l)} - \sum C_{0,corr}^{(l-1)}}{\sum C_{0,corr}^{(l-1)}}. \tag{6.6}$$

The corrected values are considered to be converged when $\varepsilon \leq 0.00001$. Once the corrected C_0 values are found, they are stored in the fifth column of the *VoidPeaks* array. The fractional errors of the corrected values are calculated using the equation

$$fe = \frac{C_{0,corr} - C_{0,direct}}{C_{0,corr}}, \tag{6.7}$$

and stored in the sixth column of the *VoidPeaks* array.

ScatterSubtract then solves for the corrected correlation peak values for the object measurement. The purpose is to solve the equation

$$C_{corr}(i) = C_{meas}(i) - \frac{NPSO}{NPSV} \sum_{\substack{j=1 \\ i \neq j}}^{n} C_{0,corr}(j) ISF(j \to i) \exp(-\tau)$$

$$- \frac{NPSO}{NPSV} \sum_{j=1}^{n} C_{0,corr}(j) PScF(j \to i).$$

(6.8)

The ratios of the NPS values in this equation are for converting the C_0 values to the object source strength. Similar to Equation 6.4, this equation contains a value, τ, which depends on the corrected correlation value. Therefore, the corrected object values must also be solved iteratively. The iterative equation is

$$C_{corr}^{(l)}(i) = C_{meas}(i) - \frac{NPSO}{NPSV} \sum_{\substack{j=1 \\ i \neq j}}^{n} C_{0,corr}(j) ISF(j \to i) \exp(-\tau^{(l-1)})$$

$$- \frac{NPSO}{NPSV} \sum_{j=1}^{n} C_{0,corr}(j) PScF(j \to i).$$

(6.9)

Before the iteration begins, the angles of the detectors are written into the first column of the *Attenuation* array. The total, direct, and no cross talk attenuation values are calculated using the exponential attenuation formula (Equation 5.8) and stored in the second through fourth columns of the attenuation array. The fifth column is used as the initial guess for the iterative procedure. It is calculated using the measured (no cross talk) value of C and the corrected value of C_0 found earlier.

At the start of each iteration, the material, object-to-detector distance, difference between scattering and receiving detector angles, and the attenuation value of the previous iteration for each detector position are submitted to the PScFGE subroutine.

140

This subroutine determines the appropriate PScF parameters using the PScFGEs and coefficients from Section 6.1.3 and returns the appropriate PScF value using Equation 5.10. The structure of the PScF array is identical to that of the ISF array – rows represent the detector being scattered into and the columns represent the detector whose neutrons are responsible for the scattering. Unlike the ISF values which are strictly a function of the detector geometry and materials, the PScF values must be recalculated with each iteration.

After the individual PScF values are computed, the sum of object scattering to each detector (the last summation in Equation 6.9) is calculated and stored in the last column of the PScF array. Simultaneously, the total inter-array scattering (the first summation in Equation 6.9) is calculated and stored in the last column of the ISF array. The corrected object correlation values are then computed by subtracting the scattering sums from the measured values. The corrected values are then stored in the sixth column of the *ObjPeaks* array. Using this value, the new attenuation values are calculated and stored in the sixth column of the *Attenuation* array.

One possible problem that can occur during the first few PScF iterations is an overcorrection of the peaks value. The overcorrection occurs because the measured attenuation values are lower than the true values. If the measured attenuation is greater than 1, the resulting PScF maximum will be larger than the true value and the object peaks value will be overcorrected. In most circumstances, this does not present a problem. If the peaks value is overcorrected in one iteration, the resulting attenuation will be too high and it will be undercorrected on the next iteration. In this case, the attenuation values should continue to oscillate around the true value, getting closer and

closer with each pass until they finally converge. However, if the data is somewhat noisy an overcorrection could be larger than the total counts at one or more detector positions. If this occurs, the final results can converge to values far from the true ones. In order to prevent this from occurring, the *ScatterSubtract* program under corrects the scattering for the first seven iterations. This is accomplished by multiplying the PScF by a constant which increases gradually during the first few iterations. The values of the constant for iterations one through seven are 0.2, 0.4, 0.6, 0.8, 0.85, 0.9, and 0.95. Iterations eight and higher receive no adjustment.

Another potential problem is values that fail to converge and cycle through a series of under and overcorrections about the true values. This is especially likely to happen with noisy data. In order to force convergence in this scenario, the corrected peaks values for each iteration after the tenth are averaged with the old ones. This is accomplished by using a weighted sum of the form

$$C_{corr}^{(l)} = \frac{(l-10)C_{corr}^{(l-1)} + C_{corr}^{(l)}}{l-9}. \tag{6.10}$$

The sum of the corrected correlation values is written in the last row/pane of the *ObjPeaks* array. It is used for checking convergence via the equation

$$\varepsilon = \frac{\sum C_{corr}^{(l)} - \sum C_{corr}^{(l-1)}}{\sum C_{corr}^{(l-1)}}. \tag{6.11}$$

The values are considered to be converged when $\varepsilon \le 0.00001$. Once the corrected values are converged, they are stored in the fifth column of the *VoidPeaks* array. The corrected attenuation values are stored in the fifth column of the *Attenuation* arrays. The sixth

column of these arrays is used to store the fractional errors that are computed in the same manner as Equation 6.7.

ScatterSubtract then calculates the uncertainty of the total, direct, no cross talk, and corrected attenuation values. These values are stored in the second through fifth columns, respectively, of the *Uncertainty* array. The detector angles are copied into the first column from the *Attenuation* array. The attenuation uncertainties are calculated using the formula

$$\sigma_\tau = \sqrt{\frac{1}{C} + \frac{1}{C_0}}, \tag{6.12}$$

which is derived by propagating the uncertainty of the exponential attenuation equation.

Next, the program calculates the fraction of scattering in the total, no cross talk, and corrected object correlation values. These values are stored in the *Scatter* array. Because scattering is the only source of error, the equation used to find them is identical to Equation 6.11, with the exception that either the total or no cross talk values are substituted for the corrected ones where appropriate.

The last series of calculations performed in *ScatterSubtract* are χ^2 goodness of fit tests on the total, no cross talk, and corrected attenuation values. The form of the χ^2 equation is given in Equation 5.6. The attenuation values are compared to the direct attenuation values. The uncertainty for each detector position is the uncertainty of the direct attenuation which is stored in the *Uncertainty* array.

Once all of the calculations are completed, the results are written out to text files. These text files can then be imported into a suitable analysis program such as Microsoft

Excel for further analysis and plotting as desired. The program output files and their

contents are listed in Table 6-5.

Table 6-5. The *ScatterSubtract* output files and their contents.

File Name	Contents
Void.iter	Records the corrected void peak values after each iteration. Primarily used for troubleshooting in the event of a convergence failure.
Obj.iter	Records the corrected object peak values after each PScF iteration. Primarily used for troubleshooting in the event of a convergence failure.
Atten.iter	Records the corrected attenuation values after each iteration. Primarily used for troubleshooting in the event of a convergence failure.
Void.out	Records the final values of the void correlation peaks at each detector position. Columns include detector angle, total correlations, direct correlations, no cross talk correlations, corrected correlations, and the fractional error of the corrected values.
Object.out	Records the final values of the object correlation peaks at each detector position. Columns include detector angle, total correlations, direct correlations, no cross talk correlations, corrected correlations, and the fractional error of the corrected values.
Attenuation.out	Records the final attenuation values at each detector position. Columns include detector angle, total attenuation, direct attenuation, no cross talk attenuation, corrected attenuation, and the fractional error of the corrected values.
Scatter.out	Records the fraction of scatter in the total, no cross talk, and corrected object correlation values.
ChiSq.out	Records the results of the χ^2 goodness of fit tests for the total, no cross talk, and corrected attenuation curves.

7. Testing of the Parameterized Scatter Removal Algorithm

With the PSRA completed, the final step is to test it and modify it if necessary. This will be done using a large number of simulated and experimental NMIS imaging scenarios. The PSRA will be applied to each of these measurements, and the resulting values will be compared to the true ones. For simulated measurements, these values are computed by the *PoliMiPP* post-processor. The values are recorded in the *Direct* column of the .peaks file. The *ScatterSubtract* output includes these values for ease of comparing the corrected values to the true ones. For experimental measurements, the true attenuation values will be calculated using *MCNP-PoliMi* simulations of the scenario. In addition to the testing, the final section of this chapter will discuss a method for integrating a generalized form of the PSRA into future NMIS imaging measurements.

7.1 Simulation Testing and Results

The first series of tests conducted was a large number of *MCNP-PoliMi* simulations. The methodology for simulating NMIS imaging measurements was presented in detail in Section 4.2 and the codes used were discussed in Sections 4.3 and 6.2. Unless noted otherwise, each simulated measurement consists of four subsamples and each subsample uses 2.5×10^7 source neutrons. This value corresponds approximately to a measurement time of 15 minutes per subsample (60 minutes total) in the laboratory using the API-120 DT neutron generator running at an output of 4×10^7 neutrons per second produced isotropically. All simulations use a source-to-detector

distance of 110 cm and an array of 32 2.54 × 2.54 × 10.16 cm plastic scintillators with a center-to-center angular separation of approximately 1.67°.

Only a small portion of the simulation results are presented in this section. Additional simulation results are shown in Appendix I. A summary of the results for all simulations are presented in Section 7.1.3.

7.1.1 Initial PSRA Testing

The first series of test simulations uses the same cylindrically symmetric geometry used to calculate the PScFs in Section 5.2. This geometry is useful for validating the PSRA methodology itself without any effects that might be caused by a different object geometry. A total of four scenarios were chosen for testing. Each scenario was based on one of the ones used for computing the PScFGEs. Each uses a different material so that all four of the materials in the library are tested. The scenarios also cover a wide range of χ^2 values on the PScF fits. Because these simulations use the PScF geometry, they will be used to adjust the PSRA algorithm if necessary. These adjustments will be tested using other simulation geometries.

The only change from the PScF modeling geometry is that the objects are complete cylinders instead of cylindrical arcs of material. This change was necessitated by a bug in the simulation software. In simulations run without the vertical collimator, no neutron collisions occurred while passing through the object initially. The source of the bug is undetermined, but it is likely related to the fact that the planes used to define the arc in the PScF geometry passed through the source location. Since the arcs used in the PScF simulations extended far beyond the angular borders of the detector arrays, the

147

use of a full cylinder will not produce any substantive changes and the problem was not pursued further. An example of the modeled geometry is depicted in Figure 7-1.

The first step of the PSRA corrects the inter-array scatter in the void measurement. Since the same void output file is used with all other simulations for determining the attenuation curves, any error in the correction of these values will result in incorrect answers for all other simulations. Thus, it is critical to ensure that the inter-array scattering is being removed from the void simulation properly before testing the correction of attenuation values.

The *ScatterSubtract* program outputs the corrected values for the void measurement in the Void.out file. In addition, the corrected values after each iteration of the ISF correction are written into the Void.iter file.
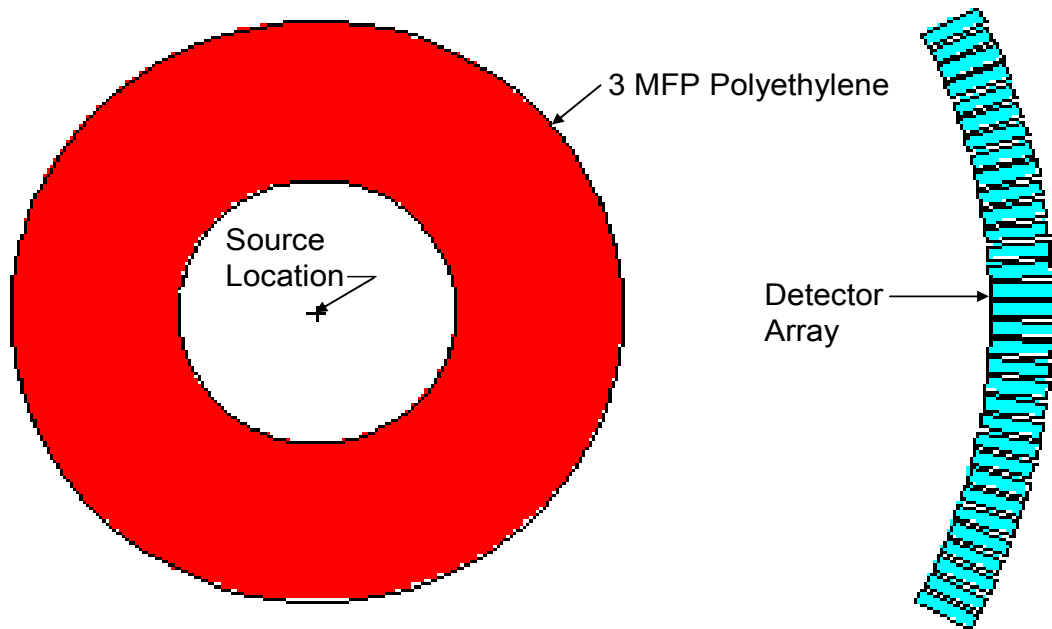
Figure 7-1. The geometry used to simulate an NMIS imaging measurement of a cylindrically symmetric object. The source-to-detector distance in all simulations is 110 cm.

The measured void correlation values and the values resulting from directly transmitted neutrons are shown in Figure 7-2. The summation of the eight pixels produces a flat neutron profile, as expected from the calculations in Section 3.1. Because the center of the pixels is offset -1.5°, the neutron correlations drop to zero at large positive detector angles while at large negative angles, the correlations drop to only 40% of the maximum value. This fact will have some bearing on the calculated attenuation curves and will be discussed further in later sections. For the uncorrected values, inter-array scatter produces extra counts which result in a neutron correlation profile that is larger than the *Direct* values. For this detector array geometry and the 1 MeV neutron energy threshold used during post-processing, the measured values are approximately 5% larger than the *Direct* ones.
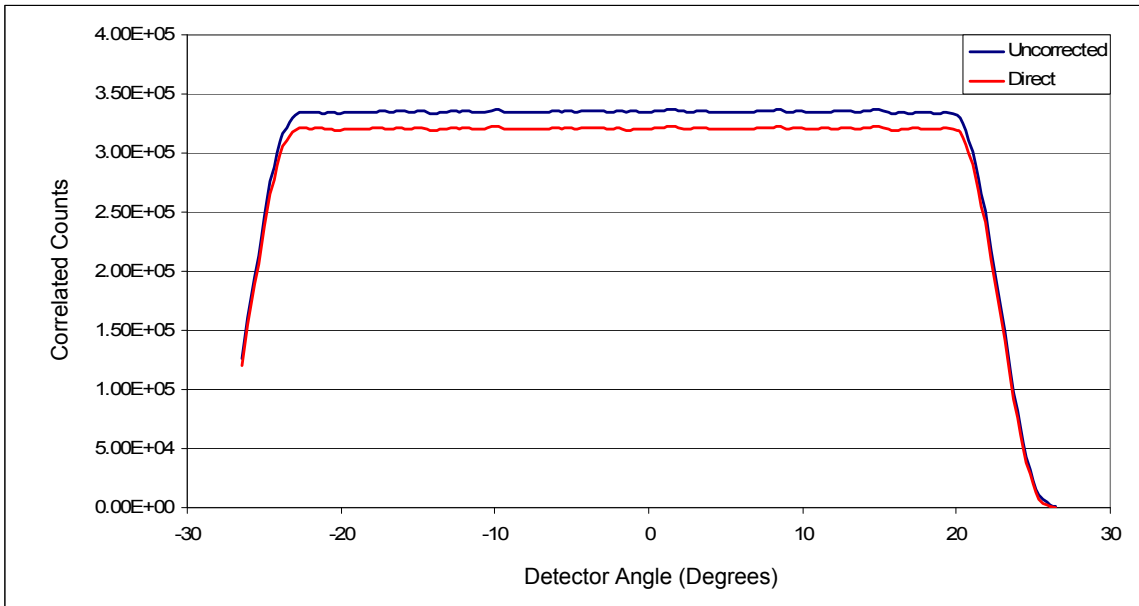
Figure 7-2. A plot of the Void simulation neutron correlation as a function of detector angle. The *Direct* curve consists of only the directly transmitted DT neutron response while the *Uncorrected* curve includes neutrons scattered from one detector in the array to another.

149

The void correlation curves after each iteration of the ISF subtraction are plotted in Figure 7-3. Note the scale on the vertical axis, which zooms in on the flat top of the neutron correlation profile. The Iteration 0 curve is the uncorrected value taken directly from the .peaks file. The Iteration 1 curve falls just below the direct values. Iterations 2-5 all overlap and are indistinguishable even with this vertical scaling. The Iteration 5 curve follows the *Direct* correlation curve extremely well and only very slight deviations of the two curves are visible. The fractional errors of the corrected values (see Equation 6.6) range between approximately 1×10^{-4} and 1×10^{-3}. These values are less than or equal to the fractional uncertainty of the correlation values. Thus, the corrected correlation values are statistically identical to the *Direct* values.

The first object simulation modeled was the Poly63 scenario. The neutron correlation curve for each of the PScF subtraction iterations is plotted in Figure 7-4. As discussed in Section 6.2, it is desirable to under correct the object scattering initially in order to prevent the possibility of the correlation curve converging to a value far from the true values. This is evident in Iterations 1-3, which progressively drop closer to the *Direct* correlation curve as scattering is removed. Note that the uncorrected value has a convex top resulting from the fact that the center detectors are receiving more scattering than those near the ends of the array. In addition to the corrected values becoming smaller with each iteration, the curve loses the domed top and takes on the correct flat shape. Iterations 8-12 were omitted for clarity. Iteration 13 is the converged result, and thus, its value represents the final corrected value.
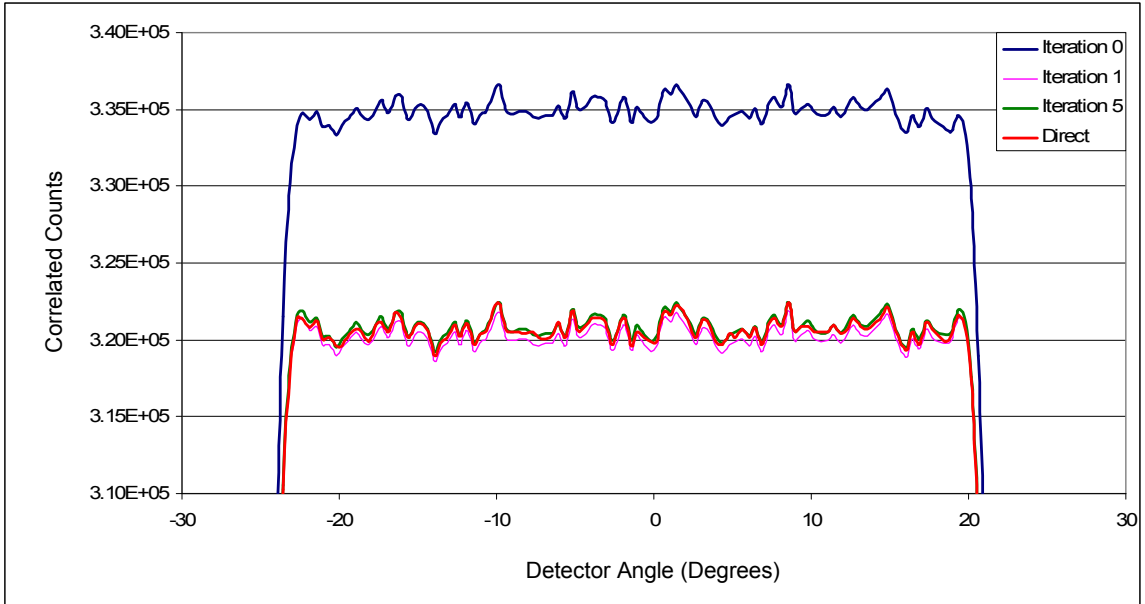
Figure 7-3. A plot of the Void neutron correlation curve for each of the ISF subtraction iterations and the *Direct* curve. Note the vertical scale, which is zoomed in on the top of the correlation profile.
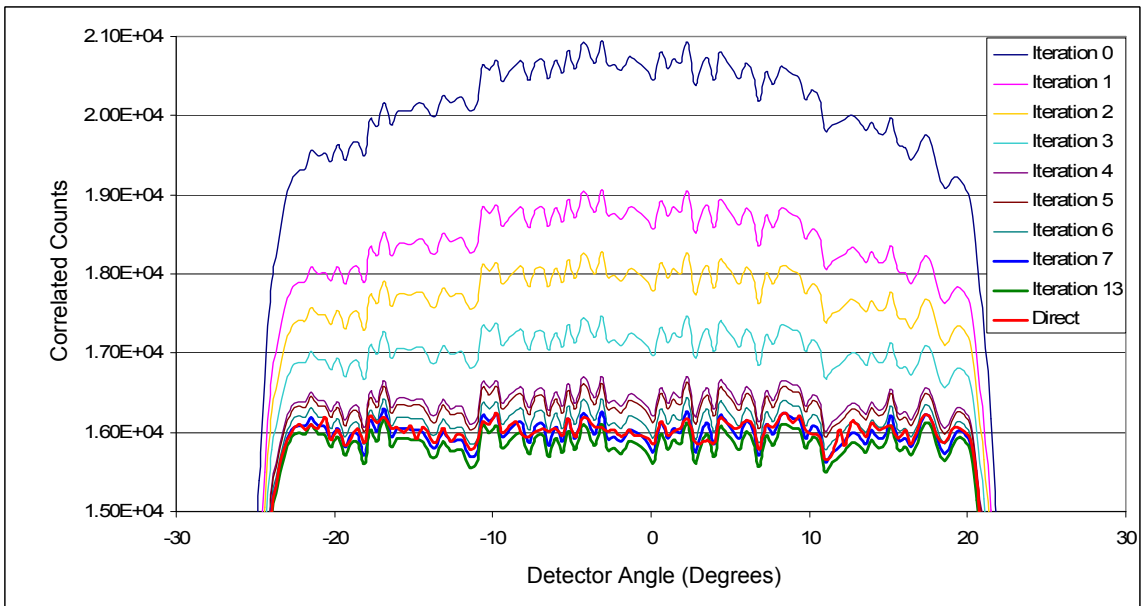
Figure 7-4. The neutron correlation curves for the Poly63 scenario. The curve resulting from each iteration of the PScF correction is shown along with the *Direct* value which includes only correlations due to uncollided 14 MeV neutrons. Iterations 8-12 are omitted for clarity.

The Poly63 correlation curves are plotted on a much narrower vertical scale in Figure 7-5. Only the seventh and thirteenth (final) iterations are shown for clarity. Although the correlation curve of Iteration 13 generally falls within the statistical fluctuations of the *Direct* curve, it appears to be systematically lower, indicating a slight overcorrection. The curve for Iteration 7 follows the *Direct* value much more closely and does not appear to be systematically biased in either direction. The carbon PScF scenario, Carb61, shows the same result with Iteration 7 being very close to the *Direct* value and the final iteration being a slight overcorrection.

The attenuation curves for the Poly63 scenario after each of the PScF subtraction iterations are shown in Figure 7-6 and Figure 7-7. These curves are the results of using the exponential attenuation equation (Equation 1.1) with the corrected object correlation values after each iteration and the corrected void iteration values shown in Figure 7-3. As the object correlation values fall with each successive iteration, the resulting attenuation values rise. The convex shapes visible in the first few iterations of Figure 7-4 result in a concave attenuation curve in Figure 7-6. Again, the Iteration 7 values appear to be very close to the *Direct* ones, while the final iteration (Iteration 13) appears to be a slight overcorrection. Iteration 7 corresponds to a PScF correction of 95% of the value calculated using the PScFGEs. This discrepancy is likely due to a slight non-Normality in the tails of the object scattering function. Although the deviation is very small for any given detector, the superposition of many PScFs produces a small but noticeable overcorrection. In order to account for this overcorrection, the *ScatterSubtract* code was modified to remove a maximum of 95% of the PScF maximum value from polyethylene or carbon objects.

152
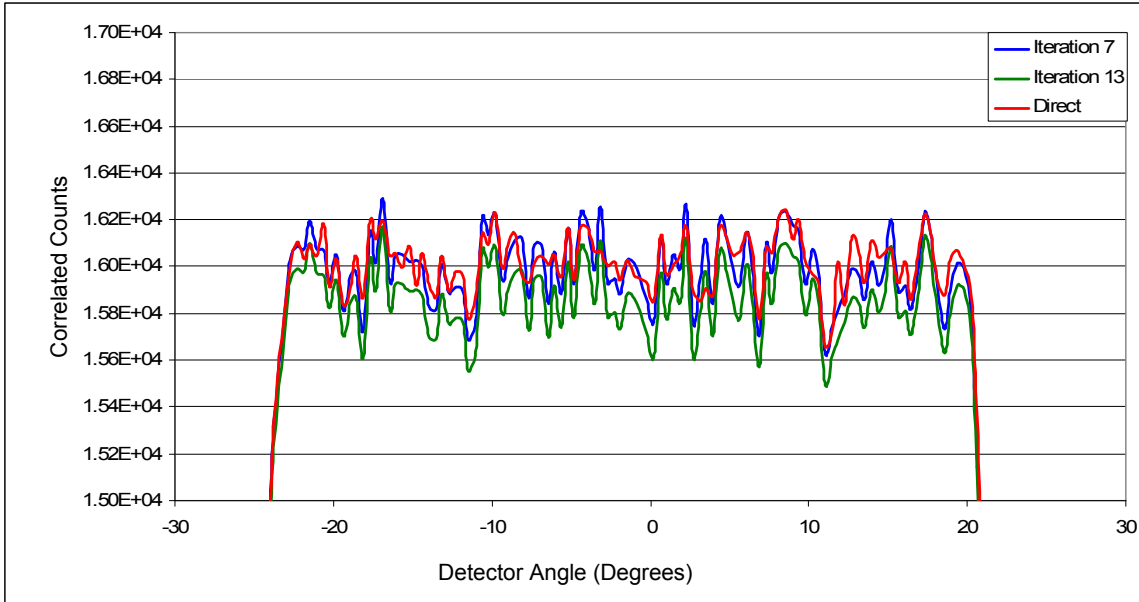
Figure 7-5. The neutron correlation curves for the Poly63 scenario plotted on a narrower vertical scale. The Iteration 7 curve represents a correction of 95% of the PScF maximum, and Iteration 13 is the final converged value. Note that the Iteration 7 curve generally follows the *Direct* curve better.
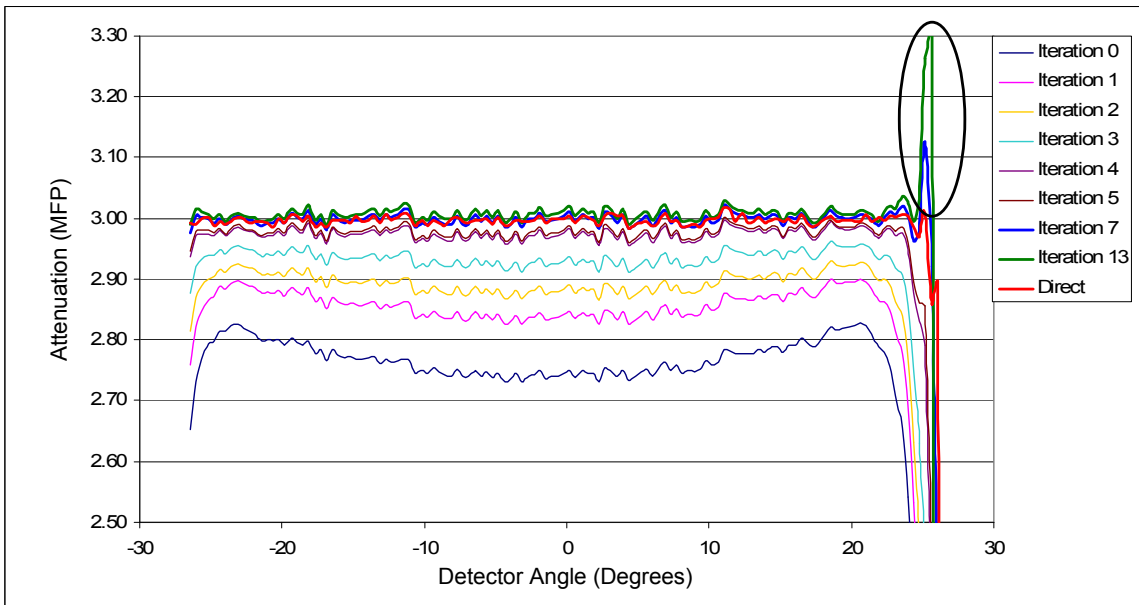
Figure 7-6. A plot of the attenuation curves for the Poly63 scenario. The *Direct* attenuation curve is shown along with the values after several of the scatter subtraction iterations. Iterations 6 and 8-12 are omitted for clarity. The large peak in the circled region is caused by poor statistics in that region due to a small number of source neutrons.
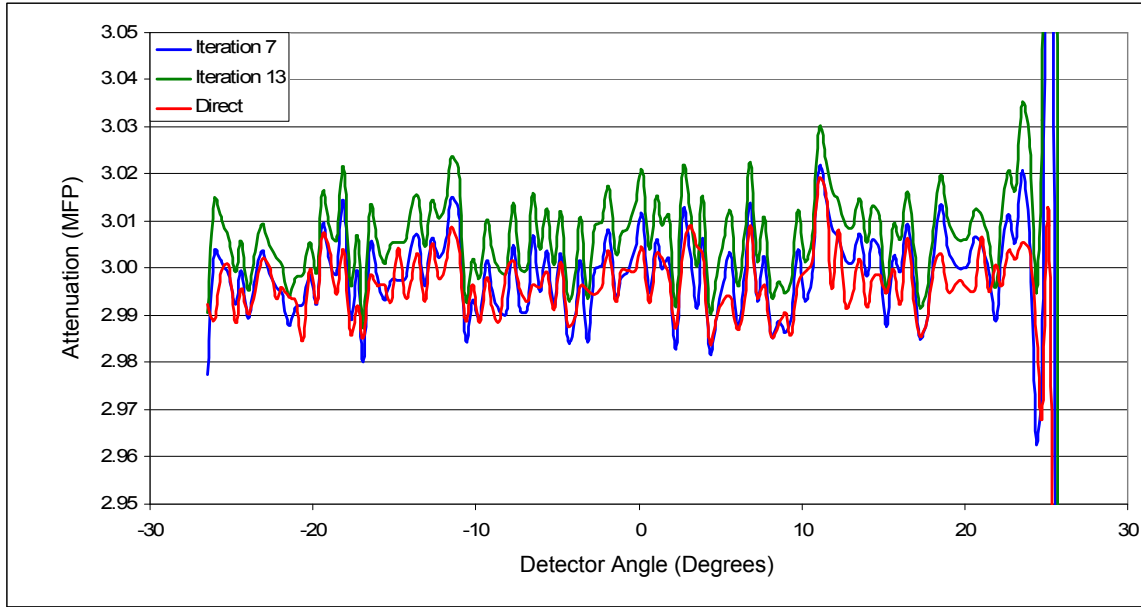
Figure 7-7. A plot of the Poly63 attenuation curves on a narrower vertical scale. As with the object correlation values, Iteration 7 matches the *Direct* values better than the final converged values corresponding to Iteration 13.

Another feature visible in Figure 7-6 is the presence of a large peak in the attenuation curve at a detector angle of approximately +25°. The peak corresponds to the region of the neutron correlation curve (see Figure 7-2) where the number of correlations has diminished to almost zero. Because of the small number of counts and corresponding large statistical uncertainty in this region, the scatter correction can result in an extremely high corrected attenuation value. Because of the small number of source neutrons, the peak has very little effect on the PSRA. In general, this area should be avoided when placing an object to be imaged.

The attenuation curves for the Iron76 and Lead32 scenarios, respectively, are shown in Figure 7-8 and Figure 7-9.
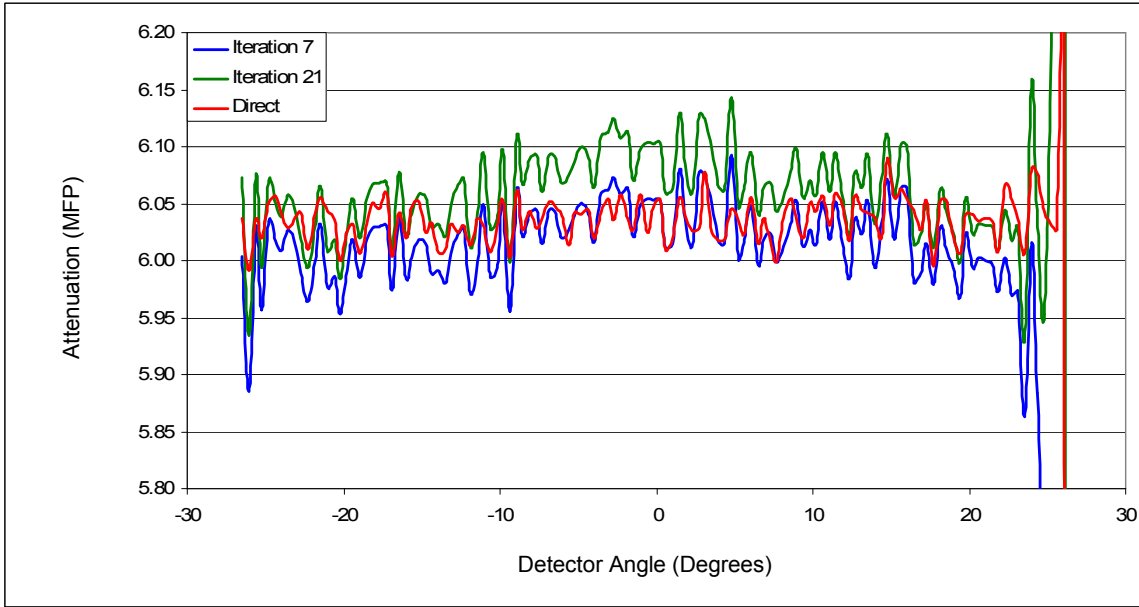
Figure 7-8. A plot of the attenuation curves for the Iron76 scenario. Note that the Iteration 7 curve is a slight undercorrection and the final converged values represented by the Iteration 21 curve are a slight overcorrection.
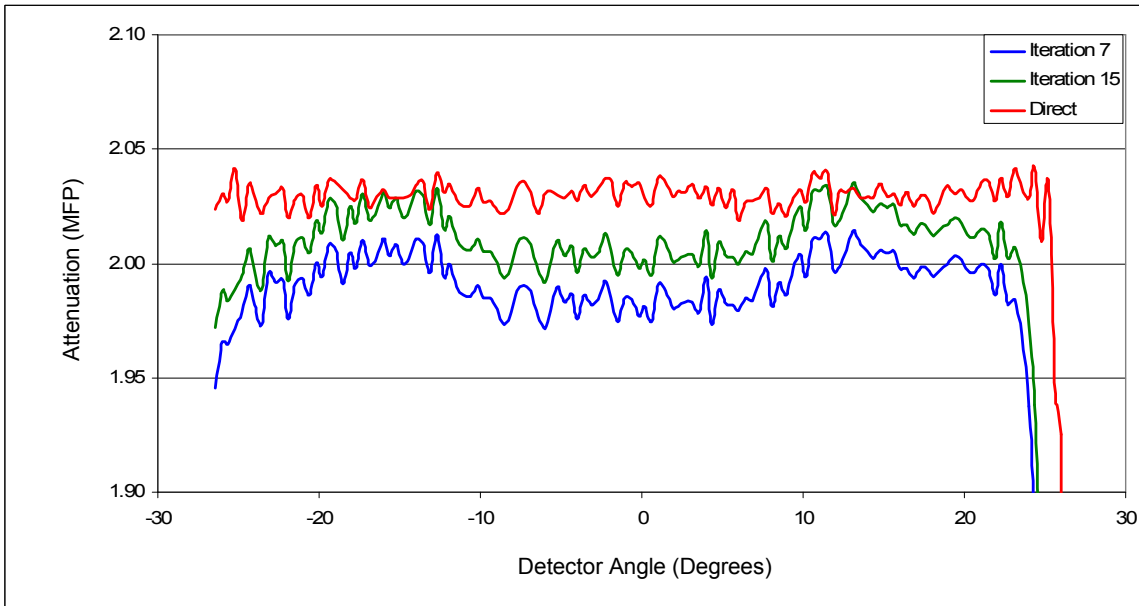
Figure 7-9. A plot of the attenuation curves for the Lead32 scenario. Note that even the final converged values represented by the Iteration 15 curve are a slight undercorrection.

155

For each, the *Direct* attenuation curve is shown along with the corrected attenuation values after Iteration 7 and the final iteration. Because of the 6 MFP thickness, the Iron76 scenario used 1×10^8 source neutrons per simulation in order to reduce the statistical fluctuations in the results. For the Iron76 attenuation curve, the Iteration 7 curve appears to be a slight undercorrection while the final corrected curve (Iteration 21) appears to be a slight overcorrection. With the Lead32 attenuation curve, even the final corrected curve (Iteration 15) is a slight undercorrection. This increasing undercorrection with the heavier scattering nuclei is likely caused by the forward peaking of the elastic scattering cross section. For lead in particular, the diffraction patterns in the elastic scattering cross section caused the Gaussian PScFs to fit somewhat poorly in the tail regions. Although the discrepancy is fairly small for any single PScF, the superposition of many similar PScFs causes an undercorrection in the lead attenuation curve.

To account for these results, iterations greater than 7 will subtract 97% of the PScF maximum for iron and 105% of the PScF maximum for lead. The likely cause of these deviations is the forward peaking of the elastic scattering cross sections for these heavier nuclei. As stated previously, the Gaussian PScF fits are not particularly good in the tails regions because of the diffraction patterns produced by higher order (p-wave and above) scattering. In order to account for these features, a much more complex model using Legendre polynomials would be required. Such a model would be much more computationally intensive than the Gaussian model and will not be explored in this work.

7.1.2 Other Simulation Results

Now that the initial testing of the PSRA is complete, some additional simulation results will be presented in this section. These simulations will test the PSRA algorithm and also the modifications made based on the results in the previous section. The first simulation used two different shell thicknesses of material in order to test the how well the PSRA performs when applied to multiple thicknesses of material. The geometry used for this first simulation, titled C2C4, is depicted in Figure 7-10. It consists of half shells of graphite 2 and 4 MFP thick. The outer edge of the cylinder is located 40 cm from the detector array. The attenuation curves produced by the *ScatterSubtract* code are plotted in Figure 7-11. The corrected attenuation values show that the scatter subtraction yields near-perfect results. The corrected curve shows the proper horizontal profile in the two plateau regions and is indistinguishable from the *Direct* curve in most places. The scatter correction also results in a much higher contrast between the two plateau regions. The improved contrast will allow for the more accurate identification of features within the object being imaged.

The next scenario use slabs of two different thicknesses joined vertically along the plane connecting the source location and the horizontal center of the detector array. The two slabs have perpendicular thicknesses of 1 and 3 MFP. The slabs used in this scenario are composed of carbon. Figure 7-12 shows the geometry used for this scenario. The resulting attenuation curves for the C13S scenario are shown in Figure 7-13. The corrected attenuation curves for this simulation follow the *Direct* attenuation curves very well and only small deviations are visible. The scatter correction also increases the contrast between the two slabs.
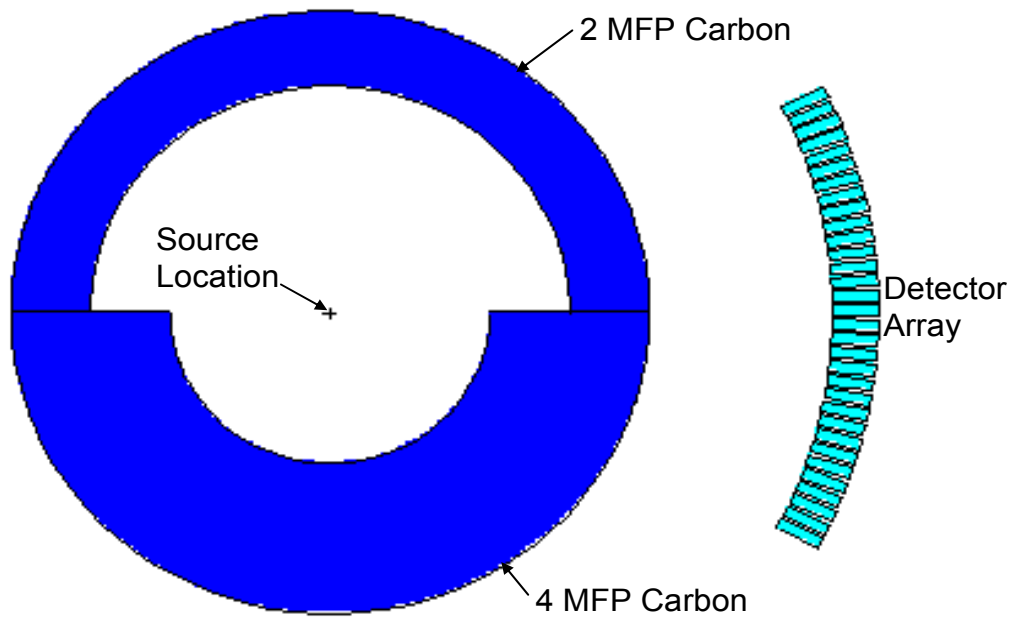
Figure 7-10. The geometry used to simulate two different thicknesses of object material. The object consists of two half shells of different thicknesses joined at the 0° and 180° positions. The source-to-detector distance in all simulations is 110 cm.
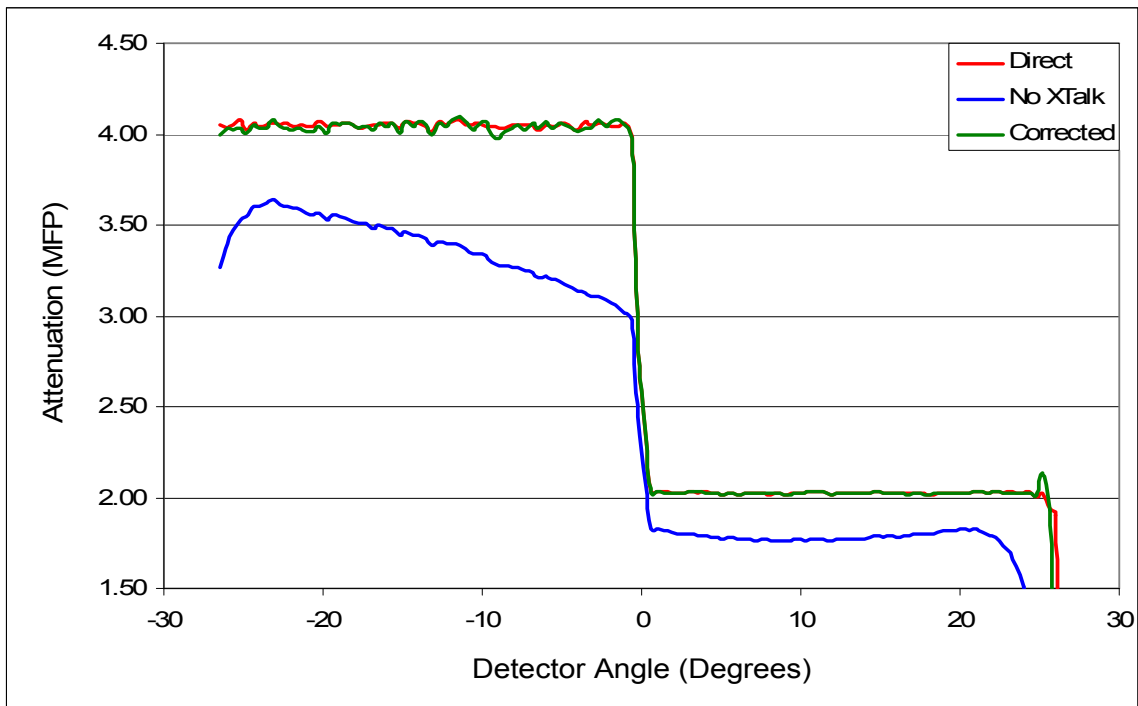
Figure 7-11. The attenuation curves for the C2C4 scenario.
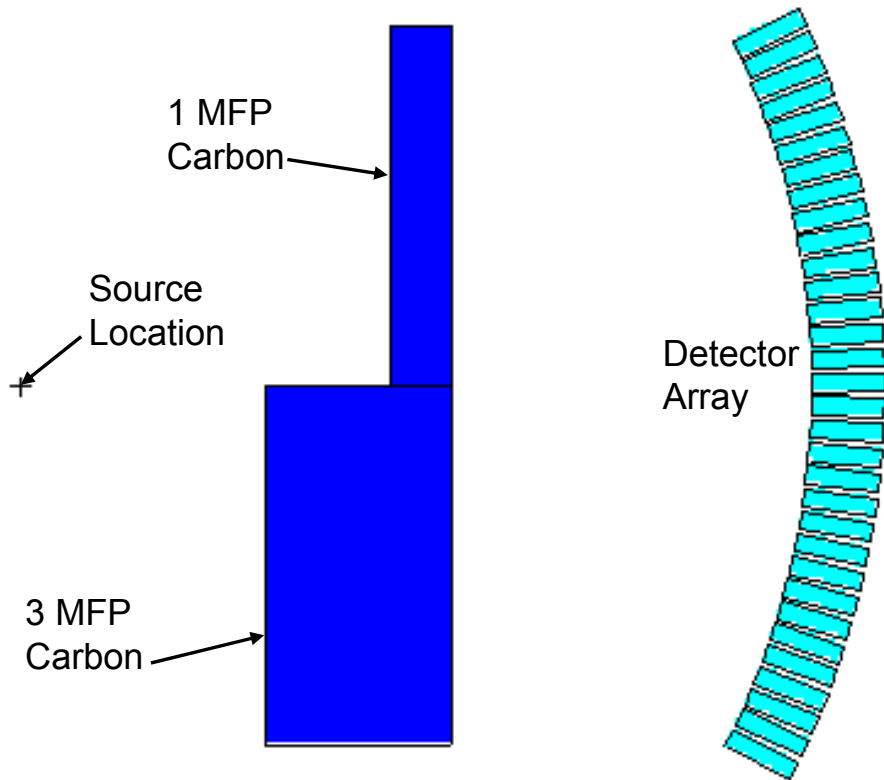
Figure 7-12. The geometry used for simulating measurements of slabs composed of two thicknesses of material. The configuration shown is the C13S scenario.
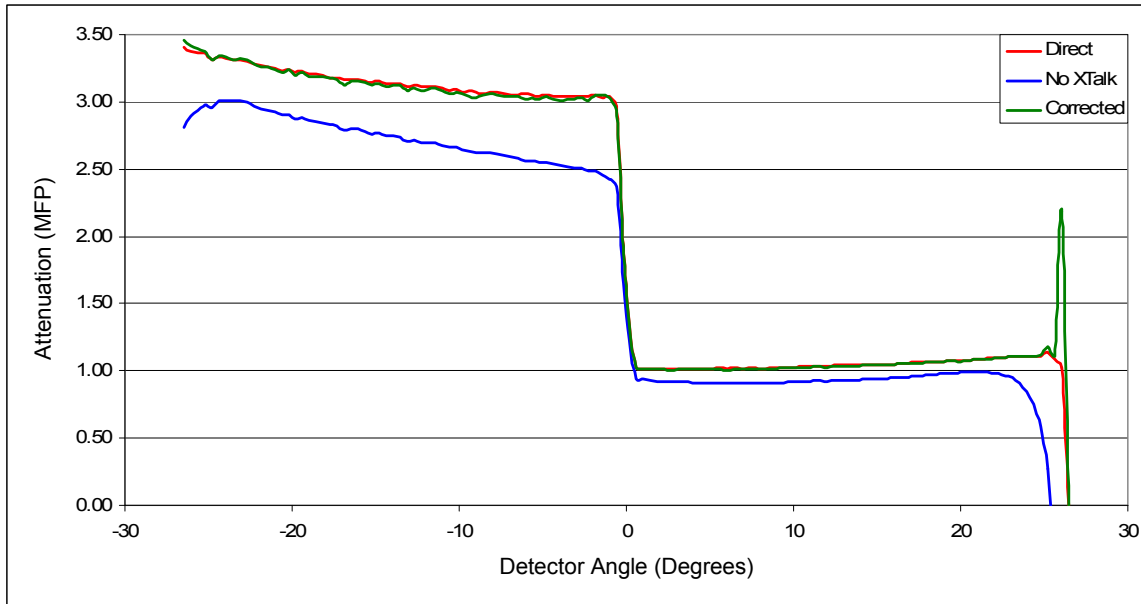
Figure 7-13. The attenuation curves for the C13S scenario.

Another scenario used for testing the PSRA on a slab geometry employs a step wedge design. The step wedge is a slab that increases its thickness in finite steps from one end to the other. The step wedge modeled here has six thicknesses of iron ranging between 1 and 3.5 MFP in steps of 0.5 MFP. Each step is 8 cm wide except for the first and last which are 34 cm wide. These wide outer steps extend the step wedge well beyond the horizontal extent of the DT pixels. The flat edge of the step is located 40 cm from the angular center of the detector array. The geometry of the step wedge scenario is depicted in Figure 7-14.

The attenuation curves for the step wedge scenario are plotted in Figure 7-15. The corrected attenuation values show excellent agreement with the *Direct* values except for a slight divergence at the two thickest steps.
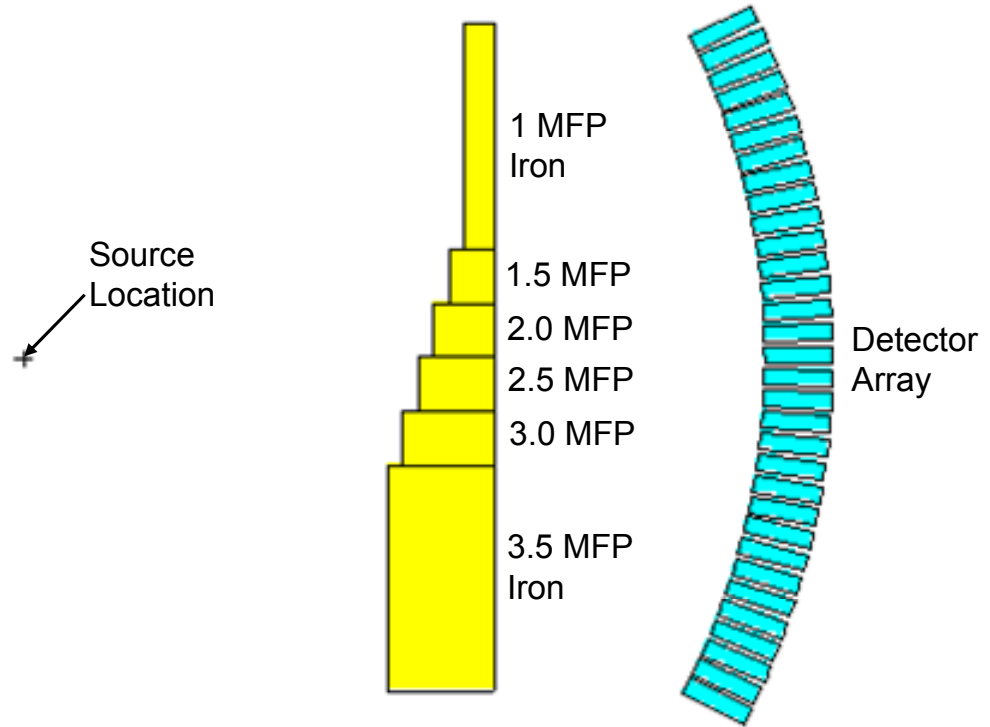
Figure 7-14. The geometry of the step wedge scenario. The source-to-detector distance is 110 cm.
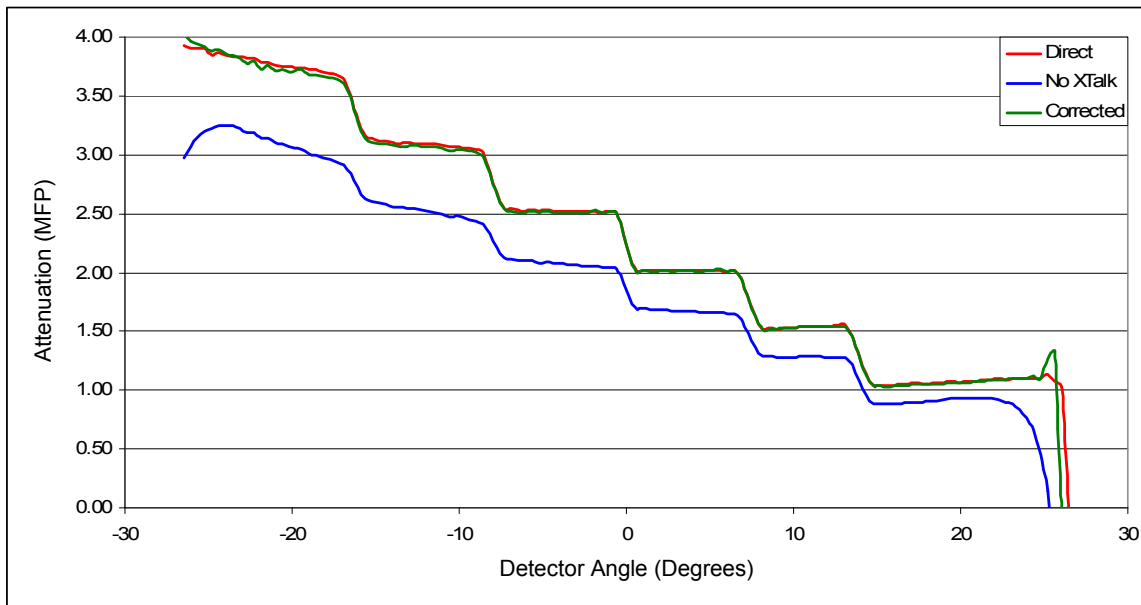
Figure 7-15. The attenuation curves for the Step Wedge scenario.

