

**Fieldable Neutron Imaging System:
Simulation and Experimentation into the Viability of
Associated Particle Imaging in Austere Field
Environments**

A Thesis Presented for a
Master of Science
Degree
The University of Tennessee, Knoxville

Benjamin Charles Troxell
May 2020

Copyright © 2020 by Benjamin Charles Troxell
All rights reserved.

ACKNOWLEDGEMENTS

A special thanks is due to my committee chair, Professor Jason P. Hayward, who to all his students appears to be omniscient in both his knowledge and organizational capacity. From the hefty requirements of being the Associate Department Head, to navigating the many organizations and channels at ORNL, to managing over 15 fellow graduate students Professor Hayward always found a way to make himself available.

I would also like to thank my committee members, Professor Howard L. Hall and Lawrence H. Heilbronn. Their classes demonstrated the many impacts nuclear engineering has had both in shaping our current world and will continue to have on the future. I would be remiss to forget Dr. Paul Hausladen – a leader in the field of Associated Particle Imaging for over two decades. For without Paul’s daily tutelage and laboratory expertise, none of this work would have been possible. From eliciting support from fellow staff at ORNL to devote time on this project, to his daily dad jokes, Paul is a professional scientist every graduate student aspires to become.

Additionally, gratitude is owed to the University of Tennessee and Oak Ridge National Laboratory for laboratory, equipment, and personnel support as well as financial support. Finally, I would like to express my gratitude to the United States Army and my loving wife Lauren for allowing me to take a two-year sabbatical to pursue this degree, moving our small family a seventh time in less than a decade.

ABSTRACT

This thesis is the culmination of an effort to develop a portable fast neutron radiography device capable of generating images of the internal structure of high Z objects non-intrusively in austere field environments. This work represents the first advancement to this end for the Fieldable Nuclear Materials Identification System (FNMIS). The effort was two pronged in its approach. First, a physical system was constructed and tested in an effort to reduce size and overall system weight. Simultaneously, a new simulation workspace was developed to test objects unavailable for physical imaging. The physics of the simulation were modeled utilizing the latest version of MCNP 6.2. Post processing and image reconstruction were completed utilizing python 3.7 employing the mcnpools module to study the content of MCNP's PTRAC file. As part of this simulation process, a new beam characterization profile for the Russian ING-27 DT generator has been developed. The result is a double gaussian fit comparable to the characterization of the Thermo Scientific API-120 system. A series of experiments on various objects and source-detector misalignments were conducted. The result of this work is an agreement between the results of simulations and those of the laboratory measurements. Significant advances in setup time, alignment techniques, and required time of exposure were all made as part of the construction process. This work paves the way for the application of neural networks to improving misalignment assessment and image quality.

TABLE OF CONTENTS

Chapter One Introduction and General Information.....	1
IED Identification Utilizing Radiographic X-Ray, Dual Energy, and Compton	
Scatter Imaging	1
Neutron Interactions with Matter	3
Advantages of Neutron Over X-ray Tomography	5
Challenges Facing Fast Neutron Tomography	6
Chapter Two Literature Review.....	8
Introduction.....	8
Categorization of Neutron Generators	9
Nuclear Reactor and Spallation Neutron Sources.....	10
Laboratory and Static Assemblies	12
Mobile / Portable Systems	21
Image Reconstruction and Post Processing	22
Scintillators and detectors	25
Gaseous Neutron Detectors.....	26
Scintillators with Intensified CCD Cameras.....	28
High Z Sensitized Scintillators / Doped Scintillators	29
Summary	31
Chapter Three Materials and Methods.....	33
Physical Materials and Methods	33
Detector Arm	33
Modular Units	35
Scintillators	35
Readout Equipment.....	36
Software and Graphical User Interface (GUI)	37
Calibration.....	39
Void Measurement.....	41
Trials – Order of Measurements	42
Simulation Method.....	44
World and Scintillator Generation	47
Source Methodology	49
Light Yield Methodology	55
PTRAC Analysis Methodology	57
Radiograph Reconstruction.....	60
Chapter Four Results and Discussion	62
Initial Steel Canister.....	63
Equidistant Measurements – Steel, Aluminum, Polyethylene, & Depleted Uranium	
.....	65
Translational Offset Void Measurements ($\pm X$ Axis).....	75
Rotational Offset Void Measurements.....	78
Translational Offset Void Measurements ($\pm Y$ Axis).....	82
Advanced Configurations	85

Chapter Five Conclusions and Recommendations	95
Recommendations	100
Future Work	103
List of References	105
Appendices.....	110
Appendix: A – Laboratory Set Up	111
Appendix: B – Obstructed View Set Up.....	114
Appendix: C – MCNP6.2 Detector Generation Code.....	116
Appendix: D – MCNP6.2 Transformation Card Generation Code.....	121
Appendix: E – MCNP6.2 Neutron Source Generation Code	124
Appendix: F – MCNP6.2 Input Deck Generation Code	126
Appendix: G – Ptrac Parser Code	131
Appendix: H – Detector Counts Summation Code.....	138
Appendix: I – Correlation Parser Code.....	141
Appendix: J – Example MCNP Input Deck.....	184
Vita.....	227