This scenario and the others presented in this section show that the PSRA can successfully correct the attenuation values for a slab geometry with a minor modification to account for the changing object-to-detector distance.

The next object geometry to test is cylindrical geometry. Unlike the PScF geometry, these cylindrical objects are not symmetric about the source location. This geometry is frequently encountered in NMIS imaging in the form of a drum or barrel. A barrel filled with steel beads, two iron pipes, a depleted uranium casting, and a polyethylene rod was modeled to simulate such an object. The object in this scenario is a based on the object shown in Figure 1-5. The barrel is located so that its center is exactly half way (55 cm) between the neutron source location and the center of the detector array. The barrel has thin steel walls approximately 1 mm thick and an outer radius of 17.88 cm. The geometry of the object and the detector array is presented in Figure 7-16. The dimensions and cross sections of the materials inside the barrel are listed in Table 7-1.

The attenuation curves for the barrel simulation are plotted in Figure 7-17. The PSRA does an excellent job of removing the scatter from the measured values. There is a slight deviation between the measured and *Direct* attenuation curves, but in general, the two follow each other very well. The scatter correction also greatly increases the contrast between the two iron pipes (the double humped regions at approximately ±10°) and between the DU casting (large peaks at approximately ±5°) and the surrounding regions of air. The presence of uranium and polyethylene in the object does not seem to negatively affect the results because they occupy a relatively small portion of the object.

162
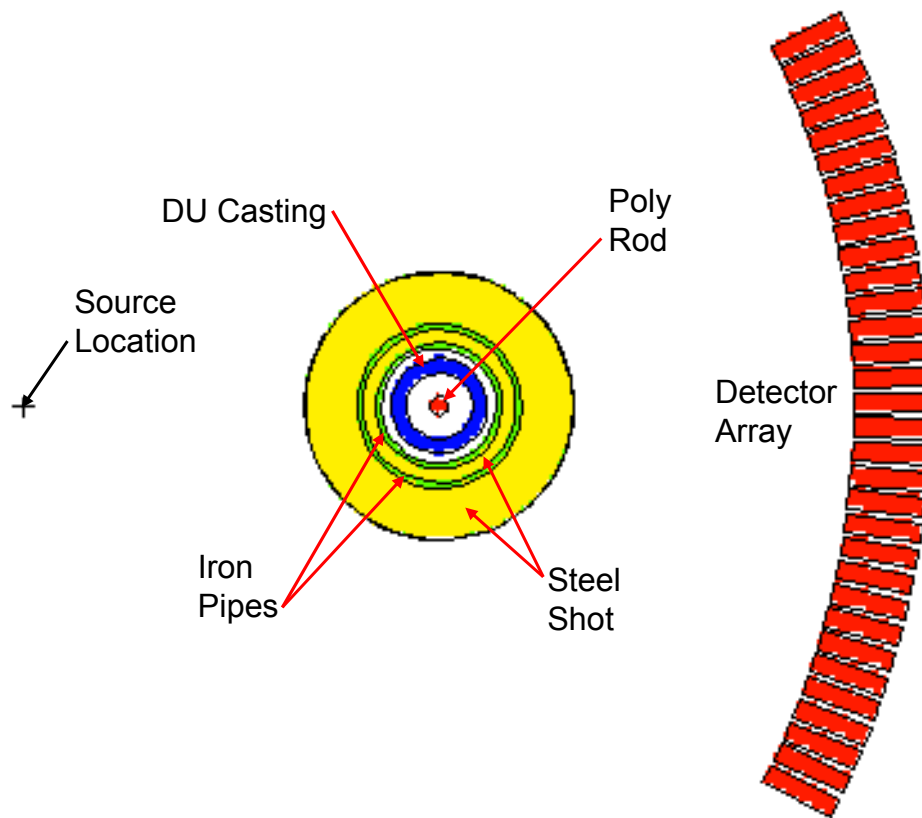
Figure 7-16. The geometry used for simulating the Barrel measurement. Dimensions for the barrel contents are listed in Table 7-1. The source-to-detector distance is 110 cm.

Table 7-1. The dimensions and cross sections of the materials inside of the barrel.

Shell	Material	Inside Diameter (cm)	Outside Diameter (cm)	Cross section (cm^{-1})
1	Polyethylene	0	2.54	0.11
2	Air	2.54	8.89	0
3	Depleted Uranium	8.89	12.70	0.28
4	Air	12.70	15.24	0
5	Iron Pipe	15.24	16.83	0.22
6	Steel Beads	16.83	20.32	0.13
7	Iron Pipe	20.32	21.91	0.22
8	Steel Beads	21.91	35.56	0.13
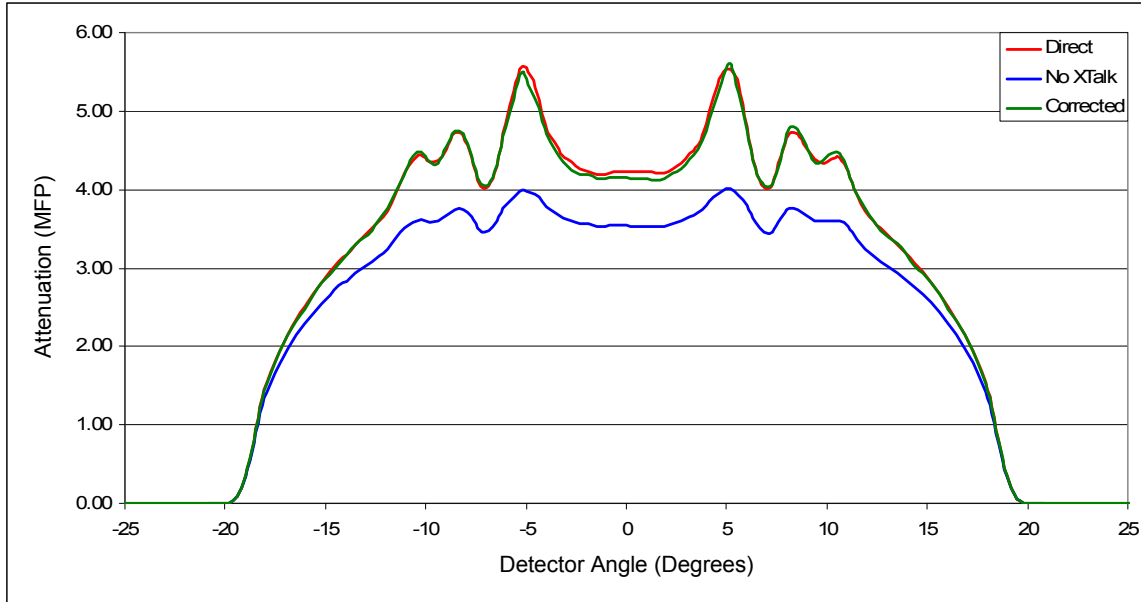
Figure 7-17. The attenuation curves for the barrel scenario.

Because the barrel and the objects inside are cylindrically symmetric, the 1D

attenuation curve can be converted to a 2D attenuation map. For NMIS imaging, the

FBPGUI program performs this task using a filtered back projection (FBP). For general

object geometries, the program requires a series of projections taken at different angles

through the object. For a cylindrically symmetric object, the *FBPGUI* code uses a single

projection and assumes that the same values are recorded all around the object. In order

to use this program, the void.out and object.out files produced by *ScatterSubtract* were

converted to the custom comma separated variable (.csv) files used by *FBPGUI* using a

small piece of Fortran-90 code. This code extracts the *Direct*, corrected, and *No XTalk*

(uncorrected) values and creates an *FBPGUI* input file for each.

The 2D reconstructions of the uncorrected (left) and corrected (right) values for

the barrel simulation are shown in Figure 7-18.
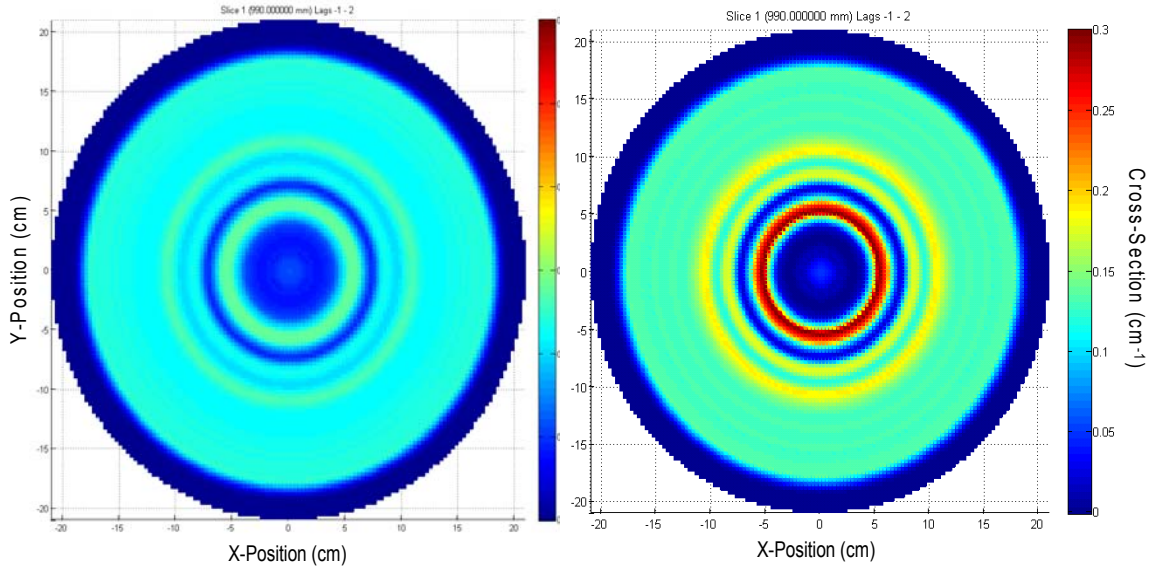
Figure 7-18. 2D Reconstructions of the barrel using uncorrected (left) and corrected (right) attenuation data.

The input files contain the location of the center of the object so the reconstruction shows the x and y positions of the various centers, in cm, relative to the center of the object. The colors show the macroscopic cross section of the various regions in units of cm^{-1}. The corrected image shows a much higher contrast between the steel beads, the iron pipes, and the depleted uranium casting. The cross section values for the iron pipes and uranium in the corrected plot are much closer to their true values of 0.22 and 0.29 than in the imaged produced by the uncorrected data. Since the goals of NMIS imaging measurements are to identify the shape and composition of the internal structure of the object, the closer the reconstructed cross section values match the real ones the more likely it is that the materials will be correctly identified.

The reconstructions of the measured and corrected data on 3D axes are presented in Figure 7-19 and Figure 7-20. Here, the cross section values are represented by both the color and the height of the various regions.
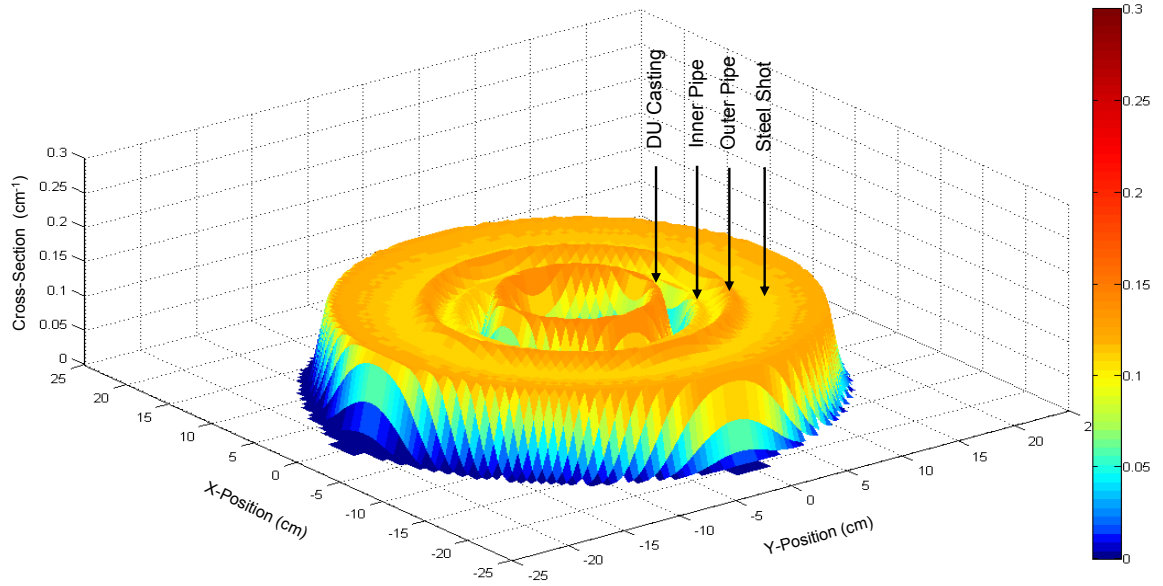
Figure 7-19. A 3D plot of the barrel reconstruction using the uncorrected attenuation values. The z-axis and color mapping show the macroscopic cross section in each region.
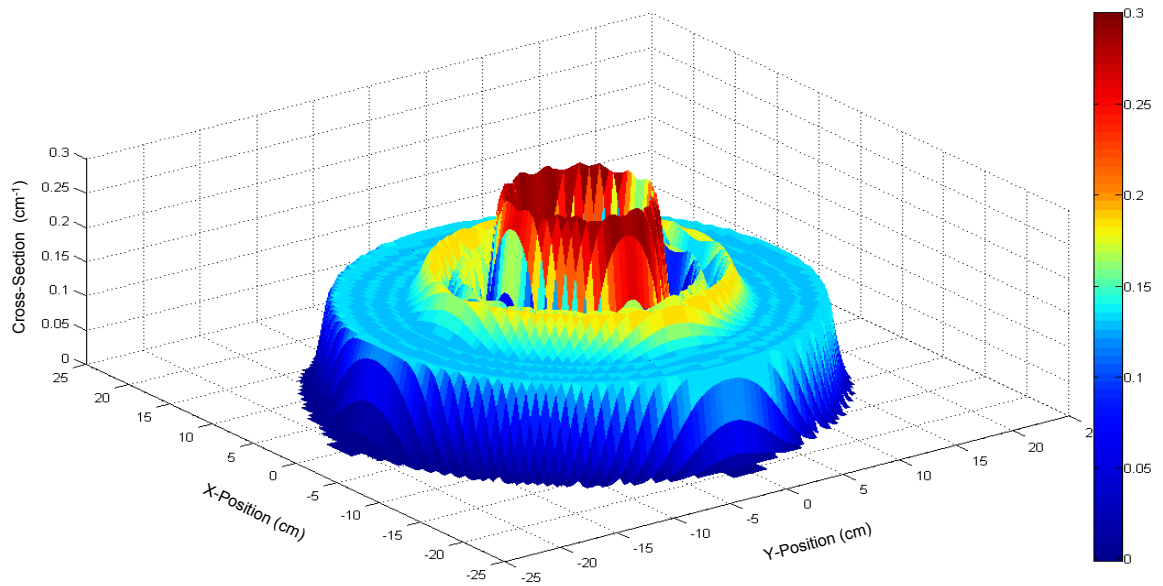
Figure 7-20. A 3D plot of the barrel reconstruction using the corrected attenuation values. The z-axis and color mapping show the macroscopic cross section in each region.

166

The positions of the DU casting, the two iron pipes, and the steel shot in the image are noted in Figure 7-19. In the uncorrected data, the cross section of the inner pipe shows a much lower value than the outer ones. The apparent cross section of the DU casting shows up at about the same level as the outer pipe. There is also a distinct concavity in the outer layer of steel beads as the cross section in that region drops as it nears the outer iron pipe. The reconstruction in Figure 7-20 shows that these errors are almost entirely corrected by the PSRA. The steel shot appears nearly flat from inside to out, the inner and outer pipes display approximately the same cross section, and the casting has a much higher value than any of the other materials.

This scenario shows that the PSRA can accurately remove the scattering from a cylindrical object. The corrected data produces a much more accurate mapping of the internal cross sections than the uncorrected data. The corrected data also shows a much higher contrast between the different materials. This will minimize the probability that regions of materials will mistakenly be combined into a single cell.

In Chapter 6, two averaged materials were defined and PScFGE coefficients for these materials were coded into the *ScatterSubtract* program. These two averaged materials were designated "material 5" (an average of iron and lead) and "material 6" (an average of carbon, iron, and lead). The purpose of these two averaged materials is to test how well the PSRA can perform when the knowledge of the material composition of the object is less than perfect. The attenuation curves using the material 5 coefficients in the *ScatterSubtract* program to correct the Iron76 and Lead32 scenarios are shown in Figure 7-21 and Figure 7-22. The use of the averaged material values overcorrects the scenario with iron and undercorrects the one with lead.
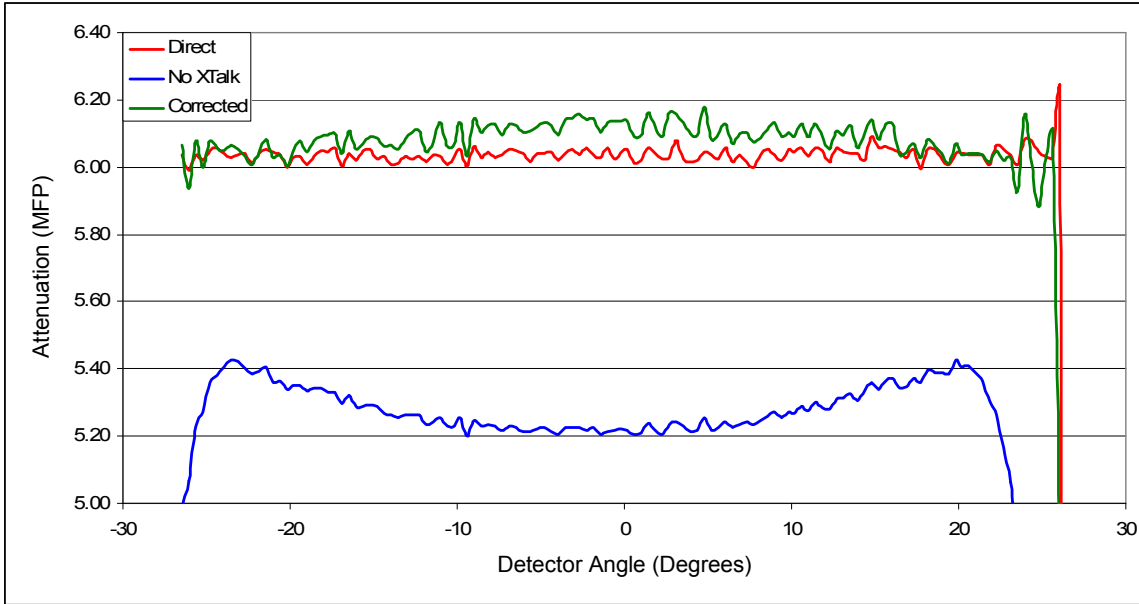
Figure 7-21. The attenuation curves for the Iron76 scenario. The PSRA used material 5 (average of iron and lead) PScF values to correct the scatter.
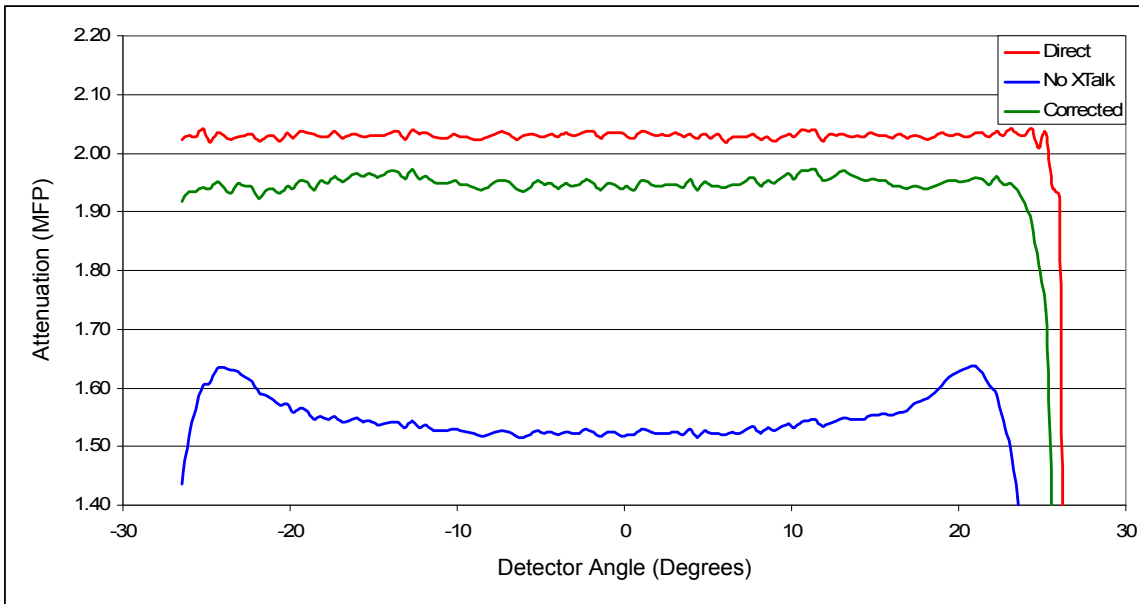
Figure 7-22. The attenuation curves for the Lead32 scenario. The PSRA used material 5 (average of iron and lead) PScF values to correct the scatter.

The corrected values diverge fairly significantly from the *Direct* values; however, the divergence of the scatter corrected values is much smaller than that of the uncorrected values and the overall shape is much more accurate.

The simulated measurement scenarios presented thus far have all consisted of only a single material in order to test the function of the PSRA. While this may accurately represent some real world scenarios (e.g., a radioactive source surrounded by a large quantity of homogeneous shielding material) most objects will be composed of more than one material. The measurement of cylindrically symmetric shells consisting of two materials was simulated in order to test how well the PSRA performs when the scatter correction for a single material is applied. Figure 7-23 shows the geometry used for one of this scenarios, the PbPo scenario, which consists of a 2 MFP layer of lead inside of a 2 MFP layer of polyethylene.

The attenuation curves for the PbPo scenario generated by using the lead and polyethylene PScF values, respectively, to remove the scatter are plotted in Figure 7-24 and Figure 7-25. Using the lead coefficients results in a significant overcorrection of the attenuation values and using polyethylene results in a significant undercorrection. The shapes of the corrected attenuation curves also differ markedly from the horizontal shape of the *Direct* curve. Overall, the use of the PScF values for a single material produces poor results for this scenario. In order to make the PSRA useful for a wide range of possible scenarios, this problem needs to be addressed. Section 7.3 will revisit this issue for the purpose of developing a method which can use the coefficients for multiple materials when applying the PSRA.

169
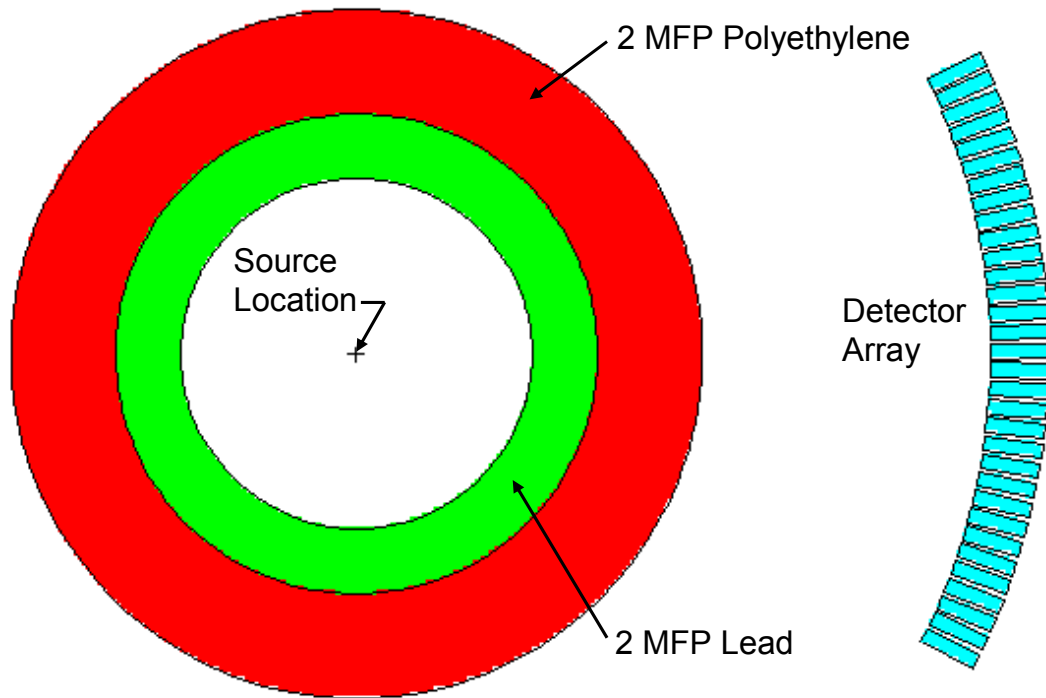
Figure 7-23. The geometry used for simulating measurements of cylindrically symmetric shells consisting of more than one material. The configuration shown here is the PbPo scenario. The source-to-detector distance in all simulations is 110 cm.
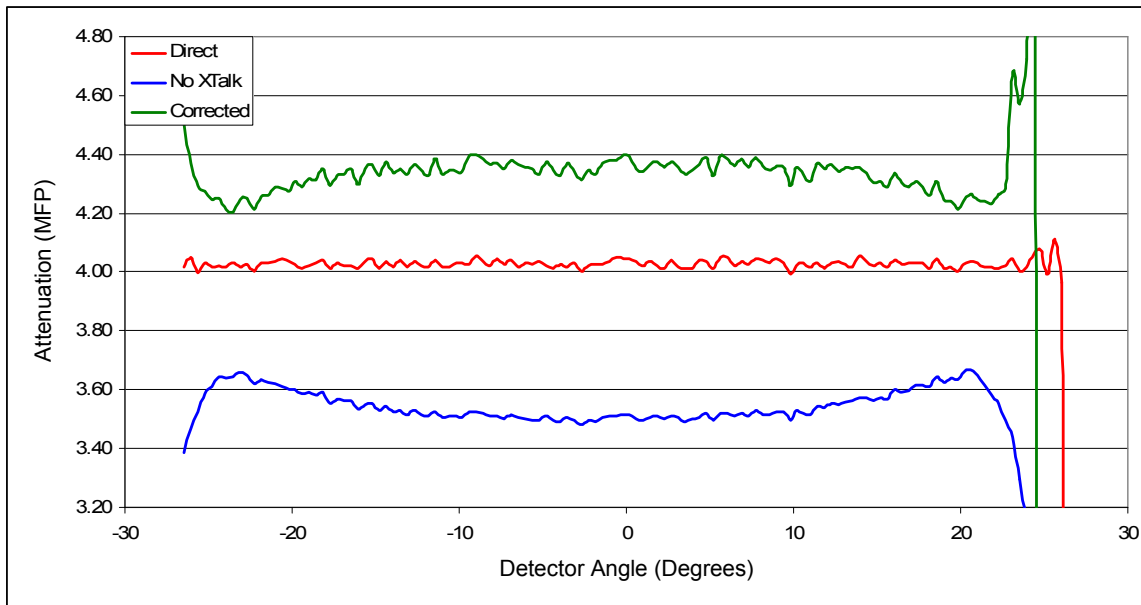
Figure 7-24. The attenuation curves for the PbPo scenario. The PSRA used lead PScF values to correct the scatter.
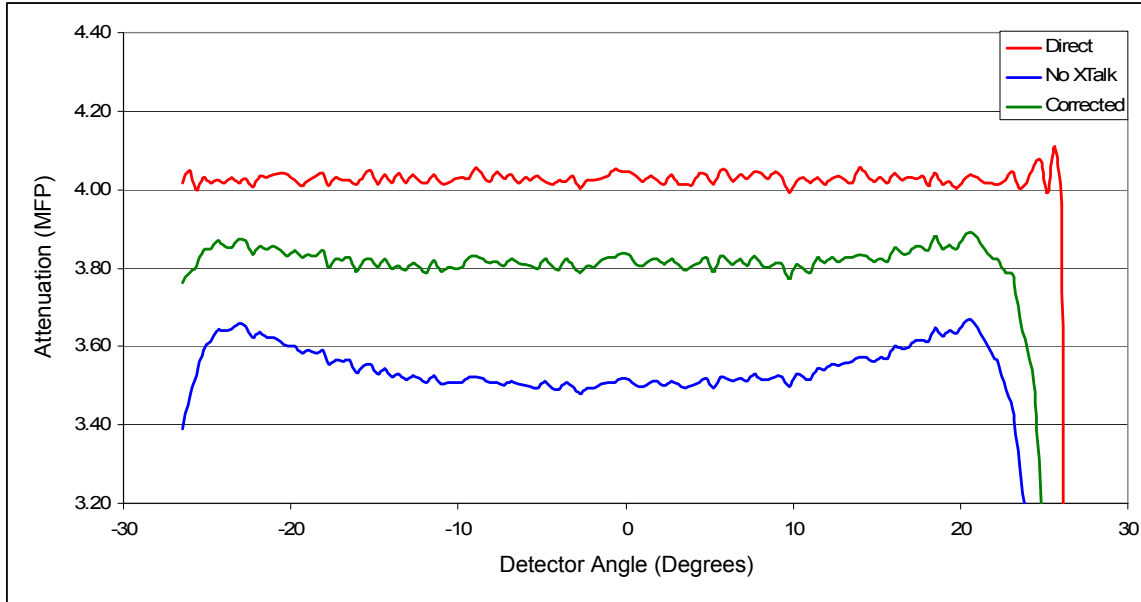
Figure 7-25. The attenuation curves for the PbPo scenario. The PSRA used polyethylene PScF values to correct the scatter.

7.1.3 A Summary of the PSRA Performance with Simulated Measurements

With a couple of exceptions (multiple materials and averaged materials), the scatter removal algorithm produced excellent results when applied to simulated scenarios. In Section 7.1.1, the PSRA produced near perfect results when removing the inter-array scatter from the void measurement. Since all of the simulation scenarios used the same void simulation, the performance of the PSRA when removing scatter from the object measurement translates directly into the attenuation profile. For this reason, the goodness of the scatter correction has been tested to this point by comparing the corrected and uncorrected attenuation curves to the *Direct* attenuation curve which is computed using only directly transmitted neutrons.

In this final section reporting on the simulated results, the corrected and uncorrected attenuation values will be compared to the *Direct* data using a more quantitative

171

methodology. As mentioned in Section 6.2, the *ScatterSubtract* program performs a χ^2 goodness of fit test comparing the corrected and uncorrected attenuation curves to the *Direct* one. These values are recorded in the ChiSq.out text file. The χ^2 values for each of the scenarios tested are listed in Table 7-2. The final column shows the ratio of the uncorrected value to the corrected value as a quantitative measure of the improvement realized by the scatter subtraction algorithm.

The results in Table 7-2 show that with the exception of a single scenario (L1P3 corrected as lead), the PSRA produced a more accurate attenuation curve than the uncorrected values. The χ^2 values represent the total number of variances (uncertainty squared) between the two curves being compared. Since both the corrected and uncorrected tests used the variance of the direct data in the denominator, the ratio of the two is the ratio of the sums of their squared errors. The square root of the ratio is, therefore, the ratio of the average deviation, e.g., a ratio of 100 indicates that the corrected values deviate from the *Direct* values 10 times less on average than the uncorrected values.

The scenarios using objects composed of a single material, whose attenuation was corrected using the material-specific PScFGE coefficients, all resulted in ratios of greater than 60, which corresponds to a decrease in the average deviation by a factor of 7.75 or greater. This indicates that the PSRA methodology is sound and that its use can greatly improves the accuracy of the attenuation values. The use of the material-averaged PScFGE coefficients produced ratios which mostly lay in the 20 to 40 range.   These values still represent a fairly significant improvement in the accuracy of the attenuation curve, but the degree of accuracy depends on the actual material of the object.

172

Table 7-2. The χ^2 goodness of fit results of the uncorrected and corrected attenuation curves for the simulation scenarios. The ratio column is the ratio of the uncorrected value to the corrected value.

Scenario	Uncorrected χ^2	Corrected χ^2	Ratio
Simple Cylindrically Symmetric Objects			
Poly63	6.579E+04	7.778E+01	845.8
Carb61	8.382E+04	1.347E+03	62.20
Iron76	2.076E+05	2.995E+02	693.1
Lead32	1.032E+06	1.108E+03	931.5
Cylindrically Symmetric Objects with More than One Material Thickness			
C2C4	2.980E+05	2.366E+02	1259
L2L4	6.055E+05	1.939E+03	312.2
Objects Corrected Using Material Averaged PScF Values			
Iron76 (Mat. 5)	2.076E+05	1.438E+03	144.3
Lead32 (Mat. 5)	1.032E+06	2.935E+04	35.19
Carb61 (Mat. 6)	8.382E+04	7.182E+03	11.67
Iron76 (Mat. 6)	2.076E+05	1.200E+04	17.29
Lead32 (Mat. 6)	1.032E+06	1.704E+04	60.59
C2C4 (Mat. 6)	2.980E+05	4.111E+04	7.250
L2L4 (Mat. 6)	6.055E+05	2.024E+04	29.91
Objects Composed of Multiple Materials			
PbPo (as Lead)	1.571E+05	7.682E+04	2.045
PbPo (as Poly)	1.571E+05	2.942E+04	5.339
PoPb (as Lead)	1.842E+05	8.432E+04	2.186
PoPb (as Poly)	1.842E+05	4.257E+04	4.328
LPoL (as Lead)	6.453E+05	2.527E+05	2.554
LPoL (as Poly)	6.453E+05	1.348E+05	4.784
L1P3 (as Lead)	9.917E+04	1.470E+05	0.6744
L1P3 (as Poly)	9.917E+04	6.876E+03	14.42
L3P1 (as Lead)	2.373E+05	3.074E+04	7.721
L3P1 (as Poly)	2.373E+05	7.289E+04	3.256
Objects with a Slab Geometry			
CaSl	2.180E+05	3.071E+03	70.99
LeSl	5.125E+05	2.423E+03	211.5
C13S	2.125E+05	6.337E+02	335.4
L13S	4.559E+05	8.530E+02	534.5
Step Wedge	4.512E+05	8.661E+02	521.0
Objects With a Cylindrical Geometry			
PoCy	7.338E+04	7.816E+02	93.89
FeCy	8.307E+04	7.304E+02	113.7
Barrel	1.782E+05	9.586E+02	185.9

Finally, most of the objects composed of multiple materials resulted in ratios below 10. While all but one of these scenarios did produce slightly better attenuation curves, they might not produce better results when determining the material composition of an object. Because the uncorrected attenuation values are systematically low, they provide a lower bound to material cross sections when the object is reconstructed. If, for example, the uncorrected cross section values indicate a region with a cross section of 0.13 cm^{-1}, polyethylene (0.11 cm^{-1}) could almost certainly be ruled out. If the scatter correction produces a modest overcorrection of the attenuation values, polyethylene might be incorrectly ruled out. Section 7.3 will examine ways in which the scatter correction might be modified to improve its accuracy in a wider range of scenarios.

7.2 Experimental Testing and Results

Now that the PSRA methodology has been tested using simulated imaging measurements, the next step is to test it with experimental data. One problematic aspect of NMIS measurements is that due to the constant research and development aimed at improving its performance, the configuration is in a constant state of change. These configuration changes include items such as the radius of the detector arm, the alpha detector PMT and light guide, the configuration of the electronics, and updates to the NMIS software packages. Because the PSRA is based on a given configuration, it is only applicable to measurements made with approximately the same setup. Because of this restriction, only three measurements are available for testing. While a greater number would be preferred, taking new data with this configuration is not feasible in the short term because of a planned series of measurements which have made the 110 cm radius

detector arm unavailable. Despite this, the three measurements available should be sufficient to test the PSRA on measured data and determine if it is a viable option for removing the scatter from experimental imaging data.

All experimental measurements were made were made using the 110 cm radius detector arm. The arm has 32 2.54 × 2.54 × 10.16 cm plastic scintillators with an angular separation between adjacent detector centers of approximately 1.67°. The API-120 was operated with an accelerator voltage of 87 kV and a current of 60 μA. At these settings, the total neutron output is approximately 4×10^7 neutrons per second produced isotropically. A bias of 1100 V was applied to the H8500 PMT attached to the alpha detector. At this setting, each of the eight pixels counted approximately 30,000 alpha particles per second.

7.2.1 Imaging Detector Efficiency

Before any imaging measurements were performed, the neutron energy threshold of the imaging detectors was tested using a ^{252}Cf spontaneous fission source. The ^{252}Cf source is mounted in an ion chamber which detects the heavy nuclides produced by each fission. This signal is used to measure the time-of-flight of the fission neutrons to the imaging detectors. Because the time-of-flight is a direct function of energy and the energy spectrum of ^{252}Cf is well documented, this information can be used to determine the detector efficiency at various neutron energy levels. The efficiency is reported as a percentage of neutrons that hit the front face of the detector which generate a count. The lower energy limit where the neutron efficiency falls to zero is the detector neutron energy threshold.

The fitted efficiency curves for the 32 imaging detectors produced by the Integrated Sata and Analysis Software (IDAS) are plotted in Figure 7-26. This data in this plot shows that the detector thresholds are all approximately 1.5 MeV. Because the PScF and the ISF fits were calculated assuming a threshold of 1.0 MeV, an attempt was made to lower the detector thresholds to this level by increasing the PMT voltage and lowering the constant fraction discriminator (CFD) threshold. However, due to the fact that the 32 detectors share a single four channel high voltage power supply, this failed. Either the maximum allowable PMT voltage or the minimum threshold was reached before the detector thresholds reached 1.0 MeV.

In order to proceed, the experiments were conducted using a detector threshold of 1.5 MeV. In order to make the PSRA applicable to these measurements, it was necessary to calculate new PScF and ISF parameters for this threshold. This was relatively easy to change using the results of the PScF simulations. The .dat files were post-processed using a 1.5 MeV threshold, and the PScF and ISF coefficients were calculated using the method described in Chapters 5 and 6. The resulting ISF coefficients for 1.5 MeV are

A = 0.03117959; B = 0.004329379; S = 1.434710°; and T = 2.981670°.

The PScF coefficients for polyethylene, carbon, iron, and lead are listed in Table 7-3 and Table 7-4. Because the averaged material coefficients were not tested with the experimental data, they were not calculated for 1.5 MeV. These coefficients were entered into the *ScatterSubtract* code and the program was recompiled for use with the experimental data.
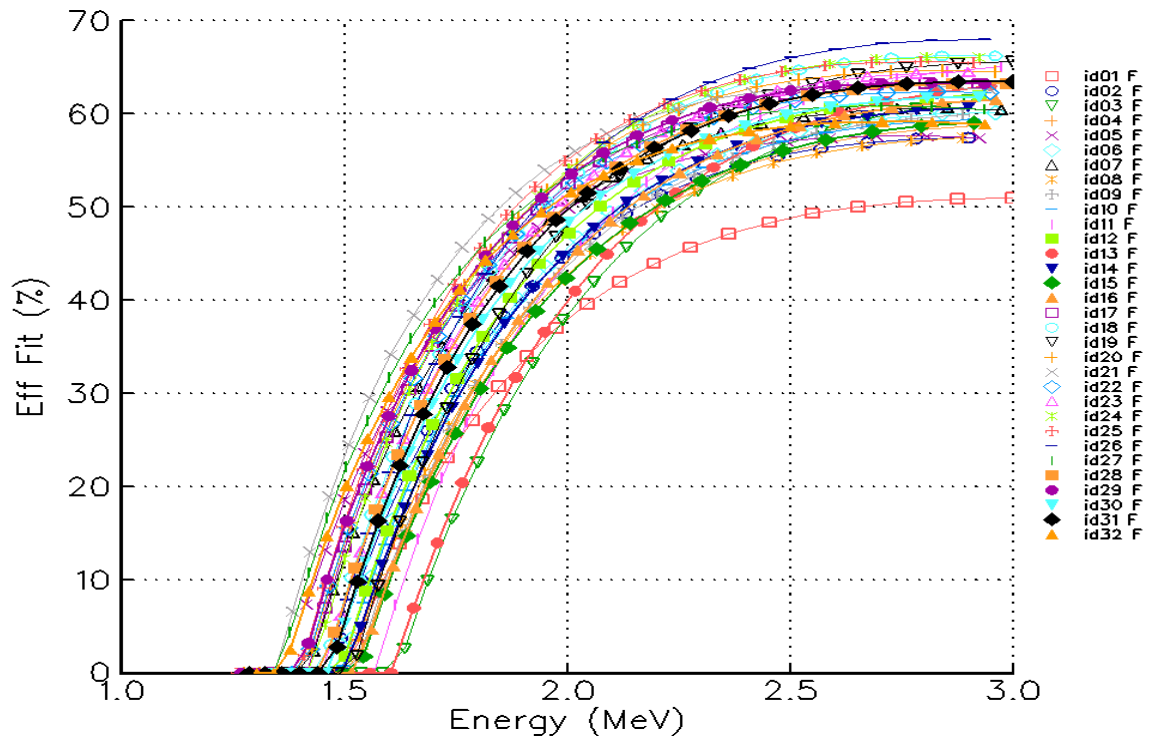
Figure 7-26. The fitted detector efficiency curves for the 32 NMIS imaging detectors generated using a ^{252}Cf spontaneous fission source.

Table 7-3. The 1.5 MeV Coefficients for the Maximum PScFGE.

Material	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
Poly	-1.970022E+00	-7.738703E-02	5.142299E-04	-1.708127E-06	1.281804E+00	1.982459E-02	-2.419642E-03	0
Carb	-2.032200E+00	-7.164776E-02	4.372984E-04	-1.288139E-06	1.113186E+00	1.389208E-02	-1.996984E-03	0
Iron	-1.579900E+00	-8.245448E-02	6.539626E-04	-2.349230E-06	1.034258E+00	7.107147E-03	-2.915926E-03	1.579430E-05
Lead	-1.147211E+00	-6.694131E-02	4.203523E-04	-1.279453E-06	1.204600E+00	2.094446E-02	-2.373614E-03	0

Table 7-4. The 1.5 MeV Coefficients for the Standard Deviation PScFGE.

Material	b_0	b1	b_2	b_3	b_4	b_5
Poly	3.993199E+00	4.843557E-02	1.056716E-03	3.282769E-01	1.245424E-02	7.561633E-03
Carb	1.928307E+00	1.338948E-01	4.388464E-04	6.694405E-01	2.914982E-02	4.600967E-03
Iron	1.128059E+00	1.220379E-01	0	5.138568E-01	9.921068E-03	3.294349E-03
Lead	-3.270967E-01	1.076363E-01	-1.807323E-04	5.710022E-01	5.119608E-02	6.383550E-04

7.2.2 Homogeneous Slabs

The first two imaging measurements available are homogeneous slabs of material similar

to those simulated in Section 7.1.6. The first slab measurement used a polyethylene slab

with dimensions 81.3 cm wide × 50.8 cm high × 11.8 cm thick. The slab was composed

of three identical sheets of polyethylene. The second measurement used a slab of

graphite measuring 45.7 cm wide × 45.7 cm high × 7.62 cm thick. Both slabs were

placed perpendicular to the line between the DT source location and the center of the

imaging detector array with an object-to-detector distance of 60 cm. Each measurement

consisted of four subsamples. Each subsample was measured for 10 minutes, resulting in

a total measurement time of 40 minutes. The void measurement used for computing

attenuation values consisted of four subsamples of five minutes each.

The attenuation curves for the polyethylene slab measurement are plotted in

Figure 7-27. The *Direct* curve was calculated by simulating the scenario with *MCNP-*

PoliMi. A quick calculation using the 14.1 MeV cross section value for polyethylene

predicts a thickness of 1.30 MFP through the center of the slab, which agrees well with

the *Direct* attenuation curve. The scatter correction produces an attenuation curve that is

significantly closer to the *Direct* values than the measured data; however, the corrected

values are still substantially under corrected.

The attenuation curves for the graphite slab measurement are shown in Figure

7-28. As with the previous measurement, the scatter correction produces a more accurate

attenuation curve, but the corrected values still fall well below the *Direct* curve.
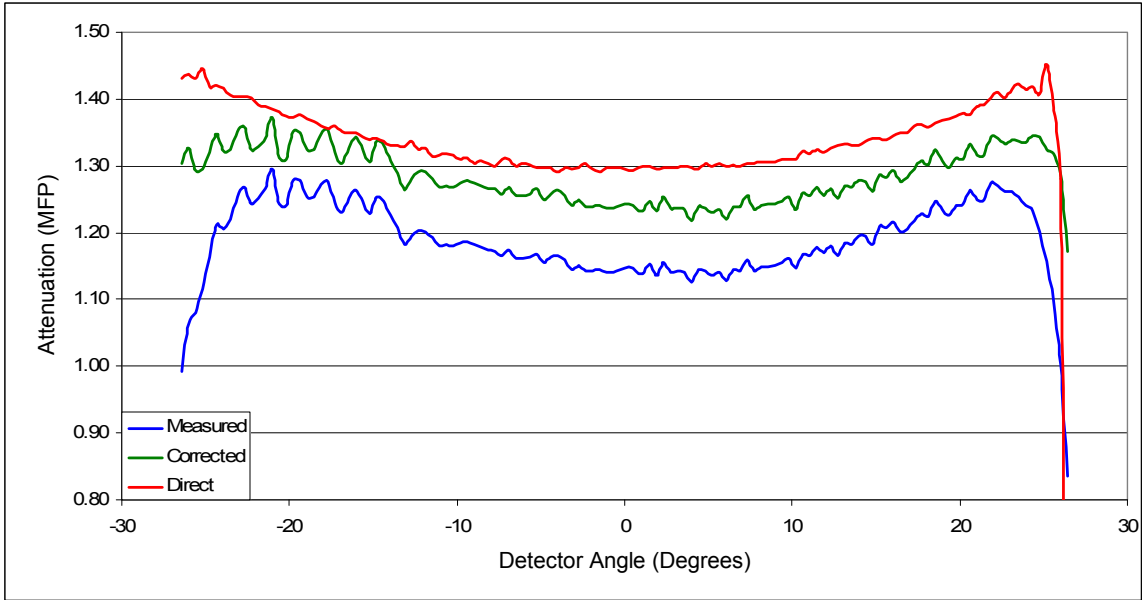
Figure 7-27. The attenuation curves for the polyethylene slab measurement. The *Direct* curve was generated by modeling the measurement with *MCNP-PoliMi*.
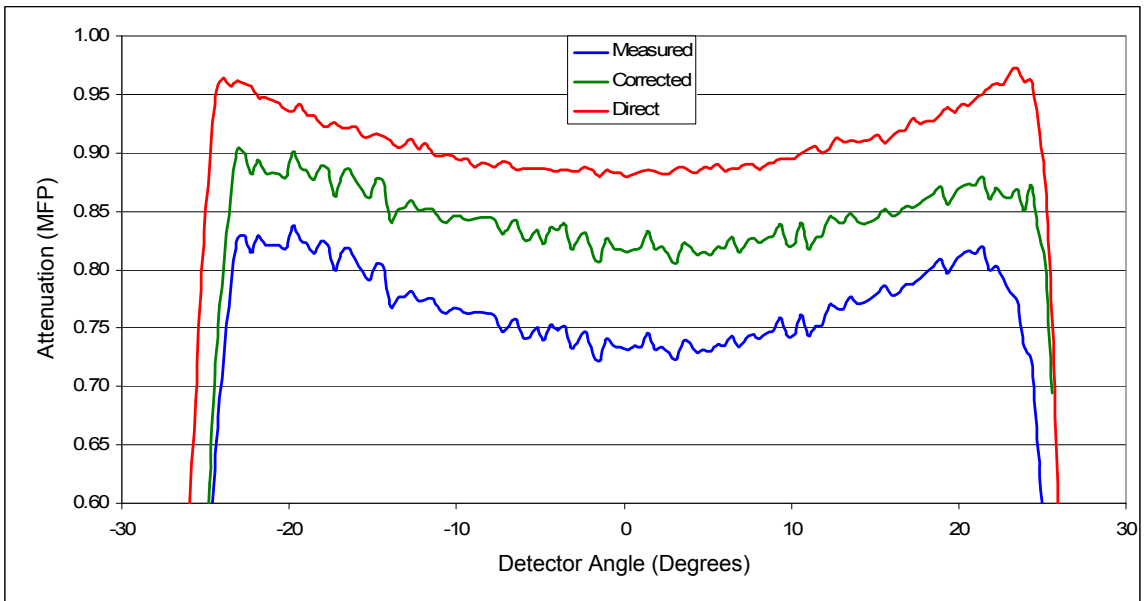
Figure 7-28. The attenuation curves for the graphite slab measurement. The *Direct* curve was generated by modeling the measurement with *MCNP-PoliMi*.

180

7.2.3 The Barrel Measurement

The final experimental imaging measurement uses the barrel configuration that was modeled in Section 7.1.2. The geometry of this measurement is identical to that simulation with the exception of the location of the barrel center. The barrel center was located at 58.3 cm from the center of the detector array.

The barrel measurement consisted of four subsamples of 20 minutes each and the void measurement consisted of four subsamples of 5 minutes each. The attenuation curves for the barrel measurement are plotted in Figure 7-29. As with the two slab measurements, the corrected values are much closer to the *Direct* curve than the measured values, but they still represent a substantial undercorrection. This systematic undercorrection, which was observed in all three experimental measurements, will be discussed in Section 7.2.4.

2D image reconstructions of the uncorrected and corrected attenuation data, respectively, are shown in Figure 7-30 and Figure 7-31. Both figures show the reconstruction using the *Direct* data (right) for reference. All plots use the same color mapping for comparison purposes. The uncorrected data shows almost no contrast between the steel beads and the iron pipes. With the exception of the two regions of air, there is no contrast between different materials. In the plot of the reconstructed data, the outer iron pipe is clearly visible and there is good contrast between it and the surrounding beads. The DU casting shows a higher cross section value than either the beads or the iron pipe, which increases the likelihood that its material would be identified correctly. Due to its position, the inner pipe is difficult to detect and is not visible in either the corrected or uncorrected reconstruction.

181
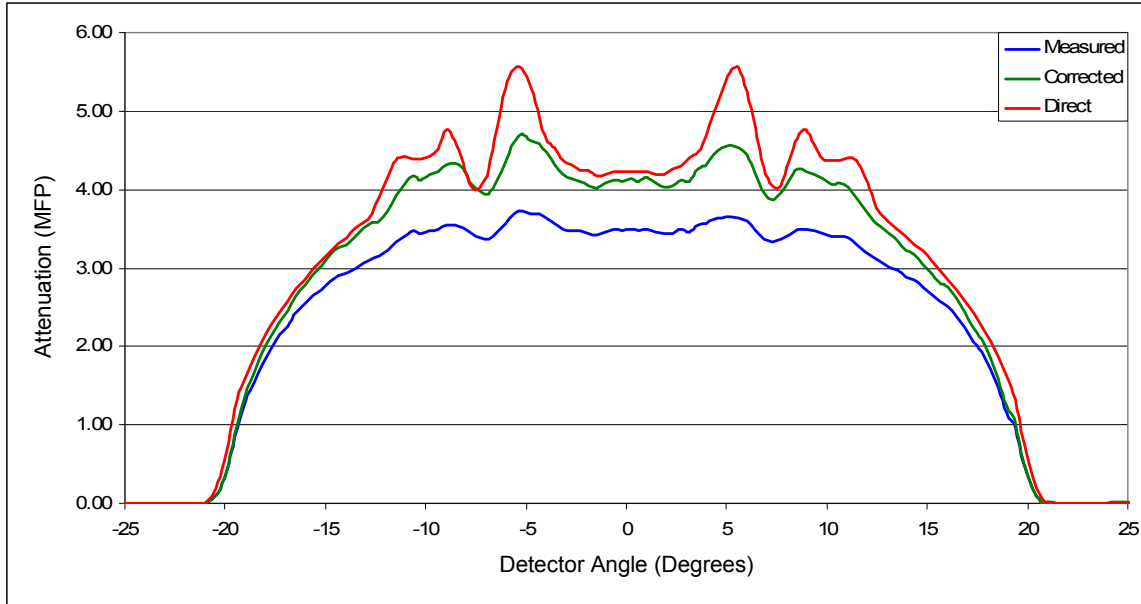
Figure 7-29. The attenuation curves for the barrel slab measurement. The *Direct* curve was generated by modeling the measurement with *MCNP-PoliMi*.
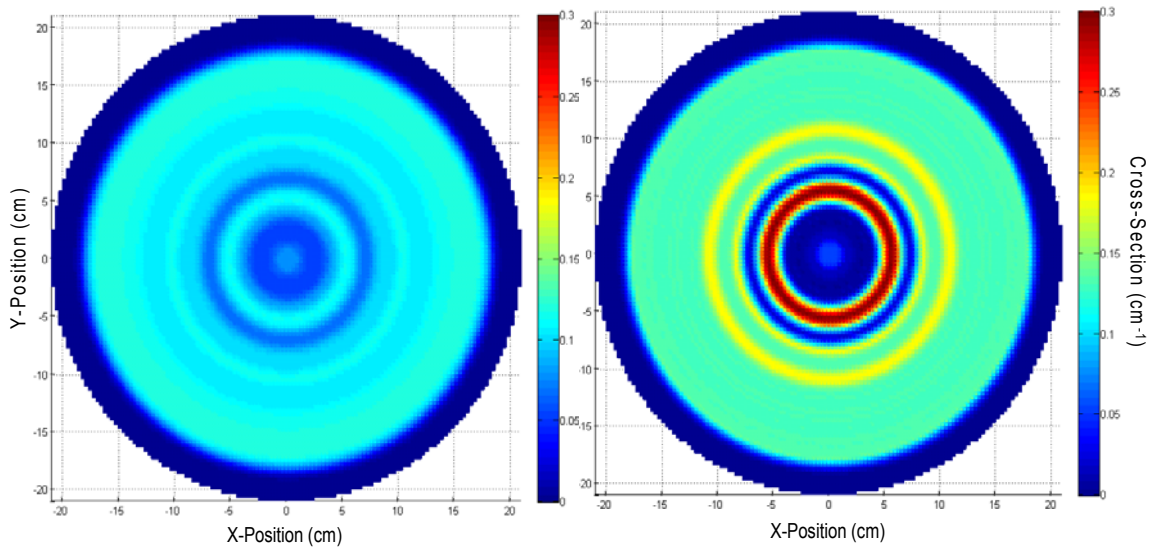
Figure 7-30. Filtered back projections of the barrel measurement generated using the measured (left) and *Direct* (right) data.
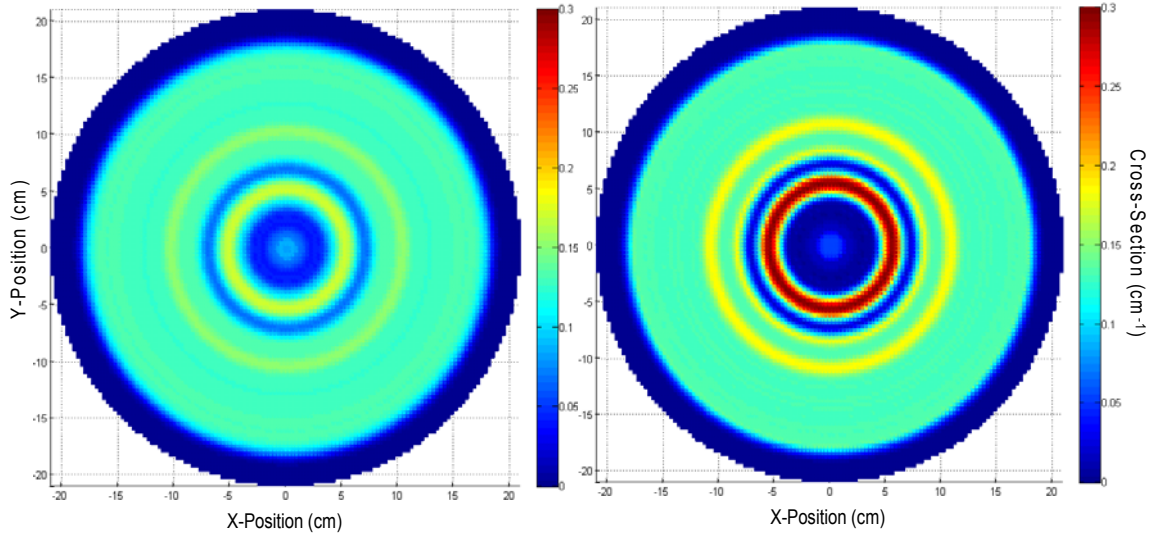
Figure 7-31. Filtered back projections of the barrel measurement generated using the corrected (left) and *Direct* (right) data.

A 3D view of the image reconstruction for the uncorrected data is shown in Figure 7-32 where the z-axis represents the cross section of each region. The color map scale has been changed from that in Figure 7-30 to enhance the contrast between materials. This plot shows that the reconstructed attenuation value for the uranium is slightly lower than that of the outer pipe or the outer layer of steel beads. The outer layer of beads also shows a distinct concavity and the inner pipe is not visible at all. A 3D view of the reconstruction using the corrected data is shown in Figure 7-33. Several improvements over the uncorrected view are visible. First, the DU casting now shows a higher cross section than the rest of the material. The outer pipe is higher than any of the surrounding beads, and the profile of the beads is almost flat. In this view, the inner pipe is just visible as a slight rise at the inner edge of the inner layer of beads.
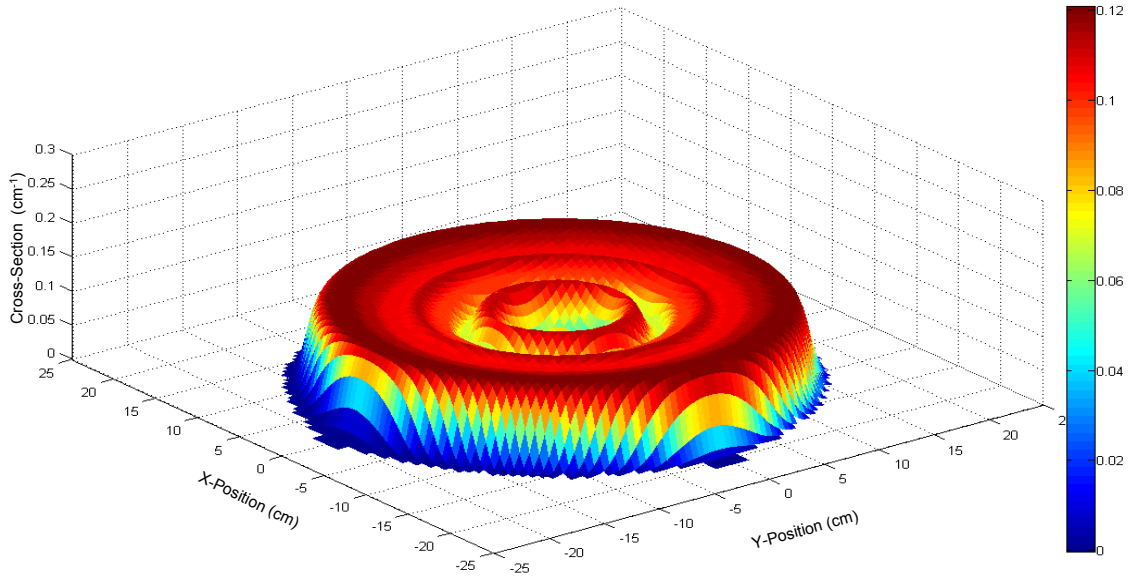
Figure 7-32. A 3D view of the barrel measurement generated using the measured data. The z-axis and color mapping indicate the estimated material cross sections.
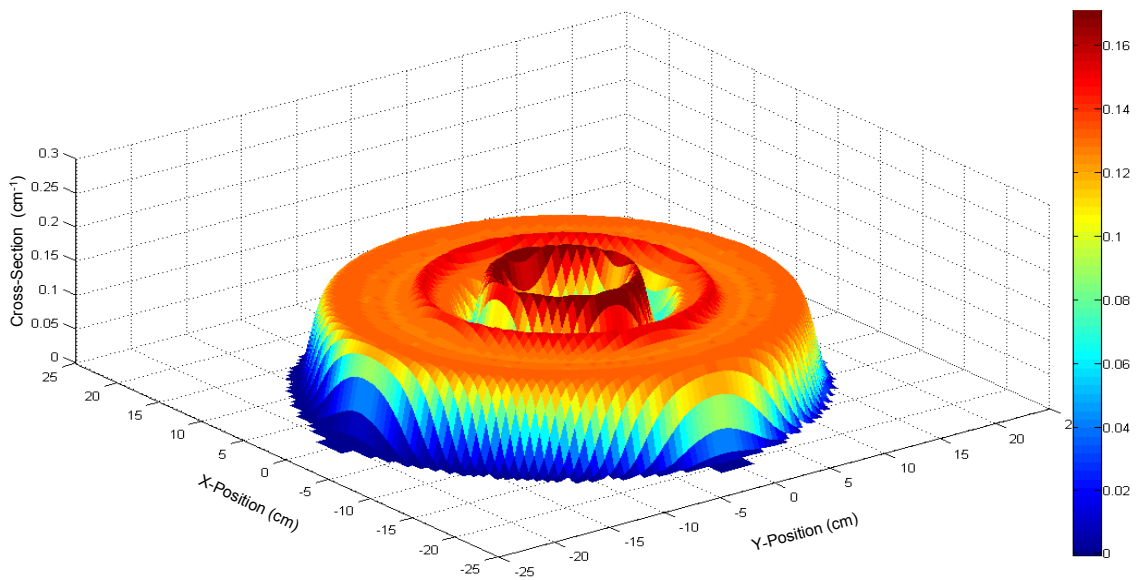
Figure 7-33. A 3D view of the barrel measurement generated using the corrected data. The z-axis and color mapping indicate the estimated material cross sections.

The last step in the NMIS object reconstruction process is the fitting of shapes to the various regions in the 2D image reconstruction. This procedure is performed by a MATLAB code named *Fitting* for NMIS measurements. The code begins with an initial guess of the number of shells and the cross section of each using the FBP, such as the ones shown in Figure 7-31. The new attenuation curve is compared to the old one and the difference between the two curves is used to adjust the initial guesses. The new guesses are then entered into TAKE again. This process continues until the new and old attenuation curves converge. The final dimensions and cross section values are then returned to the operator.

In order to test how much the PSRA correction improves the measurement results, the measured, corrected, and *Direct* filtered back projections were processed using the *Fitting* program. Although the program can generate the initial guesses for the number and locations of the initial shells automatically, it has a strong tendency to over fit the data and produce too many shells. Therefore, the standard procedure is for the user to enter the dimensions and cross section of each shell manually using the values seen on the FBP. For this test, the true dimensions of the shells were entered as the initial guesses in order to maximize the accuracy of the fitted cross section values. These fitted values can then be compared to the true values.

The fitted cross section values for the measured, corrected, and *Direct* curves are listed in Table 7-5. The fitted dimensions were all very close to the true values and were omitted for clarity. The measured data produces very poor results. In particular, the beads, the iron pipes, and the uranium all have approximately the same fitted cross section values, indicating very little contrast between these regions.

185

Table 7-5. Fitted cross section values for each of the eight regions of the barrel generated using the *Fitting* program. The fitted values using measured, corrected, and *Direct* data are shown.

Shell	Name	True Values			Fitted Cross Section Values (cm^{-1})		
		I.D. (cm)	O.D. (cm)	Σ (cm^{-1})	Measured	Corrected	*Direct*
1	Poly	0	2.54	0.11	0.089	0.097	0.061
2	Air	2.54	8.89	0	0.050	0.043	0.002
3	DU	8.89	12.70	0.28	0.106	0.171	0.281
4	Air	12.70	15.24	0	0.039	0.017	0.000
5	Iron Pipe	15.24	16.83	0.22	0.098	0.134	0.212
6	Iron Beads	16.83	20.32	0.13	0.108	0.130	0.130
7	Iron Pipe	20.32	21.91	0.22	0.111	0.190	0.243
8	Iron Beads	21.91	35.56	0.13	0.117	0.130	0.132

The two layers of beads actually have higher fitted cross section values than the pipes inside them. The corrected cross section fits show a greatly increased contrast between the outer iron pipe and the surrounding beads. The DU casting value is also greatly increased and the cross section for the outer shell of air is greatly decreased. The *Direct* cross section fit values represent a best case scenario. In order to achieve better results, either the counting time or the angular resolution of the measurement would need to be increased. The uranium value is almost perfect. The two regions of beads and the inner iron pipe are very close to the true values and the cross section in the air regions is nearly zero. Interestingly, the fit value for the outer iron pipe is slightly high.

7.2.4 Examination of the PSRA Performance

While the PSRA did produce a significant improvement over the measured attenuation values, it significantly undercorrected the data in all three scenarios. All three experimental measurements were very similar to simulations where the scatter correction performed very well. This leads to the conclusion that the source of the undercorrection lies in an underestimation of the scatter in the experimental measurements rather than a

186

failure of the PSRA itself. The most likely source of this error is additional scattering produced by structures near the detector crystals such as the photomultiplier tubes, the detector mounting frame, the support arm, and the motor that controls the angular rotation of the arm.

In the interest of developing a short-term solution to the undercorrection problem, an empirical correction is the best solution until a more rigorous solution can be developed. As a first order approximation, both the ISF and PScF amplitudes were multiplied by a common factor greater than one. This correction was predicated on the assumption that undercorrection was produced by an underestimation of the point scatter and inter-array scatter functions. A multiplicative factor of 1.55 produced the best results for all three experimental measurements. This would seem to indicate that the actual scattering fractions are 55% higher than the current PScF (and ISF) values taken from simulations.

The attenuation curves for the three measurements after the empirical correction are plotted in Figure 7-34 through Figure 7-36. The previous values without the correction are also shown for reference. The new corrected values show a much better agreement with the *Direct* attenuation curve than the old ones. For the polyethylene slab, the ratio of the corrected χ^2 value and the measured χ^2 value increased from 6.65 to 59.7. For the carbon slab, the value increased from 4.91 to 59.9. The values for the barrel measurement increased more modestly from 4.32 to 7.19.

The filtered back projection of the new corrected data (left) is shown in Figure 7-37 along with the filtered back projection of the *Direct* data (right) for reference.
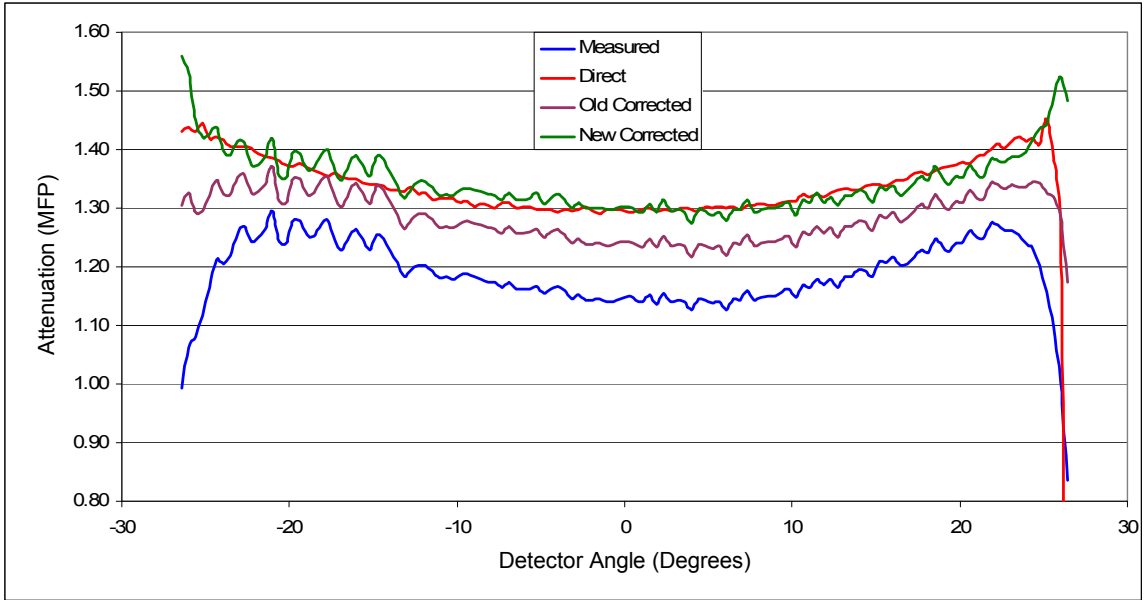
Figure 7-34. Attenuation curves for the polyethylene slab measurement. Both the old and new (with empirical correction factor) corrected curves are shown for comparison.
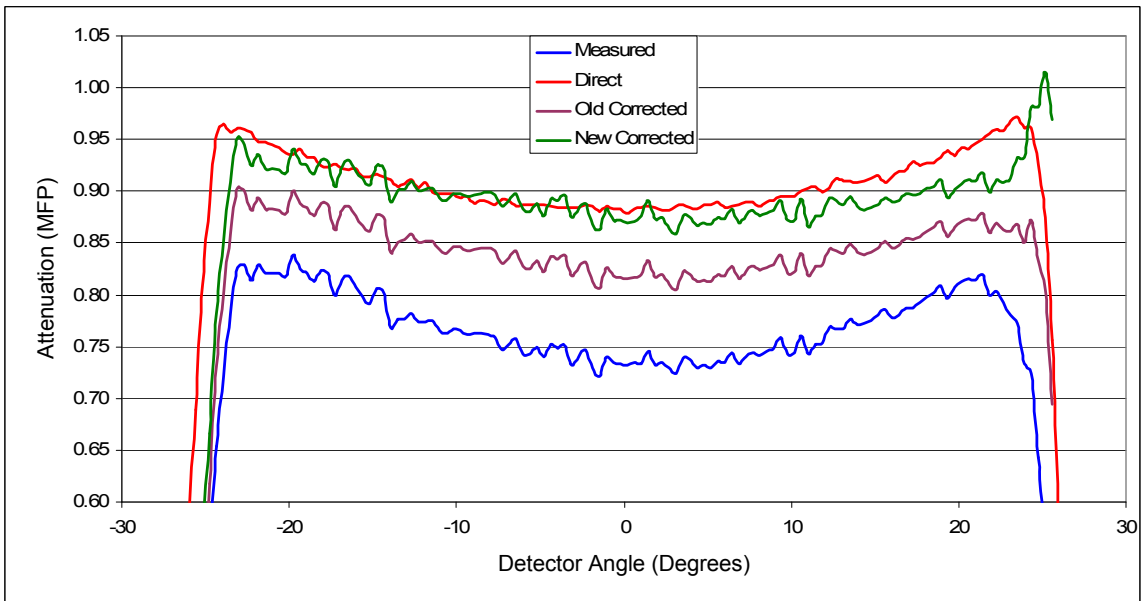
Figure 7-35. Attenuation curves for the graphite slab measurement. Both the old and new (with empirical correction factor) corrected curves are shown for comparison.
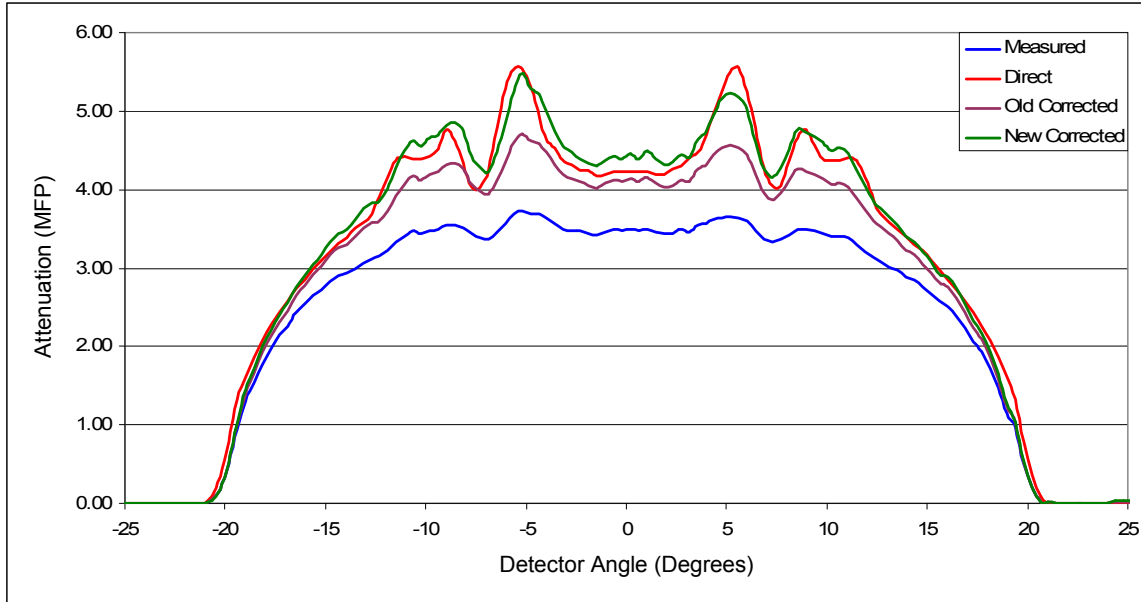
Figure 7-36. Attenuation curves for the barrel measurement. Both the old and new (with empirical correction factor) corrected curves are shown for comparison.
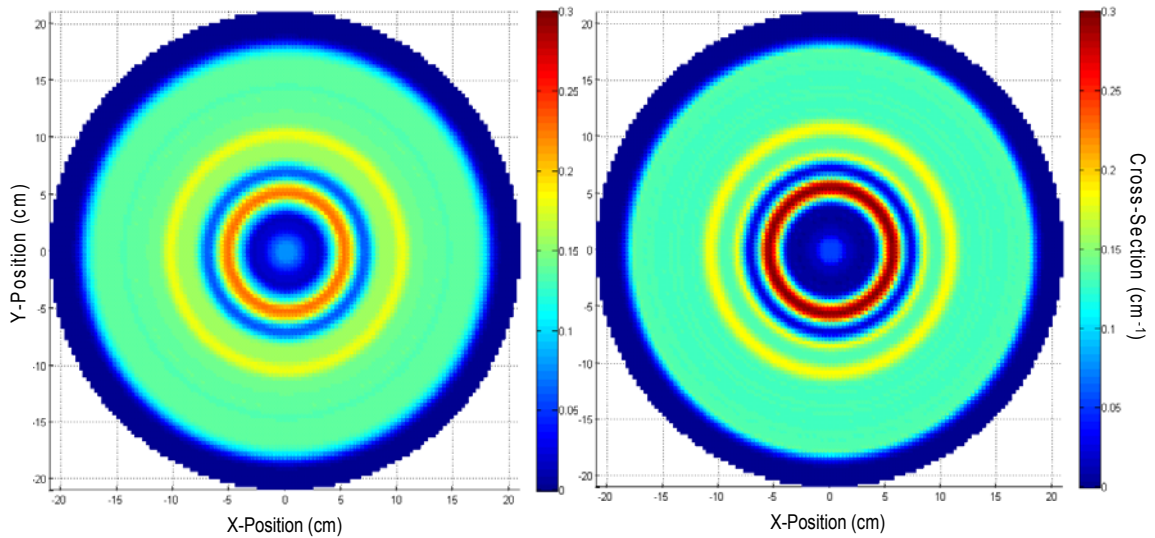
Figure 7-37. Filtered back projection using the new corrected data with the empirical correction (left) for the barrel measurement. The *Direct* filtered back projection (right) using simulated data is shown for comparison.

The cross sections for the outer iron pipe and the uranium casting show up significantly higher in this image than they did with the old corrected values (Figure 7-31). Also, the inner pipe is barely discernible in this image while it was completely invisible before. A 3D view of the data with the cross section values on the z-axis is plotted in Figure 7-38.

The results for the *Fitting* routine are presented in Table 7-6. Both the old and new corrected cross section values are shown along with the measured values. The cross section values for the air regions in the new corrected values are now very close to zero, and the cross section for the DU casting is much closer to its true value. The values for the inner shell of beads and the outer pipe are slightly inflated. While more work is required to produce a more exacting correction for experimental measurements, the PSRA with empirical correction factor produces very good results with experimental NMIS imaging measurements.
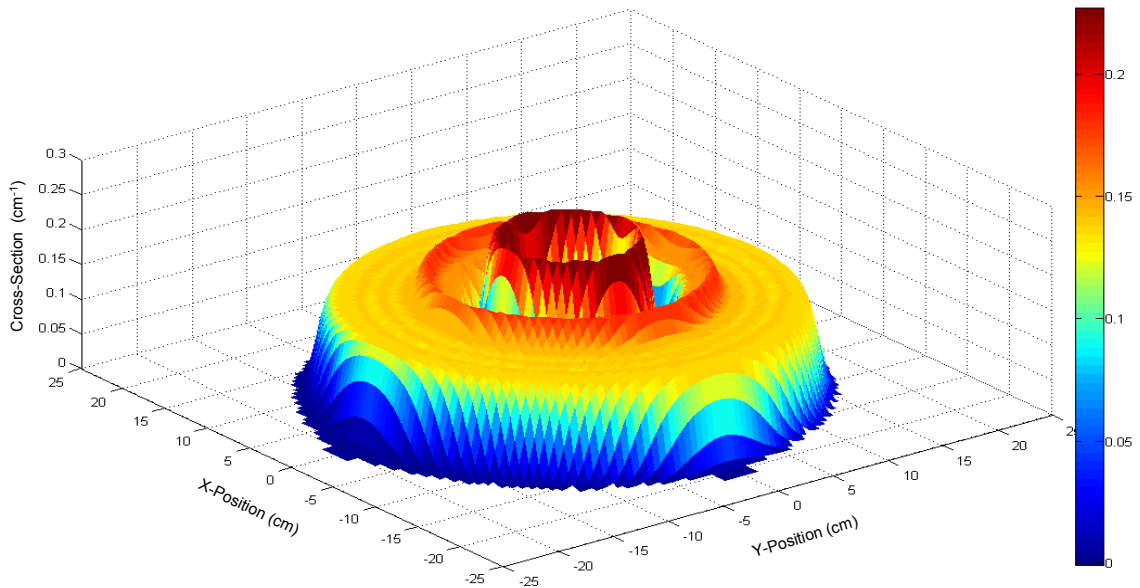
Figure 7-38. A 3D view of the new corrected filtered back projection. Both the z-axis and color mapping indicate the cross section values.

190

Table 7-6. Fit results showing the cross section values calculated using the measured data and both the old and new corrected values.

Shell	Name	True Values			Fitted Cross section Values (cm^{-1})		
		I.D. (cm)	O.D. (cm)	Σ (cm^{-1})	Measured	Old	New
1	Poly	0	2.54	0.11	0.080	0.081	0.098
2	Air	2.54	8.89	0	0.056	0.055	0.019
3	DU	8.89	12.70	0.28	0.118	0.180	0.231
4	Air	12.70	15.24	0	0.058	0.035	0.000
5	Iron Pipe	15.24	16.83	0.22	0.107	0.167	0.169
6	Iron Beads	16.83	20.32	0.13	0.104	0.132	0.152
7	Iron Pipe	20.32	21.91	0.22	0.117	0.174	0.241
8	Iron Beads	21.91	35.56	0.13	0.116	0.131	0.138

7.3 A Method for Integrating a Generalized PSRA into NMIS Imaging

In Section 7.1, the parameterized scatter removal algorithm produced excellent results when correcting scatter for objects consisting of a single known material. Its performance was marginal if the material was unknown and averaged PScF coefficient values were used. When objects consisting of multiple materials were imaged, the use of a single material's PScF coefficients produced poor results which tended to strongly under or overcorrect the attenuation values. In order to make the PSRA more useful, it needs to be modified to be able to deal with these scenarios properly.

The first task is to devise a methodology for calculating the scatter produced by neutrons traversing multiple materials. The current PSRA calculates the additional fraction of scattered neutrons that leave the outer surface of the object and reach the detectors. This value is determined by the thickness of the object measured in attenuation lengths, the material, and the distance from the outer surface to the detector array. For objects composed of multiple materials, each is generating its own scattering.

An example of a simple case where a beam of neutrons is being transmitted through multiple shells of material is depicted in Figure 7-39.
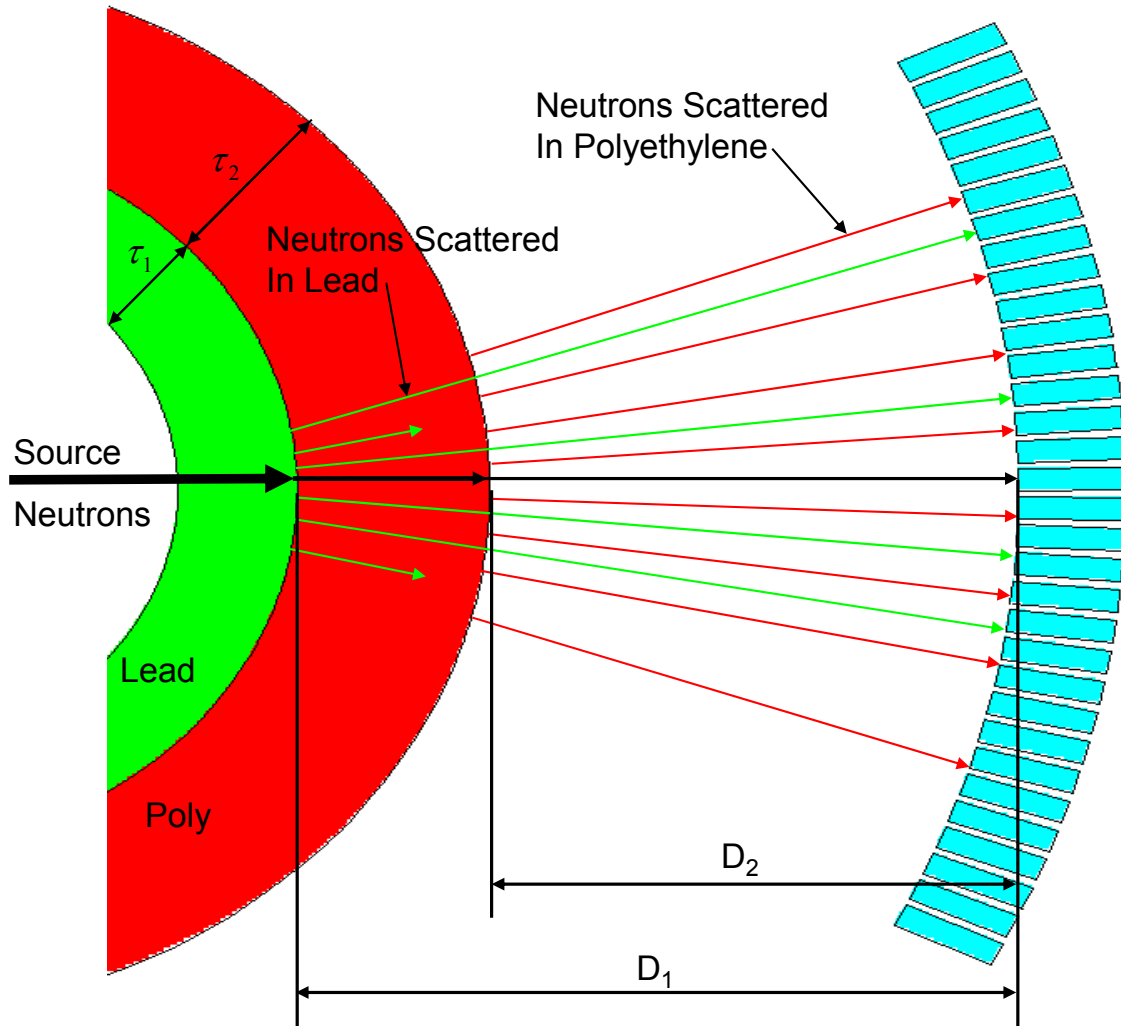
Figure 7-39. The geometry of neutrons scattering in multiple layers of material. Neutrons scattering in the first layer produce a characteristic PScF which is then further attenuated in the second layer. Simultaneously, source neutrons attenuated in the first layer go on to produce a different PScF in the second layer.

The inner layer of material is lead with a thickness of τ_1, so the neutrons passing through it will be attenuated by a factor of EXP($-\tau_1$). If there are no other layers present, those attenuated neutrons will produce a PScF for τ_1 MFP of lead located a distance D_1 from the detector array, PScF(Lead, τ_1, D_1). There is, however, a layer of polyethylene behind the lead which will further attenuate the scattered neutrons. If all of the attenuated neutrons are either absorbed or scattered in a way so that they do not produce additional fast neutron correlations in the peak region, the lead layer will contribute a scattering of EXP($-\tau_2$)PScF(Lead, τ_1, D_1) to the detectors. Simultaneously, the source neutrons were attenuated by a factor of EXP($-\tau_1$) while passing through the lead layer will produce a scatter function in the polyethylene of EXP($-\tau_1$)PScF(Poly, τ_2, D_2). Thus, the total PScF for the two shells will be EXP($-\tau_2$)PScF(Lead, τ_1, D_1) + EXP($-\tau_1$)PScF(Poly, τ_2, D_2). For an object consisting of n shells with a total thickness τ, the total PScF can be written as a superposition of the PScFs for each layer attenuated by all other layers,

$$PScF = \sum_{i=1}^{n} EXP(\tau - \tau_i)PScF(material_i, \tau_i, D_i). \tag{7.1}$$

This method for applying the PScF was tested using the lead and polyethylene scenarios from Section 7.1.2. These scenarios present the simplest possible case for testing. Unfortunately, there is no way to determine the relative thickness of each layer from the measurement data. Therefore, the actual thickness of each layer was used when calculating the PScF values. However, this was sufficient to test Equation 7.1. Initial tests resulted in an undercorrection of the data using this method. This undercorrection is undoubtedly due to the fact that some attenuated neutrons scatter back into the detectors.

Modifying the thicknesses by a factor of 0.9 seems to correct for this result. Thus, the final equation entered into the *ScatterSubtract* program was

$$PScF = \sum_{i=1}^{n} EXP[0.9(\tau - \tau_i)]PScF(material_i, 0.9\tau_i, D_i). \qquad (7.2)$$

The attenuation curves for the PbPo, LPoL, and L1P3 scenarios using the superposition of PScFs are plotted in Figure 7-40, Figure 7-41, and Figure 7-42. These scenarios represent a case where two layers are equally thick (in MFP), a case where two layers are different thicknesses, and a case where there are three layers of material. In all three cases, the corrected attenuation curves show the proper horizontal shape and lie very close to the *Direct* values.

The χ^2 goodness of fit results for the corrected and uncorrected attenuation curves and the ratio between the two for the six scenarios are listed in Table 7-7. All of these were tested previously except for the PoLP scenario.
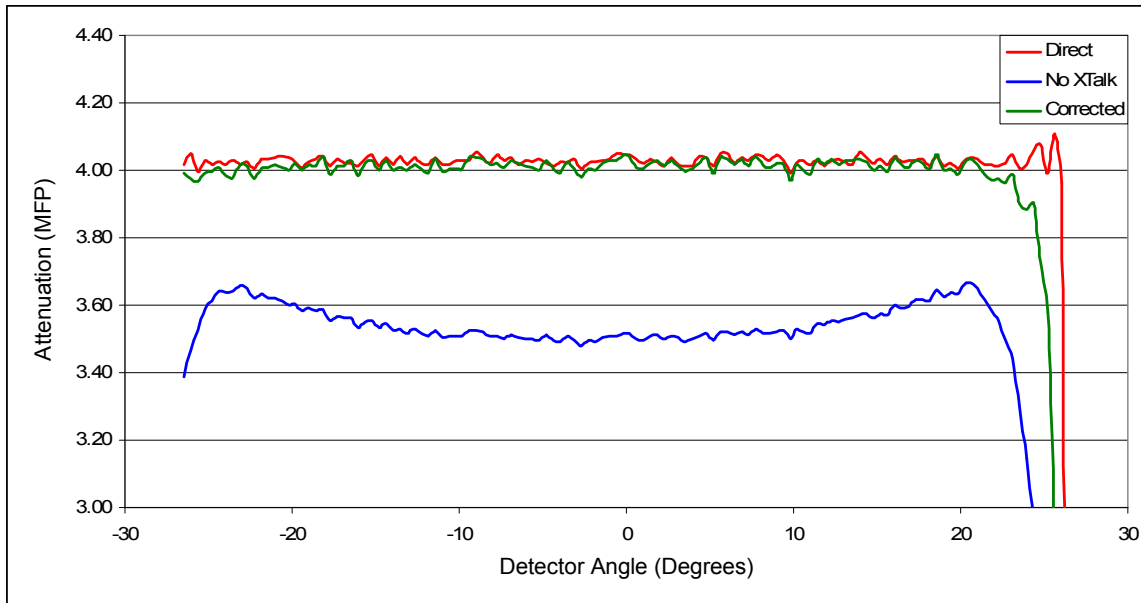
Figure 7-40. The attenuation curves for the PbPo scenario using the superposition of polyethylene and lead PScFs.

194
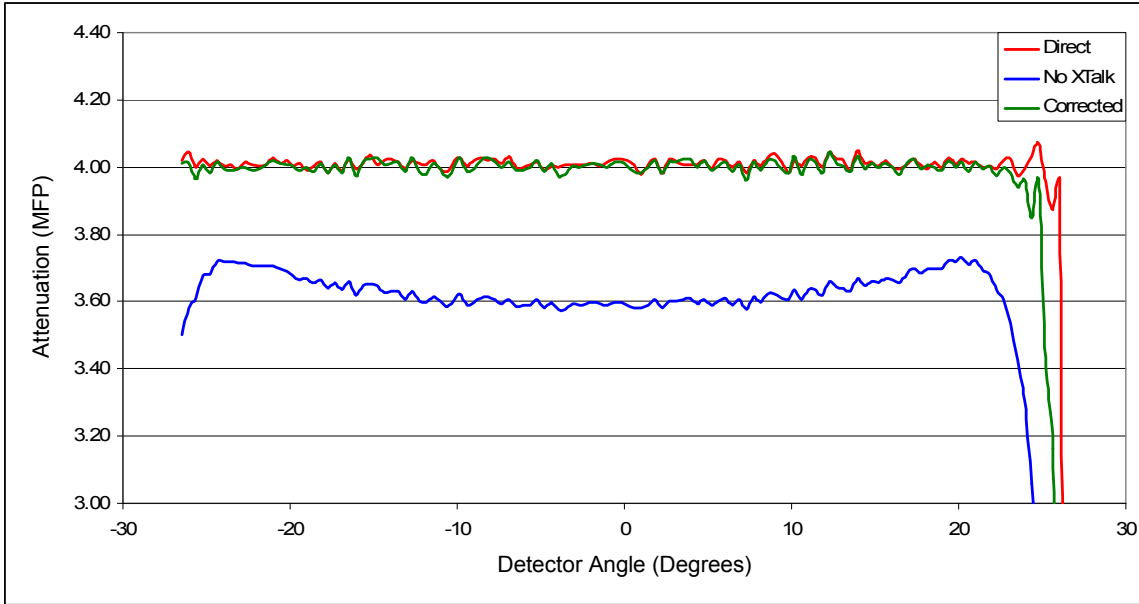
Figure 7-41. The attenuation curves for the LPoL scenario using the superposition of polyethylene and lead PScFs.
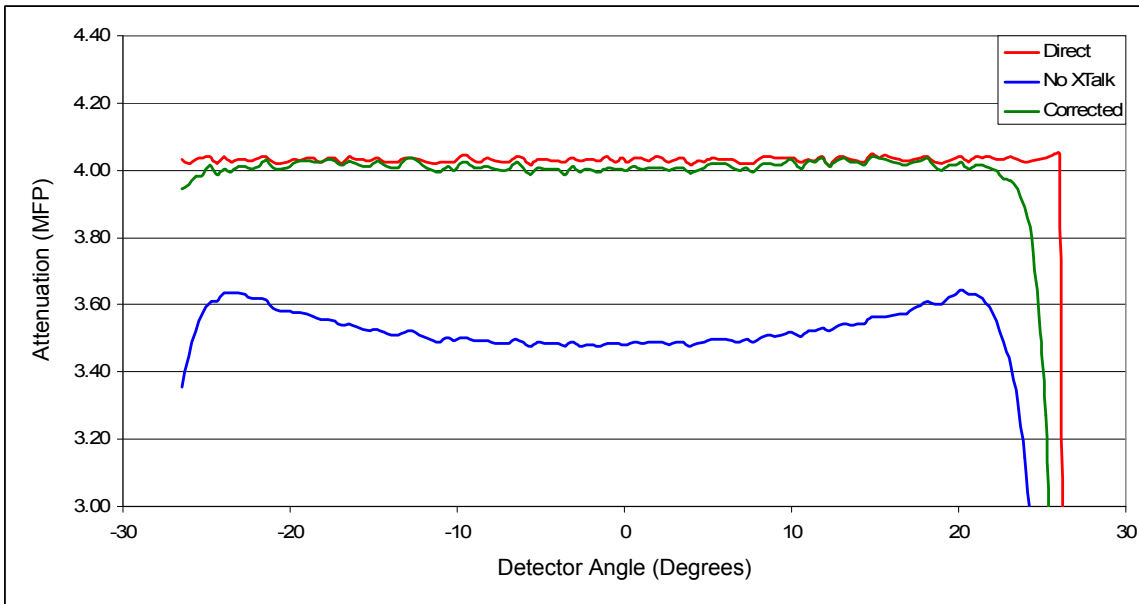
Figure 7-42. The attenuation curves for the L1P3 scenario using the superposition of polyethylene and lead PScFs.

Table 7-7. The χ^2 goodness of fit results of the uncorrected and corrected attenuation curves for the lead and polyethylene scenarios using the superposition of PScFs.

Scenario	Uncorrected χ^2	Corrected χ^2	Ratio
PbPo	1.571E+05	6.317E+02	248.7
PoPb	1.842E+05	4.412E+02	417.7
LPoL	6.453E+05	3.601E+03	179.2
PoLP	6.423E+05	1.439E+03	446.2
L1P3	9.917E+04	3.341E+02	296.8
L3P1	2.373E+05	6.374E+02	372.4

It consists of two 1 MFP layers of polyethylene surrounding a 2 MFP layer of lead.

These results show that the superposition of PScFs produces excellent results with all six

scenarios. The ratios lie between 179 and 446, indicating that the corrected attenuation

curves lie between approximately 13 and 21 times closer to the *Direct* values than the

uncorrected curves. While this is not a true test because the thickness of each material

was selected a priori, it indicates that *if* a good estimation of the relative thickness of each

layer can be determined, the PSRA can remove the scatter from the measured values and

return an accurate estimate of the true values.

7.4 Recommendations for Future Work

While this work has shown that the PSRA methodology is sound, there are still

some situations in which it performs poorly. In order to apply the superposition of PScFs

to a more generalized object where the scattering is not necessarily symmetric about the

neutron path or where the object is composed of multiple materials, a more sophisticated

method needs to be developed. If an object is cylindrically symmetric, or if several

different projections of the object are taken at different angles, the FBPGUI program can

reconstruct a filtered back projection of the internal structure of the object using the

attenuation curves. These values could be used to develop first order PScFs which could then be applied to the measured values to produce a correction. The new values could then be used to generate another filtered back projection, from which new PScF values could be developed. This process could be iterated until the filtered back projections converge to a final solution which will mirror the true internal structure of the object. In order to accomplish this integration, some additional work is required. Therefore, this section will recommend the direction in which future research efforts should proceed in order to create a general NMIS imaging solution which incorporates the PSRA.

The first area that should be investigated is the detector model used for simulating the NMIS PScFs. Experimental testing strongly suggests that the simulated PScFs presented in this work underestimated the scatter in laboratory measurements by a factor of approximately 55%. While an empirical correction factor produced results comparable to those from simulations, this is not an ideal solution. Therefore future work should focus on developing a PScF simulation that produces scattering similar to that encountered by experimental measurements. The first area of focus should be the creation of a high fidelity model of the NMIS detector array since neutrons scattering in that region have a large impact on the total scattering signal received.

Once the new PScF models are completed and the simulated PScFs match the experimental ones, the PScF scenarios can be rerun to generate new PScFGE coefficients for *ScatterSubtract*. While these simulations are being rerun, additional materials may be added to the library. The initial library was kept fairly limited since its purpose was to validate the PSRA methodology. Ideally, the final library will include all materials which NMIS imaging measurements will likely encounter.

In addition to the PScF simulations with homogeneous shells of material, simulations with multiple layers materials need to be conducted. These simulations should use the same source geometry as the ones used for extracting PScF parameters, but change the object geometry. The purpose of these simulations will be to determine how the PScFs from multiple materials are superimposed to form a single PScF in an object composed of multiple layers. Equation 7.3 produced good results when used for solving this problem, but a more thorough study is desirable. In addition, these tests should examine scenarios where the object (or the material regions inside the object) is asymmetric. Those simulations will test how the PScF contribution to individual detectors is modified as the scattered neutrons pass through different thicknesses of material. A robust method for modifying the PScFs in these scenarios is needed before the PSRA can be fully integrated into NMIS imaging.

The next area of research should focus on integrating the PSRA with the FBPGUI program. The FBPGUI generates a filtered back projection of the object being imaged. The FBP creates a matrix of x and y pixels and assigns a cross section value to each. *ScatterSubtract* could then read these values in order to determine the best PScF to use for each detector position and the modifications that should be performed based on the neutron path to other detectors in the array. Since FBPGUI is a GUI-based program and *ScatterSubtract* is executed from the command line, the most logical integration method would be for the FBPGUI code to write the FBP data to a file and call *ScatterSubtract* to read it each time it is needed.

In order for the scattering subtraction to work properly, the correct PScFs need to be selected. Integrating *ScatterSubtract* into FBPGUI allows for the possibility that the

operator can explicitly specify the material of each region based on prior knowledge, if available. Even if *a priori* knowledge of the materials is not available, the FBP will give clues which can be used to select the best PScFs. If the materials list is coupled with the nominal cross section of each material, then certain materials can be excluded if the FBP value is significantly greater the material cross section. Based on the results of this comparison, the PScF parameters can be selected by choosing the one that produces the *smallest* correction to the data to avoid an overcorrection. This correction will increase the FBP cross section values in the next iteration, which can eliminate more materials from consideration. So long as these values are carefully chosen to avoid overcorrecting the data, the process should eventually converge. Once the iterations converge to a final value, FBPGUI could tentatively identify each material based on the final PScF parameters that were used. By assigning a material to each pixel, this procedure will have successfully completed its ultimate goal of NMIS imaging, which is to properly identify the material composition and geometry of the internal structure of the object.

8. Conclusions

This work has presented the development of a parameterized scatter removal algorithm (PSRA) from initial theory all the way to its application to experimental measurements. The PSRA is a completely new approach to removing the scattering component from NMIS imaging. Earlier methods focused primarily on minimizing the number of scattered neutrons that were incorrectly identified as being directly transmitted. The PSRA approaches the problem from a different direction by using Monte Carlo simulations to estimate how large the scattering contribution will be so that it can be subtracted from the measured values directly. By parameterizing the scattering based on the location, thickness, and material of the object being imaged, the scatter correction can be applied to a much wider range of scenarios than those which were used for developing the PScFs.

The PSRA performed extremely well when correcting the scatter in simulations of simple homogeneous targets of known material. The PSRA reduced the value of the χ^2 goodness of fit test of the attenuation curve versus ideal values by a factor of 60 or more over the uncorrected results. These results confirm that the methodology works and strongly suggest that it will be a powerful tool for improving the quality of imaging measurements when integrated into the NMIS software package.

The simulation-derived PScFs initially underestimated the scatter in experimental measurements; however, an empirical correction factor improved the results significantly and χ^2 values were reduced by a factor of 7 to 60. These results strongly indicate that once the PScF simulations are improved to more accurately simulate the scattering in the

NMIS detector array, the PSRA will produce excellent results removing scatter from experimental measurements.

By increasing the accuracy of the measured attenuation profile, these improvements promise to increase the contrast between regions of dissimilar materials and more accurately determine the cross section of each region. More accurately measuring the cross sections will increase the likelihood that the materials can be correctly identified. This will obviate the need for a more obtrusive method (such as opening the container) which can be expensive and hazardous.

In addition to the PSRA, this work also led to the design of a completely new technique for simulating NMIS imaging measurements with a pixelated DT neutron beam. While previous simulations used a uniform fan beam, the new models used individual pixel shapes which are based on both analytical calculations and experimental measurements. In order to simulate measurements using these pixel models, a series of custom codes was written to automate the process. These codes include programs to generate multiple *MCNP-PoliMi* input decks, a post-processor for extracting the correlation information from the output files, and a program for applying the scattering correction to the results. Other codes were developed for converting NMIS measurement data into the format of simulation data and vice versa. These codes will allow for a closer integration of simulations and experiments in the future.

While this work has shown that the PSRA methodology can make substantial improvements in NMIS imaging measurements, some additional work is required before it can be integrated into the current NMIS software suite as a generalized imaging

solution. Recommendations for future research directed towards this goal were presented at the end of the previous chapter.

List of References

1. J. K. Shultis and R. E. Faw, *Radiation Shielding*, Prentice-Hall, Upper Saddle River, NJ, 1996.

2. J. E. Turner, *Atoms, Radiation, and Radiation Protection*, 2nd Edition, John Wiley & Sons, New York, 1995.

3. J. T. Mihalczo, et al., "Physical Description of Nuclear Materials Identification System (NMIS) Signatures," *Nuclear Instruments and Methods in Physics Research A,* **450,** 531-555 (2000).

4. D. S. Koltick, et al., "Characterization of an Associated Particle Neutron Generator with ZnO:Ga Alpha-Detector and Active Focusing," *IEEE Transactions on Nuclear Science* **56,** (3), 1301-1305 (2009).