LIST OF TABLES

Table 2.1. Performance Parameters of Neutron Detection Systems for Imaging	26
Table 3.1. Reactions of interest for a plastic scintillator at neutron energies below 15 MeV.....	58
Table 4.1. Rotation Offset Void Measurements with cm and Corresponding Degree Offsets.....	77

LIST OF FIGURES

Figure 1.1. Schematic illustration of Compton back-scatter system for landmine Detection.....	3
Figure 1.2. Mass attenuation and atomic number relationship.....	6
Figure 2.1. Background neutron count as a function data sets collected.....	10
Figure 2.2. Neutron flux intensity as a measure of energy group. Error bars are statistical uncertainties; the colored bands are the ranges obtained in the test of systematics.....	11
Figure 2.3. ManDi instrument on beamline 11B at the Spallation Neutron Source at Oak Ridge National Laboratory.....	12
Figure 2.4. Calculated ratio, R, of the 14MeV neutron to cobalt-60 γ -ray mass attenuation coefficients for a range of elements from hydrogen to plutonium.....	14
Figure 2.5. Schematic illustration of the nuclear reaction ${}^3_1\text{H}(d, n){}^4_2\text{He}$, Q value 17.59.....	16
Figure 2.6. Time of flight between neutron detectors for (solid) all scattering events and (dashed) those events where an alpha has been detected.....	17
Figure 2.7. Elastic scatter diagram with variable identification and labeling.....	20
Figure 2.8. Images of a depleted uranium casting.....	21
Figure 2.9. (a) Attenuation lengths (μx) of object 1 plotted against detector slot (correlates to lateral position within the object) for a single rotation. (b) FBP radial tomography of object 1 that plots the attenuation coefficient (cm^{-1}) as a function of distance (cm).....	23
Figure 2.10. Centerline slice of the MLEM fission reconstruction doubles tomograph with the two peaks corresponding to the cylindrical region of HEU shielded on the outside by a region of DU.....	24
Figure 2.11: Imaging of carbon rods and a steel wrench with fast neutrons.....	28
Figure 2.12: Summary of the major photophysical processes in the scintillation of organic materials.....	30
Figure 2.13: Pulse height spectra of 0–60% QD/2% FBtF/PVT nanocomposite scintillators.....	31
Figure 3.1. 3D printed graphite infused carbon nylon.....	34
Figure 3.2. 3D printed detector casings.....	35
Figure 3.3. Portable fieldable assembly.....	36
Figure 3.4. Signal processing equipment crate.....	37
Figure 3.5. ING-27 graphical user interface (GUI).....	38
Figure 3.6. Cs ¹³⁷ calibration set up.....	40
Figure 3.7. Detector drift observations.....	41
Figure 3.8. Experimental measurements chart.....	43
Figure 3.9. MCNP Input deck generation flowchart.....	46
Figure 3.10. Ptrac post processing flowchart.....	47
Figure 3.11. Detector geometry verification.....	48
Figure 3.12. MCNP sub-level universes.....	49
Figure 3.13. Single gaussian fit to the correlated count dist. for ING-27 pixel 7.....	50
Figure 3.14. Alpha pixel profiles for pixels 1-16.....	51