5. P.A. Hausladen, et al., "Portable Fast-Neutron Radiography with the Nuclear Materials Identification System for Fissile Material Transfers," *Nuclear Instruments and Methods in Physics Research B* 261, 387–390 (2007).

6. J. K. Mattingly, et al., *Use Of The Nuclear Materials Identification System (NMIS) For Enhanced Receipt Confirmation Measurements At The Oak Ridge Y-12 Plant*, Y/LB-16,025, Y-12 National Security Complex, Oak Ridge, TN, 2000.

7. V. P. Dubinin, et al., *VNIIEF-ORNL Joint Plutonium Measurements With NMIS And Results Of Plutonium Attributes Preliminary Evaluations*, Y/LB-16,075, Y-12 National Security Complex, Oak Ridge, TN, 2001.

8. J. A. Mullens, *Addition of Tomographic Capabilities to NMIS*, Y/LB-16,160, Y-12 National Security Complex, Oak Ridge, TN, 2003.

9. D. L. Chichester, et al., "Advanced Compact Accelerator Neutron Generator Technology for Active Interrogation Field Work," *Journal of Radioanalytical and Nuclear Chemistry* **271,** (3), 629-637 (2007).

10. D. L. Chichester, et al., "The API 120: A portable neutron generator for the associated particle technique," *Nuclear Instruments and Methods in Physics Research B* **241,** 753–758 (2005).

11. Thermo Scientific API 120, http://www.thermo.com/eThermo/CMA/PDFs/Various/File_29654.pdf. (Accessed January 2010).

12. P. A. Hausladen, et al., "An Alpha Particle Detector for a Portable Neutron Generator for the Nuclear Materials Identification System (NMIS)," *Nuclear Instruments and Methods in Physics Research B* **241,** 835–838 (2005).

13. Hamamatsu PMT Modules, H8500, H9500, http://sales.hamamatsu.com/en/products/electron-tube-division/detectors/photomultiplier-modules.php. (Accessed October 2009).

14. J. Jakubek, "Data Processing and Image Reconstruction Methods for Pixel Detectors," *Nuclear Instruments and Methods in Physics Research A* **576,** 223–234 (2007).

15. C. Fischer, "The History of the First Neutron Radiographs in Berlin 1935-1944," in *Neutron Radiography (4)*, 3-9, ed. J. P. Barton, Gordon and Breach Science Publishers, Yverdon, CH, 1994.

16. A. K. Heller and J. S. Brenizer, "Neutron Radiography," in *Neutron Imaging and Applications: A Reference for the Imaging Community*, 67-80, eds. I. S. Anderson, et al., Springer, New York, 2009.

17. P. von der Hardt and H. Röttger eds., *Neutron Radiography Handbook*, D. Reidel Publishing Company, Dordrecht, Holland, 1981.

18. Y. Segal, et al., "Point Spread Functions Due to Neutron Scattering in Thermal Neutron Radiography of Aluminum, Iron, Zircaloy and Polyethylene Objects," *Nuclear Instruments and Methods* **197,** 557-562 (1982).

19. Z. Hrdlicka and F. Peterka, "Neutron Radiography at the Nuclear Research Institute Rez Principles of Neutron Transmission Analysis," in *Neutron Radiography* (3), 131-143, S. Fujine, et al. eds., Kluwer Academic Publishers, Dordrecht, Holland, 1990.

20. H. Kobayashi, et al., "Macroscopic Cross Section Measurements and Defect Detection in Materials Using Neutron Radiography", *Journal of Nuclear Science and Technology* **29** (11), 1045-1053 (1992).

21. Y. Murata, et al., "Two-dimensional neutron image excluding the effect of scattered neutrons", in *Neutron Radiography* (4), 583-590, ed. J. P. Barton, Gordon and Breach Science Publishers, Yverdon, CH, 1994.

22. F. C. de Beer, et al., *Neutron Scattering Corrections for Neutron Radiography in Neutron Imaging: A Non-Destructive Tool for Materials Testing*, IAEA-TECDOC-1604, International Atomic Energy Agency, Vienna, Austria, 2008.

23. M. Tamaki, et al., "Study on Neutron Imaging Techniques and Processings for Developing Quantitative Neutron Radiography," *Nuclear Instruments and Methods in Physics Research A* **377,** 102-106 (1996).

24. H. Pleinert, et al., "Neutron Signal Transfer Analysis," *Nuclear Instruments and Methods in Physics Research A* **424,** 177–182 (1999).

25. J. Briesmeister (Ed.), *MCNP - A general Monte Carlo code, version 4A*, LA-1265, Los Alamos National Lab., Los Alamos, 1993.

26. N. Kardjilov, et al., "Representation of the image formation in applied neutron radiography in terms of a PSF superposition," *Applied Physics A* **74** (Suppl.), S228–S230 (2002).

27. J. Briesmeister (Ed.): *MCNP – A General Monte Carlo N-Particle Transport Code, Version 4B*, LA-12625-M, Los Alamos National Laboratory, Los Alamos, March 1997

28. Neutron Imaging and Activation Group at the Paul Scherrer Institute, http://neutra.web.psi.ch. (Accessed November 2009).

29. René Hassanein, et al., "Methods of scattering corrections for quantitative neutron radiography," *Nuclear Instruments and Methods in Physics Research A* **542,** 353–360, (2005).

30. L.S. Waters ed., *MCNPX User's Manual, Version 2.4.0*, LA-CP-02-408, Los Alamos National Laboratory, Los Alamos, September 2002.

31. J. F. Briesmeister ed., *MCNP - A General Monte Carlo N-Particle Transport Code, Version 4C*, LA-13709-M, Los Alamos National Laboratory, Los Alamos, April 2000.

32. B. Tang, et al., "The Pilot Experimental Study of 14 MeV Fast Neutron Digital Radiography," *Science in China Series G* **52,** 1330-1336 (2009).

33. E. Tochilin, "Photographic detection of fast neutrons: application to neutron radiography," *Physics in Medicine and Biology* **10,** 477-490 (1965).

34. H. Berger, "Image detection methods for 14.5 MeV neutrons: techniques and applications", *International Journal of Applied Radiation and Isotopes* **21,** 59-70 (1970).

35. J. S. Brzosko, et al., "Advantages and limits of 14-MeV neutron radiography," *Nuclear Instruments and Methods in Physics Research B* **72,** 119-131 (1992).

36. This software is not available to the public. According to J. S. Brzosko, et al. [35]: "The master complex is proprietary information of Neutronics / Photonics R&D, 152 Harrison Ave, Jersey City, NJ, 07304, USA".

37. E. Rhodes, et al., "APSTNG: Neutron Interrogation for Detection of Nuclear and CW Weapons, Explosives, and Drugs," in *Neutron Radiography* (4), 827-835, ed. J. P. Barton , Gordon and Breach Science Publishers, Yverdon, CH, 1994.

38. K. Yoshii and H. Kobayashi, "Correction of Fast Neutron Scattered Components from Fast Neutron Radiography Images," *Nuclear Instruments and Methods in Physics Research A* **377,** 76-79 (1996).

39. H. Rahmanian, et al., "Optimisation of resolution in accelerator-based fast neutron radiography," *Nuclear Instruments and Methods in Physics Research A* **477,** 378-382 (2002).

40. R. M. Ambrosi and J. I. W. Watterson, "The effect of the imaging geometry and the impact of neutron scatter on the detection of small features in accelerator-based fast neutron radiography," *Nuclear Instruments and Methods in Physics Research A* **524,** 340-354 (2004).

41. M. H. Hassan, "Point Scattered Function (PScF) for fast neutron radiography," *Nuclear Instruments and Methods in Physics Research B* **267,** 2545–2549 (2009).

42. J. Eberhardt, et al., "Fast-Neutron/Gamma-ray radiography scanner for the detection of contraband in air cargo containers - art. no. 621303," *Non-Intrusive Inspection Technologies* 6213: 21303-21303 2006.

43. B. D. Sowerby and J. R. Tickner, "Recent advances in fast neutron radiography for cargo inspection," *Nuclear Instruments and Methods in Physics Research A* **580,** 799–802 (2007).

44. Y. Liu, et al., "Comparison of Neutron and High-Energy X-Ray Dual-Beam Radiography for Air Cargo Inspection," *Applied Radiation and Isotopes* **66,** 463-473 (2008).

45. G. M. Hale and M. Drosg, *ENDF/B-VII.0 Library*, Accessed online through the National Nuclear Data Center, http://www.nndc.bnl.gov. (Accessed November 2009).

46. E. W. Walker, et al., *Chart of the Nuclides,* 14th Edition, General Electric, San Jose, California, 1989.

47. J. J. Duderstadt and L. J. Hamilton, *Nuclear Reactor Analysis*, Wiley, New York, New York, 1976.

48. J. F. Ziegler, et al., *The Stopping and Range of Ions in Matter, Version SRIM-2008.04*, Available online at http://www.srim.org. (Accessed October 2009).

49. J. T. Mihalczo, Personal Communication. API-120 Target Geometry, January 2009.

50. ISO/IEC 1539:1991*, Information Technology - Programming Languages - Fortran, Second Edition*, ISO Publications Department, Geneva, Switzerland, 1991.

51. G. F. Knoll, *Radiation Detection and Measurement,* 3rd Edition, John Wiley & Sons, New York, 2000.

52. SAS Institute, *JMP User Guide, Release 7*, SAS Institute, Inc., Cary, NC, 2007, http://www.jmp.com/support/downloads/pdf/jmp_user_guide.pdf. (Accessed November 2009).

53. National Nuclear Data Center, http://www.nndc.bnl.gov. (Accessed December 2009).

54. E. Padovani and S. A. Pozzi, *MCNP-PoliMi Ver. 1.0 Users Manual*, CESNEF-021125, Polytechnic of Milan, Italy, 2002.

55. Radiation Safety Information Computational Center, http://rsicc.ornl.gov/. (Accessed December 2009).

56. The Portland Group, http://www.pgroup.com/. (Accessed December 2009).

57. X-5 Monte Carlo Team, *MCNP — A General Monte CarloN-Particle Transport Code, Version 5*, LA-UR-03-1987, Los Alamos National Laboratory, Los Alamos, NM, 2005.

58. D. B. Pelowitz ed., *MCNPX User's Manual, Version 2.6.0*, LA-CP-07-1473, Los Alamos National Laboratory, Los Alamos, NM, 2008.

59. M. Marseguerra, et al., "Simulating the Wrong Physics Can Generate the Correct Results," *3rd IMACS Seminar on Monte Carlo Methods*, Salzburg, Austria, September 2001.

60. G. W. McKinney, MCNPX Team Leader, Los Alamos National Laboratory, Personal Communication. Questions Regarding MCNPX 2.6 vs. MCNP-PoliMi, July 2008.

61. MATLAB software, The Mathworks, http://www.mathworks.com/products/matlab/. (Accessed November 2009).

62. S. A. Pozzi, *Recent Developments in the MCNP-PoliMi Postprocessing Code*, ORNL/TM-2004/299, Oak Ridge National Laboratory, Oak Ridge, TN, 2004.

Appendices

Appendix A – The *MCPixelGeom* Code

```
PROGRAM MCPixelGeom
IMPLICIT NONE

REAL :: ThetaXY, ThetaXZ, x0, y0, z0, v, w, x1, y1, z1, r1, Omega, TargetRad
REAL :: PhiXY, PhiXZ, PsiXY, PsiXZ, PixelDim, x2, y2, z2, x3, y3, z3
REAL :: DetectorRad, DelPhiMax, Vn, Vcm, Valpha, Weight
REAL :: PhiAvg, ThetaAvg, PsiAvg
REAL, PARAMETER :: TwoPi = 6.2831853
REAL, PARAMETER :: Pi = 3.1415927
REAL, ALLOCATABLE :: Results(:,:,:), Results1D(:,:,:)
INTEGER (KIND=4) :: I, J, K, M, N
INTEGER :: NumPixels, HLimit, VLimit, MinHoriz, MaxHoriz
INTEGER (KIND=4) :: NumSamples
INTEGER (KIND=4), ALLOCATABLE :: PixelCounts(:)

!Initialize the random seed
CALL init_random_seed()

! Define input variables
! Positions are measured in cm from the target center
DetectorRad = 110.
TargetRad = 0.250
PixelDim = 0.608
NumPixels = 8
NumSamples = 1000000000
!Velocities, fraction of c
Vn = 0.173015
Vcm = 0.003306
Valpha = 0.043564
HLimit = 400
VLimit = 80

ALLOCATE(PixelCounts(NumPixels+1))

!Allocate Results matrix, which will hold the results.  Rows are bins for the horizontal
!angle (*10) of the neutron, Columns are vertical angle, and Panes are the pixels.  The
!penultimate pane will contain the sum of all pixels and the last contains the maximum
!pixel value for that angle.
ALLOCATE(Results(-HLimit:HLimit,-VLimit:VLimit,NumPixels+2))

! The 1D results matrix is similar, but contains the sum of +/- 0.6 degrees in
! the vertical direction, which is ~ the detector height.
ALLOCATE(Results1D(-HLimit:HLimit,NumPixels+1,2))
Results = 0.
Results1D = 0.
PixelCounts = 0

PhiAvg = 0.
ThetaAvg = 0.
PsiAvg = 0.

DO I = 1, NumSamples
  ! v, w are relative position on the pixel face (-0.5 < v,w < 0.5)
  ! Omega, r1 are radial coordinates on target face.
  CALL RandomSample(v,w,Omega,r1)
```

```
   ThetaXY = v
   ThetaXZ = W

   !Convert Omega, r1 to cartesian coordinates relative to target center
   x2 = 0.
   y2 = r1*SIN(Omega)
   z2 = r1*COS(Omega)

   !Now account for the 45 degree (xy) tilt of the target and convert
   !x,y2 to absolute coordinates.  No change in z2.
   x2 = y2*SIN(TwoPi/8.)
   y2 = y2*SIN(TwoPi/8.)

   !Calculate the Phi angles, alpha angle in the LAB CS.
   PhiXY = ATAN((VAlpha*SIN(ThetaXY)+Vcm)/(VAlpha*COS(ThetaXY)))
   PhiXZ = ThetaXZ

   ! Find x1, y1, z1 where the alpha reaches the alpha detector plane
   x1 = -5.7
   y1 = y2 + ABS(x2-x1)*TAN(-PhiXY)
   z1 = z2 + ABS(x2-x1)*TAN(-PhiXZ)

   ! Determine which pixel (if any) the alpha strikes.
   IF (ABS(z1) < PixelDim / 2.) THEN
     K = INT(REAL(NumPixels) / 2. - y1/PixelDim + 1)
     IF (K < 1 .OR. K > NumPixels) CYCLE
     PixelCounts(K) = PixelCounts(K) + 1
     PixelCounts(NumPixels+1) = PixelCounts(NumPixels+1) + 1
   ELSE
     CYCLE
   END IF

   !Now calculate neutron angle in the LAB coordinate system.
   PsiXY = ATAN((Vn*SIN(ThetaXY)-Vcm)/(Vn*COS(ThetaXY)))
   PsiXZ = PhiXZ

   M = NINT(10.*PsiXY*360./TwoPi)
   N = NINT(10.*PsiXZ*360./TwoPi)

   IF (ABS(M) > HLimit .OR. ABS(N) > VLimit) THEN
     PRINT *, "Array Bounds Exceeded"
     CYCLE
   END IF

   Results(M,N,K) = Results(M,N,K) + 1

   PhiAvg = PhiAvg + PhiXY
   ThetaAvg = ThetaAvg + ThetaXY
   PsiAvg = PsiAvg + PsiXY

END DO

PhiAvg = PhiAvg / REAL(NumSamples)*360./TwoPi
ThetaAvg = ThetaAvg / REAL(NumSamples)*360./TwoPi
PsiAvg = PsiAvg / REAL(NumSamples)*360./TwoPi


DO I = -HLimit, HLimit, 1
  DO J = -VLimit, VLimit, 1
   DO K = 1, NumPixels
```

```fortran
        Results(I,J,K) = Results(I,J,K) * NumPixels / PixelCounts(NumPixels+1)
      END DO
    END DO
END DO

DO K = 1, NumPixels+1
    PRINT *, K, PixelCounts(K)
END DO

DO I = -HLimit, HLimit, 1
  DO J = -VLimit, VLimit, 1
    DO K = 1, NumPixels
      Results(I,J,NumPixels+1) = Results(I,J,NumPixels+1) + Results(I,J,K)
      IF(Results(I,J,K) > Results(I,J,NumPixels+2)) THEN
        Results(I,J,NumPixels+2) = Results(I,J,K)
      END IF
      IF(ABS(J) <= 6) THEN
        Results1D(I,K,1) = Results1D(I,K,1) + Results(I,J,K)
        Results1D(I,NumPixels+1,1) = Results1D(I,NumPixels+1,1) +
Results(I,J,K)
      END IF
    END DO
  END DO
END DO

DO I = 1, NumPixels+1
  DO J = -HLimit, HLimit, 1
    MinHoriz = MAX(-HLimit,J-6)
    MaxHoriz = MIN(HLimit,J+6)
    DO K = MinHoriz, MaxHoriz
      Results1D(J,I,2) = Results1D(J,I,2) + Results1D(K,I,1)
    END DO
  END DO
END DO

!Write results array to a file
OPEN (UNIT=1, FILE="Results.txt", ACTION="WRITE", STATUS="REPLACE")
DO I = 1,NumPixels+2
  WRITE(1,'(A5,I3)') "Pixel", I
  WRITE (1,'(A6)', ADVANCE="NO") "      "
  DO K = -VLimit,VLimit,1
    WRITE (1,'(F10.1)', ADVANCE="NO") REAL(K)/10.
  END DO
  WRITE (1,*)
  DO J = -HLimit,HLimit,1
      WRITE (1,'(F6.1)', ADVANCE="NO") REAL(J)/10.
    DO K = -VLimit,VLimit,1
      WRITE (1,'(F10.6)', ADVANCE="NO") Results(J,K,I)
    END DO
      WRITE (1,*)
  END DO
    WRITE (1,*)
    WRITE (1,*)
END DO

CLOSE(UNIT=1)
DEALLOCATE(Results)

!Write 1D results array to a file
OPEN (UNIT=2, FILE="Results1D.txt", ACTION="WRITE", STATUS="REPLACE")
OPEN (UNIT=3, FILE="ResultsDet.txt", ACTION="WRITE", STATUS="REPLACE")
```

212

```fortran
WRITE (2,'(A6)', ADVANCE="NO") "      "
WRITE (3,'(A6)', ADVANCE="NO") "      "

DO I = 1,NumPixels
  WRITE(2,'(A7,I3)', ADVANCE="NO") "  Pixel", I
  WRITE(3,'(A7,I3)', ADVANCE="NO") "  Pixel", I
END DO

WRITE(2,'(A10)') "Total"
WRITE(3,'(A10)') "Total"

DO I = -HLimit,HLimit,1
  WRITE (2,'(F6.1)', ADVANCE="NO") REAL(I)/10.
  WRITE (3,'(F6.1)', ADVANCE="NO") REAL(I)/10.
  DO J = 1, NumPixels+1
      WRITE (2,'(F10.6)', ADVANCE="NO") Results1D(I,J,1)
      WRITE (3,'(F10.6)', ADVANCE="NO") Results1D(I,J,2)
  END DO
      WRITE (2,*)
      WRITE (3,*)
END DO

CLOSE(UNIT=2)
CLOSE(UNIT=3)
DEALLOCATE(Results1D)

CONTAINS

! This subroutine generates the seed for the random number generator from the
system clock.
SUBROUTINE init_random_seed()
INTEGER :: i, n, clock
INTEGER, DIMENSION(:), ALLOCATABLE :: seed
CALL RANDOM_SEED(size = n)
ALLOCATE(seed(n))
CALL SYSTEM_CLOCK(COUNT=clock)
seed = clock + 37 * (/ (i - 1, i = 1, n) /)
CALL RANDOM_SEED(PUT = seed)
DEALLOCATE(seed)
END SUBROUTINE

! This subroutine generates pseudorandom numbers for the position on the alpha
pixel and
! the position on the target.
SUBROUTINE RandomSample(zv,zw,zOmega,zr1)
REAL, INTENT(OUT) :: zv, zw, zOmega, zr1
REAL :: zzz

CALL RANDOM_NUMBER(zzz)
zv = (zzz - 0.5)*TwoPi/4.
CALL RANDOM_NUMBER(zzz)
zw = (zzz - 0.5)*16.*TwoPi/360.
CALL RANDOM_NUMBER(zzz)
zOmega = TwoPi*zzz
CALL RANDOM_NUMBER(zzz)
zr1 = TargetRad*SQRT(zzz)
END SUBROUTINE

END PROGRAM
```

Appendix B – The *Elastic* Code

```fortran
PROGRAM Elastic
!-------------------------------------------------------------
! Written by Brandon R. Grogan
! at the Oak Ridge National Laboratory
! Last Modified 19 November 2009
!-------------------------------------------------------------
IMPLICIT NONE

INTEGER :: A, I, J, K, ErrCode, NumLines, Angles, NumColumns
! Distance Variables
REAL(KIND=4) :: R, D, L, Dout, Lmax, DetFace
! Energy and speed variables
REAL(KIND=4) :: Ein, Eout, Ethresh, Vin, Vout, Alpha
! Angle variables
REAL(KIND=4) :: Theta, PhiLab, PhiCM, MaxTheta, Omega
! Time variables
REAL(KIND=4) :: tmin, tmax, tprime
! Detector efficiency, pulse height factor, Response, and PScF
REAL(KIND=4) :: DetEff, PHF, Response, PScF
! Cross section variables
REAL(KIND = 4) :: xs, SigmaS, SigmaT, MFP
! Diff. Scattering Cross-Section Array
REAL(KIND=4), ALLOCATABLE :: DiffXS(:,:)
! Speed of Light (cm/ns)
REAL(KIND = 4), PARAMETER :: c = 29.97925
! Main OutPut Data Array
REAL(KIND=4), ALLOCATABLE :: Output(:,:)
REAL, PARAMETER :: TwoPi = 6.2831853
! Text variables
CHARACTER :: InpFile*80, DummyChar*1, OutFile*80

! Initialize variables here
R = 110.
D = 40.
L = 10.16
DetFace = 2.54
MaxTheta = 45.
NumColumns = 11
Ein = 14.051
Ethresh = 1.0
MFP = 3.0

Vin = NeutronSpeed(Ein)
tmin = R / (Vin * c)
tmax = (R + L) / (Vin * c)




!-------------------------------------------------------------
! Get xs file name from the command prompt
!-------------------------------------------------------------
CALL GETARG(1, InpFile)

!-------------------------------------------------------------
! Open input file, read in A, and read in the xs data
!-------------------------------------------------------------
OPEN(UNIT=1, FILE=TRIM(InpFile), ACTION="READ", STATUS="OLD")
```

214

```
DO I = 1, 1000
  READ(1, '(A)', IOSTAT=ErrCode) DummyChar
  IF (ErrCode < 0) EXIT
END DO

NumLines = I - 2
REWIND 1

! First column = angle (degrees), second column = xs (b/Sr)
ALLOCATE(DiffXS(NumLines,2))

READ(1, *) A, SigmaT
WRITE(*,*) A, SigmaT
DO I = 1, NumLines
  READ(1, *) DiffXS(I,1), DiffXS(I,2)
END DO

CLOSE(UNIT=1)

! Calculate the Alpha value using the atomic mass
Alpha = (REAL(A-1)/REAL(A+1))**2

Angles = 10*INT(MaxTheta)
ALLOCATE(OutPut(0:Angles,NumColumns))
OutPut = 0.

!-----------------------------------------------------------
! Main Program Loop - Loops over Detector Angle
!-----------------------------------------------------------
DO I = 0, Angles
  Theta = REAL(I)/10.
  OutPut(I,1) = Theta
  ! Convert Theta to Radians
  Theta = Theta * TwoPi / 360.
  ! Calculate D'
  Dout = SQRT(R**2 + (R-D)**2 - 2.*R*(R-D)*COS(Theta))

  ! With D', can now compute PhiLAB and Omega, the detector solid angle
  Omega = DetFace**2 / Dout**2
  OutPut(I, 8) = Omega
  PhiLab = ASIN(R * SIN(Theta) / Dout)
  OutPut(I, 2) = PhiLab * 360. / TwoPi

  ! Converts PhiLab to CM coordinates
  CALL FindCM(PhiLab, PhiCM, A)
  OutPut(I, 3) = PhiCM * 360. / TwoPi

  ! Find Diff. Scattering Cross-Section
  CALL FindDiffXS(PhiCM, DiffXS, xs)
  OutPut(I,4) = xs

  ! Now that the differential scattering cross-section and solid angle
  ! are known, the total probability of scattering to angle theta can be
  ! computed
  SigmaS = xs * Omega
  OutPut(I,9) = SigmaS / (SigmaT - SigmaS) * (EXP(-MFP*SigmaS/SigmaT)-EXP(-
MFP))

  ! Find (LAB) energy of the outgoing neutron
  CALL NeutronEnergy(Ein, PhiCM, Alpha, Eout)
  OutPut(I,5) = Eout
```

215

```fortran
  ! Calculate the Pulse Height Factor
  PHF = (Eout - Ethresh) / Ein
  OutPut(I,7) = PHF

  ! Find the time when the neutron arrives at the front face of the detector
  Vout = NeutronSpeed(Eout)
  tprime = (R - D) / (Vin * c) + Dout / (Vout * c)

  ! Check to see if t' > tmax, and if so, end the loop
  ! Might get bad results (e.g., negative efficiencies) otherwise.
  IF (tprime > tmax) CYCLE

  ! Find the maximum depth the neutron can penetrate into the detector
  ! before the end of the correlation time window.
  Lmax = Vout * c * (tmax - tprime)

  ! Calculate detector efficiency
  CALL DetectorEfficiency(Eout, Lmax, DetEff)
  OutPut(I,6) = DetEff


  ! Calculate the response function of theta - this is normalized to
  ! per source neutron since flux is not included.
  Response = OutPut(I,6) * OutPut(I,7) * OutPut(I,9)
  OutPut(I,10) = Response

  ! Lastly, find the Point Scatter Function
  PScF = Response / (OutPut(0,6) * OutPut(0,7))
  OutPut(I,11) = PScF

END DO ! Main I Loop

!------------------------------------------------------------
! Write Output array to a file
!------------------------------------------------------------
OutFile = InpFile(1:2) // ".out"
OPEN(UNIT=2, FILE=TRIM(OutFile), ACTION="WRITE", STATUS="REPLACE")

DO I = 0, Angles
  DO J = 1, NumColumns
    WRITE(2, '(ES14.6)', ADVANCE="NO") OutPut(I,J)
  END DO
  WRITE(2,*)
END DO

CLOSE(UNIT=2)


CONTAINS

!------------------------------------------------------------
! This function returns the neutron speed (as a fraction of
! c) given the input energy (in MeV).
!------------------------------------------------------------
FUNCTION NeutronSpeed(ZEin)
REAL :: NeutronSpeed
REAL(KIND=4), INTENT(IN) :: ZEin
NeutronSpeed = SQRT(2. * ZEin / 939.5656)

END FUNCTION
```

216

```
!----------------------------------------------------------------
! This subroutine returns the neutron energy (in the LAB system)
! after a collision, given the CM scattering angle and alpha.
!----------------------------------------------------------------
SubRoutine NeutronEnergy(ZEin, ZPhiCM, ZAlpha, ZEout)
REAL(KIND=4), INTENT(IN) :: ZEin, ZPhiCM, ZAlpha
REAL(KIND=4), INTENT(OUT) :: ZEout

ZEout = ((1 + ZAlpha) + (1 - ZAlpha)*COS(ZPhiCM))*ZEin / 2.

END SUBROUTINE


!----------------------------------------------------------------
! This subroutine converts the lab scattering angle to the COM
! scattering angle through a brute force iteration.
!----------------------------------------------------------------
SUBROUTINE FindCM(ZLab, ZCM, ZA)
REAL(KIND=4), INTENT(IN) :: ZLab
REAL(KIND=4), INTENT(OUT) :: ZCM
INTEGER, INTENT(IN) :: ZA
REAL(KIND = 4) :: ZRange, ZMin, Z, ZTest, Delta, DeltaMin, ZBest

ZRange = 2. * ZLab
ZBest = ZLab
ZMin = ZBest - ZRange

DO J = 1, 12
  DeltaMin = 10.
  DO K = 0, 20
    Z = ZMin + REAL(K)*ZRange/20.
    ZTest = ATAN(SIN(Z) / (1/REAL(ZA) + COS(Z)))
    Delta = ABS(ZLab - ZTest)
    IF (Delta < DeltaMin) THEN
      DeltaMin = Delta
      ZBest = Z
    END IF
  END DO

ZRange = ZRange / 5.
ZMin = ZBest - ZRange/2.

END DO

ZCM = ZBest

END SUBROUTINE

!----------------------------------------------------------------
! This Subroutine determines the differential scattering cross-section by
! interpolating the values from the input file.
!----------------------------------------------------------------
SUBROUTINE FindDiffXS(ZPhiCM, ZDiffXS, Zxs)
REAL(KIND=4), INTENT(IN) :: ZPhiCM, ZDiffXS(:,:)
REAL(KIND=4), INTENT(OUT) :: Zxs
REAL(KIND=4) :: ZAngle, ZAngleMin, ZAngleMax, ZxsMin, ZxsMax
REAL(KIND=4) :: ZAngleDiff, ZxsDiff

ZAngle = ZPhiCM * 360. / TwoPi
```

```
DO J = 1, NumLines
  IF (ZDiffXS(J,1) > ZAngle) THEN
    ZAngleMin = ZDiffXS(J-1,1)
    ZAngleMax = ZDiffXS(J,1)
    ZxsMin = ZDiffXS(J-1,2)
    ZxsMax = ZDiffXS(J,2)
    EXIT
  END IF
END DO

ZAngleDiff = ZAngleMax - ZAngleMin
ZxsDiff = ZxsMax - ZxsMin

Zxs = ZxsMin + (ZAngle - ZAngleMin)/ZAngleDiff*ZxsDiff

END SUBROUTINE

!-----------------------------------------------------------
! This Subroutine determines the intrinsic detector efficiency
! using equation 15.8b from Knoll, 3rd Ed.  Fits for the hydrogen
! cross-section are good for 1+ MeV.  The carbon fit is only accurate
! over the range 8.5 MeV < E < 14.1 MeV.
!-----------------------------------------------------------
SUBROUTINE DetectorEfficiency(ZEout, ZLmax, ZEff)
REAL(KIND=4), INTENT(IN) :: ZEout, ZLmax
REAL(KIND=4), INTENT(OUT) :: ZEff
REAL(KIND=4) :: SigH, SigC, NH, NC, Hxs, Cxs

NH = 0.08397 * 1.023 * 0.6022 / 1.
NC = 0.91603 * 1.023 * 0.6022 / 12.
SigH = EXP(1.6431845 - 0.324906*ZEout + 0.021717*ZEout**2-0.000632*ZEout**3)
SigC = 4.4929537 - 0.906192*ZEout + 0.0718101*ZEout**2 - 0.001809*ZEout**3
Hxs = NH * SigH
Cxs = NC * SigC

ZEff = Hxs * (1 - EXP(-(Hxs + Cxs)*ZLmax)) / (Hxs + Cxs)

END SUBROUTINE

END PROGRAM
```

Appendix C – The *MakeInp* Code and Supporting Files

The *MakeInp* Code

```
PROGRAM makeinp
! Version 2.20
! Written January 5th, 2009
! By Brandon Grogan
! At The Oak Ridge National Laboratory
! Last Edited: January 24th, 2009

IMPLICIT NONE
REAL(KIND=4) :: face, Theta, phi, radius, offset, nu, Depth, x0, y0
REAL(KIND=4) :: crys, rotate, xy, yx, phi2, zplane, detsep, FWHM, CorrFact
REAL(KIND=4), ALLOCATABLE :: Detect(:,:,:), SDEF(:,:)
REAL(KIND=4) :: pixangle, xcos, ycos, PDFSum, pixcenter
REAL(KIND=4) :: threshold, deadtime, window
INTEGER :: I, J, NumDet, FirstNum, DetMat, N, DetNum, P, NumPixels
INTEGER :: ErrCode, NumLines, L, M, FancyDet, Q, NumSDEFCards, NumThisLine
INTEGER(KIND=4) :: K, NPS
!FileBase=User Input File Name; FileNme=PoliMi input file
CHARACTER :: FileBase*8, FileNme*8, FileNum*2, Hist*16
CHARACTER :: PixelNum*2, TxtOut*64, OutNme*8, DatNme*8
CHARACTER :: FileNme3*16, ParamText*64, ThisLine*80
CHARACTER(80) :: Inp(10000)

CALL GetArg(1, ParamText)
IF (LEN(TRIM(ParamText)) == 0) THEN
  PRINT *, "Enter file name (8.3) of the parameter input file."
  READ (*,*) ParamText
END IF

CALL ReadParams()

!FileBase = "void.inp"
!offset = -55.
!radius = 110.
!face = 3.0607
!detsep = 3.20
!crys = 2.54
!Depth = 10.16
!FirstNum = 401
DetMat = FirstNum
!N = 4
!NPS = 12500000
!NumPixels=16
!FWHM=3.055
!NumSDEFCards = 46
!NumDet=32
!zplane=117.5
!FancyDet=1

ALLOCATE(SDEF(NumSDEFCards,5))

phi = 2.*ATAN(detsep/radius/2.)

!First Order of business: Calculate numbers for detector surfaces.
!Calculations based on source at origin (transform from there).
ALLOCATE(Detect(NumDet, 14, 5))
```

```
nu = -(REAL(NumDet-1)/2.)*phi

DO I = 0, NumDet-1
  J = I + 1
  ! Create the surface cards for the detector crystal
  x0 = radius * cos(nu)
  y0 = radius * sin(nu)
  Detect(J, 1, 1) = FirstNum + I*N
  Detect(J, 2, 1) = x0 + (crys/2.0) * sin(nu)
  Detect(J, 3, 1) = y0 - (crys/2.0) * cos(nu)
  Detect(J, 4, 1) = -crys/2.0
  Detect(J, 5, 1) = -crys*sin(nu)
  Detect(J, 6, 1) = crys*cos(nu)
  Detect(J, 7, 1) = 0.
  Detect(J, 8, 1) = Depth*cos(nu)
  Detect(J, 9, 1) = Depth*sin(nu)
  Detect(J, 10, 1) = 0.
  Detect(J, 11, 1) = 0.
  Detect(J, 12, 1) = 0.
  Detect(J, 13, 1) = crys
  Detect(J, 14, 1) = nu

  ! Create the surface cards for the detector housing (inner surface)
  x0 = (radius-0.0001) * cos(nu)
  y0 = (radius-0.0001) * sin(nu)
  Detect(J, 1, 2) = 1000 + FirstNum + I*N
  Detect(J, 2, 2) = x0 + ((face-0.22)/2.0) * sin(nu)
  Detect(J, 3, 2) = y0 - ((face-0.22)/2.0) * cos(nu)
  Detect(J, 4, 2) = -(face-0.22)/2.0
  Detect(J, 5, 2) = -(face-0.22)*sin(nu)
  Detect(J, 6, 2) = (face-0.22)*cos(nu)
  Detect(J, 7, 2) = 0.
  Detect(J, 8, 2) = (Depth+1.17)*cos(nu)
  Detect(J, 9, 2) = (Depth+1.17)*sin(nu)
  Detect(J, 10, 2) = 0.
  Detect(J, 11, 2) = 0.
  Detect(J, 12, 2) = 0.
  Detect(J, 13, 2) = (face-0.22)
  Detect(J, 14, 2) = nu

  ! Create the surface cards for the detector housing (outer surface)
  x0 = (radius-0.1) * cos(nu)
  y0 = (radius-0.1) * sin(nu)
  Detect(J, 1, 3) = 2000 + FirstNum + I*N
  Detect(J, 2, 3) = x0 + ((face-0.02)/2.0) * sin(nu)
  Detect(J, 3, 3) = y0 - ((face-0.02)/2.0) * cos(nu)
  Detect(J, 4, 3) = -(face-0.02)/2.0
  Detect(J, 5, 3) = -(face-0.02)*sin(nu)
  Detect(J, 6, 3) = (face-0.02)*cos(nu)
  Detect(J, 7, 3) = 0.
  Detect(J, 8, 3) = (Depth+1.37)*cos(nu)
  Detect(J, 9, 3) = (Depth+1.37)*sin(nu)
  Detect(J, 10, 3) = 0.
  Detect(J, 11, 3) = 0.
  Detect(J, 12, 3) = 0.
  Detect(J, 13, 3) = (face-0.02)
  Detect(J, 14, 3) = nu

  ! Create the surface cards for the PMT housing (inner surface)
  ! These values will define a right circular cylinder
  ! Format: Surface# RCC Transform# x0 y0 z0  x y z  r
```

```
   ! Format: 1 RCC Transform# 2 3 4  8 9 10  13
   ! x0, y0, z0 are coordinates to center of one of the bases
   ! x, y, z is the vector to the center of the other bace
   ! r is the radius
   x0 = (radius+Depth+0.0001) * cos(nu)
   y0 = (radius+Depth+0.0001) * sin(nu)
   Detect(J, 1, 4) = 3000 + FirstNum + I*N
   Detect(J, 2, 4) = x0
   Detect(J, 3, 4) = y0
   Detect(J, 4, 4) = 0.
   Detect(J, 5, 4) = 0.
   Detect(J, 6, 4) = 0.
   Detect(J, 7, 4) = 0.
   Detect(J, 8, 4) = 9.475*cos(nu)
   Detect(J, 9, 4) = 9.475*sin(nu)
   Detect(J, 10, 4) = 0.
   Detect(J, 11, 4) = 0.
   Detect(J, 12, 4) = 0.
   Detect(J, 13, 4) = (face-0.12)/2
   Detect(J, 14, 4) = nu

   ! Create the surface cards for the PMT housing (outer surface)
   ! Format: 1 RCC Transform# 2 3 4  8 9 10  13
   x0 = (radius+Depth+0.0001) * cos(nu)
   y0 = (radius+Depth+0.0001) * sin(nu)
   Detect(J, 1, 5) = 4000 + FirstNum + I*N
   Detect(J, 2, 5) = x0
   Detect(J, 3, 5) = y0
   Detect(J, 4, 5) = 0.
   Detect(J, 5, 5) = 0.
   Detect(J, 6, 5) = 0.
   Detect(J, 7, 5) = 0.
   Detect(J, 8, 5) = 9.525*cos(nu)
   Detect(J, 9, 5) = 9.525*sin(nu)
   Detect(J, 10, 5) = 0.
   Detect(J, 11, 5) = 0.
   Detect(J, 12, 5) = 0.
   Detect(J, 13, 5) = (face-0.02)/2
   Detect(J, 14, 5) = nu


   nu = nu + phi
END DO

!Get the name of the input file and create a subdirectory based on the name.
!CALL GETARG(1,FileBase)
TxtOut = "md " // FileBase(1:4)
CALL SYSTEM(TRIM(TxtOut))

!Open the input file and read it into the "Inp" array.
OPEN (UNIT=66, File=TRIM(FileBase), STATUS="OLD", ACTION="READ")
DO I = 1, 1000
  READ (66, '(A)', IOSTAT=ErrCode) Inp(I)
  IF (ErrCode < 0) EXIT
END DO
NumLines = I-1
CLOSE (UNIT=66)

TxtOut = FileBase(1:4) // ".bat"
OPEN (UNIT=99, FILE=TRIM(TxtOut), STATUS = "REPLACE", ACTION = "WRITE")
OPEN (UNIT=98, FILE="PP.bat", STATUS="REPLACE", ACTION="WRITE")
```

```fortran
phi2 = phi * 360. / 6.283185307

!The main loop of the program, loops over pixel number
DO P = 1, NumPixels

IF (P < 10) THEN
  WRITE (PixelNum, '(I1)') P
ELSE
  WRITE (PixelNum, '(I2)') P
END IF


!Create the pixel subdirectory
TxtOut = "md .\" // FileBase(1:4) // "\" // TRIM(PixelNum)
CALL SYSTEM(TRIM(TxtOut))

FileNme3 = "Start" // TRIM(PixelNum) // ".sh"
OPEN(UNIT=97, FILE=TRIM(FileNme3), STATUS = "REPLACE", ACTION = "WRITE")

!Writes to the 2 .bat files
WRITE (99, '(A)') "cd " // TRIM(PixelNum)
WRITE (98, '(A)') "cd " // TRIM(PixelNum)


WRITE (97, '(A)') "cd " // TRIM(PixelNum)
WRITE (97, '(A)') "dos2unix *.*"

!J loops the subsample number (it's actually J+1).  K gives the actual SS
!number.
DO J = 0, N-1
K = J+1
IF (K < 10) THEN
  WRITE (FileNum, '(I1)') K
ELSE
  WRITE (FileNum, '(I2)') K
END IF

!PomiMi input file name
FileNme = FileBase(1:2) // TRIM(PixelNum) // TRIM(FileNum) // ".i"
!Output file name (PoliMi Output - shows subsample)
OutNme = FileBase(1:2) // TRIM(PixelNum) // TRIM(FileNum) // ".o"
!Dat file name
DatNme = FileBase(1:2) // TRIM(PixelNum) // TRIM(FileNum) // ".d"

!Write PoliMi execution statement to the batch file
WRITE (97, '(A)') "mcnp-polimi inp=" // TRIM(FileNme) // " out=" // &
& TRIM(OutNme) // " dumn1=" // TRIM(DatNme)

WRITE(98, '(A, I12, I10, F10.0, F8.4)') "PoliMiPP " // DatNme // " 0 ", NPS,
INT(window), &
& deadtime, threshold

WRITE (99, '(A)') "mcnp-polimi inp=" // TRIM(FileNme) // " out=" // &
& TRIM(OutNme) // " dumn1=" // TRIM(DatNme)


OPEN (UNIT=1, FILE = TRIM(FileNme), STATUS = "REPLACE", ACTION = "WRITE")

DO L = 1, NumLines

IF (TRIM(INP(L)) == "c 111111") THEN
```

```fortran
! Put #det nums. here

! Fancy Detectors
IF (FancyDet == 1) THEN
DO M = 3, 5, 2
 WRITE(1, '(A)', ADVANCE="NO") "       "
 DO I = 1, NumDet
   DetNum = INT(Detect(I,1,M)) + J
   WRITE (1, '(I5)', ADVANCE="NO") DetNum
   IF (MOD(I,12) == 0 .AND. I .NE. NumDet) THEN
     WRITE(1, *)
     WRITE(1,'(A)', ADVANCE="NO") "       "
   END IF
 END DO
 WRITE(1, *)
END DO

ELSE
 WRITE(1, '(A)', ADVANCE="NO") "       "
 DO I = 1, NumDet
   DetNum = INT(Detect(I,1,1)) + J
   WRITE (1, '(I5)', ADVANCE="NO") DetNum
   IF (MOD(I,12) == 0 .AND. I .NE. NumDet) THEN
     WRITE(1, *)
     WRITE(1,'(A)', ADVANCE="NO") "       "
   END IF
 END DO
 WRITE(1, *)

END IF


! Fancy Detectors

IF (FancyDet == 1) THEN
  WRITE (1, '(A)') "       #961 966 968"
END IF

WRITE (1, '(A)') "        imp:n,p=1"

WRITE (1, '(A)') "c"
WRITE (1, '(A)') "c Detectors"
WRITE (1, '(A)') "c"

!Detector Cell Cards
DO I = 1, NumDet
  DetNum = INT(Detect(I,1,1)) + J
  WRITE (1, '(I3, I4, F9.4, I5, A, I2)') DetNum, FirstNum, -1.023, -DetNum,&
  &" imp:n,p=1 $ detector ", I
END DO

! Fancy Detectors
IF (FancyDet == 1) THEN

WRITE (1, '(A)') "c"
WRITE (1, '(A)') "c Detector Housing"
WRITE (1, '(A)') "c"

DO I = 1, NumDet
  DetNum = INT(Detect(I,1,2)) + J
  WRITE (1, '(I4, I4, A, 2I6, 2(A,I4) A, I2)') DetNum, 0, "          ", -
DetNum,&
```

223

```
  &DetNum-1000, " #", DetNum+2000, " #", DetNum+3000, " imp:n,p=1 $ detector ",
I
END DO

WRITE (1, '(A)') "c"

DO I = 1, NumDet
  DetNum = INT(Detect(I,1,3)) + J
  WRITE (1, '(I4, I4, F9.4, 2I6, 2(A,I4) A, I2)') DetNum, FirstNum+2, -2.70, -
DetNum,&
  &DetNum-1000, " #", DetNum+1000, " #", DetNum+2000, " imp:n,p=1 $ detector ",
I
END DO

WRITE (1, '(A)') "c"
WRITE (1, '(A)') "c Photomultiplier Tubes"
WRITE (1, '(A)') "c"

DO I = 1, NumDet
  DetNum = INT(Detect(I,1,4)) + J
  WRITE (1, '(I4, I4, F9.4, I6, A, I2)') DetNum, FirstNum+3, -0.50, -DetNum,&
  &"                imp:n,p=1 $ detector ", I
END DO

WRITE (1, '(A)') "c"

DO I = 1, NumDet
  DetNum = INT(Detect(I,1,5)) + J
  WRITE (1, '(I4, I4, F9.4, 2I6, A, I2)') DetNum, FirstNum+1, -8.75, -DetNum,&
  &DetNum-1000, "              imp:n,p=1 $ detector ", I
END DO

END IF

! Fancy Detectors
IF (FancyDet == 1) THEN

WRITE (1, '(A)') "c"
WRITE (1, '(A)') "c Detector Frame and Mounting Arm"
WRITE (1, '(A)') "c"

! WRITE (1, '(I4, I4, F9.4, 6I6)') 951, FirstNum+2, -2.70, 951, -952,&
! &953, -954, 957, -958
! WRITE (1, '(A)') "        imp:n,p=1 $ detector frame upper"
! WRITE (1, '(I4, I4, F9.4, 6I6)') 952, FirstNum+2, -2.70, 951, -952,&
! &955, -956, 957, -958
! WRITE (1, '(A)') "        imp:n,p=1 $ detector frame lower"
WRITE (1, '(I4, I4, F9.4, 2I6)') 961, FirstNum+2, -2.70, -961, 962
WRITE (1, '(A)') "        imp:n,p=1 $ detector arm"
! WRITE (1, '(I4, I4, F9.4, I6)') 963, FirstNum+2, -2.70, -963
! WRITE (1, '(A)') "        imp:n,p=1 $ detector arm"
! WRITE (1, '(I4, I4, F9.4, I6)') 964, FirstNum+2, -2.70, -964
! WRITE (1, '(A)') "        imp:n,p=1 $ detector arm"

WRITE (1, '(A)') "c"
WRITE (1, '(A)') "c DT Generator Body"
WRITE (1, '(A)') "c"
WRITE (1, '(I4, I4, F9.4, 2I6)') 966, FirstNum+4, -8.60, -966, 967
WRITE (1, '(A)') "        imp:n,p=1 $ DT Generator Tube"
WRITE (1, '(I4, I4, F9.4, 2I6)') 967, FirstNum+2, -2.70, -968, 969
WRITE (1, '(A)') "        imp:n,p=1 $ PMT Housing"
```

224

```
WRITE (1, '(I4, A, I6)') 968, "   0          ", -967
WRITE (1, '(A)') "       imp:n,p=1 $ DT Generator Interior"
WRITE (1, '(I4, I4, F9.4, 2I6)') 969, FirstNum+3, -0.50, -969, 966
WRITE (1, '(A)') "       imp:n,p=1 $ PMT Interior"

END IF


CYCLE !Go read the next line of input
!IF 111111
END IF


IF (TRIM(INP(L)) == "c 222222") THEN
WRITE (1, '(A)') "c"
WRITE (1, '(A)') "c Detectors"
WRITE (1, '(A)') "c"

! Detector surface cards go here
DO I = 1, NumDet
  DetNum = INT(Detect(I,1,1)) + J
  WRITE (1, '(I3, I4, A5, 5F12.6)') DetNum,  J+FirstNum, " BOX ",
Detect(I,2,1), Detect(I,3,1), &
  &Detect(I,4,1), Detect(I,5,1), Detect(I,6,1)
  WRITE (1, '(A6, 6F12.6)') "      ", Detect(I,7,1), Detect(I,8,1),
Detect(I,9,1), &
  &Detect(I,10,1), Detect(I,11,1), Detect(I,12,1)
  Theta = Detect(I,14,1)*360/6.283185307
  WRITE (1, '(A6, F12.6, A, F12.6)') "      ", Detect(I,13,1), " $ Angle = ",
Theta
END DO

! Fancy Detectors
IF (FancyDet == 1) THEN

WRITE (1, '(A)') "c"
WRITE (1, '(A)') "c Detector Housing"
WRITE (1, '(A)') "c"

DO M = 2,3
 DO I = 1, NumDet
   DetNum = INT(Detect(I,1,M)) + J
   WRITE (1, '(I4, I4, A5, 5F12.6)') DetNum,  J+FirstNum, " BOX ",
Detect(I,2,M), Detect(I,3,M), &
   &Detect(I,4,M), Detect(I,5,M), Detect(I,6,M)
   WRITE (1, '(A6, 6F12.6)') "      ", Detect(I,7,M), Detect(I,8,M),
Detect(I,9,M), &
   &Detect(I,10,M), Detect(I,11,M), Detect(I,12,M)
   WRITE (1, '(A6, F12.6)') "      ", Detect(I,13,M)
 END DO
END DO

! Fancy Detectors

WRITE (1, '(A)') "c"
WRITE (1, '(A)') "c Photomultiplier Tubes"
WRITE (1, '(A)') "c"

DO M = 4,5
 DO I = 1, NumDet
   DetNum = INT(Detect(I,1,M)) + J
```

```fortran
    WRITE (1, '(I4, I4, A5, 3F12.6)') DetNum,  J+FirstNum, " RCC ",
Detect(I,2,M), Detect(I,3,M), &
    &Detect(I,4,M)
    WRITE (1, '(A6, 6F12.6)') "      ", Detect(I,8,M), Detect(I,9,M),
Detect(I,10,M), &
    &Detect(I,13,M)
 END DO
END DO

END IF

! Fancy Detectors

IF (FancyDet == 1) THEN

WRITE (1, '(A)') "c"
WRITE (1, '(A)') "c Detector Mount and Support Arm"
WRITE (1, '(A)') "c"


! WRITE (1, '(I4, I4, A5, F12.6)') 951, J+FirstNum, "  SO ", radius+4.
! WRITE (1, '(I4, I4, A5, F12.6)') 952, J+FirstNum, "  SO ", radius+14.
! WRITE (1, '(I4, I4, A5, F12.6)') 953, J+FirstNum, "  PZ ", -face/2. - 0.81
! WRITE (1, '(I4, I4, A5, F12.6)') 954, J+FirstNum, "  PZ ", -face/2. - 0.01
! WRITE (1, '(I4, I4, A5, F12.6)') 955, J+FirstNum, "  PZ ", face/2. + 0.01
! WRITE (1, '(I4, I4, A5, F12.6)') 956, J+FirstNum, "  PZ ", face/2. + 0.81

! Theta = REAL(NumDet+1)*phi/2.
! WRITE (1, '(I4, I4, A, 3F12.6)') 957, J+FirstNum, "   P 0 0 0  0 0 100 ",
radius, &
! & radius*TAN(-Theta), 0.
! WRITE (1, '(I4, I4, A, 3F12.6)') 958, J+FirstNum, "   P 0 0 0  0 0 100 ",
radius, &
! & radius*TAN(Theta), 0.

WRITE (1, '(A)') "c"
WRITE (1, '(I4, I4, A5, 3F12.6, A)') 961,  FirstNum-1, " BOX ", radius+1., -
82.5, &
&face/2. + 6., " 10.16 0 0  0 165 0  0 0 11.113"

WRITE (1, '(I4, I4, A5, 3F12.6, A)') 962,  FirstNum-1, " BOX ", radius+1.32, -
82.5, &
&face/2. + 7.27, "  9.52 0 0  0 165 0  0 0  9.52"

! WRITE (1, '(I4, I4, A5, 3F12.6, A)') 963, FirstNum-1, " RCC ", radius+6.08, -
20., &
! &face/2. + 0.81, " 0 0 5.19 1.5"
! WRITE (1, '(I4, I4, A5, 3F12.6, A)') 964, FirstNum-1, " RCC ", radius+6.08,
20., &
! &face/2. + 0.81, " 0 0 5.19 1.5"


WRITE (1, '(A)') "c"
WRITE (1, '(A)') "c DT Generator Body"
WRITE (1, '(A)') "c"
WRITE (1, '(I4, I4, A)') 966, FirstNum-1, " RCC  0 -30.48 0   0    90.54 0
3.81"
WRITE (1, '(I4, I4, A)') 967, FirstNum-1, " RCC  0 -30.28 0   0    90.14 0
3.61"
WRITE (1, '(I4, I4, A)') 968, FirstNum-1, " RCC -15   0   0 11.18 0    0
3.81"
```