Figure 3.15. 100 ps arrival time bins for pixel profiles 1-10.....	52
Figure 3.16. Double gaussian fit to the correlated count dist. for ING-27 pixel 7.....	53
Figure 3.17. Double gaussian fit to the correlated count dist. for ING-27 pixel 1.....	54
Figure 3.18. Double gaussian translation of normalized probabilities as a function of α pixel.....	54
Figure 3.19. (Left) $^1\text{H}(n,\gamma)$ ^1H reaction cross sections and (right) $^1\text{H}(n,\text{elastic})$ ^1H cross sections.....	55
Figure 3.20. $^{12}\text{C}(n,n')$ induced Reactions by 14.1 MeV Neutrons.....	58
Figure 3.21. Ptrac Parser Total Count Tracker.....	60
Figure 3.22. Simulated Steel Attenuation Profile.....	61
Figure 4.1. Material canisters made of depleted uranium (DU), tungsten, led, steel, aluminum, and polyethylene, pictured from left to right.....	63
Figure 4.2. Initial steel attenuation profile.....	64
Figure 4.3. MCNP top down view of detectors and interrogation object	65
Figure 4.4. ING-27 DT generator offset of central alpha pixel.....	65
Figure 4.5. DT-detector equidistant, centered steel attenuation profile.....	67
Figure 4.6. DT-detector equidistant, offset right (5cm) and offset left (10cm) steel attenuation profile.....	68
Figure 4.7. DT-detector equidistant, centered Al attenuation profile.....	69
Figure 4.8. DT-detector equidistant, offset right (5cm) and offset left (10cm) Al attenuation profile.....	70
Figure 4.9. DT-detector equidistant, centered polyethylene attenuation profile.....	71
Figure 4.10. DT-detector equidistant, offset right (5cm) and offset left (10cm) polyethylene attenuation profile.....	72
Figure 4.11. DT-detector equidistant, centered DU attenuation profile.....	73
Figure 4.12. DT-detector equidistant, offset right (5cm) and offset left (10cm) DU attenuation profile.....	74
Figure 4.13. Translation X axis movement diagram. Detector arm in blue. Target in teal. Source in red cross.....	75
Figure 4.14. X detector translational offsets: 5 cm, 10 cm, and 15 cm.....	76
Figure 4.15. -X detector translational offsets: -5 cm, -10 cm, and -15 cm.....	77
Figure 4.16. XY plane view of detector arm through - $\hat{x}$ offsets: -5cm, -10cm, and - 15cm.....	79
Figure 4.17. XY plane translational offsets: +5 cm, +10 cm, and +15 cm.....	80
Figure 4.18. -XY plane translational offsets: -5 cm, -10 cm, and -15 cm.....	81
Figure 4.19. DT-detector (top) 110.1 cm distance (bottom) 105.1 cm distance.....	83
Figure 4.20. DT-detector (top) 120.1 cm distance (bottom) 125.1 cm distance.....	84
Figure 4.21. Led, DU nested cylinder assembly.....	85
Figure 4.22. Led, DU nested cylinder assembly attenuation profile.....	86
Figure 4.23. (top) API results converted to linear plots. (left) The apparent thickness (in mean free paths) is shown along with (right) two lineouts whose positions are shown by the dashed lines.....	88
Figure 4.24. (Left) photograph of DU casting and (right) raw fast-neutron radiograph of the DU casting.....	89

Figure 4.25. The experimental setup with the ING-27 DT neutron generator visible in the foreground and the near-side lead shield visible in the background. In this photograph, a depleted uranium (DU) casting is positioned in front of the detector.....	90
Figure 4.26 DU and lead canister casting with +10cm $\hat{x}$ offset.....	91
Figure 4.27 DU and lead canister casting with +5cm $\hat{y}$ offset.....	92
Figure 4.28 DU and lead canister casting with a clockwise rotation in the XY plane of 6.789°.....	93

CHAPTER ONE

INTRODUCTION AND GENERAL INFORMATION

IED Identification Utilizing Radiographic X-Ray, Dual Energy, and Compton Scatter Imaging

IEDs are typically comprised of five components: activator, fuse, body, explosive and battery. In most cases these components will be hidden – either buried or disguised in a larger container to prevent identification. Once one or more of these components has been identified an attempt can be made to neutralize the object. This is typically done in one of two ways: isolation and detonation of the device or disarmament in the case that evacuation is not feasible [1]. Under normal circumstances the challenge most often lies in identification of the device from its surroundings. Once the IED has been identified, its components and geometry can be imaged using one of three methods: radiographic x-ray imaging, dual energy imaging, and Compton scatter imaging. Each method carries its own unique strengths and weaknesses.

The simplest method is radiographic x-ray imaging. Typically, a package is passed over a large area imager or across a continuous x-ray source coupled with a line sensor. This method is typically found in airport screening or in the inventories of local police and SWAT bomb squads. The main advantage of these systems is their high throughput and relative ease of use. However, their simplicity also results in a series of drawbacks. The main drawback of single radiographic line scanning is that black powder creates the same x-ray shadow as honey [2]. In addition, overlapping materials increase

the absorption coefficient and further degrade image quality. One method developed to overcome this limit is x-ray CT imaging.

By rotating the x-ray source and detector a 360 multi-slice image can be constructed. This method is very good at discriminating between different materials however it still has issues with materials with similar absorption coefficients. Ironically fruit cake and plastic explosives both appear identical to x-rays. However, scanning and image reconstruction are time consuming. The result is CT imaging is often coupled in a two-tier approach, with continuous x-ray imaging [2].

For inspections of larger containers such as shipping containers higher energy x-ray sources are required. In order to penetrate steel containers at multi-millimeter distances MeV level energies are required. This requirement naturally results in larger static emplacements which cannot be easily moved. A benefit of such high energy sources is they allow imaging at two or more different x-ray energies as such x-ray sources are not typically monoenergetic. Dual energy imaging allows for more precise differentiation between materials based upon the absorption curves of different materials [2].

Finally, there is the use of Compton backscatter in the identification of materials utilizing x-rays. In this case a primary collimator is moved over the area of interest and a preset detector system is fixed to the secondary collimator. Figure 1.1 provides an illustration of this effect.

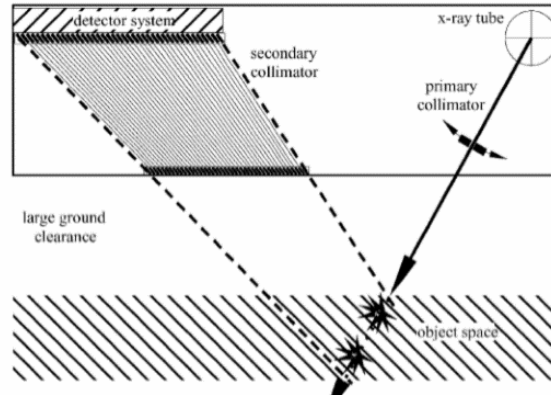


Figure 1.1. Schematic illustration of Compton back-scatter system for landmine detection [2]

A significant draw back to this system is it is depth specific. The preset position of the secondary collimator and detector system determine the interrogation depth. If the depth of interrogation is to be adjusted the secondary must be repositioned and the x-rays must be of sufficient energy to penetrate the additional distance. If the depth of the target is well established this imaging system is one of the most effective.

Nevertheless, all these systems fail to overcome the fundamental limit of the photoelectric cross section whose size scales proportionally to Z^3 . Accordingly, high Z materials tend to have extremely large attenuation coefficients [3]. IEDs continue to become more advanced and have taken advantage of the fact that high Z materials mask their content. In order to penetrate these materials a particle not susceptible to the electromagnetic force is required – the neutron.

Neutron Interactions with Matter

As previously stated, neutrons do not interact with charged particles in nature. Instead they interact with the nucleus of an atom, dominated by the strong and weak nuclear force, through one of the six following pathways. These pathways are elastic

scattering, inelastic scattering, radiative capture, charged-particle reactions, neutron-producing reactions, and fission. In all these cases the incident neutron strikes the nucleus and a follow-on reaction occurs. Many are a two-step process in which the neutron is absorbed to form a new nucleus whose atomic mass is increased by one [4].

In elastic scattering a neutron strikes the nucleus and a neutron reappears. In this case the nucleus is not excited and kinetic energy is conserved between the two particles. Elastic scattering is abbreviated as an (n,n) reaction. Inelastic scattering differs from elastic scattering in the kinetic energy of the two particles is not conserved. Instead, some of the kinetic energy is translated into excitation energy of the nucleus. This excitation energy decays resulting in the release of γ -rays or other more exotic forms of deexcitation. Inelastic scattering is referred to as (n,n') reactions [4].

Next there is radiative capture (n, γ). In this reaction the incident neutron is absorbed by the nucleus. One or more γ -rays is released as a result of the new excited state of the nucleus as it transitions to its ground state. Since the original neutron is absorbed this reaction is often referred to as an absorption reaction. Closely related to absorption is fission. In fission reactions the original neutron is absorbed by a nucleus. The resulting nucleus is unstable and as a result splits resulting in two or more fission fragments. Often additional neutrons will be released in this process – an essential component of nuclear fission reactors [4].

Finally, there are charged particle reactions and neutron-producing reactions. In both reactions the neutron is initially assimilated into the nucleus. In charged particle reactions an alpha particle α or proton may be ejected. These charged reactions can be exothermic or endothermic and are typically referred to as (n, α) or (n,p) depending on the

particle ejected. In neutron producing reactions additional neutrons are ejected from the nucleus. These reactions are always endothermic and are often found in moderators of reactors such as Deuterium and Beryllium [11] and are an essential component to nuclear power generation and the neutron lifecycle inside a reactor. It is important to note that these reactions, while uncommon, may include several variations such as $(n,2n)$ where multiple particles are ejected.

In the specific case of polyvinyl toluene (PVT) and a neutron the most dominate of these reactions is the elastic scattering of neutrons off hydrogen nuclei. This is result of the kinematics in which a neutron can transfers all its kinetic energy to the proton of a hydrogen atom in one collision. As result, a relatively difficult to identify particle (the neutron) imparts all its energy to an easily identifiable particle (the proton). The process is aided by the large cross section of hydrogen to 14.1 MeV neutrons of 20.48 barns [5]. This ability to detect a neutral particle coupled with the wide-ranging cross sections of materials for neutrons permits neutron tomography. To a lesser extent inelastic scattering and radiative capture both play a role in the indirect creation of electrons which are also detected in PVT.

Advantages of Neutron Over X-ray Tomography

Since neutrons only react with the nucleus of the atom and the cross sections of these reactions is dependent on both the neutrons energy and the specific isotope of the nucleus, unique imaging possibilities become possible. One of these capabilities is the ability to penetrate high Z materials and image delicate high hydrogen, carbon rich

materials within them. When properly calibrated, imaging a rose inside a lead flask becomes possible as in Figure 1.2.