226

```
          WRITE (1, '(I4, I4, A)') 969, FirstNum-1, " RCC -14.4 0   0   9.98  0     0
          3.61"

          END IF !Fancy Detectors

          CYCLE !Go read the next line of input
          !IF 222222
          END IF

          IF (TRIM(INP(L)) == "c 333333") THEN
          ! Detector Material Specification
          WRITE (1, '(A,I3,A)') "m", FirstNum, " NLIB=60C $ organic scintillator"
          WRITE (1, '(A)') "      6000 10"
          WRITE (1, '(A)') "      1001 11"

          ! Fancy Detectors

          IF (FancyDet == 1) THEN

          WRITE (1, '(A,I3,A)') "m", FirstNum+1, " NLIB=70C $ Mu Metal, rho = 8.75 g/cm3"
          WRITE (1, '(A)') "      6000   -0.000200"
          WRITE (1, '(A)') "      14028  -0.003228    14029  -0.000163"
          WRITE (1, '(A)') "      14030  -0.000109    25055  -0.005000"
          WRITE (1, '(A)') "      26054  -0.008734    26056  -0.136983"
          WRITE (1, '(A)') "      26057  -0.003165    26058  -0.000418"
          WRITE (1, '(A)') "      28058  -0.544640    28060  -0.209760"
          WRITE (1, '(A)') "      28061  -0.009120    28062  -0.029040"
          WRITE (1, '(A)') "      28064  -0.007440"
          WRITE (1, '(A)') "      42092  -0.006233"
          WRITE (1, '(A)') "      42094  -0.003885    42095  -0.006686"
          WRITE (1, '(A)') "      42096  -0.007006    42097  -0.004011"
          WRITE (1, '(A)') "      42098  -0.010135    42100  -0.004044"

          WRITE (1, '(A,I3,A)') "m", FirstNum+2, " NLIB=70C $ Aluminum 6061-T6, rho=2.70
          g/cm3"
          WRITE (1, '(A)') "      12024  -0.007899    12025  -0.001000"
          WRITE (1, '(A)') "      12026  -0.001101    13027  -0.980000"
          WRITE (1, '(A)') "      14028  -0.005534    14029  -0.000280"
          WRITE (1, '(A)') "      14030  -0.000186"
          WRITE (1, '(A)') "      24050  -0.000087    24052  -0.001676"
          WRITE (1, '(A)') "      24053  -0.000190    24054  -0.000047"
          WRITE (1, '(A)') "      29063  -0.001383    29065  -0.000617"

          WRITE (1, '(A,I3,A)') "m", FirstNum+3, " NLIB=70C $ Photomultiplier Tube volume
          average, rho ~ 0.5 g/cm3"
          WRITE (1, '(A)') "      1001   -0.051892    1002   -0.000008"
          WRITE (1, '(A)') "      6000   -0.303200"
          WRITE (1, '(A)') "      8016   -0.096600    9019   -0.076000"
          WRITE (1, '(A)') "      13027  -0.150000"
          WRITE (1, '(A)') "      14028  -0.112797    14029  -0.005712"
          WRITE (1, '(A)') "      14030  -0.003791"
          WRITE (1, '(A)') "      29063  -0.138340    29065  -0.061660"

          WRITE (1, '(A,I3,A)') "m", FirstNum+4, " NLIB=70C $ Common Brass, rho = 8.6
          g/cm3"
          WRITE (1, '(A)') "      29063  -0.435771    29065  -0.194229"
          WRITE (1, '(A)') "      30000  -0.370000"
          END IF


          WRITE (1, '(A, I3, F8.2, A, F8.2)') "*TR", FirstNum-1, offset, " 0 ", zplane
```

```
! Detector Transforms go here
DO I = 1, 2*N-1, 2
K = FirstNum + (I+1)/2 - 1
  rotate = phi*360*REAL(I-N)/REAL(N)/6.283185307/2.
  xy = 90.-rotate
  yx = 90.+rotate
    WRITE (1, '(A, I3, F8.2, A, 3F10.6, A, 2F10.6, A)') "*TR", K, offset, " 0
", zplane, -rotate, xy
    WRITE (1, '(A, 2F10.6, A)')  "        90 ", yx, -rotate, " 90 90 90 0"
END DO

WRITE (1, '(A,F8.2,A)') "*TR999", offset, " 0 0"

! Calculate the angle to the center of the pixel
!IF (NumPixels == 8) THEN
!  pixangle = -29.03 + 6.11*REAL(P)
!ELSE IF (NumPixels == 16) THEN
!  pixangle = -25.975 + 3.055*REAL(P)
!ELSE
!  PRINT *, "Unknown pixel configuration"
!  PRINT *, "Only 8 or 16 pixels are supported."
!  STOP
!END IF
pixangle = pixcenter - (REAL(NumPixels + 1)/2. - REAL(P))*FWHM

! Now compute the values for the SI & SP cards for the SDEF
! CorrFact is a correction factor that narrows the pixel slightly.
! This correction is necessary to account for the fact that the SDEF card is a
! discrete function.  Without it, the pixels will be slightly more tail heavy
! than expected and produce a 'wavy' void measurement.
CorrFact = REAL(NumSDEFCards - 2) / REAL(NumSDEFCards - 1)
PDFSum = 0.
DO Q = 1, NumSDEFCards
  SDEF(Q, 1) = REAL(Q-1) * 90. / REAL(NumSDEFCards-1)
  SDEF(Q, 2) = COS(FWHM*SDEF(Q,1)/90.*6.283185307/360.*CorrFact)
  SDEF(Q, 3) = (COS(SDEF(Q,1)*6.283185307/360.)) ** 2
  SDEF(Q, 4) = REAL(Q**2 - (Q-1)**2)
  PDFSum = PDFSum + SDEF(Q,3) * SDEF(Q,4)
END DO

DO Q = 1, NumSDEFCards
  SDEF(Q, 5) = SDEF(Q, 3) * SDEF(Q, 4) / PDFSum
END DO

xcos = COS(pixangle*6.283185307/360.)
ycos = COS((90.-pixangle)*6.283185307/360.)
WRITE (1, '(A,F8.2,A,F8.2,A)') "SDEF pos=", offset, " 0 ", zplane," ERG=14.100
DIR=d1 "
WRITE (1, '(A, 2F10.6, A)') "        PAR=1 VEC=", xcos, ycos, " 0"

! Write out the SI Card
WRITE(1, '(A,4F12.8)') "SI1    -1           ", SDEF(NumSDEFCards,2),
SDEF(NumSDEFCards-1,2), &
& SDEF(NumSDEFCards-2,2), SDEF(NumSDEFCards-3,2)
WRITE(1, '(A)', ADVANCE="NO") "         "
NumThisLine = 0
DO Q = NumSDEFCards-4, 1, -1
  WRITE(1, '(F12.8)', ADVANCE="NO") SDEF(Q,2)
  NumThisLine = NumThisLine + 1
  IF (NumThisLine == 5 .AND. Q /= 1) THEN
    NumThisLine = 0
```

228

```
      WRITE(1, *)
      WRITE(1, '(A)', ADVANCE="NO") "        "
    END IF
  END DO
END DO
WRITE(1, *)

! Now write the SP Card
WRITE(1, '(A,4F12.8)') "SP1     0           ", SDEF(NumSDEFCards,5),
SDEF(NumSDEFCards-1,5), &
& SDEF(NumSDEFCards-2,5), SDEF(NumSDEFCards-3,5)
WRITE(1, '(A)', ADVANCE="NO") "        "
NumThisLine = 0
DO Q = NumSDEFCards-4, 1, -1
  WRITE(1, '(F12.8)', ADVANCE="NO") SDEF(Q,5)
  NumThisLine = NumThisLine + 1
  IF (NumThisLine == 5 .AND. Q /= 1) THEN
    NumThisLine = 0
    WRITE(1, *)
    WRITE(1, '(A)', ADVANCE="NO") "        "
  END IF
END DO

WRITE(1, *)
WRITE (1, '(A)') "phys:n J 20."
WRITE (1, '(A)') "phys:p 0 1 1"
WRITE (1, '(A,F12.4)') "cut:n ", window*0.1
WRITE (1, '(A,F12.4,A)') "cut:p ", window*0.1," J 0"
WRITE (1, '(A)') "rdum 0.000150 0.001000"
! IDUM Card
WRITE (1, '(A, I3)') "idum 0 1 2 1 J 1 ", NumDet
WRITE (1, '(A)', Advance="NO") "        "
DO I = 1, NumDet
  DetNum = INT(Detect(I,1,1)) + J
  WRITE (1, '(I4)', ADVANCE="NO") DetNum
  IF (MOD(I,10) == 0) THEN
    WRITE(1, *)
    IF(I .NE. NumDet) WRITE(1,'(A)', ADVANCE="NO") "        "
  ELSEIF (I == NumDet) THEN
    WRITE(1,*)
  END IF
END DO
! DBCN Card
WRITE (1, '(A)', ADVANCE="NO") "DBCN 74822145211985 6J "
K = 1 + NPS * J * P
WRITE (Hist, '(I12)') K
Hist = TRIM(Hist)
! PRINT *, Hist
WRITE (1, '(A)', ADVANCE="NO") Hist
WRITE (1, '(A)') " 4J 12851"
WRITE (1, '(A,I12)') "nps ", NPS
WRITE (1, '(A)') "FILES 21 DUMN1"

CYCLE
!IF 333333
END IF

WRITE (1, '(A)') Inp(L)

!End of L Loop (L = Line of input deck)
END DO
CLOSE (UNIT=1)
```

229

```fortran
TxtOut = "move " // TRIM(FileNme) // " .\" // FileBase(1:4) &
& // "\" // TRIM(PixelNum)
CALL SYSTEM(TRIM(TxtOut))


!End of J (subsample) loop
END DO

WRITE (97, '(A)') "rm -f runtp*"
CLOSE(UNIT=97)
TxtOut = "move " // TRIM(FileNme3) // " .\" // FileBase(1:4)
CALL SYSTEM(TRIM(TxtOut))

WRITE (98, '(A, 3I4, 2F10.5)') "JoinSS " // FileBase(1:2) // TRIM(PixelNum), N,
NumDet, &
& FirstNum, radius, DetSep
WRITE (98, '(A)') "copy " // FileBase(1:2) // TRIM(PixelNum) &
& // ".peaks .."
WRITE (98, '(A)') "cd .."

WRITE (99, '(A)') "cd .."

!End P (pixels) loop
END DO

WRITE (98, '(A, I4)') "JoinPixels " // FileBase(1:2) , NumPixels
WRITE (98, '(A)') "pause"
CLOSE(UNIT=98)

WRITE (99, '(A)') "pause"
CLOSE(UNIT=99)



TxtOut = "move " // FileBase(1:4) // ".bat .\" // FileBase(1:4)
CALL SYSTEM(TRIM(TxtOut))
TxtOut = "move PP.bat .\" // FileBase(1:4)
CALL SYSTEM(TRIM(TxtOut))

CONTAINS

SUBROUTINE ReadParams()

OPEN(UNIT=101, FILE=TRIM(ParamText), ACTION="READ", STATUS="OLD",
IOSTAT=ErrCode)
IF (ErrCode > 0) STOP "Could not find specified input parameter file."

DO I = 1, 10000
  READ(101, '(A)', IOSTAT=ErrCode) ThisLine
  IF (ErrCode < 0) EXIT !End of parameter file

  IF (ThisLine(1:4) == "BASE") THEN
  DO J = 5, 80
    IF (ThisLine(J:J) /= " ") THEN
      FileBase = TRIM(ThisLine(J:80))
      EXIT
    END IF
  END DO
  ELSE IF (ThisLine(1:3) == "NPS") THEN
```

```
      READ(ThisLine(4:80), '(I11)') NPS
   ELSE IF (ThisLine(1:5) == "NUMSS") THEN
     READ(ThisLine(6:80), '(I11)') N
   ELSE IF (ThisLine(1:4) == "SRCX") THEN
     READ(ThisLine(5:80), '(F16.8)') offset
   ELSE IF (ThisLine(1:6) == "ZPLANE") THEN
     READ(ThisLine(7:80), '(F16.8)') zplane
   ELSE IF (ThisLine(1:6) == "STODET") THEN
     READ(ThisLine(7:80), '(F16.8)') radius
   ELSE IF (ThisLine(1:6) == "NUMDET") THEN
     READ(ThisLine(7:80), '(I11)') NumDet
   ELSE IF (ThisLine(1:6) == "1STDET") THEN
     READ(ThisLine(7:80), '(I11)') FirstNum
   ELSE IF (ThisLine(1:6) == "DETDIM") THEN
     READ(ThisLine(7:80), '(F16.8)') crys
   ELSE IF (ThisLine(1:6) == "DETDEP") THEN
     READ(ThisLine(7:80), '(F16.8)') Depth
   ELSE IF (ThisLine(1:4) == "FACE") THEN
     READ(ThisLine(5:80), '(F16.8)') face
   ELSE IF (ThisLine(1:6) == "DETSEP") THEN
     READ(ThisLine(7:80), '(F16.8)') detsep
   ELSE IF (ThisLine(1:5) == "FANCY") THEN
     READ(ThisLine(6:80), '(I11)') FancyDet
   ELSE IF (ThisLine(1:6) == "NUMPIX") THEN
     READ(ThisLine(7:80), '(I11)') NumPixels
   ELSE IF (ThisLine(1:6) == "PIXCEN") THEN
     READ(ThisLine(7:80), '(F16.8)') pixcenter
   ELSE IF (ThisLine(1:4) == "FWHM") THEN
     READ(ThisLine(5:80), '(F16.8)') FWHM
   ELSE IF (ThisLine(1:6) == "SIBINS") THEN
     READ(ThisLine(7:80), '(I11)') NumSDEFCards
   ELSE IF (ThisLine(1:6) == "WINDOW") THEN
     READ(ThisLine(7:80), '(F16.8)') window
   ELSE IF (ThisLine(1:6) == "DEADTM") THEN
     READ(ThisLine(7:80), '(F16.8)') deadtime
   ELSE IF (ThisLine(1:6) == "THRESH") THEN
     READ(ThisLine(7:80), '(F16.8)') threshold
   END IF
END DO

CLOSE(UNIT=101)

END SUBROUTINE ReadParams

END PROGRAM makeinp
```

A Sample Parameters File

```
-! The input name of the base MCNP deck
BASE Barr.inp
! The number of source particles
NPS 25000000
! The number of subsamples
NUMSS 4

! The x position of the source, in cm.
SRCX  -55.0
! The z position of the source and the detector array
ZPLANE 0.
```

```
! The source to detector distance.  Standard NMIS distances are 85, 110,
! and 217.17 cm.
STODET 110.0


! The number of imaging detectors in the array
NUMDET 32
! The cell number of the first imaging detector.  1STDET + NUMSS * NUMDET must
! be < 1000 or the resulting decks crash.
1STDET 401
! The size of the front face of the plastic scintillator crystal.
! Only crystals with a square face are supported at this time.
DETDIM 2.54
! The depth (long dimension) of the scintillator crystal
DETDEP 10.16
! The dimension of the front face of the detector housing.
! Only square facess are supported.
FACE 3.0607
! The separation between adjacent detector centers, in cm.
! DETSEP =~ STODET * TAN(Angle between detector centers)
DETSEP 3.20
! Setting FANCY to 1 will generate detectors with housings and photo-multiplier
! tubes.  Setting it to 0 omits these.  The RSICC version of mcnp-polimi will
! probably not be able to handle FANCY=1 with > 16 detectors.
FANCY 0


! The number of DT Generator Pixels
NUMPIX 8
! The offset of the center of the pixels from the center of the detector array.
PIXCEN -1.535
! The full width at half maximum of the pixels.
FWHM 6.11
! The number of bins that will be used on the SI and SP cards to define the
! initial neutron directions.  More bins will more closely approximate a
! continuous function.
SIBINS 46


! The size of the correlation window, in ns.
WINDOW 256.
! The detector deadtime, in ns.
DEADTM 35.
The neutron threshold of the array detectors, in MeV.
THRESH 1.0
```

An Example Base Input Deck

```
Barrel Simulation Base File
C
C       CELL CARDS *********************************************************
c
c
c Object being scanned - Barrel with iron pipes and steel shot
c
101    3 -7.86       -101 102                                   IMP:N,P=1
102    0             -102 #111 #121 #131 #141 #151              IMP:N,P=1
103    6 -0.25       -103  104                                  IMP:N,P=1
104    0             -104                                       IMP:N,P=1
111    2 -0.95       -102 -111 -152                             IMP:N,P=1
121    5 -18.90      -102  121 -122 -152                        IMP:N,P=1
131    3 -7.86       -102  131 -132 -153                        IMP:N,P=1
```

```
141    3 -7.86       -102  141 -142 -153                           IMP:N,P=1
151    3 -4.75       -102  132 -151 #141                           IMP:N,P=1
c
c
c Problem boundary and 'everything else' cell
c
99     0             99                                            imp:n,p=0
98     0    -99 101 103
c 111111

C        BLANK LINE DELIMITER --------------------------------------------------
C
C        SURFACE CARDS ******************************************************
C
c
c Problem Boundary
99    BOX    -400 -400 -100 800 0 0 0 800 0 0 0 400
c
c
c Object Being Scanned - Barrel with iron pipes and steel shot
c
101 3 RCC  0 0 10.0    0 0 47.00   17.88
102 3 RCC  0 0 10.1    0 0 46.80   17.78
103 3 RCC  0 0 0       0 0 10      16.88
104 3 RCC  0 0 0       0 0 10      6.72
c
111 3 CZ   1.27
121 3 CZ   4.445
122 3 CZ   6.35
131 3 CZ   7.62
132 3 CZ   8.41375
141 3 CZ   10.16
142 3 CZ   10.95375
151 3 PZ   23
152 3 PZ   25.24
153 3 PZ   28.0
c
c 222222

C        BLANK LINE DELIMITER --------------------------------------------------
C
C
C        DATA CARDS ******************************************************
C
MODE n p
c
c Geometric Transforms
c
*TR3  0 0 -17   0.000000 90.000000 90  90.000000  0.000000 90 90 90 0
TR201  -55.00 0   0
C
C MATERIALS
C
c
c Polyethylene, Rho=0.95 g/cc
c
M1    NLIB=60c
      6000   2
      1001   4
c
c Graphite, Rho = 2.20 g/cc
```

233

```
c
M2    NLIB=60c
      6000   1
c
c Elemental Iron, Rho = 7.86 g/cc
c
M3    NLIB=60c
      26054  -0.058500
      26056  -0.917500
      26057  -0.021200
      26058  -0.002800
c
c Elemental Lead, Rho = 11.6 g/cc
c
M4    NLIB=60c
      82206    -0.244400
      82207    -0.224100
      82208    -0.531500
c
c Depleted Uranium, Rho = 18.9
c
M5    NLIB=60c
       92235   -0.002
       92238   -0.998
c
c Cellulose (Wood)
c
M6    NLIB=70c
        1001   0.499925
        1002   0.000075
        6000   0.25
        8016   0.25
c
c 333333
```

A Sample MCNP Input Deck Created by *MakeInp*

```
Barrel Simulation Base File
C
C       CELL CARDS *********************************************************
c
c
c Object being scanned - Barrel with iron pipes and steel shot
c
101    3 -7.86       -101 102                                        IMP:N,P=1
102    0             -102 #111 #121 #131 #141 #151                   IMP:N,P=1
103    6 -0.25       -103  104                                       IMP:N,P=1
104    0             -104                                            IMP:N,P=1
111    2 -0.95       -102 -111 -152                                  IMP:N,P=1
121    5 -18.90      -102  121 -122 -152                             IMP:N,P=1
131    3 -7.86       -102  131 -132 -153                             IMP:N,P=1
141    3 -7.86       -102  141 -142 -153                             IMP:N,P=1
151    3 -4.75       -102  132 -151 #141                             IMP:N,P=1
c
c
c Problem boundary and 'everything else' cell
c
99     0                  99                                         imp:n,p=0
98     0    -99 101 103
```

```
            401   405   409   413   417   421   425   429   433   437   441   445
            449   453   457   461   465   469   473   477   481   485   489   493
            497   501   505   509   513   517   521   525
        imp:n,p=1
c
c Detectors
c
401 401  -1.0230  -401 imp:n,p=1 $ detector  1
405 401  -1.0230  -405 imp:n,p=1 $ detector  2
409 401  -1.0230  -409 imp:n,p=1 $ detector  3
413 401  -1.0230  -413 imp:n,p=1 $ detector  4
417 401  -1.0230  -417 imp:n,p=1 $ detector  5
421 401  -1.0230  -421 imp:n,p=1 $ detector  6
425 401  -1.0230  -425 imp:n,p=1 $ detector  7
429 401  -1.0230  -429 imp:n,p=1 $ detector  8
433 401  -1.0230  -433 imp:n,p=1 $ detector  9
437 401  -1.0230  -437 imp:n,p=1 $ detector 10
441 401  -1.0230  -441 imp:n,p=1 $ detector 11
445 401  -1.0230  -445 imp:n,p=1 $ detector 12
449 401  -1.0230  -449 imp:n,p=1 $ detector 13
453 401  -1.0230  -453 imp:n,p=1 $ detector 14
457 401  -1.0230  -457 imp:n,p=1 $ detector 15
461 401  -1.0230  -461 imp:n,p=1 $ detector 16
465 401  -1.0230  -465 imp:n,p=1 $ detector 17
469 401  -1.0230  -469 imp:n,p=1 $ detector 18
473 401  -1.0230  -473 imp:n,p=1 $ detector 19
477 401  -1.0230  -477 imp:n,p=1 $ detector 20
481 401  -1.0230  -481 imp:n,p=1 $ detector 21
485 401  -1.0230  -485 imp:n,p=1 $ detector 22
489 401  -1.0230  -489 imp:n,p=1 $ detector 23
493 401  -1.0230  -493 imp:n,p=1 $ detector 24
497 401  -1.0230  -497 imp:n,p=1 $ detector 25
501 401  -1.0230  -501 imp:n,p=1 $ detector 26
505 401  -1.0230  -505 imp:n,p=1 $ detector 27
509 401  -1.0230  -509 imp:n,p=1 $ detector 28
513 401  -1.0230  -513 imp:n,p=1 $ detector 29
517 401  -1.0230  -517 imp:n,p=1 $ detector 30
521 401  -1.0230  -521 imp:n,p=1 $ detector 31
525 401  -1.0230  -525 imp:n,p=1 $ detector 32

C        BLANK LINE DELIMITER ------------------------------------------------
C
C        SURFACE CARDS ****************************************************
C
c
c Problem Boundary
99     BOX     -400 -400 -100 800 0 0 0 800 0 0 0 400
c
c
c Object Being Scanned - Barrel with iron pipes and steel shot
c
101 3 RCC  0 0 10.0    0 0 47.00   17.88
102 3 RCC  0 0 10.1    0 0 46.80   17.78
103 3 RCC  0 0 0       0 0 10      16.88
104 3 RCC  0 0 0       0 0 10       6.72
c
111 3 CZ    1.27
121 3 CZ    4.445
122 3 CZ    6.35
131 3 CZ    7.62
132 3 CZ    8.41375
```

```
141 3 CZ    10.16
142 3 CZ    10.95375
151 3 PZ    23
152 3 PZ    25.24
153 3 PZ    28.0
c
c
c Detectors
c
401 401 BOX     98.453758  -49.076168   -1.270000     1.106819    2.286165
            0.000000     9.144662   -4.427274    0.000000    0.000000    0.000000
            2.540000 $ Angle =   -25.833366
405 401 BOX     99.839478  -46.191906   -1.270000     1.039858    2.317390
            0.000000     9.269559   -4.159431    0.000000    0.000000    0.000000
            2.540000 $ Angle =   -24.166698
409 401 BOX    101.140717  -43.268559   -1.270000     0.972017    2.346653
            0.000000     9.386614   -3.888068    0.000000    0.000000    0.000000
            2.540000 $ Angle =   -22.500029
413 401 BOX    102.356384  -40.308598   -1.270000     0.903354    2.373932
            0.000000     9.495727   -3.613416    0.000000    0.000000    0.000000
            2.540000 $ Angle =   -20.833361
417 401 BOX    103.485451  -37.314537   -1.270000     0.833927    2.399201
            0.000000     9.596805   -3.335707    0.000000    0.000000    0.000000
            2.540000 $ Angle =   -19.166693
421 401 BOX    104.526955  -34.288902   -1.270000     0.763794    2.422441
            0.000000     9.689763   -3.055175    0.000000    0.000000    0.000000
            2.540000 $ Angle =   -17.500023
425 401 BOX    105.480019  -31.234255   -1.270000     0.693015    2.443631
            0.000000     9.774523   -2.772058    0.000000    0.000000    0.000000
            2.540000 $ Angle =   -15.833355
429 401 BOX    106.343842  -28.153181   -1.270000     0.621649    2.462753
            0.000000     9.851012   -2.486596    0.000000    0.000000    0.000000
            2.540000 $ Angle =   -14.166686
433 401 BOX    107.117676  -25.048288   -1.270000     0.549757    2.479792
            0.000000     9.919167   -2.199029    0.000000    0.000000    0.000000
            2.540000 $ Angle =   -12.500018
437 401 BOX    107.800880  -21.922199   -1.270000     0.477401    2.494732
            0.000000     9.978929   -1.909603    0.000000    0.000000    0.000000
            2.540000 $ Angle =   -10.833349
441 401 BOX    108.392883  -18.777563   -1.270000     0.404640    2.507562
            0.000000    10.030248   -1.618560    0.000000    0.000000    0.000000
            2.540000 $ Angle =    -9.166680
445 401 BOX    108.893158  -15.617040   -1.270000     0.331537    2.518270
            0.000000    10.073079   -1.326148    0.000000    0.000000    0.000000
            2.540000 $ Angle =    -7.500012
449 401 BOX    109.301315  -12.443302   -1.270000     0.258154    2.526847
            0.000000    10.107388   -1.032614    0.000000    0.000000    0.000000
            2.540000 $ Angle =    -5.833344
453 401 BOX    109.616982   -9.259036   -1.270000     0.184552    2.533287
            0.000000    10.133146   -0.738206    0.000000    0.000000    0.000000
            2.540000 $ Angle =    -4.166675
457 401 BOX    109.839905   -6.066936   -1.270000     0.110794    2.537582
            0.000000    10.150330   -0.443174    0.000000    0.000000    0.000000
            2.540000 $ Angle =    -2.500006
461 401 BOX    109.969894   -2.869702   -1.270000     0.036942    2.539731
            0.000000    10.158925   -0.147767    0.000000    0.000000    0.000000
            2.540000 $ Angle =    -0.833337
465 401 BOX    110.006836    0.329960   -1.270000    -0.036941    2.539731
            0.000000    10.158925    0.147766    0.000000    0.000000    0.000000
            2.540000 $ Angle =     0.833332
469 401 BOX    109.950699    3.529342   -1.270000    -0.110793    2.537582
```

236

```
              0.000000    10.150330     0.443173     0.000000     0.000000     0.000000
              2.540000 $ Angle =       2.500000
473 401 BOX   109.801537     6.725738    -1.270000    -0.184551     2.533287
              0.000000    10.133146     0.738205     0.000000     0.000000     0.000000
              2.540000 $ Angle =       4.166669
477 401 BOX   109.559464     9.916443    -1.270000    -0.258153     2.526847
              0.000000    10.107388     1.032613     0.000000     0.000000     0.000000
              2.540000 $ Angle =       5.833337
481 401 BOX   109.224701    13.098758    -1.270000    -0.331537     2.518270
              0.000000    10.073079     1.326147     0.000000     0.000000     0.000000
              2.540000 $ Angle =       7.500006
485 401 BOX   108.797516    16.269991    -1.270000    -0.404640     2.507562
              0.000000    10.030248     1.618559     0.000000     0.000000     0.000000
              2.540000 $ Angle =       9.166675
489 401 BOX   108.278282    19.427456    -1.270000    -0.477400     2.494732
              0.000000     9.978929     1.909602     0.000000     0.000000     0.000000
              2.540000 $ Angle =      10.833344
493 401 BOX   107.667435    22.568483    -1.270000    -0.549757     2.479792
              0.000000     9.919167     2.199028     0.000000     0.000000     0.000000
              2.540000 $ Angle =      12.500011
497 401 BOX   106.965485    25.690416    -1.270000    -0.621649     2.462753
              0.000000     9.851012     2.486595     0.000000     0.000000     0.000000
              2.540000 $ Angle =      14.166680
501 401 BOX   106.173035    28.790615    -1.270000    -0.693014     2.443631
              0.000000     9.774523     2.772057     0.000000     0.000000     0.000000
              2.540000 $ Angle =      15.833349
505 401 BOX   105.290749    31.866451    -1.270000    -0.763794     2.422441
              0.000000     9.689763     3.055174     0.000000     0.000000     0.000000
              2.540000 $ Angle =      17.500019
509 401 BOX   104.319374    34.915325    -1.270000    -0.833926     2.399201
              0.000000     9.596805     3.335706     0.000000     0.000000     0.000000
              2.540000 $ Angle =      19.166687
513 401 BOX   103.259743    37.934658    -1.270000    -0.903354     2.373932
              0.000000     9.495727     3.613415     0.000000     0.000000     0.000000
              2.540000 $ Angle =      20.833355
517 401 BOX   102.112740    40.921894    -1.270000    -0.972017     2.346653
              0.000000     9.386614     3.888068     0.000000     0.000000     0.000000
              2.540000 $ Angle =      22.500025
521 401 BOX   100.879333    43.874504    -1.270000    -1.039858     2.317390
              0.000000     9.269560     4.159430     0.000000     0.000000     0.000000
              2.540000 $ Angle =      24.166693
525 401 BOX    99.560577    46.789997    -1.270000    -1.106818     2.286165
              0.000000     9.144662     4.427273     0.000000     0.000000     0.000000
              2.540000 $ Angle =      25.833361

C       BLANK LINE DELIMITER -------------------------------------------------
C
C
C       DATA CARDS *********************************************************
C
MODE n p
c
c Geometric Transforms
c
*TR3  0 0 -17   0.000000 90.000000 90  90.000000  0.000000 90 90 90 0
TR201  -55.00 0   0
C
C MATERIALS
C
c
c Polyethylene, Rho=0.95 g/cc
```

237

```
c
M1     NLIB=60c
       6000    2
       1001    4
c
c Graphite, Rho = 2.20 g/cc
c
M2     NLIB=60c
       6000    1
c
c Elemental Iron, Rho = 7.86 g/cc
c
M3     NLIB=60c
       26054  -0.058500
       26056  -0.917500
       26057  -0.021200
       26058  -0.002800
c
c Elemental Lead, Rho = 11.6 g/cc
c
M4     NLIB=60c
       82206    -0.244400
       82207    -0.224100
       82208    -0.531500
c
c Depleted Uranium, Rho = 18.9
c
M5     NLIB=60c
        92235   -0.002
        92238   -0.998
c
c Cellulose (Wood)
c
M6     NLIB=70c
         1001   0.499925
         1002   0.000075
         6000   0.25
         8016   0.25
c
m401 NLIB=60C $ organic scintillator
       6000 10
       1001 11
*TR400  -55.00 0      0.00
*TR401  -55.00 0    0.000000  0.625001 90.625000
       90  89.375000  0.625001 90 90 90 0
*TR402  -55.00 0    0.000000  0.208334 90.208336
       90  89.791664  0.208334 90 90 90 0
*TR403  -55.00 0    0.000000 -0.208334 89.791664
       90  90.208336 -0.208334 90 90 90 0
*TR404  -55.00 0    0.000000 -0.625001 89.375000
       90  90.625000 -0.625001 90 90 90 0
*TR999  -55.00 0 0
SDEF pos=  -55.00 0      0.00 ERG=14.100 DIR=d1
       PAR=1 VEC=  0.921050 -0.389445 0
SI1    -1         0.99456882  0.99480730  0.99504048  0.99526829
          0.99549073  0.99570787  0.99591964  0.99612612  0.99632716
          0.99652290  0.99671328  0.99689835  0.99707800  0.99725235
          0.99742132  0.99758494  0.99774319  0.99789608  0.99804366
          0.99818581  0.99832267  0.99845415  0.99858022  0.99870098
          0.99881637  0.99892640  0.99903107  0.99913037  0.99922431
          0.99931288  0.99939603  0.99947387  0.99954635  0.99961346
```

238

```
          0.99967521   0.99973154   0.99978256   0.99982822   0.99986845
          0.99990338   0.99993289   0.99995703   0.99997586   0.99998927
          0.99999732   1.00000000
SP1       0            0.00000000   0.00017344   0.00067736   0.00148600
          0.00257230   0.00390802   0.00546408   0.00721063   0.00911736
          0.01115372   0.01328905   0.01549287   0.01773509   0.01998616
          0.02221729   0.02440068   0.02650962   0.02851873   0.03040410
          0.03214340   0.03371609   0.03510346   0.03628879   0.03725742
          0.03799684   0.03849673   0.03874905   0.03874798   0.03849005
          0.03797406   0.03720104   0.03617428   0.03489926   0.03338354
          0.03163673   0.02967037   0.02749782   0.02513415   0.02259600
          0.01990143   0.01706977   0.01412148   0.01107793   0.00796129
          0.00479429   0.00160004
phys:n J 20.
phys:p 0 1 1
cut:n      25.6000
cut:p      25.6000 J 0
rdum 0.000150 0.001000
idum 0 1 2 1 J 1  32
        401 405 409 413 417 421 425 429 433 437
        441 445 449 453 457 461 465 469 473 477
        481 485 489 493 497 501 505 509 513 517
        521 525
DBCN 74822145211985 6J            1     4J 12851
nps     25000000
FILES 21 DUMN1
```

Appendix D – The *PoliMiPP* Code and Supporting Files

The *PoliMiPP* Code

```fortran
PROGRAM PoliMiPP
! Version 3.00
! Written January 7th, 2009
! By Brandon Grogan
! At The Oak Ridge National Laboratory
! Last Edited: December 21st, 2009

!=============================================================================
============
! Variable declaration
!=============================================================================
============

IMPLICIT NONE

TYPE DatFile
  INTEGER(KIND=4) Col1, Col2, Col3, Col4, Col5, Col6
  REAL(KIND=4) Col7, Col8, Col9, Col10, Col11, Col12
  INTEGER(KIND=4) Col13, Col14, Col15
  ! REAL(KIND=4) Col16
END TYPE DatFile

TYPE DetectorPulse
  INTEGER(KIND=4) HistNo, DetNo, ParNo, Direct, SubHist, XTalk
  REAL(KIND=4) :: Time, PulseHeight
END TYPE DetectorPulse

TYPE PeaksFile
  INTEGER(Kind=4) DetNo, TotalPeak
  REAL(KIND=4) TotalMean
  INTEGER(KIND=4) DirectPeak
  REAL(KIND=4) DirectMean
  INTEGER(KIND=4) XTPeak
  REAL(KIND=4) XTMean
END TYPE PeaksFile

! Removed - qqqqq
! The DatFile array holds all of the data from the PoliMi .DAT file
! It's dimensions will be the number of rows in the .DAT file x the number of
columns
! in the PoliMi output (16).
! REAL, ALLOCATABLE :: DatFile(:,:)

TYPE(DatFile), ALLOCATABLE :: ThisHist(:)
TYPE(DatFile) :: HistSwap, TestInput

TYPE(DetectorPulse), ALLOCATABLE :: TempPulses(:), Pulses(:)
INTEGER(KIND=4) :: NumPulses

! This array will temporarily store the data on the starting histories until
the total
! number of histories in the .DAT file is known, at which time the data will be
moved
! to the (smaller) DatHist array.
INTEGER(KIND=4), ALLOCATABLE :: TempHist(:,:)
```

```fortran
! The DatHist array contains the history the history number, starting line
number (in the
! DatFile array) of each history number, and the number of events for that
history.
INTEGER(KIND=4), ALLOCATABLE :: DatHist(:,:), PulseHist(:,:)

! Records the number of detectors (NumDet) and detector cell numbers found in
the .DAT file.
! CurrentDet holds the cell number of the detector in the current history that
is being
! manipulated by the program.
INTEGER(KIND=4) :: NumDet, Detector(50), CurrentDet
! NewDet indicates whether a detector cell number has been seen in the .DAT
file previously.
LOGICAL :: NewDet

! Integer variables used for DO loops and swapping
INTEGER(KIND=4) :: I, J, K, L, SwapInt, SubHistNo, NumSubHist
REAL(KIND=4) :: MinTime

! NumEvents is the number of rows in the .DAT file.  NumHist is the number of
histories
! in the .DAT file.  MaxEvt is the maximum number of events recorded for a
single history.
! MaxEvtHist is the history number with the maximum number of events.
INTEGER(KIND=4) :: NumEvents, NumHist, MaxEvt, MaxEvtHist, NumHistP, MaxPulse,
MaxPulseHist

! This variable is used to sort the events in a history by interaction time.
INTEGER(KIND=4) :: MinRowLoc(1)

! 'History' holds the history number of the line currently being read in the
.DAT file.
! 'OldHistory' holds the history number of the previous line
INTEGER(KIND=4) :: History, OldHistory

! 'InputStatus' holds the IO status of a file read.  IO status < 0 indicates
end of file.
INTEGER :: InputStatus, FileBaseSize
CHARACTER :: DatName*64, FileBase*64, DetectName*4, NPSText*11, OneLetter*1
CHARACTER :: CorrWindowText*11, DeadTimeText*11, DatFormatText*128,
nThreshText*11

! These variables store the time at the beginning and end of the program in
order to
! measure the execution time of the program.
REAL(Kind = 4) :: StartTime, FinishTime

! Dummy Variables
INTEGER(KIND=4) :: DummyInt
REAL(KIND=4) :: DummyReal

! DeadStart records the time that the detector deadtime window opens.
PulseStart records
! the time the pulse generation window opens.  PHThisEvent is the total light
output
! attributed to a particular event in the .DAT file.  PHTotal is the total
light output
! generated during the pulse generation window so far.
REAL(Kind = 4) :: DeadStart, PulseStart, PHThisEvent
```

```fortran
REAL(KIND=4), ALLOCATABLE :: PHTotal(:)

! Input parameters.  These are currently hard-coded, but may be placed off-line
later.
! Pulse generation time.  This is the amount of time it takes the detector to
generate a
! pulse once the initial energy is deposited in the detector cell.  If two
events occur
! within pgentime of each other, their light output will combine.
REAL :: pgentime = 10.0
! The time after a pulse in a detector in which all further events are lost.
! REAL, PARAMETER :: deadtime = 80.0
REAL :: deadtime
! The light output (in MeVee) required to produce a pulse in the detector.
REAL :: threshold, nthresh
! INTEGER, PARAMETER :: CorrWindow = 1024
INTEGER(KIND=4) :: CorrWindow
! The cell number of the start detector which will be used for cross-
correlation
! calculations
INTEGER :: StartDet, StartDetRow, StopDetRow

! Used to hold the integer (rather than real) pulse time.
INTEGER :: IntTime, StopIntTime, TimeLag, CorrWindowOverflow

! Arrays used to hold the program output.
INTEGER(KIND=4), ALLOCATABLE :: Correlation(:,:,:), CrossCorr(:,:,:)

! Variables used for calculating multiplicities
! qqqqq - Made this a command line input for now.
! INTEGER(KIND=4), PARAMETER :: nps = 18600000
INTEGER(KIND=4) :: nps
INTEGER(KIND=4) :: TotalMult, TotalNeutronMult
INTEGER(KIND=4), ALLOCATABLE :: Multiplicity(:,:)

TYPE(PeaksFile), ALLOCATABLE :: Peaks(:)
INTEGER, PARAMETER :: PeakWidth = 5
INTEGER(KIND=4) :: PeakSumTotal, PeakSumDirect, PeakSumNoXT, PeakStart, PeakOld

! PH Spectrum parameters
REAL, PARAMETER :: PHSIncrement = 0.05, PHSMax = 10.00
INTEGER :: PHSNumBins, PHSBin
INTEGER(KIND=4), ALLOCATABLE :: PHSpectrum(:,:,:)

INTEGER :: NumNeutronPulses

! Variables Used for assigning the minimum particle number and subhist to a
pulse if
! it is the aggregation of more than 1 event
INTEGER(KIND=4) :: MinParNo, MinSubHist, MinCollisions, MinGen, MinCode
REAL :: MinWeight

! These values are used to control the maximum number of lines and histories
the program
! will read from the .DAT files.  If these values are too small, the entire
.DAT file
! will not be processed.  If they are set too large, a huge .DAT file may
overflow
! system memory and cause a crash.  Values as large as MaxHist = 10000000 and
! MaxLines = 24000000 have been tested.
```

242

```
INTEGER(KIND=4) :: MaxHist, MaxLines
MaxHist = 20000000
MaxLines = 40000000

!=============================================================================
============
! Read variables from command line, open .DAT file, initialize variables
!=============================================================================
============

CALL CPU_TIME(StartTime)


PRINT *, "PolimiPP, Version 3.00"
PRINT *, "Modified 26 October 2009"


! Reads the .DAT filename from the command line
CALL GetArg(1, DatName)
CALL GetArg(2, DetectName)
CALL GetArg(3, NPSText)
CALL GetArg(4, CorrWindowText)
CALL GetArg(5, DeadTimeText)
CALL GetArg(6, nThreshText)

IF (LEN(TRIM(DatName)) == 0) THEN
 Print *, "PoliMiPP Syntax:"
 PRINT *,
 PRINT *, "PoliMiPP <.dat Filename> <Start Det Cell #> <nps> <Correlation
Window Size>,"
 PRINT *, "<Detector Dead Time>, <Neutron Threshold>"
 STOP
END IF

READ (DetectName, '(I4)') StartDet
IF (LEN(TRIM(NPSText)) > 0) THEN
  READ (NPSText, '(I11)') nps
ELSE
  nps = 1
END IF

IF (LEN(TRIM(CorrWindowText)) > 0) THEN
  READ (CorrWindowText, '(I11)') CorrWindow
ELSE
  CorrWindow = 256
END IF

IF (CorrWindow > 2048) THEN
  PRINT *, "Correlation Window exceeds maximum value of 2048 ns.  Correlation
Window size, dead time, "
  PRINT *, "and pulse generation time will be converted to microseconds by
dividing by 1000."
END IF

! qqqqq
!PRINT *, CorrWindow

IF (LEN(TRIM(DeadTimeText)) > 0) THEN
  READ (DeadTimeText, '(F11.0)') deadtime
ELSE
  deadtime = 35.0
```

```
      END IF

      IF (LEN(TRIM(nThreshText)) > 0) THEN
        READ (nThreshText, '(F11.0)') nthresh
      ELSE
        nthresh = 1.0
      END IF

      threshold = 0.0364*nthresh**2 + 0.125*nthresh


      ! qqqqq
      !PRINT *, deadtime

      FileBase = " "

      J = ICHAR(".")
      DO I = 1,LEN(TRIM(DatName))
        IF (DatName(I:I) == ".") THEN
          EXIT
        ELSE
          OneLetter = DatName(I:I)
          FileBase(I:I) = OneLetter
        END IF
      END DO

      ! qqqqq
      !PRINT *, TRIM(FileBase)


      !FileBase = DatName(1:4)

      ! Opens .DAT file
      OPEN(UNIT=1, FILE=TRIM(DatName), ACTION="READ", STATUS="OLD",
      POSITION="REWIND", IOSTAT = InputStatus)
      IF (InputStatus > 0) STOP "Error opening specified .DAT file"

      NumHist = 0
      OldHistory = 0
      NumDet = 0
      Detector = 0

      ! TempHist will record history numbers and starting line of all histories in
      the .DAT file.
      ! Because the number of histories will not be known until after all of the
      histories are
      ! read in, it must be allocated with more rows than the maximum number of
      histories any
      ! reasonably (< 2 GB) sized .DAT file could contain.
      ALLOCATE(TempHist(MaxHist,2))


      !============================================================================
      ============
      ! Step 1 : Process the raw .DAT file
      !============================================================================
      ============

      !120 FORMAT(I11, I5, I3, I5, I6, I4)

      ! This loop runs through the .DAT file to record the number of events and
      histories.
```

244

```fortran
! The loop will terminate when the end of the file is reached.
DO I=1,MaxLines

  NewDet = .TRUE.
  ! Reads the history number of each line
  READ(1,*, IOSTAT=InputStatus) History, DummyInt, DummyInt, DummyInt,
DummyInt, CurrentDet

  ! Terminates the loop once the end of the .DAT file is reached.
  IF (InputStatus < 0) THEN
   PRINT *, "Last History in .DAT file: ", History
   EXIT
  END IF
  ! Stops the program if the .DAT file cannot be read.
  IF (InputStatus > 0) STOP "*** Error Reading DAT file ***  Unrecognized .DAT
file format"

  IF(I == MaxLines) THEN
    PRINT *, "Warning, .DAT file is too large.  Only histories up to number "
    PRINT *,  TempHist(NumHist, 1), " were processed."

  END IF

  IF (MOD(I, 1000000)==0) PRINT *, I, " lines read"


  DO J = 1, NumDet
    IF (CurrentDet == Detector(J)) THEN
      NewDet = .FALSE.
      EXIT
    END IF
  END DO

  IF (NewDet .EQV. .TRUE.) THEN
    NumDet = NumDet + 1
    Detector(NumDet) = CurrentDet
  END IF


  ! Checks to see if this is a new history number.  If so, NumHist is increased
by
  ! one and this history number becomes the 'OldHistory'.
  IF (History == OldHistory) THEN
    CYCLE
  ELSE
    NumHist = NumHist + 1
      ! Check to see if the maximum number of histories has been exceeded.  If
so,
      ! stop processing and
      IF (NumHist > MaxHist) THEN
        NumHist = MaxHist
        PRINT *, "Warning, .DAT file too large.  Maximum history number
processed"
        PRINT *, "is number ", TempHist(NumHist, 1)
        EXIT
      END IF
    OldHistory = History
    TempHist(NumHist,1) = History
    TempHist(NumHist,2) = I
  END IF
```

245

```
      END DO


      NumEvents = I-1


      DO I = 1, NumDet-1
        MinRowLoc = I + MINLOC(Detector(I+1:NumDet))
        IF (Detector(I) .GT. Detector(MinRowLoc(1))) THEN
          SwapInt = Detector(I)
          Detector(I) = Detector(MinRowLoc(1))
          Detector(MinRowLoc(1)) = SwapInt
        END IF
      END DO

      StartDetRow = 0
      DO I = 1, NumDet
        IF (Detector(I) == StartDet) StartDetRow = I
      END DO


      IF (StartDetRow == 0) THEN
        PRINT *, "Starting Detector Cell not found in .DAT file.  Detector-detector
      cross-&
        &correlations will not be computed for this file."
      END IF

      ! Allocates the arrays now that the number of events and histories is known
      ! Removed DatFile array - qqqqq
      ! ALLOCATE(DatFile(NumEvents,16))
      ALLOCATE(DatHist(NumHist,3))


      MaxEvt = 0

      ! This loop writes the data stored in the TempHist array into the DatHist
      array.
      ! TempHist is (probably much) larger than DatHist because it could not be
      allocated
      ! with the correct number of histories until after the entire DAT file had been
      read.
      DO I = 1, NumHist
        DatHist(I,1) = TempHist(I,1)
        DatHist(I,2) = TempHist(I,2)
        IF (I < NumHist) THEN
          DatHist(I,3) = TempHist(I+1,2) - TempHist(I,2)
        ELSE
          DatHist(I,3) = NumEvents+1 - TempHist(I,2)
        END IF
        IF (DatHist(I,3) > MaxEvt) THEN
          MaxEvt = DatHist(I,3)
          MaxEvtHist = DatHist(I,1)
        END IF
      END DO

      ! Now that the data has been written to the smaller DatHist array, this array
      is
      ! no longer needed.
      DEALLOCATE(TempHist)

      !
      ! Used for testing - qqqqq
      !
```

```
!OPEN(UNIT=2, FILE="TEST.OUT", ACTION="WRITE", STATUS="REPLACE")
!DO I = 1, NumHist
!   WRITE (2, '(3I13)') DatHist (I,1), DatHist(I,2), DatHist(I,3)
!END DO
!CLOSE(UNIT=2)


!==============================================================================
============
! Step 2 : Extract pulses for each history
!==============================================================================
============

! Return to the start of the .DAT file
REWIND 1

110 FORMAT(I11,I5,I3,I5,I6,I4,F10.5,F10.3,F9.2,F8.2,F8.2,F7.3,I5,I6,I4)
130 FORMAT(I11, I4, I5, I4, I5, I3, F10.3, F8.3)

ALLOCATE(TempPulses(NumEvents))
ALLOCATE(ThisHist(MaxEvt))
ALLOCATE(PHTotal(MaxEvt))
PHTotal = 0.

NumPulses = 0

! Convert deadtime and pgentime to microseconds if necessary.
IF (CorrWindow > 2048) THEN
  deadtime = deadtime / 1000.
  pgentime = pgentime / 1000.
END IF


! qqqqq
! OPEN(UNIT=8, FILE="mlge.dat", ACTION="WRITE", STATUS="REPLACE")

! Loop through the history file.
DO I = 1, NumHist
  ! Reads all of the events for history 'I' into the ThisHist array.
  DO J = 1, DatHist(I,3)
    READ (1, *) ThisHist(J)%Col1, ThisHist(J)%Col2, ThisHist(J)%Col3,
ThisHist(J)%Col4, &
    & ThisHist(J)%Col5, ThisHist(J)%Col6, ThisHist(J)%Col7, ThisHist(J)%Col8, &
    & ThisHist(J)%Col9, ThisHist(J)%Col10, ThisHist(J)%Col11,
ThisHist(J)%Col12, &
    & ThisHist(J)%Col13, ThisHist(J)%Col14, ThisHist(J)%Col15


!==============================================================================
=======
    ! Converts the time from shakes (10^-8 s.) to ns if CorrWindow is <= 2048.
    ! If CorrWindow > 2048, converts the time to microseconds instead.

    IF (CorrWindow <= 2048) THEN
      ThisHist(J)%Col8 = 10. * ThisHist(J)%Col8
    ELSE
      ThisHist(J)%Col8 = ThisHist(J)%Col8 / 100.
    END IF

    ! qqqqq
    ! IF (ThisHist(J)%Col1 == 31461277) WRITE(8,110) ThisHist(j)
```

```
      END DO


!==============================================================================
============
  ! Step ?? : Sort Histories ascending by detector cell and then by time within
the detector
  ! cell.  If there is a tie for both detector cell and time, the event with
the lowest
  ! collision number and then code is brought up.

!==============================================================================
============

  IF(DatHist(I,3) > 1) THEN
  DO J = 1, DatHist(I,3)-1
    DO K = J, DatHist(I,3)
      IF (ThisHist(J)%Col6 > ThisHist(K)%Col6) THEN
        HistSwap = ThisHist(J)
        ThisHist(J) = ThisHist(K)
        ThisHist(K) = HistSwap
      ELSEIF(ThisHist(J)%Col6 == ThisHist(K)%Col6 .AND. &
      & ThisHist(J)%Col8 > ThisHist(K)%Col8) THEN
        HistSwap = ThisHist(J)
        ThisHist(J) = ThisHist(K)
        ThisHist(K) = HistSwap
      ELSEIF(ThisHist(J)%Col6 == ThisHist(K)%Col6 .AND. &
      & ThisHist(J)%Col8 == ThisHist(K)%Col8 .AND. ThisHist(J)%Col14 >
ThisHist(K)%Col14) THEN
        HistSwap = ThisHist(J)
        ThisHist(J) = ThisHist(K)
        ThisHist(K) = HistSwap
      ELSEIF(ThisHist(J)%Col6 == ThisHist(K)%Col6 .AND. &
      & ThisHist(J)%Col8 == ThisHist(K)%Col8 .AND. ThisHist(J)%Col14 ==
ThisHist(K)%Col14 &
      & .AND. ThisHist(J)%Col15 > ThisHist(K)%Col15) THEN
        HistSwap = ThisHist(J)
        ThisHist(J) = ThisHist(K)
        ThisHist(K) = HistSwap
      END IF
    END DO
  END DO
  END IF



  DO J = 1, DatHist(I,3)
    CALL PulseHeight(ThisHist(J)%Col3, ThisHist(J)%Col4, ThisHist(J)%Col5,
ThisHist(J)%Col7, PHThisEvent)
    PHTotal(J) = PHThisEvent

    ! qqqqq
    ! PRINT '(I11,F10.5)', ThisHist(J)%Col1, PHTotal(J)
    ! IF (DatHist(I,3) > 1) PRINT 110, ThisHist(J)
    ! qqqqq
    ! Used for splitting up big .DAT files
    ! IF (I <= 50000) WRITE (8, 110) ThisHist(J)


  END DO
```