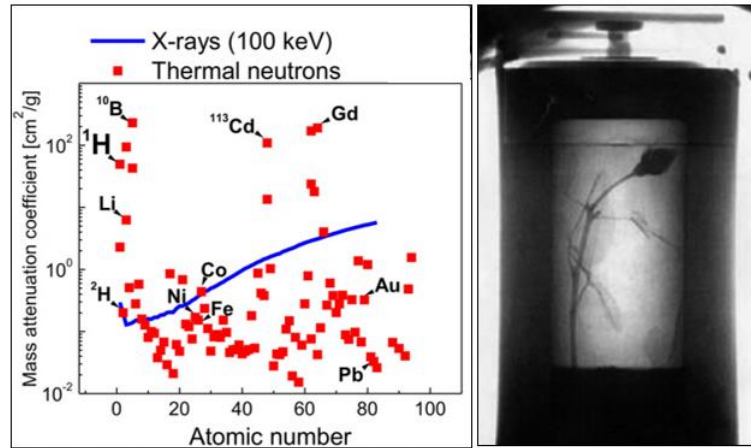


Figure 1.2. Mass attenuation and atomic number relationship [6]

These images are possible due to the fundamental cross section properties of materials of differing atomic number. Whereas this coefficient is relatively stable with x-rays, the values range wildly for neutrons of varying energies.

Challenges Facing Fast Neutron Tomography

Under ideal circumstances, images such as Figure 1.2 can be obtained. Currently, these ideal circumstances restrict capabilities to the laboratory – a similar image collected by the author and presented in chapter four highlights this. Factors as simple as the humidity in the atmosphere can impact imagery, as high hydrogen content of a moist atmosphere plays a contributing role in attenuation and scatter. The challenge facing neutron tomography is incorporating resiliency into systems currently laboratory bound in order to mitigate field conditions. To this end, advanced photodetectors such as Silicon

Photomultiplier (SiPM) detectors and artificial neural networks are being explored as means to improve system weight and reduce dependency on ideal conditions.

CHAPTER TWO

LITERATURE REVIEW

Introduction

Only three years after the neutron's discovery in 1932 by English physicist James Chadwick [7], the US Patent office issued the first neutron imaging patent to Hartmut Kallmann and Ernst Kuhn [8]. Over the next twenty years, the foundations of neutron imaging took root. All systems would come to have a neutron source, a detector, a target and an imager. Neutron sources have advanced much from the first high energy neutron beam used by Thewlis in his 1950s experiments at the BEPO reactor at Harwell [9]. Neutron sources have undergone massive miniaturization from 200kW reactor sized devices [10] capable of only providing 1×10^5 n/s to football sized generators such as the Thermo Scientific minigene capable of providing 3×10^8 n/s [11]. Detectors have progressed from simple x-ray film to large area amorphous silicon imagers. Advances in CCD imaging and low-light cameras have achieved spatial resolutions over large image sizes previously thought unachievable just a decade ago. Most often overlooked are the advances in computer processing speed, RAM, and data storage – all of which are essential with the massive amounts of data generated in today's experiments.

Neutron flux measurements are generally organized by their kinetic energies into the following spectrum macro-groups: (1) thermal neutrons with energies less than 0.5 eV, (2) epithermal neutrons in the range of 0.5 eV to 100 eV, (3) intermediate and fast neutrons in the range of 0.1 keV to 10 MeV, and (4) very fast neutrons with energies greater than 10 MeV.

Each of these macro groups has unique characteristics that can be used for imaging and identification purposes. Typically, fast neutrons are synonymous with 14.1 MeV neutrons in the field of API a term which may confuse those outside of the field.

While there have been many advances in neutron imaging over the last seventy years, there remain many challenges and obstacles to be overcome in the field. System size/portability, resolution, and imaging time all present the end user with unavoidable tradeoffs that must be weighted in each application. As a result, systems are designed to specialize in one application.

Categorization of Neutron Generators

Neutrons are not an entirely artificial particle despite their naturally short lifetimes as they decay into protons over a period of about 887 seconds [12]. The background field experienced on Earth's surface is primarily the result of neutrons created by the interaction of cosmic rays with elements within the earth's atmosphere. High energy cosmic rays – up to 10^{14} eV interact with the atmosphere on a continual basis. As the cosmic rays interact with the geomagnetic field of the earth they produce a cascade of exotic, high energy particles to include $e^+ e^-$ pairs, muons, and neutrons. These particles rapidly transfer their energy as they penetrate the atmosphere and the resulting cascade of particles eventually arrive at sea level where we can observe neutrons [13]. Most of the neutrons generated in this manner originate from nitrogen and oxygen spallation by high energy protons. Figure 2.1 provides an example of this background field.