248

```
   CurrentDet = 0

  DO J = 1, DatHist(I,3)

    IF (ThisHist(J)%Col6 /= CurrentDet) THEN
      ! Set the start of the deadtime and pulse generation time to a negative
number large
      ! enough that the first event will not be in either window.
      DeadStart = -2.*deadtime
      PulseStart = -2.*pgentime
      CurrentDet = ThisHist(J)%Col6
    END IF



    IF (ThisHist(J)%Col8 < DeadStart + deadtime) CYCLE

    IF (PHTotal(J) >= threshold) THEN
      NumPulses = NumPulses + 1
      TempPulses(NumPulses)%HistNo = ThisHist(J)%Col1
      TempPulses(NumPulses)%DetNo = ThisHist(J)%Col6
      TempPulses(NumPulses)%ParNo = ThisHist(J)%Col3
      TempPulses(NumPulses)%Time = ThisHist(J)%Col8
      TempPulses(NumPulses)%PulseHeight = PHTotal(J)
      TempPulses(NumPulses)%SubHist = ThisHist(J)%Col2


      ! Check to see if the event that generated the pulse was a directly
transmitted DT neutron.
      ! SubHistory = 1, Particle Type = 1 (neutron), Weight > 0.95, Generation
= 0,
      ! # Collisions = 0, Code = 0
      IF(ThisHist(J)%Col2 == 1 .AND. ThisHist(J)%Col3 == 1 .AND.
ThisHist(J)%Col12 >= 0.95 &
      & .AND. ThisHist(J)%Col13 == 0 .AND. ThisHist(J)%Col14 == 0 .AND.
ThisHist(J)%Col15 == 0) THEN
        TempPulses(NumPulses)%Direct = 1
      ELSE
        TempPulses(NumPulses)%Direct = 0
      END IF

      DeadStart = ThisHist(J)%Col8
      PulseStart = -2.*pgentime
      CYCLE
    ELSE
      PulseStart = ThisHist(J)%Col8
    END IF

    DO K = J+1, DatHist(I,3)
      IF (ThisHist(K)%Col6 /= CurrentDet) EXIT

      ! qqqqq
      !     PRINT *, "J= ", J, "K= ", K, "PHTotal= ", PHTotal(J)

      IF (ThisHist(K)%Col8 < PulseStart + pgentime) THEN
        PHTotal(J) = PHTotal(J) + PHTotal(K)
      ELSE
        EXIT
      END IF

      IF (PHTotal(J) >= threshold) THEN
```

249

```
          MinParNo = 99999
          MinSubHist = 99999
          MinCollisions = 99999
          MinGen = 99999
          MinCode = 99999
          NumPulses = NumPulses + 1
          TempPulses(NumPulses)%HistNo = ThisHist(J)%Col1
          TempPulses(NumPulses)%DetNo = ThisHist(J)%Col6
          TempPulses(NumPulses)%Time = ThisHist(J)%Col8
          TempPulses(NumPulses)%PulseHeight = PHTotal(J)

          DO L = J, K
             IF (ThisHist(L)%Col2 <= MinSubHist .AND. ThisHist(L)%Col3 <= MinParNo
  .AND. &
             & ThisHist(L)%Col13 <= MinGen .AND. ThisHist(L)%Col14 <=
  MinCollisions .AND. &
             & ThisHist(L)%Col15 <= MinCode) THEN
               MinSubHist = ThisHist(L)%Col2
               MinParNo = ThisHist(L)%Col3
               MinGen = ThisHist(L)%Col13
               MinCollisions = ThisHist(L)%Col14
               MinCode = ThisHist(L)%Col15
               MinWeight = ThisHist(L)%Col12
             END IF
          END DO
          TempPulses(NumPulses)%ParNo = MinParNo
          TempPulses(NumPulses)%SubHist = MinSubHist

          ! Check to see if the event that generated the pulse was a directly
  transmitted DT neutron.
          IF(MinSubHist == 1 .AND. MinParNo == 1 .AND. MinWeight >= 0.95 &
          & .AND. MinGen == 0 .AND. MinCollisions == 0 .AND. MinCode == 0) THEN
            TempPulses(NumPulses)%Direct = 1
          ELSE
            TempPulses(NumPulses)%Direct = 0
          END IF

          DeadStart = ThisHist(J)%Col8
          PulseStart = -2.*pgentime
          EXIT
        END IF
      END DO
    END DO

    PHTotal = 0.

END DO

! Now convert CorrWindow to microseconds if need be.
IF (CorrWindow > 2048) THEN
  CorrWindow = CorrWindow / 1000 + 1
END IF

CLOSE(UNIT=1)
! qqqqq
! CLOSE(UNIT=8)

ALLOCATE(Pulses(NumPulses))

DO I = 1, NumPulses
  Pulses(I) = TempPulses(I)
```

```
   Pulses(I)%XTalk = 0
  ! This loop changes the detector number value from the actual cell number to
the line
  ! referencing that cell number in the Detector() array.  This will make
matching the
  ! detector cell faster in the correlation section.
  DO J = 1, NumDet
    IF (Pulses(I)%DetNo == Detector(J)) THEN
      Pulses(I)%DetNo = J
      EXIT
    END IF
  END DO
END DO

! qqqqq
! DO I = 1, MaxEvt
!   WRITE (*, 110) ThisHist(I)
! END DO

! DO I = 1,NumDet
!   PRINT *, Detector(I)
! END DO


DEALLOCATE(ThisHist)
DEALLOCATE(PHTotal)
DEALLOCATE(TempPulses)
DEALLOCATE(DatHist)

!==============================================================================
============
! Step 3 : Extract multiplicities and correlations from pulse data
!==============================================================================
============

ALLOCATE(TempHist(NumPulses,2))

NumHistP = 0
OldHistory = 0
MaxPulse = 0

PHSNumBins = INT(PHSMax/PHSIncrement)
ALLOCATE(PHSpectrum(-1:PHSNumBins,0:NumDet,3))
PHSpectrum = 0

DO I = 1, 3
  PHSpectrum(-1,0,I) = nps
  DO J = 1, NumDet
    PHSpectrum(-1,J,I) = Detector(J)
  END DO
END DO

DO I = 1, NumPulses

PHSBin = INT(Pulses(I)%PulseHeight / PHSIncrement)
IF (PHSBin > PHSNumBins) PHSBin = PHSNumBins
PHSpectrum(PHSBin,Pulses(I)%DetNo,1) = PHSpectrum(PHSBin,Pulses(I)%DetNo,1) + 1
IF (Pulses(I)%ParNo == 1) THEN
  PHSpectrum(PHSBin,Pulses(I)%DetNo,2) = PHSpectrum(PHSBin,Pulses(I)%DetNo,2) +
1
ELSE IF (Pulses(I)%ParNo == 2) THEN
```

```fortran
    PHSpectrum(PHSBin,Pulses(I)%DetNo,3) = PHSpectrum(PHSBin,Pulses(I)%DetNo,3) +
1
END IF

! qqqqq
! IF (Pulses(I)%Direct == 1 .AND. Pulses(I)%Time > 50) PRINT *,
Pulses(I)%HistNo


  IF (Pulses(I)%HistNo == OldHistory) CYCLE
  OldHistory = Pulses(I)%HistNo
  NumHistP = NumHistP + 1
  TempHist(NumHistP,1) = Pulses(I)%HistNo
  TempHist(NumHistP,2) = I
END DO

ALLOCATE(PulseHist(NumHistP,3))

DO I = 1, NumHistP
  PulseHist(I,1) = TempHist(I,1)
  PulseHist(I,2) = TempHist(I,2)
  IF (I < NumHistP) THEN
    PulseHist(I,3) = TempHist(I+1,2) - TempHist(I,2)
  ELSE
    PulseHist(I,3) = NumPulses+1 - TempHist(I,2)
  END IF
  IF (PulseHist(I,3) > MaxPulse) THEN
    MaxPulse = PulseHist(I,3)
    MaxPulseHist = PulseHist(I,1)
  END IF
END DO

DEALLOCATE(TempHist)




!===============================================================================
============
! Step 3a : Calculate multiplicities
!===============================================================================
============

PRINT *, "Calculating Multiplicities"

! Multiplicity structure: Rows = number of source triggered multiplicities;
Columns -
! Column 1 = total multiplicities; Column 2 = neutron multiplicities
ALLOCATE(Multiplicity(0:MaxPulse,2))
Multiplicity = 0

! Go through the pulse file and record the number of pulses created for each
history in
! the Multiplicity array.
DO I = 1, NumHistP
  Multiplicity(PulseHist(I,3),1) = Multiplicity(PulseHist(I,3),1) + 1

  NumNeutronPulses = 0
  DO J = 1, PulseHist(I,3)
    IF (Pulses(PulseHist(I,2)+J-1)%ParNo == 1) NumNeutronPulses =
NumNeutronPulses + 1
```

```
    END DO
    Multiplicity(NumNeutronPulses, 2) = Multiplicity(NumNeutronPulses, 2) + 1

  END DO

  TotalMult = 0
  TotalNeutronMult = 0

  ! Assigns to 'TotalMult' the total number of histories which have >= 1 pulse.
  DO I = 1, MaxPulse
    TotalMult = TotalMult + Multiplicity(I, 1)
    TotalNeutronMult = TotalNeutronMult + Multiplicity(I, 2)
  END DO

  ! Total histories minus all histories with >= 1 pulse = no. of histories with
  no pulses.
  Multiplicity(0, 1) = nps - TotalMult
  Multiplicity(0, 2) = nps - TotalNeutronMult

  !============================================================================
  ============
  ! Step 3b : Calculate source-detector correlations
  !
  ! NOTE: If CorrWindow is still > 2048 then no correlations are computed
  !============================================================================
  ============

  ! Begin CorrWindow IF Statement
  IF (CorrWindow <= 2048) THEN

  ! Correlation Structure - Rows = Time, Columns = Det #, Panes = Total, n,
  gamma, direct, no x-talk
  ALLOCATE(Correlation(-1:CorrWindow,0:NumDet,5))
  ! CrossCorr Structure - Rows = Time Lag, Columns = Det #, Panes = Total, nn,
  gg, np, pn
  ALLOCATE(CrossCorr(-CorrWindow-1:CorrWindow,0:NumDet,5))

  Correlation = 0
  CrossCorr=0

  ! Input detector cell numbers and time steps into correlation arrays
  DO I = 1, 5
    CrossCorr(-CorrWindow-1,0,I) = nps
    DO J = 1, NumDet
      CrossCorr(-CorrWindow-1,J,I) = Detector(J)
    END DO
    DO J = -CorrWindow,CorrWindow
      CrossCorr(J,0,I) = J
    END DO
  END DO

  DO I = 1, 5
    Correlation(-1,0,I) = nps
    DO J = 1, NumDet
      Correlation(-1,J,I) = Detector(J)
    END DO
    DO J = 0, CorrWindow
      Correlation(J,0,I) = J
    END DO
  END DO
```

```
! qqqqq
! OPEN (UNIT=99, FILE="error.out", STATUS="REPLACE", ACTION="WRITE")

PRINT *, "Calculating Source-Detector Correlations"

CorrWindowOverflow = 0

! This is where the magic happens!!!!1111!!!  Correlations and cross-
correlations are
! computed here.
DO I = 1, NumHistP
  DO J = 1, PulseHist(I,3)
    IntTime = INT(Pulses(PulseHist(I,2)+J-1)%Time)
    IF (ABS(IntTime) > CorrWindow) THEN
      CorrWindowOverflow = CorrWindowOverflow + 1
      CYCLE
    END IF
    CurrentDet = Pulses(PulseHist(I,2)+J-1)%DetNo
    Correlation(IntTime,CurrentDet,1) = Correlation(IntTime,CurrentDet,1) + 1
    IF (Pulses(PulseHist(I,2)+J-1)%Direct == 1) THEN
      Correlation(IntTime,CurrentDet,4) = Correlation(IntTime,CurrentDet,4) + 1
    END IF
    IF (Pulses(PulseHist(I,2)+J-1)%ParNo == 1) THEN
      Correlation(IntTime,CurrentDet,2) = Correlation(IntTime,CurrentDet,2) + 1
    ELSE IF (Pulses(PulseHist(I,2)+J-1)%ParNo == 2) THEN
      Correlation(IntTime,CurrentDet,3) = Correlation(IntTime,CurrentDet,3) + 1
    ELSE
      PRINT *, "Error!  Particle type other than a photon or a neutron
encountered &
      &in .DAT file, history number ", PulseHist(I,1), ".  Check .DAT file."
      WRITE (*, 130) Pulses(PulseHist(I,2)+J-1)
      STOP
    END IF

!===============================================================================
============
! Step 3c : Calculate detector-detector correlations if start detector is
specified
!===============================================================================
============

    ! Check cross-correlations and fill in the cross-correlation arrays.
    IF (CurrentDet == StartDetRow) THEN
      DO K = 1, PulseHist(I,3)
        IF (K == J) CYCLE
        StopDetRow = Pulses(PulseHist(I,2)+K-1)%DetNo
        StopIntTime = NINT(Pulses(PulseHist(I,2)+K-1)%Time)
        TimeLag = StopIntTime - IntTime
        IF (ABS(TimeLag) > CorrWindow) THEN
          PRINT *, "ERROR!! Start Time= ", IntTime, "Stop Time= ", StopIntTime,
&
          & "TimeLag= ", TimeLag
          CYCLE
        END IF
        CrossCorr(TimeLag,StopDetRow,1) = CrossCorr(TimeLag,StopDetRow,1) + 1
        ! qqqqq
        ! IF (ABS(TimeLag) <= 5) WRITE(99,'(I11)') PulseHist(I,1)
        IF (Pulses(PulseHist(I,2)+J-1)%ParNo == 1 .AND. &
        & Pulses(PulseHist(I,2)+K-1)%ParNo == 1) THEN
          CrossCorr(TimeLag,StopDetRow,2) = CrossCorr(TimeLag,StopDetRow,2) + 1
```

254

```
            ELSE IF (Pulses(PulseHist(I,2)+J-1)%ParNo == 2 .AND. &
            & Pulses(PulseHist(I,2)+K-1)%ParNo == 2) THEN
              CrossCorr(TimeLag,StopDetRow,3) = CrossCorr(TimeLag,StopDetRow,3) + 1
            ELSE IF (Pulses(PulseHist(I,2)+J-1)%ParNo == 1 .AND. &
            & Pulses(PulseHist(I,2)+K-1)%ParNo == 2) THEN
              CrossCorr(TimeLag,StopDetRow,4) = CrossCorr(TimeLag,StopDetRow,4) + 1
            ELSE IF (Pulses(PulseHist(I,2)+J-1)%ParNo == 2 .AND. &
            & Pulses(PulseHist(I,2)+K-1)%ParNo == 1) THEN
              CrossCorr(TimeLag,StopDetRow,5) = CrossCorr(TimeLag,StopDetRow,5) + 1
            END IF
            ! IF (ABS(TimeLag) < NINT(deadtime) .AND. ABS(TimeLag) > 0 .AND. &
            !    & StopDetRow == StartDetRow) THEN
            !  WRITE (99,'(4I11)') PulseHist(I,1), StopIntTime, IntTime, TimeLag
            ! END IF
        END DO
      END IF
    END DO
END DO

! qqqqq
! CLOSE(UNIT=99)

!=============================================================================
============
! Step ?? : Determine the Fast Neutron Time Window.  This is determined by
finding the
! largest source-detector correlations in the entire measurement.  All
detectors have the
! same time window.
!=============================================================================
============


PRINT *, "Calculating Peak Values"

ALLOCATE(Peaks(1:NumDet))
PeakOld = 0
PeakStart = 0
PeakSumTotal = 0

DO I = 1, NumDet
  Peaks(I)%DetNo = Detector(I)
  DO J = 0, CorrWindow
    PeakSumTotal = Correlation(J,I,1)
    IF (PeakSumTotal >= PeakOld) THEN
      PeakOld = PeakSumTotal
      PeakStart = J
    END IF
  END DO
END DO
  PeakStart = PeakStart - PeakWidth / 2

PRINT *, "Fast Neutron Time Window: ", PeakStart, " to ", PeakStart+Peakwidth-1

!=============================================================================
============
! Step ?? : Now that the fast neutron time window is known, mark the cross talk
between
! detectors.  Pulses are sorted by detector and then by time, so if more than
one
```

255

```
! correlation occurs in the fast time window for a given history, the one with
the larger
! time is marked as cross-talk regardless of detector cell.
!============================================================================
============

PRINT *, "Calculating Cross-Talk"

DO I = 1, NumHistP
  IF (PulseHist(I,3) == 1) THEN
      IF (Pulses(PulseHist(I,2))%Time < REAL(PeakStart) .OR. &
      & Pulses(PulseHist(I,2))%Time >= REAL(PeakStart + PeakWidth)) THEN
         Pulses(PulseHist(I,2))%XTalk = 1
      END IF
      CYCLE
  END IF
  DO J = 1, PulseHist(I,3)
     ! Cycle if Pulse J not in time window or if it has already been marked as
XTalk
     ! Note that the window is 1 smaller in integer math than in Real math
because of
     ! rounding.
     IF (Pulses(PulseHist(I,2)+J-1)%Time < REAL(PeakStart) .OR. &
     & Pulses(PulseHist(I,2)+J-1)%Time >= REAL(PeakStart + PeakWidth)) THEN
       Pulses(PulseHist(I,2)+J-1)%XTalk = 1
       CYCLE
     END IF
     IF (Pulses(PulseHist(I,2)+J-1)%XTalk == 1) CYCLE
     DO K = J+1, PulseHist(I,3)
        IF (Pulses(PulseHist(I,2)+K-1)%Time < REAL(PeakStart) .OR. &
        & Pulses(PulseHist(I,2)+K-1)%Time >= REAL(PeakStart+PeakWidth)) CYCLE
       ! If we have reached this point, both Pulse J and K are in the time
window.  The
       ! one with the higher time gets marked a XTalk.  PoliMi only gives time
to 0.1 ns,
       ! so it's possible (but very unlikely) the times could be equal.  If so,
the
       ! detector with the higher number gets counted as XTalk because of the
order of
       ! pulses in the array.
       IF (Pulses(PulseHist(I,2)+J-1)%Time > Pulses(PulseHist(I,2)+K-1)%Time)
THEN
         Pulses(PulseHist(I,2)+J-1)%XTalk = 1
       ELSE
         Pulses(PulseHist(I,2)+K-1)%XTalk = 1
       END IF
     END DO
  END DO
END DO

! Unfortunately, in order to find the fast neutron window to remove the cross-
talk, I had
! to compute correlations and now I have to go through again to compute the no
cross-talk
! correlations.  These no cross-talk correlations should equal the directs plus
scatter
! in the object being imaged inside of the fast neutron window.  If other
features are in
! the geometry as well, such as the fancy detectors or the detector arm,
scattered
```

```
! neutrons and induced gammas from those objects can contribute as well.  No
cross-talk
! is computed outside of the peaks window, so those values should be exactly
the same
! as the measured.

DO I = 1, NumHistP
  DO J = 1, PulseHist(I,3)
    IntTime = INT(Pulses(PulseHist(I,2)+J-1)%Time)
    IF (ABS(IntTime) > CorrWindow) CYCLE
    CurrentDet = Pulses(PulseHist(I,2)+J-1)%DetNo
    IF (Pulses(PulseHist(I,2)+J-1)%XTalk == 0) THEN
      Correlation(IntTime,CurrentDet,5) = Correlation(IntTime,CurrentDet,5) + 1
    END IF
  END DO
END DO

PeakSumTotal = 0
PeakSumDirect = 0
PeakSumNoXT = 0

DO I = 1, NumDet
  DO J = PeakStart, PeakStart + PeakWidth - 1
    PeakSumTotal = PeakSumTotal + Correlation(J,I,1)
    PeakSumDirect = PeakSumDirect + Correlation(J,I,4)
    PeakSumNoXT = PeakSumNoXT + Correlation(J,I,5)
  END DO

  Peaks(I)%TotalPeak = PeakSumTotal
  Peaks(I)%DirectPeak = PeakSumDirect
  Peaks(I)%XTPeak = PeakSumNoXT

  Peaks(I)%TotalMean = 0.
  Peaks(I)%DirectMean = 0.
  Peaks(I)%XTMean = 0.

  DO J = PeakStart, PeakStart + PeakWidth - 1

    IF (PeakSumTotal > 0) THEN
      Peaks(I)%TotalMean = Peaks(I)%TotalMean + REAL(J * Correlation(J,I,1))
    END IF

    IF (PeakSumDirect > 0) THEN
      Peaks(I)%DirectMean = Peaks(I)%DirectMean + REAL(J * Correlation(J,I,4))
    END IF

    IF (PeakSumNoXT > 0) THEN
      Peaks(I)%XTMean = Peaks(I)%XTMean + REAL(J * Correlation(J,I,5))
    END IF
  END DO

IF (PeakSumTotal > 0) THEN
  Peaks(I)%TotalMean = Peaks(I)%TotalMean / REAL(Peaks(I)%TotalPeak)
END IF
IF (PeakSumDirect > 0) THEN
  Peaks(I)%DirectMean = Peaks(I)%DirectMean / REAL(Peaks(I)%DirectPeak)
END IF
IF (PeakSumNoXT > 0) THEN
  Peaks(I)%XTMean = Peaks(I)%XTMean / REAL(Peaks(I)%XTPeak)
END IF
```

```
PeakSumTotal = 0
PeakSumDirect = 0
PeakSumNoXT = 0

END DO


! Outputs a list of pulses for debug purposes.
! qqqqq
! OPEN(UNIT=5, FILE="Pulses.out", STATUS="REPLACE", ACTION="WRITE")
! DO I = 1, NumPulses
!    WRITE (5,130) Pulses(I)
! END DO
! qqqqq
! CLOSE(UNIT=5)

! qqqqq
! Outputs histories which contain 2 or more pulses for troubleshooting purposes
! OPEN(UNIT=7, FILE="MultiPulse.out", ACTION="WRITE", STATUS="REPLACE")
! DO I = 1, NumHistP
!    IF (PulseHist(I,3) > 1) THEN
!      DO J = 1, PulseHist(I,3)
!        WRITE(7,130) Pulses(PulseHist(I,2)+J-1)
!      END DO
!    END IF
! END DO
! CLOSE(UNIT=7)



DEALLOCATE(PulseHist)
DEALLOCATE(Pulses)

ELSE
 PRINT *, "Correlation Window exceeds 2048 microseconds.  source-detector and
detector-detector"
 PRINT *, "correlations will not be calculated.  Lower correlation window size
to 2,048,000 ns"
 PRINT *, "or less if correlations are desired."
! End CorrWindow IF Statement
END IF

!==============================================================================
============
! Step 4 : Print Output to screen and file(s).
!==============================================================================
============

PRINT *,
PRINT *, "The .DAT file contains ", NumEvents," lines and ", NumHist, "
histories."
PRINT *,
PRINT *, "The largest history is number ", MaxEvtHist, ".  It has ", MaxEvt, "
events."
PRINT *,
PRINT *, "The .DAT file has records for ", NumDet, " detectors."
PRINT *,
PRINT *, "A total of ", NumPulses, " pulses were recorded."
PRINT *,
PRINT *, "The largest history is number ", MaxPulseHist, ".  It has ",
MaxPulse, " pulses."
```

258

```
PRINT *,
IF (CorrWindowOverflow > 0) THEN
  PRINT '(I10, A)', CorrWindowOverflow, " pulses were discarded because the
correlation window &
  &was too small.  Consider increasing the size of the window if these losses
are large."
  PRINT *,
END IF


! Begin CorrWindow IF Statement
IF (CorrWindow <= 2048) THEN


! Writes the correlation output to files.
OPEN(UNIT=11, FILE=TRIM(FileBase)//".total.corr", STATUS="REPLACE",
ACTION="WRITE")
OPEN(UNIT=12, FILE=TRIM(FileBase)//".neutron.corr", STATUS="REPLACE",
ACTION="WRITE")
OPEN(UNIT=13, FILE=TRIM(FileBase)//".gamma.corr", STATUS="REPLACE",
ACTION="WRITE")
OPEN(UNIT=14, FILE=TRIM(FileBase)//".direct.corr", STATUS="REPLACE",
ACTION="WRITE")
OPEN(UNIT=15, FILE=TRIM(FileBase)//".noXT.corr", STATUS="REPLACE",
ACTION="WRITE")
DO I = -1, CorrWindow
  DO J = 0, NumDet-1
    WRITE(11,'(I10)', ADVANCE="NO") Correlation(I,J,1)
    WRITE(12,'(I10)', ADVANCE="NO") Correlation(I,J,2)
    WRITE(13,'(I10)', ADVANCE="NO") Correlation(I,J,3)
    WRITE(14,'(I10)', ADVANCE="NO") Correlation(I,J,4)
    WRITE(15,'(I10)', ADVANCE="NO") Correlation(I,J,5)
  END DO
  WRITE(11,'(I10)') Correlation(I,NumDet,1)
  WRITE(12,'(I10)') Correlation(I,NumDet,2)
  WRITE(13,'(I10)') Correlation(I,NumDet,3)
  WRITE(14,'(I10)') Correlation(I,NumDet,4)
  WRITE(15,'(I10)') Correlation(I,NumDet,5)
END DO
CLOSE(UNIT=11)
CLOSE(UNIT=12)
CLOSE(UNIT=13)
CLOSE(UNIT=14)
CLOSE(UNIT=15)

IF (StartDetRow /= 0) THEN
! Writes the cross-correlation output to files.
OPEN(UNIT=21, FILE=TRIM(FileBase)//".total.cc", STATUS="REPLACE",
ACTION="WRITE")
OPEN(UNIT=22, FILE=TRIM(FileBase)//".nn.cc", STATUS="REPLACE", ACTION="WRITE")
OPEN(UNIT=23, FILE=TRIM(FileBase)//".pp.cc", STATUS="REPLACE", ACTION="WRITE")
OPEN(UNIT=24, FILE=TRIM(FileBase)//".np.cc", STATUS="REPLACE", ACTION="WRITE")
OPEN(UNIT=25, FILE=TRIM(FileBase)//".pn.cc", STATUS="REPLACE", ACTION="WRITE")

DO I = -CorrWindow - 1, CorrWindow
  DO J = 0, NumDet - 1
    WRITE(21,'(I10)', ADVANCE="NO") CrossCorr(I,J,1)
    WRITE(22,'(I10)', ADVANCE="NO") CrossCorr(I,J,2)
    WRITE(23,'(I10)', ADVANCE="NO") CrossCorr(I,J,3)
    WRITE(24,'(I10)', ADVANCE="NO") CrossCorr(I,J,4)
    WRITE(25,'(I10)', ADVANCE="NO") CrossCorr(I,J,5)
```

```fortran
    END DO
    WRITE(21,'(I10)') CrossCorr(I,NumDet,1)
    WRITE(22,'(I10)') CrossCorr(I,NumDet,2)
    WRITE(23,'(I10)') CrossCorr(I,NumDet,3)
    WRITE(24,'(I10)') CrossCorr(I,NumDet,4)
    WRITE(25,'(I10)') CrossCorr(I,NumDet,5)
  END DO

  CLOSE(UNIT=21)
  CLOSE(UNIT=22)
  CLOSE(UNIT=23)
  CLOSE(UNIT=24)
  CLOSE(UNIT=25)
END IF

! End CorrWindow IF Statement
END IF


! Writes the Multiplicity array to file.
OPEN(UNIT=31, FILE=TRIM(FileBase)//".multip", STATUS="REPLACE", ACTION="WRITE")
WRITE(31,'(A)') " N       Total    Neutrons"
DO I = 0, MaxPulse
  WRITE (31, '(I3, 2I11)') I, Multiplicity(I,1), Multiplicity(I,2)
END DO
CLOSE(UNIT=31)

OPEN(UNIT=41, FILE=TRIM(FileBase)//".peaks", STATUS="REPLACE", ACTION="WRITE")
  WRITE (41,'(I12, A)') nps, "  Total        Mean(T)       Direct        Mean(D)
No XTalk     Mean(N)"
DO I = 1, NumDet
  WRITE (41, '(2I12, ES12.4, I12, ES12.4, I12, ES12.4)') Peaks(I)
END DO
CLOSE(UNIT=41)

OPEN(UNIT=51, FILE=TRIM(FileBase)//".total.ph", STATUS="REPLACE",
ACTION="WRITE")
OPEN(UNIT=52, FILE=TRIM(FileBase)//".neutron.ph", STATUS="REPLACE",
ACTION="WRITE")
OPEN(UNIT=53, FILE=TRIM(FileBase)//".gamma.ph", STATUS="REPLACE",
ACTION="WRITE")

DO I = 0, NumDet - 1
  WRITE(51,'(I11)', ADVANCE="NO") PHSpectrum(-1,I,1)
  WRITE(52,'(I11)', ADVANCE="NO") PHSpectrum(-1,I,2)
  WRITE(53,'(I11)', ADVANCE="NO") PHSpectrum(-1,I,3)
END DO
  WRITE(51,'(I11)') PHSpectrum(-1,NumDet,1)
  WRITE(52,'(I11)') PHSpectrum(-1,NumDet,2)
  WRITE(53,'(I11)') PHSpectrum(-1,NumDet,3)

DO I = 0, PHSNumBins
  WRITE(51,'(F11.5)', ADVANCE="NO") REAL(I) * PHSIncrement
  WRITE(52,'(F11.5)', ADVANCE="NO") REAL(I) * PHSIncrement
  WRITE(53,'(F11.5)', ADVANCE="NO") REAL(I) * PHSIncrement
  DO J = 1, NumDet - 1
  WRITE(51,'(I11)', ADVANCE="NO") PHSpectrum(I,J,1)
  WRITE(52,'(I11)', ADVANCE="NO") PHSpectrum(I,J,2)
  WRITE(53,'(I11)', ADVANCE="NO") PHSpectrum(I,J,3)
  END DO
  WRITE(51,'(I11)') PHSpectrum(I,NumDet,1)
```

```
  WRITE(52,'(I11)') PHSpectrum(I,NumDet,2)
  WRITE(53,'(I11)') PHSpectrum(I,NumDet,3)

END DO

CLOSE(UNIT=51)
CLOSE(UNIT=52)
CLOSE(UNIT=53)

DEALLOCATE (Correlation, CrossCorr, Multiplicity, Peaks, PHSpectrum)

CALL CPU_TIME(FinishTime)
WRITE (*, '(A, F8.3, A)') " The program took ", FinishTime - StartTime, "
seconds to finish."

CONTAINS

!=============================================================================
============
! Subroutine PulseHeight converts the energy deposited in a collision (MeV) to
light
! output in the detector (in MeVee)
!=============================================================================
============
SUBROUTINE PulseHeight(Zprojectile, ZRx, Ztarget, ZEin, Zpheight)

INTEGER(KIND=4), INTENT(IN) :: Zprojectile, Ztarget, ZRx
REAL(KIND=4), INTENT(IN) :: ZEin

REAL(KIND=4), INTENT(OUT) :: Zpheight

IF (Ztarget == 6000 .AND. Zprojectile == 1) THEN
  Zpheight = 0.02 * ZEin
  RETURN
ELSEIF (Ztarget == 6 .AND. Zprojectile == 1) THEN
  Zpheight = 0.02 * ZEin
  RETURN
ELSEIF (Zprojectile == 1 .AND. Ztarget == 1001 .AND. ZRx == -99) THEN
  Zpheight = 0.0364 * ZEin ** 2 + 0.125 * ZEin
  RETURN
ELSEIF (Zprojectile == 1 .AND. Ztarget == 2003 .AND. ZRx == 0) THEN
  Zpheight = ZEin
  RETURN
ELSEIF (Zprojectile == 2 .AND. Ztarget == 1) THEN
  Zpheight = ZEin
  RETURN
ELSEIF (Zprojectile == 2 .AND. Ztarget == 6) THEN
  Zpheight = ZEin
  RETURN
ELSE
  Zpheight = 0
  RETURN
END IF

END SUBROUTINE PulseHeight

END PROGRAM
```

A Sample .peaks File

25000000	Total	Mean(T)	Direct	Mean(D)	No XTalk	Mean(N)
401	2249	2.2075E+01	0	0.0000E+00	2190	2.2055E+01
405	2648	2.2104E+01	0	0.0000E+00	2496	2.2039E+01
409	4072	2.2219E+01	221	2.1688E+01	3438	2.2065E+01
413	43140	2.1797E+01	37934	2.1740E+01	42054	2.1772E+01
417	66022	2.1774E+01	60767	2.1734E+01	64800	2.1755E+01
421	35946	2.1810E+01	30868	2.1732E+01	34763	2.1774E+01
425	22037	2.1820E+01	18115	2.1737E+01	21324	2.1785E+01
429	11670	2.1820E+01	8727	2.1730E+01	11287	2.1785E+01
433	4909	2.1900E+01	2671	2.1748E+01	4687	2.1851E+01
437	1832	2.2011E+01	150	2.1727E+01	1719	2.1955E+01
441	1303	2.2039E+01	0	0.0000E+00	1256	2.2018E+01
445	955	2.2134E+01	0	0.0000E+00	913	2.2102E+01
449	765	2.2207E+01	0	0.0000E+00	750	2.2188E+01
453	605	2.2326E+01	0	0.0000E+00	593	2.2314E+01
457	419	2.2456E+01	0	0.0000E+00	404	2.2441E+01
461	297	2.2572E+01	0	0.0000E+00	288	2.2542E+01
465	218	2.2674E+01	0	0.0000E+00	209	2.2641E+01
469	158	2.2759E+01	0	0.0000E+00	152	2.2730E+01
473	118	2.3000E+01	0	0.0000E+00	116	2.2991E+01
477	76	2.3145E+01	0	0.0000E+00	74	2.3135E+01
481	80	2.3237E+01	0	0.0000E+00	76	2.3224E+01
485	50	2.3380E+01	0	0.0000E+00	50	2.3380E+01
489	36	2.3639E+01	0	0.0000E+00	36	2.3639E+01
493	32	2.3281E+01	0	0.0000E+00	32	2.3281E+01
497	28	2.3464E+01	0	0.0000E+00	28	2.3464E+01
501	14	2.3643E+01	0	0.0000E+00	14	2.3643E+01
505	13	2.3538E+01	0	0.0000E+00	12	2.3500E+01
509	9	2.2778E+01	0	0.0000E+00	9	2.2778E+01
513	6	2.3000E+01	0	0.0000E+00	6	2.3000E+01
517	8	2.3750E+01	0	0.0000E+00	8	2.3750E+01
521	4	2.2250E+01	0	0.0000E+00	4	2.2250E+01
525	5	2.2800E+01	0	0.0000E+00	5	2.2800E+01

Appendix E – The *JoinSS* and *JoinPixels* Codes

The *JoinSS* Code

```
PROGRAM JoinSS
! Version 2.00
! Written February 2nd, 2009
! By Brandon Grogan
! At The Oak Ridge National Laboratory
! Last Edited: November 28th, 2009

IMPLICIT NONE
CHARACTER :: FileBase*8, NumSSTxt*4, DetsPerSSTxt*4, FirstDetTxt*4,
FileNumTxt*2
CHARACTER :: StoDetTxt*10, DetSepTxt*10
INTEGER :: I, J
INTEGER :: NumSS, DetsPerSS, FirstDet, TotalDet, DetNum
INTEGER(KIND=4), ALLOCATABLE :: PeaksSS(:,:)
INTEGER(KIND=4) :: nps
REAL(KIND=4) :: DummyReal, StoDet, DetSep, Phi, AngleStart
REAL(KIND=4), ALLOCATABLE :: Angles(:)
INTEGER :: ErrorCode


! Get input from the command line
CALL GETARG(1, FileBase)
CALL GETARG(2, NumSSTxt)
CALL GETARG(3, DetsPerSSTxt)
CALL GETARG(4, FirstDetTxt)
CALL GETARG(5, StoDetTxt)
CALL GETARG(6, DetSepTxt)

READ (NumSSTxt, '(I4)') NumSS
READ (DetsPerSSTxt, '(I4)') DetsPerSS
READ (FirstDetTxt, '(I4)') FirstDet
READ (StoDetTxt, '(F10.5)') StoDet
READ (DetSepTxt, '(F10.5)') DetSep

TotalDet = NumSS*DetsPerSS
ALLOCATE(PeaksSS(FirstDet-1:FirstDet+TotalDet-1,4))
ALLOCATE(Angles(FirstDet:FirstDet+TotalDet-1))
PeaksSS = 0

Phi = 2.*ATAN(DetSep/2./StoDet)*360./6.283185
AngleStart = (-DetsPerSS/2. + 1/2./NumSS)*Phi

DO I = FirstDet, FirstDet+TotalDet-1
  PeaksSS(I,1) = I
  Angles(I) = AngleStart + REAL(I-FirstDet)*Phi/REAL(NumSS)
END DO

DO I = 1, NumSS
  IF (I < 10) THEN
    WRITE(FileNumTxt, '(I1)') I
  ELSE
    WRITE(FileNumTxt, '(I2)') I
  END IF
  OPEN(UNIT = I, FILE=TRIM(FileBase) // TRIM(FileNumTxt) // ".peaks",
ACTION="READ", &
```

263

```
  & STATUS="OLD", IOSTAT=ErrorCode)
  IF (ErrorCode > 0) STOP "Error!  Cannot open .DAT file.  Check file name."
  READ(I, '(I12)') nps
  IF (I == 1) PeaksSS(FirstDet-1,1) = nps
    DO J = 1, DetsPerSS
      READ(I, '(I12)', ADVANCE="NO", IOSTAT=ErrorCode) DetNum
      ! Exits the loop if the end of the peaks file is reached before DetsPerSS
      IF (ErrorCode < 0) EXIT
      IF (DetNum < FirstDet .OR. DetNum > FirstDet + TotalDet) THEN
        PRINT *, "Error!  Unexpected detector cell number encountered in peaks
file."
        STOP
      END IF
      READ(I, '(I12, ES12.4, I12, ES12.4, I12)') PeaksSS(DetNum,2), DummyReal,
&
      & PeaksSS(DetNum,3), DummyReal, PeaksSS(DetNum,4)
    END DO
  CLOSE(UNIT=I)
END DO


OPEN(UNIT = 99, FILE=TRIM(FileBase) // ".peaks", STATUS="REPLACE",
ACTION="WRITE")
WRITE (99, '(I12,A)') PeaksSS(FirstDet-1,1), "    Detector        Total
Direct   No X-Talk"
DO I = FirstDet, FirstDet+TotalDet-1
  WRITE(99, '(F12.6,4I12)') Angles(I), PeaksSS(I,1), PeaksSS(I,2),
PeaksSS(I,3), PeaksSS(I,4)
END DO
CLOSE(UNIT=99)
END PROGRAM
```

The *JoinPeaks* Code

```
PROGRAM JoinPixels
! Version 2.00
! Written February 2nd, 2009
! By Brandon Grogan
! At The Oak Ridge National Laboratory
! Last Edited: November 28th, 2009

IMPLICIT NONE

INTEGER(KIND=4), ALLOCATABLE :: Peaks(:,:,:)
CHARACTER :: FileBase*8, NumPixelsTxt*4, FileNumTxt*2
INTEGER :: I, J, K, NumPixels, NumDets
INTEGER :: ErrorCode
INTEGER(KIND=4) :: DummyInt, nps, PeakSum
REAL(KIND=4), ALLOCATABLE :: Angles(:)
REAL(KIND=4) :: DummyReal

CALL GETARG(1,FileBase)
CALL GETARG(2,NumPixelsTxt)

READ (NumPixelsTxt, '(I4)') NumPixels

OPEN(UNIT=99, File=TRIM(FileBase) // "1.peaks", ACTION="READ", STATUS="OLD", &
& IOSTAT=ErrorCode)
```

```fortran
IF (ErrorCode > 0) STOP "Error!  Input file could not be read.  Check
filename."
DO I = 1, 1000
  READ(99,'(I12)', IOSTAT=ErrorCode) DummyInt
  IF (ErrorCode < 0) EXIT
END DO
CLOSE(UNIT=99)

NumDets = I - 2

ALLOCATE(Peaks(NumDets,0:NumPixels+1,3))
ALLOCATE(Angles(NumDets))

DO I = 1, NumPixels
  IF (I < 10) THEN
    WRITE(FileNumTxt, '(I1)') I
  ELSE
    WRITE(FileNumTxt, '(I2)') I
  END IF

  OPEN(UNIT=I, FILE=TRIM(FileBase) // TRIM(FileNumTxt) // ".peaks",
STATUS="OLD", &
  & ACTION="READ", POSITION="REWIND", IOSTAT=ErrorCode)
  IF (ErrorCode > 0) STOP "Error! Missing one or more .peaks files."
  READ (I, '(I12)') DummyInt
  IF (I == 1) nps = DummyInt
  DO J = 1, NumDets
    READ (I, '(F12.6,4I12)') DummyReal, Peaks(J, 0, 1), Peaks(J, I, 1), &
    & Peaks(J, I, 2), Peaks(J,I,3)
    IF (I == 1) Angles(J) = DummyReal
  END DO
  CLOSE(UNIT=I)
END DO

DO I = 1,3
  DO J = 1, NumDets
    PeakSum = 0
    DO K = 1, NumPixels
      PeakSum = PeakSum + Peaks(J,K,I)
    END DO
    Peaks(J,NumPixels+1,I) = PeakSum
  END DO
END DO

OPEN(UNIT=98, FILE=TRIM(FileBase) // ".peaks", STATUS="REPLACE",
ACTION="WRITE")

! Writes the header row into the output file
DO I = 1,3
  WRITE(98, '(I12, A)', ADVANCE="NO") nps, "    Detector"
  DO J = 1, NumPixels
    WRITE(98, '(A, I2)', ADVANCE="NO") "     Pixel", J
  END DO
  IF (I .LE. 2) THEN
    WRITE(98, '(A)', ADVANCE="NO") "        Total  Unc. Total          "
  ELSE
    WRITE(98, '(A)') "        Total  Unc. Total"
  END IF
END DO

DO J = 1, NumDets
```

```fortran
  DO I = 1, 3
    WRITE(98, '(F12.6)', ADVANCE="NO") Angles(J)
    WRITE(98, '(I12)', ADVANCE="NO") Peaks(J,0,1)
    DO K = 1, NumPixels
      WRITE(98, '(I12)', ADVANCE="NO") Peaks(J,K,I)
    END DO
    WRITE(98, '(I12)', ADVANCE="NO") Peaks(J,NumPixels+1,I)
    WRITE(98, '(ES12.4)', ADVANCE="NO") SQRT(REAL(Peaks(J,NumPixels+1,I)))
    IF (I .LE. 2) THEN
      WRITE(98, '(A)', ADVANCE="NO") "              "
    ELSE
      WRITE(98, *)
    END IF
  END DO
END DO

CLOSE(UNIT=98)

END PROGRAM JoinPixels
```

Appendix F – The *GaussFit* Code

```
PROGRAM GaussFit
! Version 1.00
! By Brandon Grogan
! At The Oak Ridge National Laboratory
! Last Edited: January 24th, 2010

IMPLICIT NONE

INTEGER(KIND=4) :: I, J, K, L, M, NPSObject, NPSVoid
REAL :: DummyReal
! Peaks layout - Rows = detectors, Column 0 = angle, Col 1 = Total, Col 2 =
Direct, Col 3 =
! NoXTalk, Col 4 = Scattered, Col 5 = Fitted, Col6 = PScF, Col7 = Scatter in
Array,
! Col8 = Scatter in Object, Col9 = raw object scatter; Pane 1 = object, Pane 2
= void
REAL(KIND=4) :: Peaks(32,0:9,2)
! Variables used for finding the Gaussian function
REAL(KIND=4) :: FitMax1, FitSD1
REAL(KIND=4) :: FitMax2, FitSD2, FitMax3, FitSD3
REAL(KIND=4) :: VarSum, NumCounts, ChiSquared
! The Measured Attenuation at the Center Detector
REAL(KIND=4) :: MeasAtt
CHARACTER :: ObjectFile*80, VoidFile*80, DummyChar*1, StoC*1, MFP*3, OutFile*80


! Read the Object and Void .peaks File Names from the command line
CALL GETARG(1, ObjectFile)
CALL GETARG(2, VoidFile)

! Read the Object to Center Distance and MFP of material
CALL GETARG(3, StoC)
CALL GETARG(4, MFP)

! Open .peaks files and read the data into the .peaks array
OPEN(UNIT=1, FILE=TRIM(ObjectFile), STATUS="OLD", ACTION="READ",
POSITION="REWIND")
OPEN(UNIT=2, FILE=TRIM(VoidFile), STATUS="OLD", ACTION="READ",
POSITION="REWIND")

! Reads the number of source histories from the .peaks files
READ (1,*) NPSObject
READ (2,*) NPSVoid

! Read data from the .peaks files
DO I = 1, 32

  READ (1,*) DummyReal, Peaks(I,1,1), DummyReal, Peaks(I,2,1), DummyReal,
Peaks(I,3,1)
  ! Calculate the Detector Angle
  Peaks(I,0,1) = -25.0005 + 1.6667 * REAL(I-1)
  ! Calculate the Number of Scattered Counts in each detector
  Peaks(I,4,1) = Peaks(I,3,1) - Peaks(I,2,1)

  READ (2,*) DummyReal, Peaks(I,1,2), DummyReal, Peaks(I,2,2), DummyReal,
Peaks(I,3,2)
  ! Calculate the Detector Angle
  Peaks(I,0,2) = -25.0005 + 1.6667 * REAL(I-1)
```

```fortran
  ! Calculate the Number of Scattered Counts in each detector
  Peaks(I,4,2) = Peaks(I,3,2) - Peaks(I,2,2)

END DO


CLOSE(UNIT=1)
CLOSE(UNIT=2)

! Calculate the true attenuation for the center detector.  This uses the
MeasAtt = -LOG(Peaks(16,2,1) * NPSVoid / Peaks(16,2,2) / NPSObject)

! Now Calculate the Maximum and std. deviation of the scattered column.
! These values will be used as the initial guess of the fitted parameters

FitMax1 = 0.
FitMax2 = 0.
NumCounts = 0.
VarSum = 0.

DO I = 1, 32

  NumCounts = NumCounts + Peaks(I,4,1)
  VarSum = VarSum + Peaks(I,4,1) * Peaks(I,0,1)**2
  IF(Peaks(I,4,1) > FitMax1) FitMax1 = Peaks(I,4,1)

END DO

FitSD1 = SQRT(VarSum / NumCounts)

! The values from the object fit are taken directly from a gaussian fit of
! a void measurement.
! 1 MeV Threshold Values
FitSD2 = 1.420884
FitMax2 = 0.02969844*Peaks(16,2,1)
FitSD3 = 2.911001
FitMax3 = 0.004047225*Peaks(16,2,1)

! 1.5 MeV Threshold Values
! FitSD2 = 1.434710
! FitMax2 = 0.03117959*Peaks(16,2,1)
! FitSD3 = 2.981670
! FitMax3 = 0.004329379*Peaks(16,2,1)

! This subroutine calculates the optimal parameters of a Gaussian fit using
! an iterative least squares method.
CALL DoFit(Peaks, FitMax1, FitMax2, FitMax3, FitSD1, FitSD2, FitSD3, &
& ChiSquared, NPSObject, NPSVoid)


! The output file contains the original .peaks data plus the scattered values,
! the fit, and the PScF
OutFile = ObjectFile(1:4) // StoC // TRIM(MFP) // ".peaks2"

OPEN(UNIT=3, FILE=TRIM(OutFile), STATUS="REPLACE", ACTION="WRITE")
OPEN(UNIT=4, FILE="Params.out", STATUS="UNKNOWN", ACTION="WRITE",
POSITION="APPEND")

WRITE(3, '(I16, 6A16)') NPSObject, "Total", "Direct", "No XTalk", "Scattered",
&
& "Fitted", "PScF"
```

```fortran
DO I = 1, 32
  DO J = 0, 6, 1
    WRITE(3, '(F16.6)', ADVANCE="NO") Peaks(I,J,1)
  END DO
  WRITE(3,*)
END DO

WRITE(4, '(3A4, 8ES16.7)') ObjectFile(1:4), StoC // "0", TRIM(MFP), MeasAtt,
FitMax1, &
& FitSD1, FitMax2, FitSD2, FitMax3, FitSD3, ChiSquared

CLOSE(UNIT=3)
CLOSE(UNIT=4)

OPEN(UNIT=5, FILE="PScF.out", STATUS="UNKNOWN", ACTION="WRITE",
POSITION="APPEND")
OPEN(UNIT=6, FILE="FittedTotal.out", STATUS="UNKNOWN", ACTION="WRITE",
POSITION="APPEND")
OPEN(UNIT=7, FILE="FittedArray.out", STATUS="UNKNOWN", ACTION="WRITE",
POSITION="APPEND")
OPEN(UNIT=8, FILE="FittedObject.out", STATUS="UNKNOWN", ACTION="WRITE",
POSITION="APPEND")
OPEN(UNIT=9, FILE="ObjectRaw.out", STATUS="UNKNOWN", ACTION="WRITE",
POSITION="APPEND")

WRITE (5, '(A9)', ADVANCE="NO") ObjectFile(1:4) // StoC // TRIM(MFP)
WRITE (6, '(A9)', ADVANCE="NO") ObjectFile(1:4) // StoC // TRIM(MFP)
WRITE (7, '(A9)', ADVANCE="NO") ObjectFile(1:4) // StoC // TRIM(MFP)
WRITE (8, '(A9)', ADVANCE="NO") ObjectFile(1:4) // StoC // TRIM(MFP)
WRITE (9, '(A9)', ADVANCE="NO") ObjectFile(1:4) // StoC // TRIM(MFP)

DO I = 1, 32
  WRITE (5, '(ES16.7)', ADVANCE="NO")
Peaks(I,4,1)*REAL(NPSVoid)/Peaks(16,2,2)/REAL(NPSObject)
  WRITE (6, '(ES16.7)', ADVANCE="NO") Peaks(I,6,1)
  WRITE (7, '(ES16.7)', ADVANCE="NO") Peaks(I,7,1)
  WRITE (8, '(ES16.7)', ADVANCE="NO") Peaks(I,8,1)
  WRITE (9, '(ES16.7)', ADVANCE="NO") Peaks(I,9,1)
END DO

WRITE (5,*)
WRITE (6,*)
WRITE (7,*)
WRITE (8,*)
WRITE (9,*)

CLOSE(UNIT=5)
CLOSE(UNIT=6)
CLOSE(UNIT=7)
CLOSE(UNIT=8)
CLOSE(UNIT=9)

CONTAINS

SUBROUTINE DoFit(ZPeaks, C, D, E, S, T, U, ChiSq, NPSO, NPSV)

REAL(KIND=4), INTENT(INOUT) :: C, S, D, T, E, U, ChiSq, ZPeaks(32,0:9,2)
INTEGER(KIND=4), INTENT(IN) :: NPSO, NPSV
! The XTalk Scatter Array holds the part of the scattering from the object.
! It needs to be identified separately in order to fit one Gaussian to the
! XTalk and another to the Object Scatter.
```

269

```fortran
REAL(KIND=4) :: ObjectScatter(32)
REAL(KIND=4) :: RangeC, RangeS, ChiSqMin, CMin, SMin, CInit, SInit
REAL(KIND=4) :: FofTheta
! These variables are used for smoothing out the center of the scattered counts
! to account for the fact that MCNP doesn't handle small angle scattering well.

RangeC = 2.0 * C
RangeS = 5.0 * S
CInit = C
SInit = 2.5*S
ChiSq = 0.
ChiSqMin = 1.0E+25

PRINT *, C, S, D, T, E, U

! Subtracts the Object Scatter Gaussian from the Total Scattering to leave only
! the scattering due to cross-talk.
  DO L = 1, 32
    IF (L == 16) THEN
     ObjectScatter(L) = ZPeaks(L,4,1)
    ELSE
     FofTheta = D*EXP(-ZPeaks(L,0,1)**2/(2.*T**2)) + E*EXP(-
ABS(ZPeaks(L,0,1))/U)
     ObjectScatter(L) = ZPeaks(L,4,1) - FofTheta
    END IF
  END DO


DO I = 1, 10

  DO J = 0, 200, 1
    C = (-0.5 + 0.005*REAL(J))*RangeC + CInit
    IF (C <= 0.) CYCLE
    DO K = 0, 200, 1
     S = (-0.5 + 0.005*REAL(K))*RangeS + SInit
     IF (S <= 0.) CYCLE
     DO L = 1, 32
      ! IF (L == 16) CYCLE
      FofTheta = C*EXP(-ZPeaks(L,0,1)**2/(2.*S**2))
      ChiSq = ChiSq + ((ObjectScatter(L) - FofTheta) / SQRT(ZPeaks(L,4,1)))**2
!      ChiSq = ChiSq + (ObjectScatter(L) - FofTheta)**2
     END DO
     IF (ChiSq < ChiSqMin) THEN
      ChiSqMin = ChiSq
      CMin = C
      SMin = S
     END IF
    ChiSq = 0.
    END DO ! K Loop
  END DO ! J Loop


  ChiSqMin = 1.0E+25


RangeC = RangeC / 5.0
RangeS = RangeS / 5.0
C = CMin
S = SMin
CInit = C
SInit = S
```

270

```fortran
      ChiSqMin = 1.0E+25

      WRITE (*,'(I2, 6F16.6)') I, C, S, D, T, E, U

      END DO ! I Loop


      DO L = 1, 32
        IF (L == 16) THEN
          FofTheta = C*EXP(-ZPeaks(L,0,1)**2/(2.*S**2))
        ELSE
          FofTheta = C*EXP(-ZPeaks(L,0,1)**2/(2.*S**2)) + &
          & D*EXP(-ZPeaks(L,0,1)**2/(2.*T**2)) + E*EXP(-ABS(ZPeaks(L,0,1))/U)
        END IF
          ZPeaks(L,5,1) = FofTheta
          ZPeaks(L,6,1) = FofTheta * REAL(NPSV) / ZPeaks(16,2,2) / REAL(NPSO)
          ZPeaks(L,9,1) = ObjectScatter(L) * REAL(NPSV) / ZPeaks(16,2,2) / REAL(NPSO)
        ChiSq = ChiSq + ((ZPeaks(L,4,1) - FofTheta) / SQRT(ZPeaks(L,4,1)))**2
!    ChiSq = ChiSq + (ZPeaks(L,4,1) - FofTheta)**2
      END DO

! Convert magnitudes of the fits to a per source neutron basis.
      C = C * REAL(NPSV) / ZPeaks(16,2,2) / REAL(NPSO)
      D = D * REAL(NPSV) / ZPeaks(16,2,2) / REAL(NPSO)
      E = E * REAL(NPSV) / ZPeaks(16,2,2) / REAL(NPSO)


      DO L = 1, 32
        ZPeaks(L,8,1) = C*EXP(-ZPeaks(L,0,1)**2/(2.*S**2))
        IF (L == 16) THEN
          ZPeaks(L,7,1) = 0.
          CYCLE
        END IF
        ZPeaks(L,7,1) = D*EXP(-ZPeaks(L,0,1)**2/(2.*T**2))+E*EXP(-
      ABS(ZPeaks(L,0,1))/U)
      END DO

      END SUBROUTINE DoFit


      END PROGRAM GaussFit
```

Appendix G – The PScF Parameters for a 1 MeV Detector Threshold

Scenario	Attenuation	PScF Maximum	PScF Standard Deviation
Poly30.5	0.500	5.37E-03	6.511
Poly31	0.999	6.17E-03	6.800
Poly31.5	1.498	5.33E-03	7.107
Poly32	1.997	4.11E-03	7.447
Poly32.5	2.496	3.00E-03	7.763
Poly33	2.995	2.10E-03	8.094
Poly33.5	3.496	1.44E-03	8.403
Poly34	3.996	9.69E-04	8.724
Poly34.5	4.494	6.40E-04	9.090
Poly35	4.989	4.21E-04	9.388
Poly35.5	5.489	2.75E-04	9.731
Poly36	5.991	1.77E-04	10.071
Poly36.5	6.485	1.14E-04	10.479
Poly37	6.985	7.35E-05	10.779
Poly40.5	0.500	3.35E-03	7.931
Poly41	0.999	3.88E-03	8.240
Poly41.5	1.498	3.39E-03	8.571
Poly42	1.997	2.65E-03	8.905
Poly42.5	2.496	1.96E-03	9.236
Poly43	2.995	1.39E-03	9.582
Poly43.5	3.496	9.53E-04	9.937
Poly44	3.996	6.45E-04	10.244
Poly44.5	4.494	4.29E-04	10.640
Poly45	4.989	2.85E-04	10.961
Poly45.5	5.490	1.87E-04	11.358
Poly46	5.992	1.21E-04	11.731
Poly46.5	6.485	7.88E-05	12.128
Poly47	6.986	5.07E-05	12.581
Poly50.5	0.500	2.25E-03	9.429
Poly51	0.999	2.64E-03	9.773
Poly51.5	1.498	2.32E-03	10.116
Poly52	1.997	1.83E-03	10.475
Poly52.5	2.496	1.36E-03	10.835
Poly53	2.995	9.68E-04	11.205
Poly53.5	3.496	6.70E-04	11.590
Poly54	3.996	4.57E-04	11.927
Poly54.5	4.494	3.04E-04	12.436
Poly55	4.989	2.05E-04	12.681
Poly55.5	5.490	1.35E-04	13.191
Poly56	5.992	8.85E-05	13.617
Poly60.5	0.500	1.59E-03	11.052
Poly61	0.999	1.88E-03	11.396
Poly61.5	1.498	1.68E-03	11.746
Poly62	1.997	1.33E-03	12.135
Poly62.5	2.496	9.85E-04	12.581

Scenario	Attenuation	PScF Maximum	PScF Standard Deviation
Poly63	2.995	7.08E-04	12.979
Poly63.5	3.496	4.94E-04	13.427
Poly64	3.996	3.39E-04	13.847
Poly64.5	4.494	2.30E-04	14.166
Poly65	4.989	1.54E-04	14.650
Poly70.5	0.500	1.18E-03	12.677
Poly71	0.999	1.40E-03	13.068
Poly71.5	1.498	1.25E-03	13.555
Poly72	1.997	9.94E-04	13.962
Poly72.5	2.496	7.45E-04	14.521
Poly73	2.995	5.36E-04	14.981
Poly73.5	3.496	3.75E-04	15.536
Poly80.5	0.500	9.04E-04	14.613
Poly81	0.999	1.07E-03	15.076
Poly81.5	1.498	9.68E-04	15.547
Poly82	1.997	7.71E-04	16.120
Poly82.5	2.496	5.74E-04	16.883
Poly90.5	0.500	6.99E-04	17.266
Poly91	0.999	8.36E-04	17.907
Poly91.5	1.498	7.57E-04	18.319
Carb30.5	0.505	6.61E-03	6.693
Carb31	1.009	7.72E-03	7.130
Carb31.5	1.513	6.85E-03	7.568
Carb32	2.016	5.41E-03	8.022
Carb32.5	2.521	4.05E-03	8.495
Carb33	3.024	2.92E-03	8.988
Carb33.5	3.529	2.07E-03	9.475
Carb34	4.034	1.44E-03	9.964
Carb34.5	4.537	9.90E-04	10.522
Carb35	5.037	6.75E-04	11.090
Carb35.5	5.545	4.62E-04	11.656
Carb36	6.051	3.12E-04	12.283
Carb36.5	6.553	2.10E-04	13.026
Carb37	7.055	1.42E-04	13.755
Carb40.5	0.505	4.17E-03	8.267
Carb41	1.009	4.95E-03	8.710
Carb41.5	1.513	4.44E-03	9.162
Carb42	2.017	3.54E-03	9.646
Carb42.5	2.521	2.68E-03	10.124
Carb43	3.024	1.96E-03	10.605
Carb43.5	3.530	1.39E-03	11.120
Carb44	4.035	9.72E-04	11.666
Carb44.5	4.537	6.74E-04	12.209
Carb45	5.037	4.61E-04	12.820
Carb45.5	5.545	3.16E-04	13.390
Carb46	6.050	2.14E-04	14.022
Carb46.5	6.552	1.44E-04	14.665
Carb47	7.054	9.85E-05	15.173

Scenario	Attenuation	PScF Maximum	PScF Standard Deviation
Carb50.1	0.101	8.28E-04	9.895
Carb50.2	0.202	1.52E-03	9.829
Carb50.3	0.303	2.07E-03	9.863
Carb50.4	0.404	2.49E-03	9.930
Carb50.5	0.505	2.81E-03	9.991
Carb50.6	0.605	3.04E-03	10.081
Carb50.7	0.706	3.21E-03	10.155
Carb50.8	0.807	3.31E-03	10.261
Carb50.9	0.908	3.37E-03	10.344
Carb51	1.009	3.38E-03	10.434
Carb51.5	1.513	3.06E-03	10.915
Carb52	2.017	2.47E-03	11.412
Carb52.5	2.521	1.87E-03	11.954
Carb53	3.024	1.36E-03	12.499
Carb53.5	3.530	9.80E-04	13.020
Carb54	4.035	6.93E-04	13.540
Carb54.5	4.537	4.82E-04	14.120
Carb55	5.037	3.34E-04	14.721
Carb55.5	5.545	2.30E-04	15.308
Carb56	6.051	1.56E-04	15.962
Carb60.5	0.505	2.00E-03	11.793
Carb61	1.009	2.42E-03	12.273
Carb61.5	1.513	2.20E-03	12.809
Carb62	2.017	1.79E-03	13.358
Carb62.5	2.521	1.36E-03	13.925
Carb63	3.024	1.01E-03	14.445
Carb63.5	3.530	7.26E-04	15.001
Carb64	4.035	5.19E-04	15.565
Carb64.5	4.969	2.37E-04	16.190
Carb65	5.037	2.50E-04	17.078
Carb70.5	0.505	1.47E-03	13.715
Carb71	1.009	1.80E-03	14.175
Carb71.5	1.513	1.66E-03	14.732
Carb72	2.017	1.35E-03	15.378
Carb72.5	2.521	1.04E-03	16.028
Carb73	3.024	7.75E-04	16.492
Carb73.5	3.530	5.63E-04	17.108
Carb74	4.034	4.03E-04	17.744
Carb80.5	0.505	1.13E-03	15.705
Carb81	1.009	1.40E-03	16.083
Carb81.5	1.513	1.29E-03	16.716
Carb82	2.017	1.06E-03	17.454
Carb82.5	2.521	8.22E-04	18.080
Carb90.5	0.505	9.04E-04	17.700
Carb91	1.009	1.12E-03	18.126
Carb91.5	1.513	1.04E-03	18.880
Iron30.5	0.505	9.53E-03	4.997
Iron31	1.009	1.15E-02	5.314

Scenario	Attenuation	PScF Maximum	PScF Standard Deviation
Iron31.5	1.513	1.04E-02	5.666
Iron32	2.016	8.40E-03	6.033
Iron32.5	2.520	6.44E-03	6.380
Iron33	3.023	4.72E-03	6.752
Iron33.5	3.527	3.40E-03	7.136
Iron34	4.031	2.39E-03	7.530
Iron34.5	4.533	1.69E-03	7.899
Iron35	5.032	1.17E-03	8.309
Iron35.5	5.548	8.12E-04	8.687
Iron36	6.056	5.63E-04	9.048
Iron36.5	6.550	3.92E-04	9.361
Iron37	7.048	2.69E-04	9.706
Iron40.5	0.505	6.01E-03	6.320
Iron41	1.009	7.32E-03	6.638
Iron41.5	1.513	6.69E-03	7.001
Iron42	2.016	5.44E-03	7.365
Iron42.5	2.520	4.19E-03	7.721
Iron43	3.023	3.11E-03	8.117
Iron43.5	3.527	2.26E-03	8.472
Iron44	4.031	1.61E-03	8.850
Iron44.5	4.533	1.14E-03	9.229
Iron45	5.032	7.98E-04	9.617
Iron45.5	5.548	5.55E-04	10.037
Iron46	6.057	3.82E-04	10.353
Iron46.5	6.551	2.71E-04	10.637
Iron47	7.050	1.86E-04	11.039
Iron50.5	0.505	4.13E-03	7.645
Iron51	1.009	5.07E-03	7.944
Iron51.5	1.513	4.66E-03	8.310
Iron52	2.016	3.81E-03	8.670
Iron52.5	2.520	2.97E-03	9.018
Iron53	3.023	2.21E-03	9.409
Iron53.5	3.528	1.61E-03	9.766
Iron54	4.031	1.15E-03	10.163
Iron54.5	4.534	8.23E-04	10.575
Iron55	5.033	5.80E-04	10.956
Iron55.5	5.549	4.09E-04	11.336
Iron56	6.058	2.82E-04	11.647
Iron56.5	6.552	1.97E-04	12.053
Iron57	7.051	1.36E-04	12.599
Iron60.5	0.505	3.03E-03	8.864
Iron61	1.009	3.72E-03	9.204
Iron61.5	1.513	3.44E-03	9.555
Iron62	2.016	2.83E-03	9.937
Iron62.5	2.520	2.20E-03	10.311
Iron63	3.023	1.65E-03	10.672
Iron63.5	3.527	1.21E-03	11.064
Iron64	4.031	8.68E-04	11.483

Scenario	Attenuation	PScF Maximum	PScF Standard Deviation
Iron64.5	4.534	6.17E-04	11.937
Iron65	5.033	4.36E-04	12.441
Iron65.5	5.549	3.08E-04	12.782
Iron66	6.058	2.15E-04	13.068
Iron66.5	6.553	1.49E-04	13.609
Iron67	7.050	1.05E-04	14.080
Iron70.5	0.505	2.31E-03	10.111
Iron71	1.009	2.85E-03	10.414
Iron71.5	1.513	2.65E-03	10.759
Iron72	2.016	2.18E-03	11.175
Iron72.5	2.520	1.70E-03	11.555
Iron73	3.023	1.28E-03	11.938
Iron73.5	3.527	9.41E-04	12.332
Iron74	4.031	6.77E-04	12.800
Iron74.5	4.533	4.83E-04	13.258
Iron75	5.033	3.47E-04	13.622
Iron75.5	5.549	2.43E-04	14.123
Iron76	6.058	1.68E-04	14.688
Iron76.5	6.552	1.17E-04	15.274
Iron77	7.051	8.18E-05	15.469
Iron80.5	0.505	1.81E-03	11.295
Iron81	1.009	2.25E-03	11.634
Iron81.5	1.513	2.09E-03	11.983
Iron82	2.016	1.73E-03	12.403
Iron82.5	2.520	1.35E-03	12.791
Iron83	3.023	1.02E-03	13.251
Iron83.5	3.527	7.48E-04	13.688
Iron84	4.031	5.39E-04	14.202
Iron84.5	4.534	3.84E-04	14.706
Iron85	5.033	2.74E-04	15.232
Iron85.5	5.549	1.94E-04	15.497
Iron90.5	0.505	1.45E-03	12.561
Iron91	1.009	1.81E-03	12.881
Iron91.5	1.513	1.68E-03	13.265
Iron92	2.016	1.40E-03	13.716
Iron92.5	2.520	1.09E-03	14.130
Iron93	3.023	8.28E-04	14.591
Lead30.5	0.507	1.67E-02	3.269
Lead31	1.014	1.97E-02	3.532
Lead31.5	1.521	1.75E-02	3.815
Lead32	2.027	1.38E-02	4.121
Lead32.5	2.534	1.02E-02	4.454
Lead33	3.040	7.27E-03	4.862
Lead33.5	3.547	5.06E-03	5.294
Lead34	4.053	3.41E-03	5.844
Lead34.5	4.559	2.28E-03	6.464
Lead35	5.062	1.52E-03	7.099
Lead35.5	5.581	1.01E-03	7.782

Scenario	Attenuation	PScF Maximum	PScF Standard Deviation
Lead36	6.089	6.89E-04	8.273
Lead36.5	6.588	4.70E-04	8.741
Lead37	7.086	3.19E-04	9.305
Lead40.5	0.507	1.11E-02	4.076
Lead41	1.014	1.32E-02	4.372
Lead41.5	1.521	1.17E-02	4.698
Lead42	2.027	9.39E-03	5.012
Lead42.5	2.534	7.00E-03	5.390
Lead43	3.040	5.03E-03	5.815
Lead43.5	3.547	3.51E-03	6.292
Lead44	4.053	2.38E-03	6.903
Lead44.5	4.559	1.61E-03	7.516
Lead45	5.062	1.09E-03	8.170
Lead45.5	5.580	7.36E-04	8.802
Lead46	6.089	4.94E-04	9.418
Lead46.5	6.587	3.41E-04	9.922
Lead47	7.086	2.33E-04	10.471
Lead50.5	0.507	7.82E-03	4.923
Lead51	1.014	9.34E-03	5.244
Lead51.5	1.521	8.37E-03	5.581
Lead52	2.027	6.67E-03	5.962
Lead52.5	2.535	5.02E-03	6.361
Lead53	3.040	3.63E-03	6.826
Lead53.5	3.547	2.54E-03	7.338
Lead54	4.053	1.76E-03	7.901
Lead54.5	4.559	1.20E-03	8.536
Lead55	5.062	8.23E-04	9.137
Lead55.5	5.580	5.59E-04	9.781
Lead56	6.089	3.83E-04	10.238
Lead56.5	6.588	2.63E-04	10.791
Lead57	7.087	1.79E-04	11.393
Lead60.5	0.507	5.77E-03	5.764
Lead61	1.014	6.90E-03	6.133
Lead61.5	1.521	6.20E-03	6.505
Lead62	2.027	4.98E-03	6.880
Lead62.5	2.534	3.78E-03	7.289
Lead63	3.040	2.74E-03	7.791
Lead63.5	3.547	1.95E-03	8.315
Lead64	4.052	1.36E-03	8.866
Lead64.5	4.559	9.49E-04	9.361
Lead65	5.062	6.48E-04	9.971
Lead65.5	5.580	4.44E-04	10.628
Lead66	6.089	3.08E-04	10.996
Lead66.5	6.587	2.12E-04	11.459
Lead67	7.086	1.44E-04	12.313
Lead70.5	0.507	4.41E-03	6.607
Lead71	1.014	5.31E-03	6.976
Lead71.5	1.521	4.79E-03	7.382

Scenario	Attenuation	PScF Maximum	PScF Standard Deviation
Lead72	2.027	3.87E-03	7.772
Lead72.5	2.534	2.96E-03	8.192
Lead73	3.040	2.16E-03	8.701
Lead73.5	3.547	1.55E-03	9.206
Lead74	4.052	1.09E-03	9.677
Lead74.5	4.559	7.64E-04	10.220
Lead75	5.062	5.29E-04	10.763
Lead75.5	5.581	3.63E-04	11.320
Lead76	6.090	2.55E-04	11.727
Lead80.5	0.507	3.50E-03	7.404
Lead81	1.014	4.22E-03	7.801
Lead81.5	1.521	3.83E-03	8.220
Lead82	2.027	3.10E-03	8.637
Lead82.5	2.534	2.38E-03	9.042
Lead83	3.040	1.75E-03	9.538
Lead83.5	3.547	1.27E-03	9.983
Lead84	4.052	9.00E-04	10.451
Lead84.5	4.559	6.32E-04	10.955
Lead90.5	0.507	2.83E-03	8.184
Lead91	1.014	3.43E-03	8.599
Lead91.5	1.521	3.13E-03	9.003
Lead92	2.027	2.55E-03	9.418
Lead92.5	2.534	1.97E-03	9.827

Appendix H – The *ScatterSubtract* Code

The *ScatterSubtract* Code

```fortran
PROGRAM ScatterSubtract
!-----------------------------------------------------------------------
! Written by Brandon R. Grogan
! at the Oak Ridge National Laboratory
! Last Modified 13 January 2010
!-----------------------------------------------------------------------
IMPLICIT NONE
INTEGER(KIND=4) :: I, J, K, L, M, ErrorCode, NumDets, NumSS
INTEGER(KIND=4) :: MatNum, LOK
REAL(KIND=4) :: DR, NPSO, NPSV, ObjtoDet, ChiSq(3)
REAL(KIND=4), ALLOCATABLE :: VoidPeaks(:,:,:), ObjPeaks(:,:,:)
REAL(KIND=4), ALLOCATABLE :: ISF(:,:,:), Attenuation(:,:,:)
REAL(KIND=4), ALLOCATABLE :: Uncertainty(:,:,:), Scatter(:,:,:)
CHARACTER :: ObjFile*80, VoidFile*80, SSText*4, OtoDText*4, Material*16
CHARACTER :: NumDetText*4, LOKText*1
REAL(KIND=4) :: Epsilon

Epsilon = 0.00001


!-----------------------------------------------------------------------
! STEP 1: Read user input from the command line.
! SYNTAX: ScatterSubtract <object .peaks file> <void .peaks file> <# SS>
!         <# Detectors> <Object to Center Distance> <Material>
!-----------------------------------------------------------------------
CALL GETARG(1, ObjFile)
CALL GETARG(2, VoidFile)
CALL GETARG(3, SSText)
CALL GETARG(4, NumDetText)
CALL GETARG(5, OtoDText)
CALL GETARG(6, Material)
CALL GETARG(7, LOKText)

IF (LEN(TRIM(ObjFile)) == 0) THEN
 Print *, "ScatterSubtract Syntax:"
 PRINT *,
 PRINT *, "ScatterSubtract <object .peaks file> <void .peaks file> <# SS>"
 PRINT *, "<Object to Center Distance> <Material>"
 STOP
END IF


READ(SSText, '(I4)') NumSS
READ(NumDetText, '(I4)') NumDets
READ(OtoDText, '(F4.0)') ObjtoDet
READ(LOKText, '(I1)') LOK

IF (Material(1:2) == "Po" .OR. Material(1:2) == "PO" .OR. &
& Material(1:2) == "CH" .OR. Material(1:2) == "CH") THEN
  MatNum = 1
ELSEIF(LOK == 3) THEN
  MatNum = 6
ELSEIF (Material(1:1) == "C" .OR. Material(1:1) == "c") THEN
  MatNum = 2
ELSEIF(LOK == 2) THEN
  MatNum = 5
```

```fortran
      ELSEIF (Material(1:2) == "Ir" .OR. Material(1:2) == "IR" .OR. &
      & Material(1:2) == "ir" .OR. Material(1:2) == "Fe") THEN
        MatNum = 3
      ELSEIF (Material(1:2) == "Le" .OR. Material(1:2) == "LE" .OR. &
      & Material(1:2) == "le" .OR. Material(1:2) == "Pb") THEN
        MatNum = 4
      ELSE
        PRINT *, "Invalid Material!  Program Halted."
        STOP
      END IF

      PRINT *, ObjtoDet, MatNum

      ! Allocate and initialize arrays
      ALLOCATE(ObjPeaks(NumDets+1,6,NumSS))
      ALLOCATE(VoidPeaks(NumDets+1,6,NumSS))
      ALLOCATE(Attenuation(NumDets+1,6,NumSS))
      ALLOCATE(ISF(NumDets,NumDets+1,NumSS))
      ALLOCATE(Uncertainty(NumDets,5,NumSS))
      ALLOCATE(Scatter(NumDets,5,NumSS))

      ObjPeaks = 0.
      VoidPeaks = 0.
      Attenuation = 0.
      ISF = 0.
      Uncertainty = 0.


      !------------------------------------------------------------------------
      ! STEP 2: Read the .peaks files into memory.
      !------------------------------------------------------------------------

      OPEN(UNIT=1, FILE=TRIM(ObjFile), STATUS="OLD", ACTION="READ", IOSTAT=ErrorCode)
      OPEN(UNIT=2, FILE=TRIM(VoidFile), STATUS="OLD", ACTION="READ", &
      IOSTAT=ErrorCode)

      READ(1, *) NPSO
      READ(2, *) NPSV

      DO I = 1, NumDets
        DO K = 1, NumSS
            READ(1, *) ObjPeaks(I,1,K), DR, DR, DR, DR, DR, DR, DR, DR, DR, &
            & ObjPeaks(I,2,K), DR, DR, DR, DR, DR, DR, DR, DR, DR, DR, DR, &
            & ObjPeaks(I,3,K), DR, DR, DR, DR, DR, DR, DR, DR, DR, DR, DR, &
            & ObjPeaks(I,4,K)
            READ(2, *) VoidPeaks(I,1,K), DR, DR, DR, DR, DR, DR, DR, DR, DR, &
            & VoidPeaks(I,2,K), DR, DR, DR, DR, DR, DR, DR, DR, DR, DR, DR, &
            & VoidPeaks(I,3,K), DR, DR, DR, DR, DR, DR, DR, DR, DR, DR, DR, &
            & VoidPeaks(I,4,K)
            VoidPeaks(I,5,K) = VoidPeaks(I,4,K)
            ObjPeaks(I,5,K) = ObjPeaks(I,4,K)
            Attenuation(I,1,K) = VoidPeaks(I,1,K)
        END DO
      END DO

      CLOSE(UNIT=1)
      CLOSE(UNIT=2)

      PRINT *, NPSO, NPSV

      !------------------------------------------------------------------------
```

```fortran
! STEP 3: Convert I0measured to I0corrected
!------------------------------------------------------------------------------

CALL FindI0(VoidPeaks, ISF)

! Now normalize the ISF Array to the object counts.

DO K = 1, NumSS
 DO I = 1, NumDets
  DO J = 1, NumDets+1
    ISF(I,J,K) = ISF(I,J,K) * NPSO / NPSV
   END DO
  END DO
END DO

!------------------------------------------------------------------------------
! STEP 4: Convert Imeasured to Icorrected and use it to find the corrected
! attenuation.
!------------------------------------------------------------------------------

CALL FindAttenuation(VoidPeaks, ObjPeaks, ISF, Attenuation, NPSO, NPSV, &
& ObjtoDet, MatNum)

!------------------------------------------------------------------------------
! STEP 5: Find the uncertainty on the attenuation values.
!------------------------------------------------------------------------------

CALL FindUncertainty(VoidPeaks, ObjPeaks, Uncertainty)

!------------------------------------------------------------------------------
! STEP 6: Find the fraction of scattering in the total, NoXTalk, and
! corrected object transmission values.
!------------------------------------------------------------------------------

CALL FindScattering(ObjPeaks,Scatter)

!------------------------------------------------------------------------------
! STEP 7: Find the Chi-Squared goodness of fit values for the total, NoXtalk,
! and Corrected attenuation values.
!------------------------------------------------------------------------------

CALL FindChiSq(Attenuation, Uncertainty, ChiSq)

!------------------------------------------------------------------------------
! STEP 8: Write Output to Files.  The 5 files are:
!
! (3) void.out - contains the void correlation values (incl. corrected)
! (4) object.out - contains the object correlation values (incl. corrected)
! (5) attenuation.out - contains the attenuation values
! (6) scatter.out - contains the scatter fractions
! (7) ChiSq.out - contains the chi squared goodness of fit results
!------------------------------------------------------------------------------

OPEN(UNIT=3, FILE="Void.out", STATUS="REPLACE", ACTION="WRITE")
OPEN(UNIT=4, FILE="Object.out", STATUS="REPLACE", ACTION="WRITE")
OPEN(UNIT=5, FILE="Attenuation.out", STATUS="REPLACE", ACTION="WRITE")
OPEN(UNIT=6, FILE="Scatter.out", STATUS="REPLACE", ACTION="WRITE")
OPEN(UNIT=7, FILE="ChiSq.out", STATUS="REPLACE", ACTION="WRITE")

WRITE(3, '(I16,5A16)') INT(NPSV),"Total","Direct","No XTalk","Corrected", &
& "Frac. Error"
```

```fortran
WRITE(4, '(I16,5A16)') INT(NPSO),"Total","Direct","No XTalk","Corrected", &
& "Frac. Error"
WRITE(5, '(7A16)', ADVANCE="NO") "Angle", "Total", "Direct", "No XTalk", &
& "Corrected", "Frac. Error", " "
WRITE(5, '(4A16)') "Total", "Direct", "No XTalk", "Corrected"
WRITE(6, '(4A16)') "Angle", "Total", "No XTalk", "Corrected"
WRITE(7, '(3A16)') "Total", "No XTalk", "Corrected"

DO I = 1, NumDets
  DO K = 1, NumSS
   DO L = 1,6
     WRITE(3, '(ES16.6)', ADVANCE="NO") VoidPeaks(I,L,K)
     WRITE(4, '(ES16.6)', ADVANCE="NO") ObjPeaks(I,L,K)
     WRITE(5, '(ES16.6)', ADVANCE="NO") Attenuation(I,L,K)
     IF (L == 3) CYCLE
     IF (L == 6) EXIT
     WRITE(6, '(ES16.6)', ADVANCE="NO") Scatter(I,L,K)
    END DO
     WRITE(5, '(A16)', ADVANCE="NO") " "
    DO L = 2,5
     WRITE(5, '(ES16.6)', ADVANCE="NO") Uncertainty(I,L,K)
    END DO

    WRITE(3,*)
    WRITE(4,*)
    WRITE(5,*)
    WRITE(6,*)
  END DO
END DO

WRITE(7,'(3ES16.6)') ChiSq(1), ChiSq(2), ChiSq(3)

CLOSE(UNIT=3)
CLOSE(UNIT=4)
CLOSE(UNIT=5)
CLOSE(UNIT=6)
CLOSE(UNIT=7)


!-----------------------------------------------------------------------------
! Subroutines
!-----------------------------------------------------------------------------
CONTAINS

!-----------------------------------------------------------------------------
! Subroutine FindI0 - This subroutine calculates the corrected values of I0
! from the measured.  Reutrns the I0 values and the ISF.
!-----------------------------------------------------------------------------
SUBROUTINE FindI0(ZVoid, ISFCounts)
REAL(KIND=4), INTENT(INOUT) :: ZVoid(NumDets+1,6,NumSS)
REAL(KIND=4), INTENT(INOUT) :: ISFCounts(NumDets, NumDets+1,NumSS)
REAL(KIND=4) :: ZISF(NumDets, NumDets+1, NumSS)
REAL(KIND=4) :: ConvCheck

OPEN(UNIT=99, FILE="Void.iter", STATUS="REPLACE", ACTION="WRITE")
WRITE(99,'(A13)', ADVANCE="NO") "             "
DO I = 1, NumDets
  DO K = 1, NumSS
    WRITE(99,'(F13.6)', ADVANCE="NO") ZVoid(I,1,K)
  END DO ! K Loop
END DO ! I Loop
WRITE(99,*)
```

```fortran
WRITE (99, '(A13)', ADVANCE="NO") "Iteration   0"
DO I = 1, NumDets
  DO K = 1, NumSS
    WRITE(99,'(ES13.6)', ADVANCE="NO") ZVoid(I,5,K)
  END DO ! K Loop
END DO ! I Loop
WRITE(99,*)



! Find the normalized ISF Values (Scattered Counts in det. I per
! direct count in det J.
DO K = 1, NumSS
  DO I = 1, NumDets
    DO J = 1, NumDets
      CALL GetISF(ZVoid(J,1,K), ZVoid(I,1,K), ZISF(I,J,K))
    END DO ! J Loop
  END DO ! I Loop
END DO ! K Loop

! Maximum of 100 iterations, although convergence will probably come much
faster.
DO L = 1, 100
  DO K = 1, NumSS
   DO I = 1, NumDets
    DO J = 1, NumDets

      ! The actual number of scattered counts produced in I by det. J
      ISFCounts(I,J,K) = ZISF(I,J,K) * ZVoid(J,5,K)

      ! Total counts scattered into detector I
      ISFCounts(I,NumDets+1,K) = ISFCounts(I,NumDets+1,K) + ISFCounts(I,J,K)

    END DO ! J Loop
   END DO ! I Loop
  END DO ! K Loop

  DO K = 1, NumSS
   DO I = 1, NumDets

    ZVoid(I,6,K) = ZVoid(I,4,K) - ISFCounts(I,NumDets+1,K)
    IF (ZVoid(I,6,K) < 0.) ZVoid(I,6,K) = 0.

    ! The sum of the difference in counts between iteration will be the
    ! criterion used to determine convergence.
    ZVoid(NumDets+1,6,NumSS) = ZVoid(NumDets+1,6,NumSS) + &
    & ABS(ZVoid(I,4,K) - ZVoid(I,6,K))

   END DO ! I Loop
  END DO ! K Loop


WRITE (99, '(A9,I4)', ADVANCE="NO") "Iteration", L
DO I = 1, NumDets
  DO K = 1, NumSS
    WRITE(99,'(ES13.6)', ADVANCE="NO") ZVoid(I,6,K)
  END DO ! K Loop
END DO ! I Loop
WRITE(99,*)
```

```fortran
   ! Now check for convergence.
   ConvCheck = ABS((ZVoid(NumDets+1,6,NumSS)-ZVoid(NumDets+1,5,NumSS)) / &
   & ZVoid(NumDets+1,5,NumSS))

   IF(ConvCheck < Epsilon) THEN

    ! Get rid of the convergence testing values because they're not need any
more.
    ZVoid(NumDets+1,5,NumSS) = 0.
    ZVoid(NumDets+1,6,NumSS) = 0.
    DO K = 1, NumSS
     DO I = 1, NumDets

      ! Column 5 now becomes the corrected I0 values.
      ZVoid(I,5,K) = ZVoid(I,6,K)

      ! Column 6 will become the fractional error of the corrected values.
      ZVoid(I,6,K) = (ZVoid(I,5,K)-ZVoid(I,3,K))/ZVoid(I,5,K)

      ! This cell will hold the average fractional error for the entire
      ! void run.
      ZVoid(NumDets+1,6,NumSS) = ZVoid(NumDets+1,6,NumSS) + ZVoid(I,6,K)
     END DO
    END DO

    ! Divide by total number of dets. to find average frac. error.
    ZVoid(NumDets+1,6,NumSS) = ZVoid(NumDets+1,6,NumSS) / REAL(NumDets*NumSS)

   WRITE (99, '(A13)', ADVANCE="NO") "Direct       "
   DO I = 1, NumDets
     DO K = 1, NumSS
       WRITE(99,'(ES13.6)', ADVANCE="NO") ZVoid(I,3,K)
     END DO ! K Loop
   END DO ! I Loop
   WRITE(99,*)


    EXIT

   ! If not converged, the values from column 6 (present iteration) are copied
   ! into column 5 (old iteration) and column 6 is set to 0 for the next loop.
   ELSE
    ISFCounts = 0.
    DO K = 1, NumSS
     DO I = 1, NumDets+1
      ZVoid(I,5,K) = ZVoid(I,6,K)
      ZVoid(I,6,K) = 0.
     END DO
    END DO

   WRITE(*, '(I6, 2ES16.6)') L, ZVoid(NumDets+1,5,NumSS), ConvCheck

   END IF

END DO ! L Loop

CLOSE(UNIT=99)
```

284

```
    END SUBROUTINE FindI0

    !-----------------------------------------------------------------------------
    ! Subroutine GetISF - This subroutine returns the ISF scattering fraction
    ! for a given combination of detector angles.
    !-----------------------------------------------------------------------------
    SUBROUTINE GetISF(ThetaFrom, ThetaTo, ISFVal)
    REAL(KIND=4), INTENT(IN) :: ThetaFrom, ThetaTo
    REAL(KIND=4), INTENT(OUT) :: ISFVal
    REAL(KIND=4) :: AngleDiff, ZIter


    AngleDiff = ThetaFrom - ThetaTo

    IF (ABS(AngleDiff) < 0.01) THEN
      ISFVAL = 0.
    ELSE
     ISFVal = 0.02969844 * EXP(-AngleDiff**2 / (2. * 1.420884**2)) + &
     & 0.004047225 * EXP(-ABS(AngleDiff) / 2.911001)
    END IF

    END SUBROUTINE GetISF


    !-----------------------------------------------------------------------------
    ! Subroutine FindAttenuation - This subroutine subtracts the object scatter
    ! from I to find the corrected values and uses them to calculate the corrected
    ! attenuation.
    !-----------------------------------------------------------------------------
    SUBROUTINE FindAttenuation(ZVoid, ZObj, ZISF, ZAtten, ZNPSO, ZNPSV, &
    & ZObjDet, ZMat)
    REAL(KIND=4), INTENT(IN) :: ZVoid(NumDets+1,6,NumSS)
    REAL(KIND=4), INTENT(INOUT) :: ZISF(NumDets,NumDets+1,NumSS)
    REAL(KIND=4), INTENT(INOUT) :: ZObj(NumDets+1,6,NumSS),
    ZAtten(NumDets+1,6,NumSS)
    REAL(KIND=4), INTENT(IN) :: ZNPSO, ZNPSV, ZObjDet
    INTEGER(KIND=4), INTENT(IN) :: ZMat
    REAL(KIND=4) :: ZPScF(NumDets, NumDets+1, NumSS), ConvCheck

    ! Calculates the total, direct, and measured attenuation values.
    ! Also calculates the first guess for corrected attenuation.
    DO K = 1, NumSS
      DO I = 1, NumDets
        ZAtten(I,2,K) = -LOG(ZObj(I,2,K)*ZNPSV/ZVoid(I,2,K)/ZNPSO)
        IF(ZVoid(I,2,K) == 0. .OR. ZObj(I,2,K)==0.) ZAtten(I,2,K) = 0.
        ZAtten(I,3,K) = -LOG(ZObj(I,3,K)*ZNPSV/ZVoid(I,3,K)/ZNPSO)
        IF(ZVoid(I,3,K) == 0. .OR. ZObj(I,3,K)==0.) ZAtten(I,3,K) = 0.
        ZAtten(I,4,K) = -LOG(ZObj(I,4,K)*ZNPSV/ZVoid(I,4,K)/ZNPSO)
        IF(ZVoid(I,4,K) == 0. .OR. ZObj(I,4,K)==0.) ZAtten(I,4,K) = 0.
        ZAtten(I,5,K) = -LOG(ZObj(I,5,K)*ZNPSV/ZVoid(I,5,K)/ZNPSO)
        IF(ZVoid(I,5,K) == 0. .OR. ZObj(I,5,K)==0.) ZAtten(I,5,K) = 0.
        ZObj(NumDets+1,5,NumSS) = ZObj(NumDets+1,5,NumSS) + ZObj(I,5,K)
      END DO ! I Loop
    END DO ! K Loop


    OPEN(UNIT=99, FILE="Obj.iter", STATUS="REPLACE", ACTION="WRITE")
    OPEN(UNIT=98, FILE="Atten.iter", STATUS="REPLACE", ACTION="WRITE")

    WRITE(99,'(A13)', ADVANCE="NO") "             "
    WRITE(98,'(A13)', ADVANCE="NO") "             "
    DO I = 1, NumDets
```

285

```
   DO K = 1, NumSS
      WRITE(99,'(F13.6)', ADVANCE="NO") ZObj(I,1,K)
      WRITE(98,'(F13.6)', ADVANCE="NO") ZAtten(I,1,K)
   END DO ! K Loop
END DO ! I Loop
WRITE(99,*)
WRITE(98,*)

WRITE (99, '(A13)', ADVANCE="NO") "Iteration    0"
WRITE (98, '(A13)', ADVANCE="NO") "Iteration    0"
DO I = 1, NumDets
   DO K = 1, NumSS
      WRITE(99,'(ES13.6)', ADVANCE="NO") ZObj(I,5,K)
      WRITE(98,'(ES13.6)', ADVANCE="NO") ZAtten(I,5,K)
   END DO ! K Loop
END DO ! I Loop
WRITE(99,*)
WRITE(98,*)


! Maximum of 100 iterations, although convergence will probably come much
faster.
DO L = 1, 100

! After each 100 iterations, decrease convergence criteria by a factor of 10.
! IF(MOD(L,100) == 0) Epsilon = 10. * Epsilon

   ! Clear the PScF array at the beginning of each iteration.
   ZPScF = 0.

   DO K = 1, NumSS
    DO I = 1, NumDets

      ! Clear the last column of the ISF array so that it can hold the total
      ! Inter-array scattering into that detector.
      ISF(I,NumDets+1,K) = 0.

      DO J = 1, NumDets

       ! Get the PScF Values using the PScFGEs
       CALL PScFGE(ZObj(J,1,K), ZObj(I,1,K), ZPScF(I,J,K), ZMat, ZObjDet, &
       & ZAtten(J,5,K), L)

       ! Convert the PScF value to counts and sum the total scattered
       ! counts into each detector
       ZPScF(I,J,K) = ZPScF(I,J,K) * ZVoid(J,5,K) * ZNPSO / ZNPSV
       ZPScF(I,NumDets+1,K) = ZPScF(I,NumDets+1,K) + ZPScF(I,J,K)

       ! ISF counts are summed across contributing detectors.  Each value is
weighted
       ! by the attenuation value from the previous iteration.
       ZISF(I,NumDets+1,K) = ZISF(I,NumDets+1,K) + EXP(-
ZAtten(J,5,K))*ZISF(I,J,K)


      END DO ! J Loop
     END DO ! I Loop
   END DO ! K Loop

   DO K = 1, NumSS
    DO I = 1, NumDets
```

286

```
    ! Find Icorr by subtracting the ISF and PScF
    ZObj(I,6,K) = ZObj(I,4,K) - ZISF(I,NumDets+1,K) - ZPScF(I,NumDets+1,K)
    IF(ZObj(I,6,K) <= 0.) ZObj(I,6,K) = 0.

    ! This IF statement is used to force convergence by averaging the results
    ! of each iteration past the 10th.  This is useful for getting measurements
    ! with poor statistics to converge.  Otherwise, there is a tendancy to
    ! oscillate between two values.
    IF (L > 10) THEN
     ZObj(I,6,K) = (ZObj(I,6,K) + REAL(L-10)*ZObj(I,5,K))/REAL(L-9)
    END IF


    ! Now calculate the new attenuation.
    IF(ZObj(I,6,K) == 0. .OR. ZVoid(I,5,K) == 0.) THEN
     ZAtten(I,6,K) = 0.
    ELSE
     ZAtten(I,6,K) = -LOG(ZObj(I,6,K)*ZNPSV/ZVoid(I,5,K)/ZNPSO)
    END IF
    IF(ZAtten(I,6,K) < 0.) ZAtten(I,6,K) = 0.

    ! This value is the total object counts for the entire measurement.  It is
used
    ! for checking convergence.
    ZObj(NumDets+1,6,NumSS) = ZObj(NumDets+1,6,NumSS) + ZObj(I,6,K)
   END DO ! I Loop
  END DO ! K Loop

  ! Writes the corrected value for this iteration to a file.
  WRITE (99, '(A9,I4)', ADVANCE="NO") "Iteration", L
  WRITE (98, '(A9,I4)', ADVANCE="NO") "Iteration", L
  DO I = 1, NumDets
   DO K = 1, NumSS
    WRITE(99,'(ES13.6)', ADVANCE="NO") ZObj(I,6,K)
    WRITE(98,'(ES13.6)', ADVANCE="NO") ZAtten(I,6,K)
   END DO ! K Loop
  END DO ! I Loop
  WRITE(99,*)
  WRITE(98,*)


  IF (L >= 5) THEN
   ! Now check for convergence.
   ConvCheck = ABS((ZObj(NumDets+1,6,NumSS)-ZObj(NumDets+1,5,NumSS)) / &
  & ZObj(NumDets+1,5,NumSS))
  ELSE
   ConvCheck = 1.
  END IF

  IF(ConvCheck < Epsilon) THEN

   ! Get rid of the convergence testing values because they're not need any
more.
   ZObj(NumDets+1,5,NumSS) = 0.
   ZObj(NumDets+1,6,NumSS) = 0.
   DO K = 1, NumSS
    DO I = 1, NumDets

     ! Column 5 now becomes the corrected I values.
     ZObj(I,5,K) = ZObj(I,6,K)
```

```fortran
      ZAtten(I,5,K) = ZAtten(I,6,K)

      ! Column 6 will become the fractional error of the corrected values.
      IF (ZObj(I,5,K) == 0.) THEN
       ZObj(I,6,K) = 0.
       ZAtten(I,6,K) = 0.
      ELSE
       ZObj(I,6,K) = (ZObj(I,5,K)-ZObj(I,3,K))/ZObj(I,5,K)
       ZAtten(I,6,K) = (ZAtten(I,5,K)-ZAtten(I,3,K))/ZAtten(I,5,K)
      END IF

      ! This cell will hold the average fractional error for the entire
      ! void run.
      ZAtten(NumDets+1,6,NumSS) = ZAtten(NumDets+1,6,NumSS) + ZAtten(I,6,K)
     END DO
    END DO

    ! Divide by total number of dets. to find average frac. error.
    ZObj(NumDets+1,6,NumSS) = ZObj(NumDets+1,6,NumSS) / REAL(NumDets*NumSS)

    WRITE (99, '(A13)', ADVANCE="NO") "Direct       "
    WRITE (98, '(A13)', ADVANCE="NO") "Direct       "
    DO I = 1, NumDets
     DO K = 1, NumSS
       WRITE(99,'(ES13.6)', ADVANCE="NO") ZObj(I,3,K)
       WRITE(98,'(ES13.6)', ADVANCE="NO") ZAtten(I,3,K)
     END DO ! K Loop
    END DO ! I Loop
    WRITE(99,*)
    WRITE(98,*)


    EXIT

  ! If not converged, the values from column 6 (present iteration) are copied
  ! into column 5 (old iteration) and column 6 is set to 0 for the next loop.
  ELSE
   DO K = 1, NumSS
    DO I = 1, NumDets+1
     ZObj(I,5,K) = ZObj(I,6,K)
     ZAtten(I,5,K) = ZAtten(I,6,K)
     ZObj(I,6,K) = 0.
    END DO
   END DO

  WRITE(*, '(I6, 2ES16.6)') L, ZObj(NumDets+1,5,NumSS), ConvCheck

  END IF

END DO ! L Loop

CLOSE(UNIT=98)
CLOSE(UNIT=99)

END SUBROUTINE FindAttenuation

!-----------------------------------------------------------------------------
! Subroutine PScFGE - This subroutine returns the appropriate PScF value for
! a given combination of material, Obj to Det distance, and detector angles.
!-----------------------------------------------------------------------------
SUBROUTINE PScFGE(ThetaFrom, ThetaTo, PScFVal, MatNo, ODD, Tau, ZIter)
```

```
REAL(KIND=4), INTENT(IN) :: ThetaFrom, ThetaTo, ODD, Tau
REAL(KIND=4), INTENT(OUT) :: PScFVal
INTEGER(KIND=4), INTENT(IN) :: MatNo, ZIter
REAL(KIND=4) :: AngleDiff, PScFMax, PScFSD, Factor
REAL(KIND=4) :: a0, a1, a2, a3, a4, a5, a6, a7
REAL(KIND=4) :: b0, b1, b2, b3, b4, b5, Beta

AngleDiff = ThetaFrom - ThetaTo

IF (ZIter < 5) THEN
 Factor = REAL(ZIter)/5.
ELSEIF (ZIter < 8) THEN
 Factor = 0.80 + 0.05*(REAL(ZIter-4))
END IF

Beta = LOG(Tau) - Tau

IF (MatNo == 1) THEN

IF(ZIter >= 8) Factor = 0.95

a0 = -2.015225E+00
a1 = -7.679420E-02
a2 = 5.112759E-04
a3 = -1.711049E-06
a4 = 1.288199E+00
a5 = 2.053442E-02
a6 = -2.418549E-03
a7 = 0.000000E+00

b0 = 3.954940E+00
b1 = 4.910408E-02
b2 = 1.018272E-03
b3 = 3.312332E-01
b4 = 1.360807E-02
b5 = 7.200204E-03

ELSEIF(MatNo == 2) THEN

IF(ZIter >= 8) Factor = 0.95

a0 = -2.028606E+00
a1 = -7.184288E-02
a2 = 4.461472E-04
a3 = -1.348499E-06
a4 = 1.131994E+00
a5 = 1.638951E-02
a6 = -1.976645E-03
a7 = 0.000000E+00

b0 = 2.199167E+00
b1 = 1.227376E-01
b2 = 4.967885E-04
b3 = 6.212162E-01
b4 = 3.649710E-02
b5 = 4.528034E-03

ELSEIF(MatNo == 3) THEN

IF(ZIter >= 8) Factor = 0.97
```

289

```
a0 = -1.582257E+00
a1 = -8.241122E-02
a2 = 6.540168E-04
a3 = -2.353720E-06
a4 = 1.037106E+00
a5 = 8.299626E-03
a6 = -2.858238E-03
a7 = 1.541697E-05

b0 = 1.120331E+00
b1 = 1.216453E-01
b2 = 0.000000E+00
b3 = 5.337523E-01
b4 = 1.092541E-02
b5 = 3.126423E-03

ELSEIF(MatNo == 4) THEN

IF(ZIter >= 8) Factor = 1.05

a0 = -1.128685E+00
a1 = -6.752032E-02
a2 = 4.280224E-04
a3 = -1.315156E-06
a4 = 1.214529E+00
a5 = 2.257471E-02
a6 = -2.402139E-03
a7 = 0.000000E+00

b0 = -5.296592E-01
b1 = 1.124436E-01
b2 = -2.060663E-04
b3 = 6.195354E-01
b4 = 4.911718E-02
b5 = 0.000000E+00

ELSEIF(MatNo == 5) THEN

IF(ZIter >= 8) Factor = 1.00

a0 = -2.112510E+00
a1 = -4.644079E-02
a2 = 1.537551E-04
a3 = 0.000000E+00
a4 = 9.314129E-01
a5 = 0.000000E+00
a6 = 0.000000E+00
a7 = 0.000000E+00

b0 = -1.590860E-01
b1 = 1.125601E-01
b2 = 0.000000E+00
b3 = 9.099116E-01
b4 = 0.000000E+00
b5 = 0.000000E+00

ELSEIF(MatNo == 6) THEN

IF(ZIter >= 8) Factor = 1.00

a0 = -1.647906E+00
```

```
a1 = -7.204015E-02
a2 = 3.299292E-04
a3 = 0.000000E+00
a4 = 1.077152E+00
a5 = 0.000000E+00
a6 = -3.809334E-03
a7 = 0.000000E+00

b0 = 1.889415E+00
b1 = 1.123858E-01
b2 = 0.000000E+00
b3 = 7.124436E-01
b4 = 0.000000E+00
b5 = 0.000000E+00

END IF


IF(Tau < 0.1) THEN
  PScFMax = 0.
  PScFSD = 1.0
ELSE
  PScFMax = EXP(a0 + a1*ODD + a2*ODD**2 + a3*ODD**3 + a4*Beta + a5*Beta**2 + &
  & a6*Beta*ODD + a7*Beta*ODD**2)
  PScFSD = b0 + b1*ODD + b2*ODD**2 + b3*Tau + b4*Tau**2 + b5*Tau*ODD
END IF

! PRINT *, PScFMax, PScFSD

PScFVal = Factor * PScFMax * EXP(-AngleDiff**2 / (2.*PScFSD**2))

END SUBROUTINE PScFGE

!-----------------------------------------------------------------------------
! Subroutine FindUncertainty - This subroutine calculates the uncertainty
! in the total, direct, No XTalk, and Corrected attenuation values.
!-----------------------------------------------------------------------------
SUBROUTINE FindUncertainty(ZVoid, ZObj, ZUnc)
REAL(KIND=4), INTENT(IN) :: ZVoid(NumDets+1,6,NumSS), ZObj(NumDets+1,6,NumSS)
REAL(KIND=4), INTENT(INOUT) :: ZUnc(NumDets,5,NumSS)

DO I = 1, NumDets
  DO K = 1, NumSS
    ZUnc(I,1,K) = ZObj(I,1,K)
    DO J = 2, 5
     IF(ZObj(I,J,K) <= 0. .OR. ZVoid(I,J,K) <= 0.) THEN
      ZUnc(I,J,K) = 0.
     ELSE
      ZUnc(I,J,K) = SQRT(1./ZObj(I,J,K) + 1./ZVoid(I,J,K))
     END IF
    END DO
  END DO
END DO

END SUBROUTINE FindUncertainty

!-----------------------------------------------------------------------------
! Subroutine FindScattering - This subroutine calculates the fraction of
! scattering in the total, direct, No XTalk, and Corrected object correlation
! values.  Obviously, direct will be 0, but it's easier to just include it.
!-----------------------------------------------------------------------------
```

```fortran
SUBROUTINE FindScattering(ZObj, ZScat)
REAL(KIND=4), INTENT(IN) :: ZObj(NumDets+1,6,NumSS)
REAL(KIND=4), INTENT(INOUT) :: ZScat(NumDets,5,NumSS)

DO I = 1, NumDets
  DO K = 1, NumSS
    ZScat(I,1,K) = ZObj(I,1,K)
    DO J = 2, 5
     IF(ZObj(I,J,K) <= 0.) THEN
      ZScat(I,J,K) = 0.
     ELSE
      ZScat(I,J,K) = (ZObj(I,J,K)-ZObj(I,3,K)) / ZObj(I,J,K)
     END IF
    END DO
  END DO
END DO

END SUBROUTINE FindScattering

!-------------------------------------------------------------------------
! Subroutine FindScattering - This subroutine calculates the fraction of
! scattering in the total, direct, No XTalk, and Corrected object correlation
! values.  Obviously, direct will be 0, but it's easier to just include it.
!-------------------------------------------------------------------------
SUBROUTINE FindChiSq(ZAtten, ZUnc, ZChi)
REAL(KIND=4), INTENT(IN) :: ZAtten(NumDets+1,6,NumSS), ZUnc(NumDets,5,NumSS)
REAL(KIND=4), INTENT(INOUT) :: ZChi(3)

ZChi = 0.

DO I = 1, NumDets
  DO K = 1, NumSS
   IF(ZUnc(I,3,K) == 0.) CYCLE
   ZChi(1) = ZChi(1) + ((ZAtten(I,2,K) - ZAtten(I,3,K))/ZUnc(I,3,K))**2
   ZChi(2) = ZChi(2) + ((ZAtten(I,4,K) - ZAtten(I,3,K))/ZUnc(I,3,K))**2
   ZChi(3) = ZChi(3) + ((ZAtten(I,5,K) - ZAtten(I,3,K))/ZUnc(I,3,K))**2
  END DO
END DO

END SUBROUTINE FindChiSq

END PROGRAM ScatterSubtract
```

Appendix I – Simulation Testing and Results

The attenuation curves for a scenario consisting of 2 and 4 MFP of lead titled L2L4 are shown in Figure I-1.