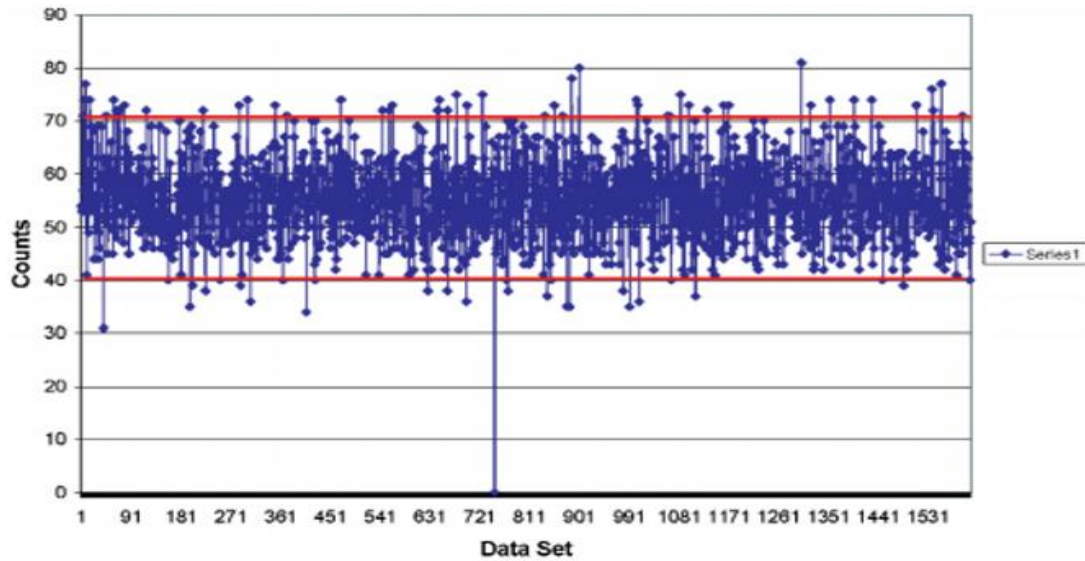


Figure 2.1. Background neutron count as a function data sets collected.

Passive neutron counting at sea level often reveals large counts for such a short-lived particle. It is not uncommon to observe one to five neutrons a second in detection studies involving the use of polyethylene block detectors and He^3 gas detectors [14]. With such high background counts the first neutron generators were hardly capable of overcoming this background field. Since the first generation of generators there have emerged three primary categories of neutron generators – reactor/spallation source generators, laboratory/static sources, and mobile/portable sources.

Nuclear Reactor and Spallation Neutron Sources

Reactors and spallation neutron sources provide the highest neutron generation source strength counts. Spallation sources generate neutrons using a particle accelerator. The accelerator utilizes magnetic field manipulation to contain and accelerate proton up to an average energy of 1.6 GeV. Ring designs are often utilized as they provide the most efficient means of containment and acceleration capabilities. Sections of the ring will

provide an electromagnetic boost in a pulsed manner to attain these high energies. Once accelerated, the proton beam is split off into a target material. Often a high Z material is selected due to its naturally high neutron content – such as Tungsten in the case of the Chinese Spallation Neutron Source [15]. From the Tungsten target body neutrons spall off in all directions. The result, as can be seen in Figure 2.2, is a wide neutron spectrum that extends over all orders of magnitude up to that of the accelerator’s energy.

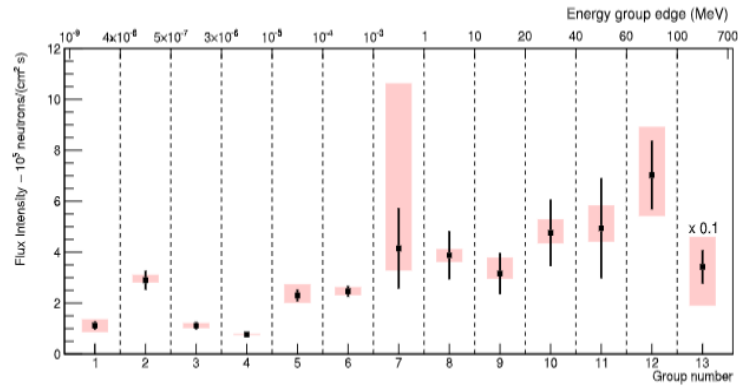


Figure 2.2. Neutron flux intensity as a measure of energy group. Error bars are statistical uncertainties; the colored bands are the ranges obtained in the test of systematics [16].

The collimated neutron beam is subsequently split into “beamlines.” Each of these beamlines allows for an independent experiment to be conducted simultaneously. Specific instrumentation utilized is covered in the neutron detection section of this review.

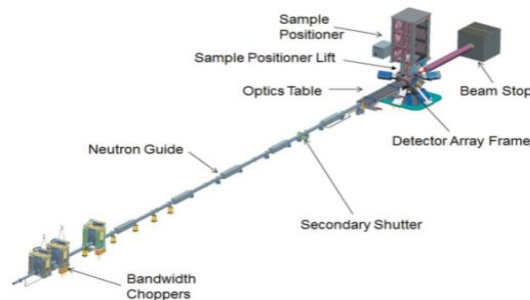


Figure 2.3. ManDi instrument on beamline 11B at the Spallation Neutron Source at Oak Ridge National Laboratory (2014) [17].

Each beamline consists of a neutron guide, detector array frame, and beam stop at a minimum. In most experiments the beamline first transits bandwidth choppers which allow time-of-flight tagging for neutron scattering spectrometers. The beamline then transits a secondary shutter – which can be closed to allow access to the array frame. The beamline then undergoes final optimization adjustments as well as slight reposition on the optics table before interaction in the array frame. Finally, a large beam backstop absorbs the remaining neutron beam to minimize dose to workers and unnecessary damage to equipment. Figure 2.3 graphically depicts this system.