The attenuation curves for the Carb61, Iron76, and Lead32 scenarios using the material 6 PScFGE coefficients are shown in Figure I-2, Figure I-3, and Figure I-4. The use of the averaged values results in an overcorrection for carbon and iron scenarios and an undercorrection for lead. As with the material 5 results, there is a significant divergence from the *Direct* values, but the corrected values are still considerably closer than the uncorrected ones. The shape of the attenuation curve is generally improved; however, there is some significant deviation from horizontal, particularly in the Lead32 scenario.
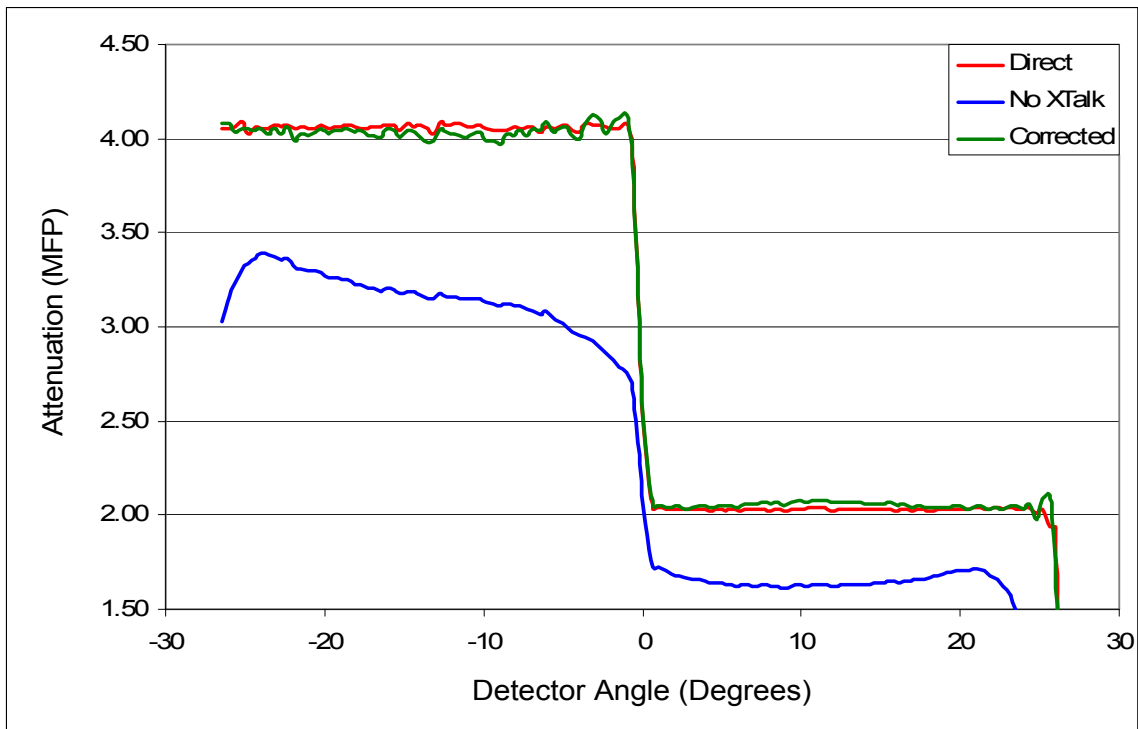
Figure I-1. The attenuation curves for the L2L4 scenario.
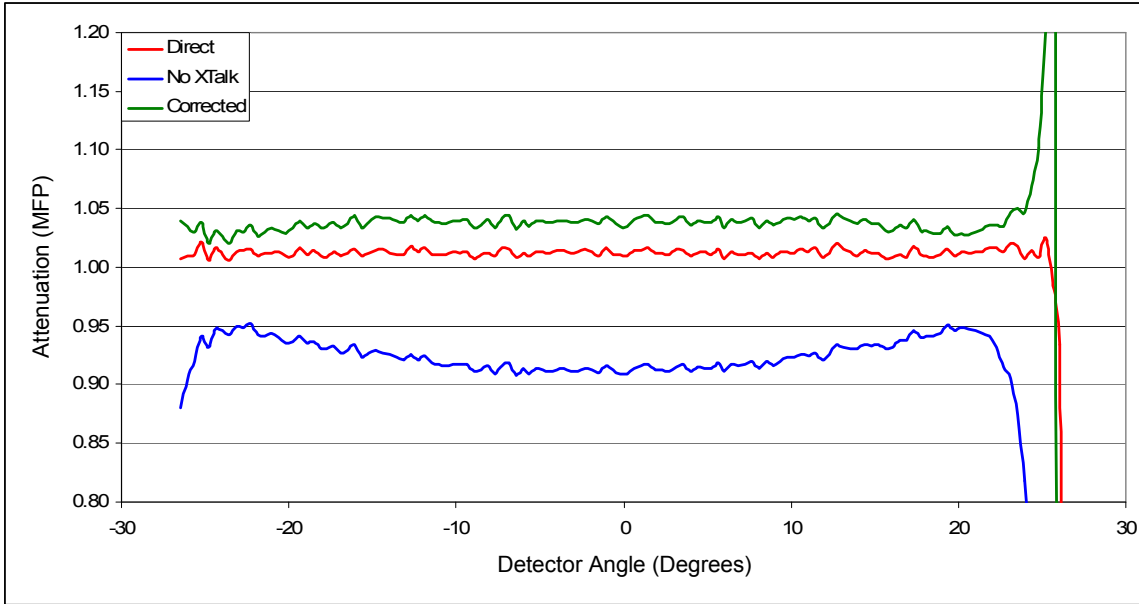
Figure I-2. The attenuation curves for the Carb61 scenario. The PSRA used material 6 (average of carbon, iron, and lead) PScF values to correct the scatter.
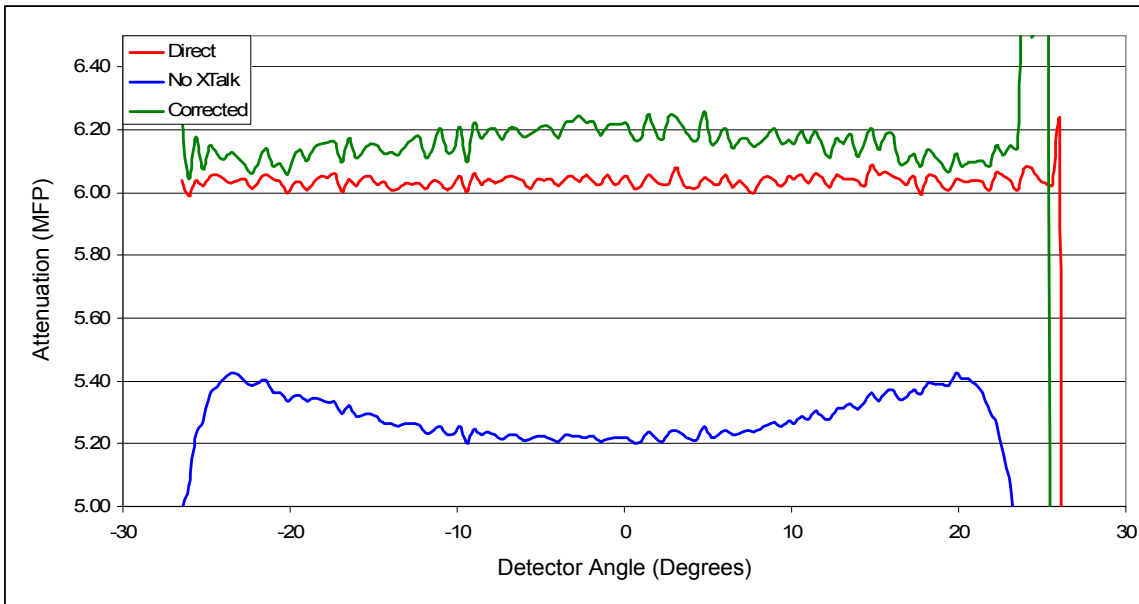
Figure I-3. The attenuation curves for the Iron76 scenario. The PSRA used material 6 (average of carbon, iron, and lead) PScF values to correct the scatter.
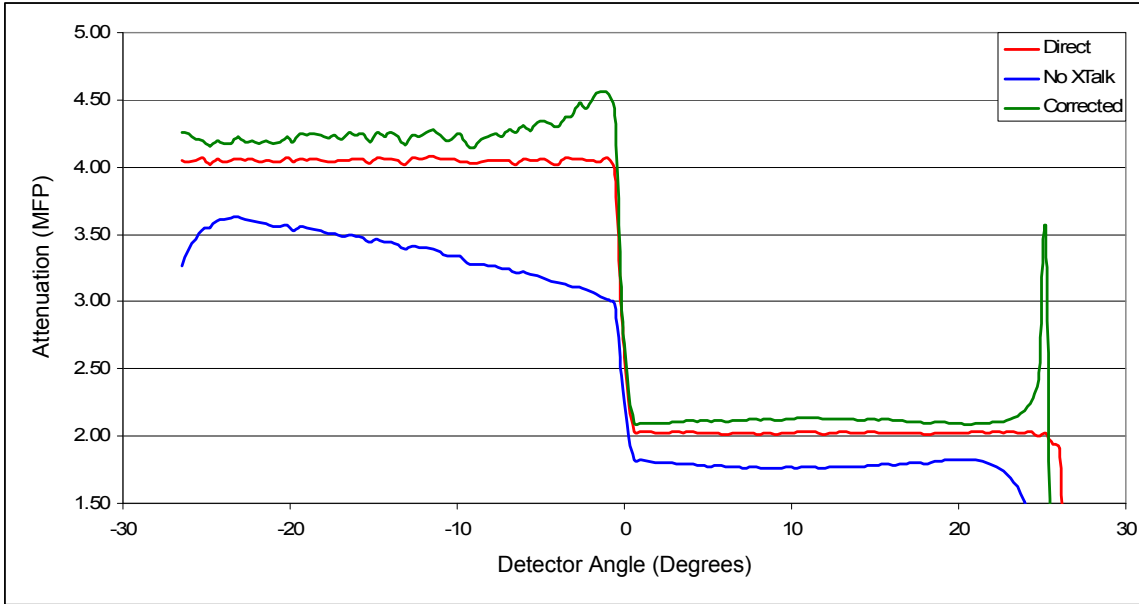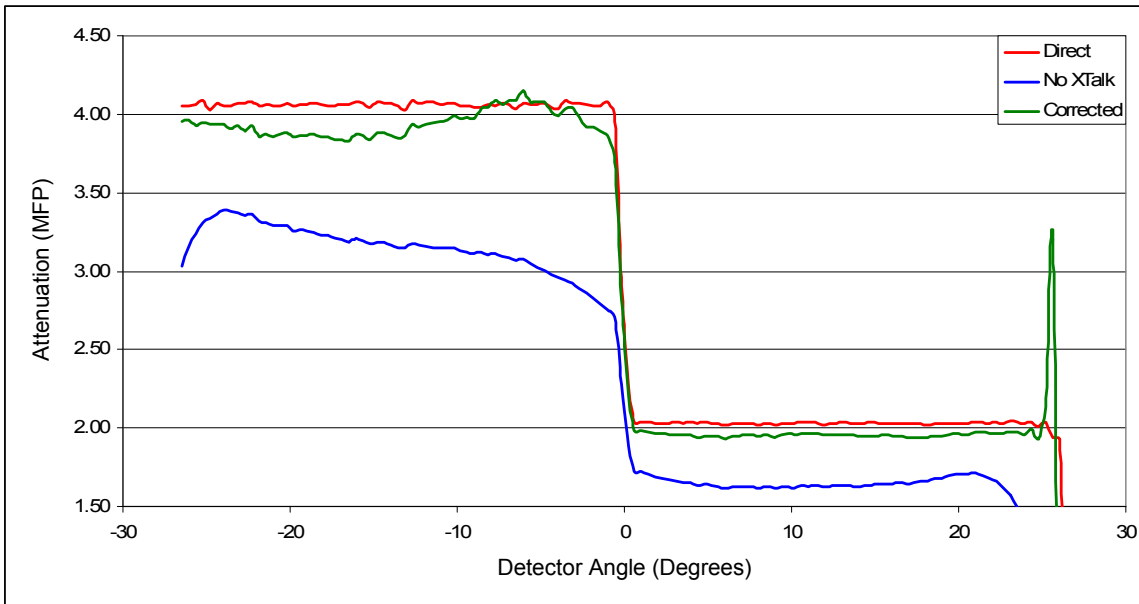
294
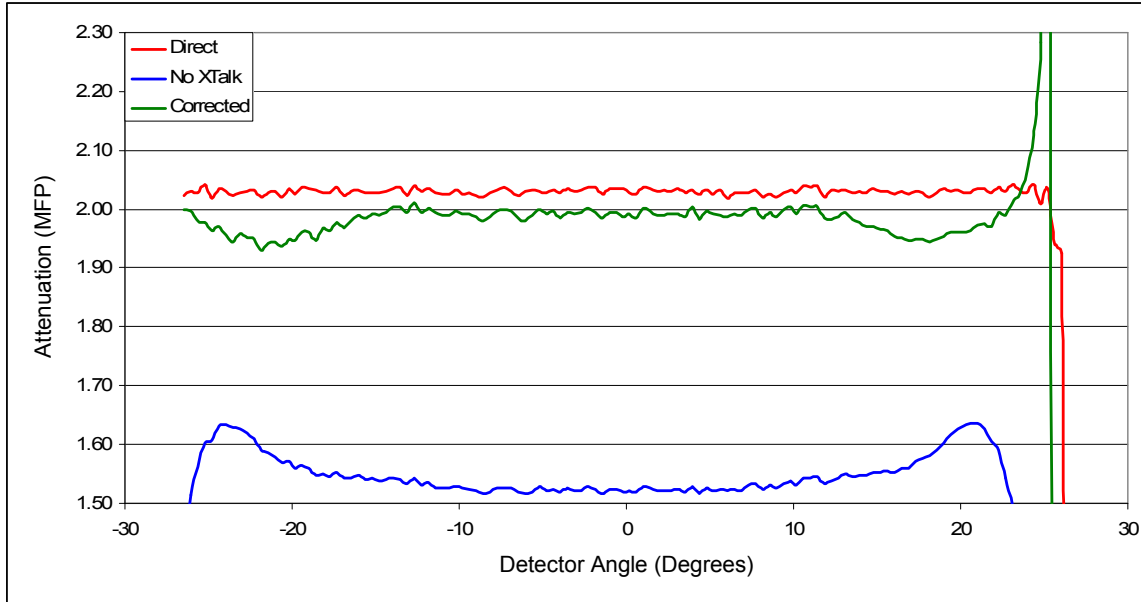
Figure I-4. The attenuation curves for the Lead32 scenario. The PSRA used material 6 (average of carbon, iron, and lead) PScF values to correct the scatter.

The attenuation curves resulting from the use of the material 6 coefficients to correct the C2C4 and L2L4 (two different thicknesses of carbon and lead, respectively) scenarios are presented in Figure I-5 and Figure I-6. As before, the scenario with carbon is overcorrected and the scenario with lead is undercorrected. Here, the scatter correction does a fairly poor job correcting the attenuation values, and the shape of the corrected curves differ significantly from the *Direct* curves, particularly in the higher attenuation areas. Overall, the corrected values are only marginally better than the uncorrected ones particularly for carbon.

The attenuation curves resulting from the use of lead and polyethylene PScF values, respectively, to correct the PoPb scenario are shown in Figure I-7 and Figure I-8. This scenario has 2 MFP of polyethylene on the inside 2 MFP of lead on the outside.

Figure I-5. The attenuation curves for the C2C4 scenario. The PSRA used material 6 (average of carbon, iron, and lead) PScF values to correct the scatter.

Figure I-6. The attenuation curves for the L2L4 scenario. The PSRA used material 6 (average of carbon, iron, and lead) PScF values to correct the scatter.
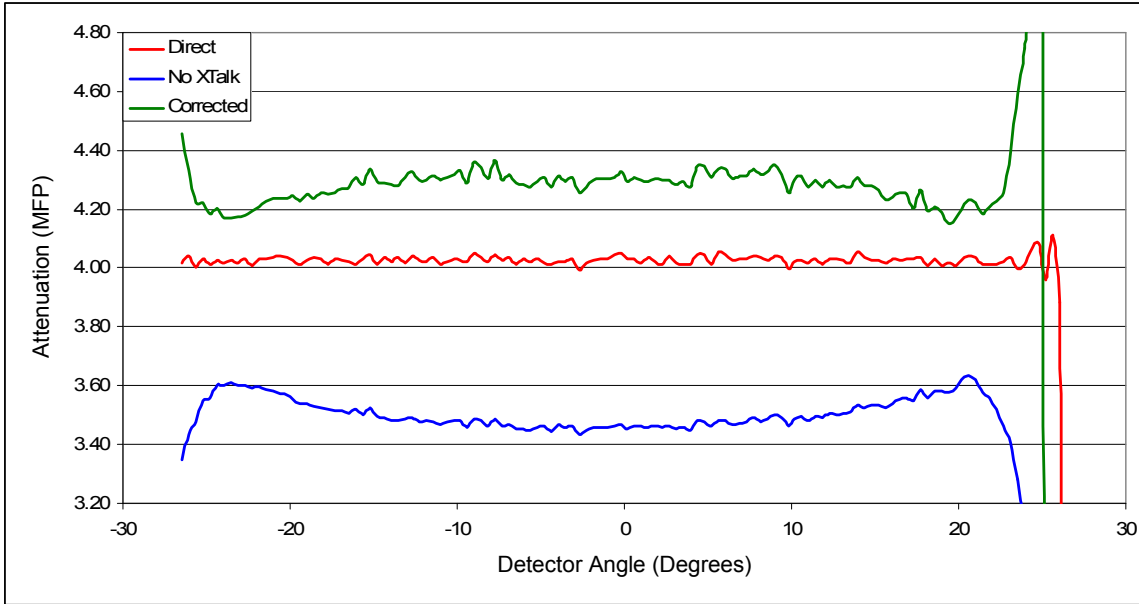
Figure I-7. The attenuation curves for the PoPb scenario. The PSRA used lead PScF values to correct the scatter.
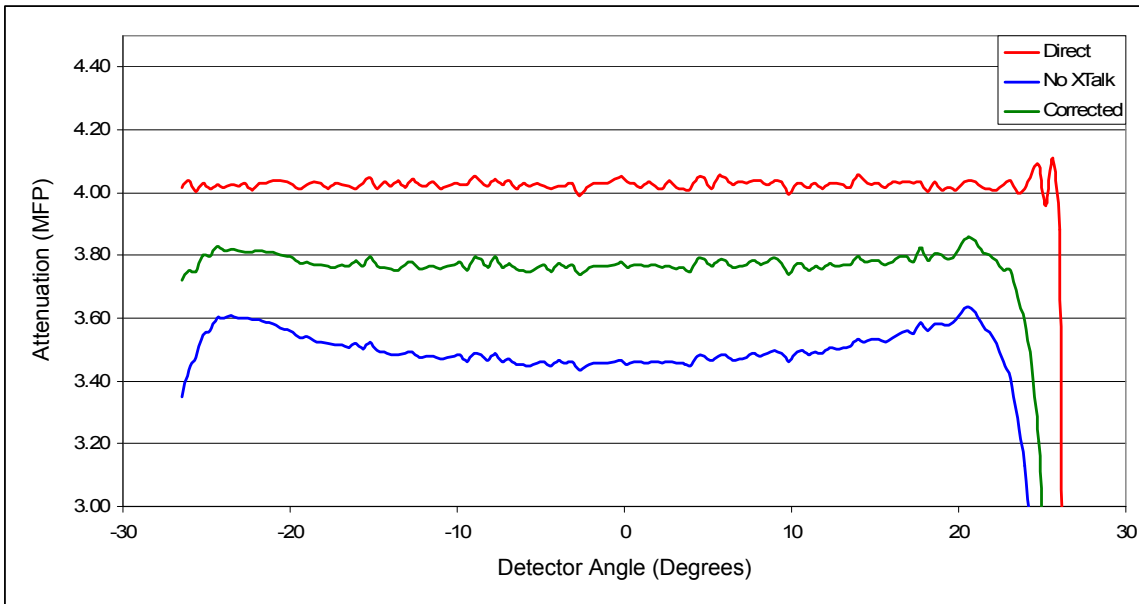

Figure I-8. The attenuation curves for the PoPb scenario. The PSRA used polyethylene PScF values to correct the scatter.

As with the previous scenario, the use of a single material in the PSRA produces poor results when removing the scatter from the attenuation values.

The next scenario simulated consisted of a 2 MFP thick layer of polyethylene sandwiched between two 1 MFP layers of lead. This scenario was given the designation LPoL. It was designed to test if the scattering in either the outer layers or the interior of the object dominates the other. The attenuation curves resulting from using lead and polyethylene coefficients, respectively, when applying the PSRA are plotted in Figure I-9 and Figure I-10. As with the previous two scenarios, the correction produces poor results using both materials. This indicates that scatter all through the object contributes significantly to the PScF and not just a single region.

Another series of scenarios was used to test the effect of varying the relative amount of the two materials on the scatter correction. The thicknesses of each material were 1, 2, or 3 MFP while the total thickness of 4 MFP of material was kept constant.   The scenario designations indicate the thickness of each material, e.g., L1P3 has 1 MFP of lead inside 3 MFP of polyethylene. The L2P2 scenario is identical to the PbPo scenario. It was renamed here for clarity.

The attenuation curves resulting from using the lead and polyethylene coefficients, respectively, are shown in Figure I-11 and Figure I-12. Note that although the *Direct* and uncorrected (*No XTalk*) values differ slightly between scenarios, only the values for the L2P2 scenario are shown for clarity. These figures show that the corrected attenuation values improve as the fraction of the material used for the scatter correction increases. However, even when the selected material comprises 75% of the object thickness, the result is rather poor.
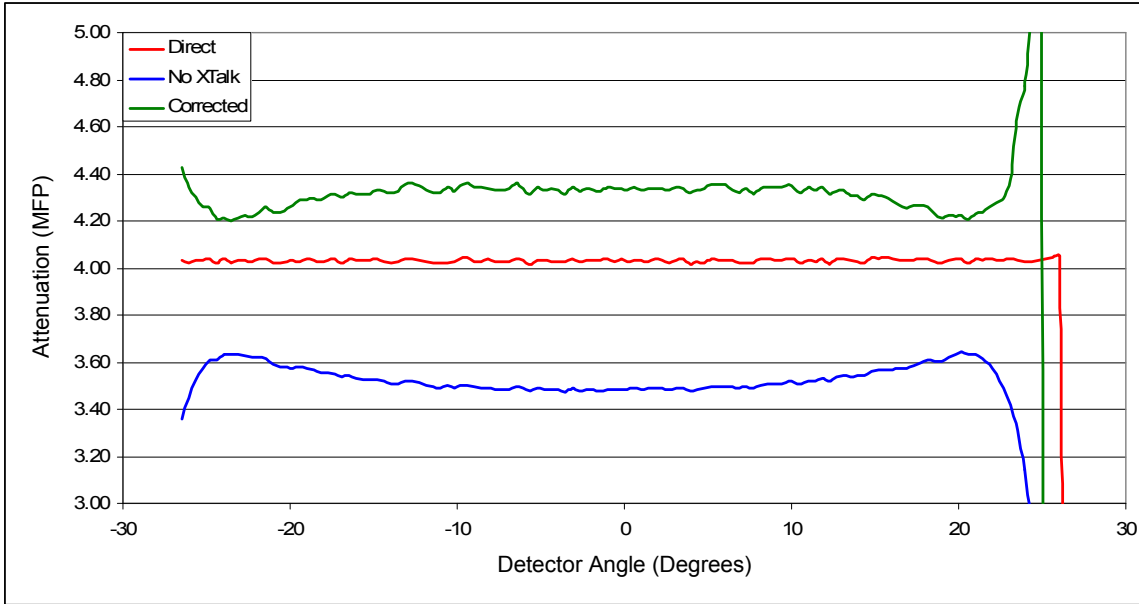
298

Figure I-9. The attenuation curves for the LPoL scenario. The PSRA used lead PScF values to correct the scatter.
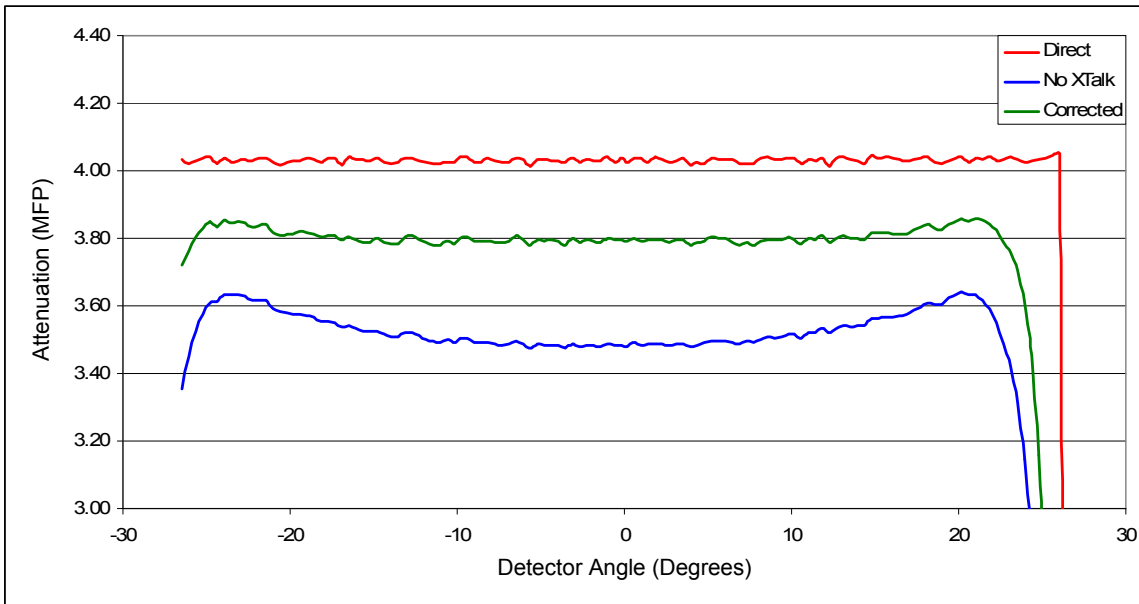

Figure I-10. The attenuation curves for the LPoL scenario. The PSRA used polyethylene PScF values to correct the scatter.
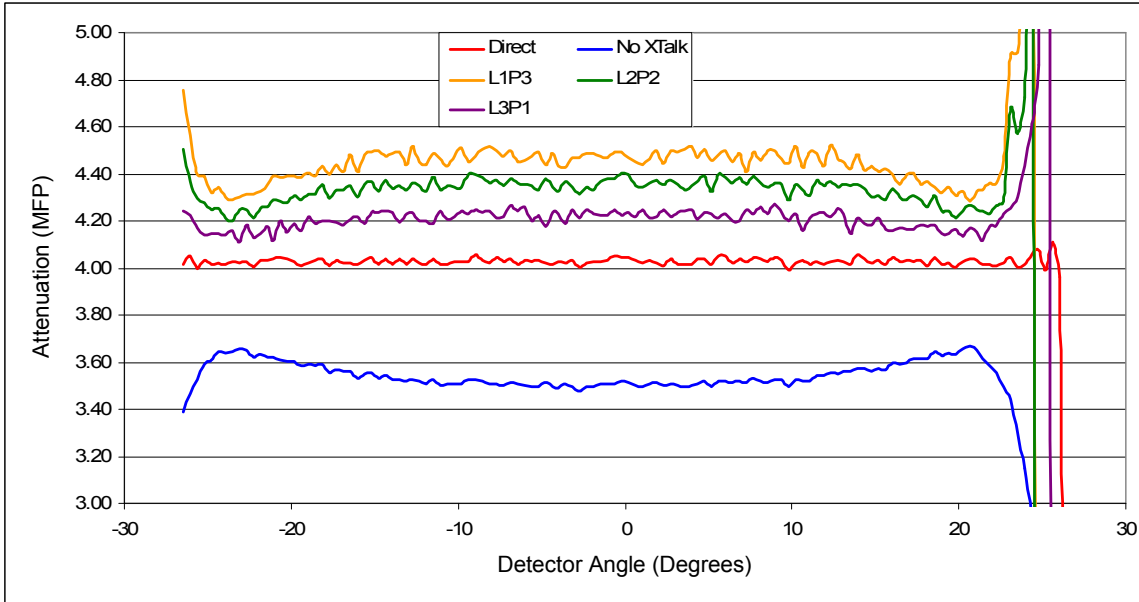
Figure I-11. Attenuation curves for the L1P3, L2P2, and L3P1 scenarios. The PSRA used lead PScF values to correct the scatter for each scenario. Note that the *Direct* and uncorrected (*No XTalk)* values differ slightly between scenarios, but only the L2P2 values are shown for clarity.
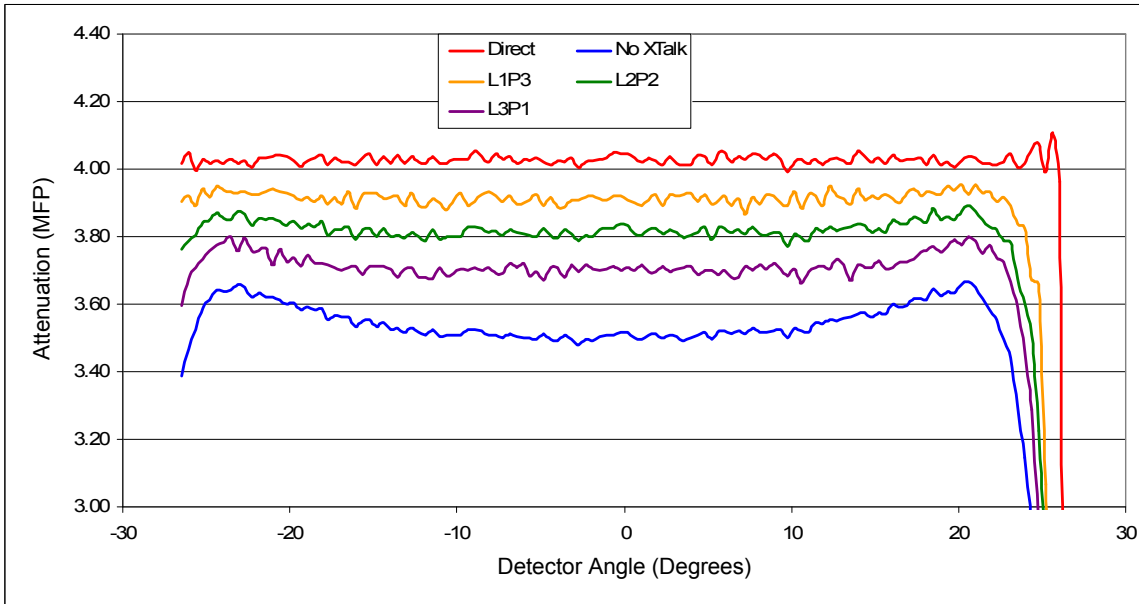

Figure I-12. Attenuation curves for the L1P3, L2P2, and L3P1 scenarios. The PSRA used polyethylene PScF values to correct the scatter for each scenario. Note that the *Direct* and uncorrected (*No XTalk)* values differ slightly between scenarios, but only the L2P2 values are shown for clarity.

These two scenarios used a simple slab geometry. Each slab had a perpendicular thickness of 3 MFP and horizontal and vertical dimensions of 1 m. The slabs were located 50 cm from the center of the detector array. At this distance, the edges of the slabs extend well outside the horizontal extent of the neutron pixels. In the first scenario, designated CaSl, the slab is composed of carbon and in the second, LeSl, the slab is composed of lead. The attenuation curves for the CaSl and LeSl scenarios are plotted in Figure I-13 and Figure I-14. Note that, unlike the cylindrically symmetric geometry, the slab geometry produces a concave attenuation curve even after the scatter is removed. There is a very slight undercorrection at the center of the CaSl corrected attenuation curve, but otherwise the modified *ScatterSubtract* program does a very good job of removing the scatter.

The next scenario uses slabs of two different thicknesses joined vertically along the plane connecting the source location and the horizontal center of the detector array. The two slabs have perpendicular thicknesses of 1 and 3 MFP of lead. The scenario is titled L13S. The resulting attenuation curves L13S scenario are shown in Figure I-15. The corrected attenuation curves follow the *Direct* attenuation curves very well and only small deviations are visible. The scatter correction also increases the contrast between the two slabs.

The next scenario simulated a homogeneous polyethylene cylinder with a radius of 18.24 cm, which is approximately 2 MFP of material. The cylinder is placed so that the minimum distance between its outer surface and the detector array is 40 cm.
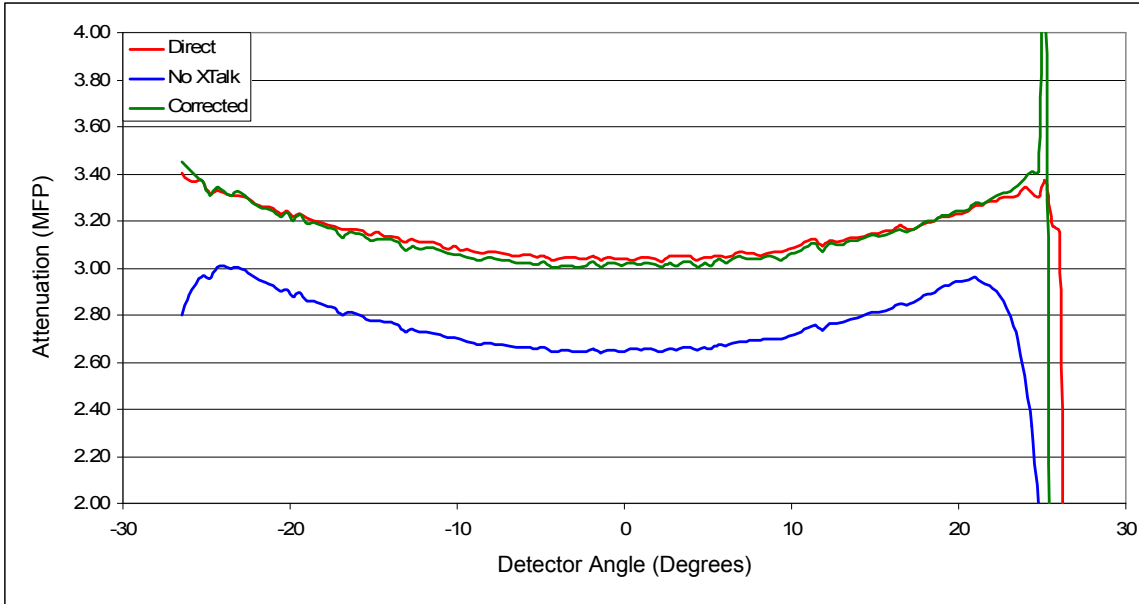
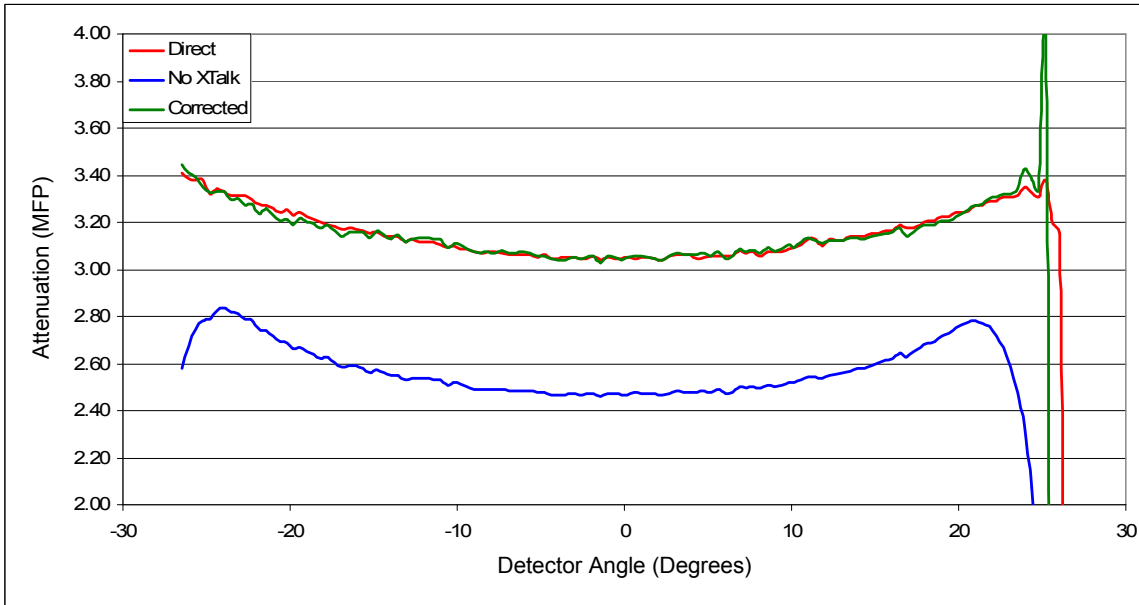**Figure I-13. The attenuation curves for the CaSl scenario.**

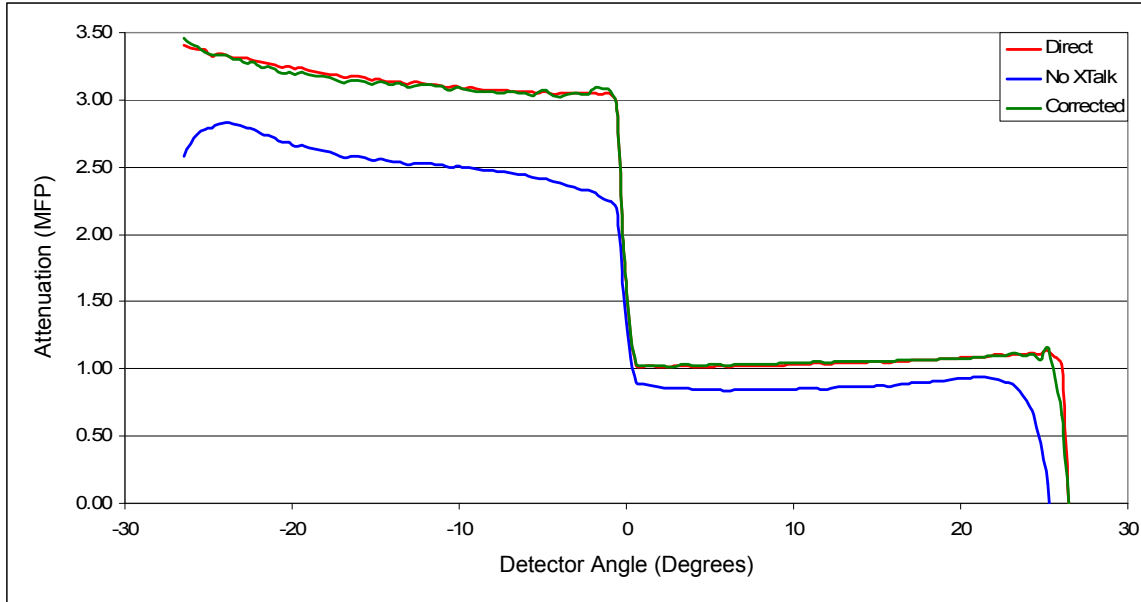**Figure I-14. The attenuation curves for the LeSl scenario.**

302

Figure I-15. The attenuation curves for the L13S scenario.

This distance was chosen so that the cylinder would cover the majority of the horizontal extent of the DT neutron cones, but not the regions beyond approximately ±24° where the correlation statistics are very poor. The attenuation curves for this scenario, which has been designated PoCy, are shown in Figure I-16. The scatter correction does an excellent job removing the scatter from the PoCy values. There is a very slight undercorrection near the center of the cylinder, but otherwise the corrected attenuation curve matches the *Direct* values very well.

The next scenario tested was a homogenous iron cylinder with a radius of 9.19 cm designated FeCy. As with the PoCy scenario, the radius of the iron cylinder is equivalent to approximately 2 MFP of material.
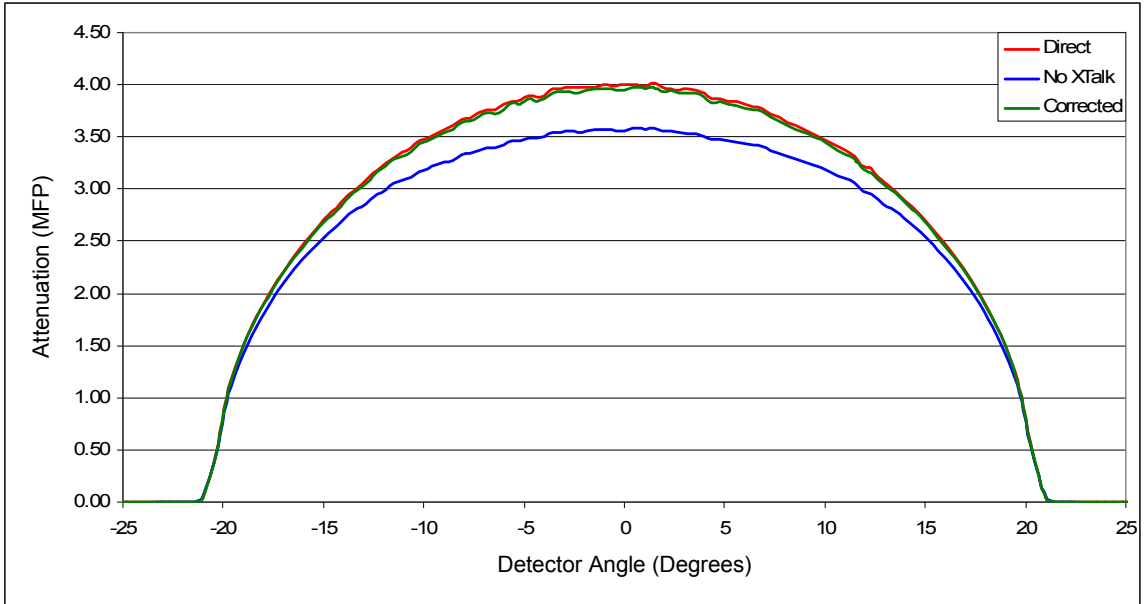
Figure I-16. The attenuation curves for the PoCy scenario.

The outside surface of the cylinder is located 70 cm from the center of the detector array in order to cover the majority of the horizontal extent of the DT neutron pixels. The attenuation curves for the FeCy scenario are plotted in Figure I-17. As with the polyethylene cylinder, the PSRA does an excellent job of removing the scatter from the measured values. There is a slight overcorrection at the center of the attenuation curve, but other than that the corrected values line up with the *Direct* attenuation curve very well.
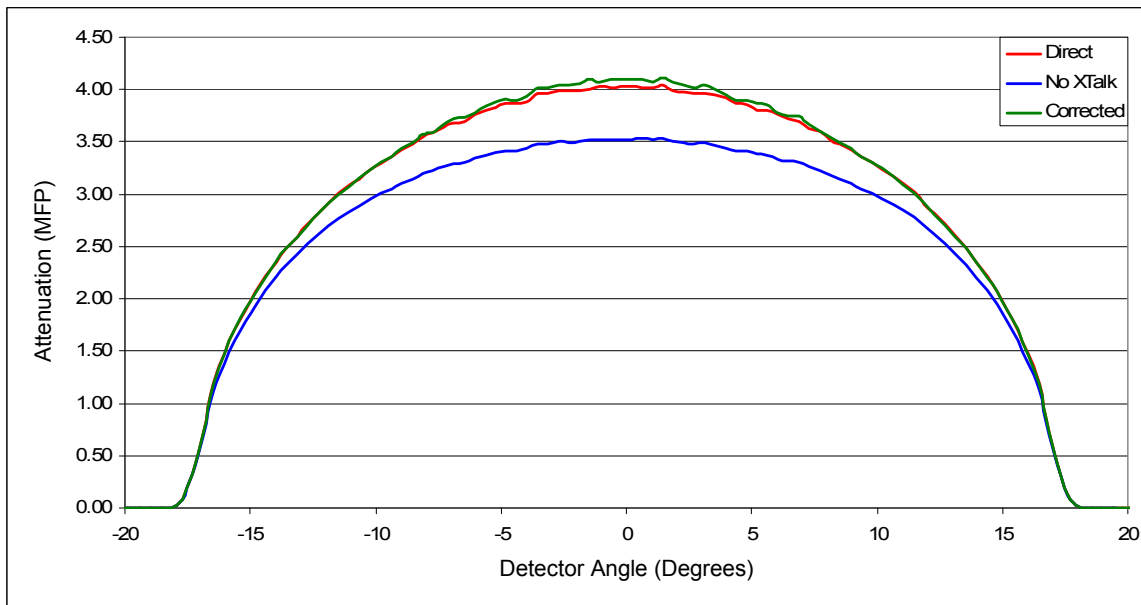

Figure I-17. The attenuation curves for the FeCy scenario.

Vita

Brandon Robert Grogan was born April 12, 1973 to Max Eugene and Bobbi Jean Grogan. In 1981, he and his family moved to Knoxville, TN. There, he graduated from Farragut High School with honors in May 1991.

In 1991, Brandon entered the United States Military Academy (West Point). While there, he earned the Army Air Assault badge at Camp Smith, New York. In the summer of 1994, he interned at the Los Alamos National Laboratory in the Nuclear Accident Response Group. He graduated with a Bachelor of Science in Engineering Physics in 1995.

After graduation from West Point, Brandon entered active duty with the United States Army as a Second Lieutenant in the Signal Corp. He first attended the Army Parachutist School at Fort Benning, GA. He then attended the Signal Officer Basic Course at Fort Gordon, GA and graduated with honors. He served in the 16th Signal Battalion at Fort Hood, TX from 1996 until 1998. His assignments included tactical satellite platoon leader, company executive officer, and assistant battalion operations officer. In 1997, he deployed to Bosnia-Herzegovina for seven months as a communications-electronics officer with the 1st Medical Group. Shortly after returning to Fort Hood, he became the communications-electronics officer for the 720th Military Police Battalion and later the 78th Military Police Brigade.

Brandon was honorably discharged from active duty in June 2000 with the rank of Captain. His awards and decorations include the Army Parachutist Badge, the Army Air Assault Badge, 2 Army Commendation Medals, 1 Army Achievement Medal, the

National Defense Service Medal, the Armed Forces Expeditionary Medal, the Armed Forces Service Medal, the Army Service Ribbon, and the NATO Medal.

From 2000-2003, Brandon worked for the Computer Sciences Corporation on the Joint Battle Damage Assessment program in Suffolk, VA as a Department of Defense contractor. While there, he served as a network administrator for classified and unclassified computer networks. In 2003, Brandon returned to Knoxville, TN where he initially worked with his father managing and refurbishing rental properties.

In 2005, he entered the University of Tennessee as a graduate student in Nuclear Engineering. In late 2006, Brandon began working for the Nuclear Materials Detection and Characterization group at the Oak Ridge National Laboratory as a graduate research assistant under Dr. John Mihalczo. Since that time, he has worked on several projects for verifying or detecting special nuclear material using active and passive interrogation techniques. His work focuses primarily on the computer modeling of measurements. Since coming to the lab, he has authored or co-authored over a dozen papers for journals and conference proceedings.

In 2008, Brandon received a Master of Science Degree in Nuclear Engineering from the University of Tennessee. His MS engineering project, "Estimating the Mass of UO_3 Powder in a Container Using MCNP-PoliMi Simulations of NMIS Measurements," was published as Oak Ridge National Laboratory Report ORNL/TM-2008/020. In August 2008, he passed the written Nuclear Engineering department PhD qualifying examinations. He completed the last of his required coursework in the summer semester of 2009. He earned a 4.0 GPA in all graduate and undergraduate coursework at the University of Tennessee.