The extremely high neutron fluxes allow for precise and detailed experiments. The number of beam lines also allows for simultaneous experiments. However, systems are expensive with few in existence. Consequently, competition for beam time remains fierce and other systems are needed in order to satiate demand.

Laboratory and Static Assemblies

Two types of facilities often utilize neutron imaging for nuclear materials identification – laboratories and freight-inspection facilities. Each facility has its own

strengths and weaknesses due to differing demands on its sources. A cargo inspection site will need a rugged system. The system must be simple to operate, not have a high false positive rate, and be capable of performing thousands of scans a day without maintenance. On the other hand, a laboratory system requires extremely precise and accurate results. For the laboratory false positives are not as difficult to deal with given the technically educated operators of the sources. Two state of the art systems are detailed below, one for each of these two static scenarios. The Commonwealth Science and Industrial Research Organization Fast Neutron and γ Radiography Technique (FNGR) is one of many dual radiography scanning systems prevalent at many ports worldwide. The system relies on the combination of two techniques to image and identify nuclear material – fast neutron and γ ray transmission [18].

The system measures transmission ratios and creates an image from the resulting ratio through the following: assuming a narrow-beam geometry in which scattered radiation does not reach the detector, the transmission of fast neutrons through an object of density ρ and thickness x is [18]

$$I_n / I_n^0 = e^{-\mu_n \rho x} \quad (1)$$

where I_n is the measured neutron intensity transmitted through the object, I_n^0 is the measured neutron intensity over the same path with the object removed and μ_n is the neutron attenuation coefficient. In the same manner, the γ -ray transmission is [18]

$$I_g / I_g^0 = e^{-\mu_g \rho x} \quad (2)$$

where I_g is the measured γ -ray intensity transmitted through the object, I_g^0 is the measured γ -ray intensity over the same path with the object removed and μ_g is the γ -ray

attenuation coefficient. The FNGR utilizes the ratio, R , of these two attenuation coefficients to determine the atomic number of a suspicious freight container as [18]

$$R = \frac{\mu_n}{\mu_g} = \frac{\ln I_n / I_n^0}{\ln I_g / I_g^0} \quad (3)$$

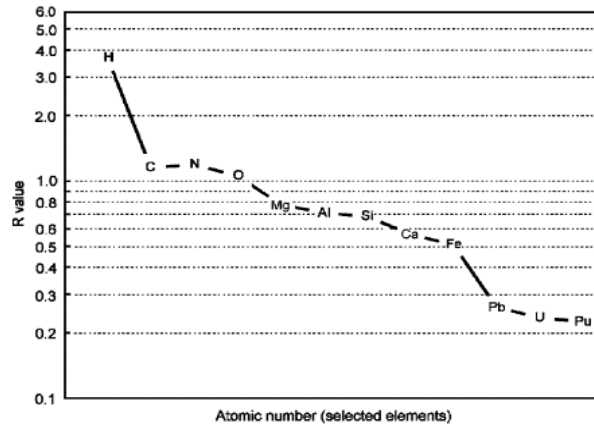


Figure 2.4. Calculated ratio R of the 14MeV neutron to cobalt-60 γ -ray mass attenuation coefficients for a range of elements from hydrogen to plutonium [18].

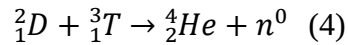
This system provides a powerful tool for comparison when the specific energies of neutrons and cosmic rays are known – as can be seen in the materials identified in Figure 2.4. However, ensuring specific energies requires additional shielding of the system from background sources discussed previously – such as cosmic rays. Furthermore, the measured attenuation coefficients are rarely the result of only one element. There are often many differing packages and materials along a path to each detector element.

Scintillator size also comprises a significant component in stationary systems. Efficient 14.1 MeV neutron detectors rely on liquid scintillators, hydrogen heavy plastics, and polymers which are bulky and weight tons on their own. The FNGR model detector array is comprised of over 700 neutron detectors [18]. The detectors in the FNGR are relatively small each one only 20mm x 20mm x 75mm. In most other cases the detectors

are much larger and relatively few. Such as the Nuclear Enterprises NE102A plastic scintillator which measures 2 meters long by 10 cm thick by 1 meter wide [19]. All arrays must incorporate algorithms to overcome baseline suppression. This is done by employing algorithms trained to account for vehicle passage or measuring different statistical features and energy distribution such as variance or skewedness. These automated systems are essential to keep false positive rates low while enabling interdiction of illicit goods.

Juxtaposed to inspection facilities are laboratory sources. These are sources and systems custom designed and built for their individual experiment's requirement or mission set. For this reason, no two systems are identical but seek to identify materials utilizing neutrons.

The fundamental operations of a laboratory associated particle neutron imaging system are described as follows. First, a deuteron is accelerated to approximately 100 keV over a pathlength typically between 7 to 10 cm in length using a compact particle accelerator at a thin Tritium rich target plate mounted on a copper heat sink rod. The Deuteron and Tritium nucleus interact in the follow manner to release a 14.1 MeV neutron [20]:



This reaction is exothermic with a Q value of 17.590 MeV. This energy is split between the α -particle 3.49 (MeV) and neutron (14.10 MeV) as can be seen in Figure 2.5 [20].

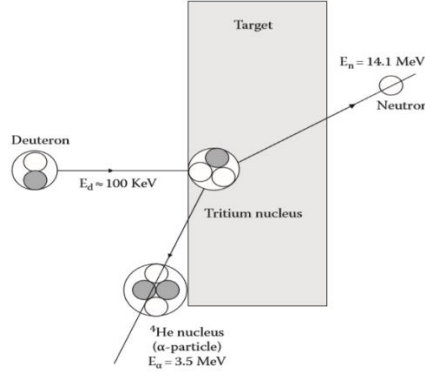


Figure 2.5. Schematic illustration of the nuclear reaction ${}^3_1\text{H}(d,n){}^4_2\text{He}$, Q value 17.590 MeV [20].

The α -particle is then detected utilizing a yttrium aluminum perovskite (YAP:Ce) scintillator within the sealed tube of the generator. The YAP scintillator is physically flush against a silicone light guide within the ING-27. This faceplate preserves position information while allowing light to exit the pressurized tube. The exiting light is then viewed by a Hamamatsu H9500 multi-anode photomultiplier tube (PMT). This PMT divides the signal into four quadrant readouts. Arrival times of the light at each of these four quadrants identifies the position of interaction. The entire system covers a solid angle of 4.25° [21].

Neutron detection varies according to the required application of the system. The most basic application is the measurement of transmitted neutrons. In transmission imaging, the number of transmitted neutrons N_{trans} is determined by

$$N_{trans} = N_0 e^{-\Sigma_t x} \quad (5)$$

where the Σ_t is the total macroscopic cross section corresponding to the total cross section of distance x covered by the neutron detected and N_0 is the incident neutron flux with no object present over the path [21]. Neutron detection in this manner takes

advantage of the time of flight difference between the γ 's and 14.1 MeV neutrons. Both γ 's and neutrons cause the PMT to trigger. The time of flight (TOF) information from the associated α -particle provide the software a means to identify incorrect γ triggers. Figure 2.6 shows the count profile for a neutron detector in an API device. One can see the initial arrival of γ 's is evident at (c) and 15 ns later at (d) the arrival of tagged 14.1 MeV neutrons.

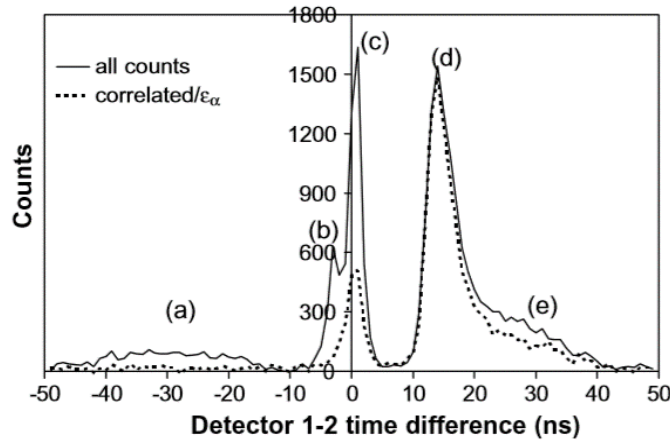


Figure 2.6. Time of flight between neutron detectors for (solid) all scattering events and (dashed) those events where an alpha has been detected. The features of the spectrum include: (a) neutrons hitting the far detector first, (b) γ -rays originating in or hitting the far detector first, (c) γ -rays originating in or hitting the near detector first, (d) 14.1 MeV neutrons hitting the near detector first by a single scatter off hydrogen and continuing to the second at 7 MeV, and (e) neutrons scattering multiple times in or before the near detector [22].

This delay allows for a time gap window of detection of only transmitted neutrons, without interference of coincident γ 's. TOF measurements can be calculated using the relativistic derivation

$$p = \frac{m_0 v}{\sqrt{1 - \frac{v^2}{c^2}}} \quad (6)$$

where m_o is the rest mass of the particle adjusted for the increase due to the particle's motion and the total energy, E , is the summation of the particle's kinetic energy plus rest mass:

$$KE + m_o c^2 = E \quad (7)$$

In a relativistic environment momentum is defined as:

$$p^2 c^2 = E^2 - m_o^2 c^4 \quad (8)$$

Substituting E from above we solve for the momentum:

$$pc = \sqrt{(KE + m_o c^2)^2 - m_o^2 c^4} \quad (9)$$

$$pc = \sqrt{KE^2 + 2KE m_o c^2} \quad (10)$$

Solving for velocity, v , one arrives at

$$v = c \sqrt{\frac{2KE}{m_o c^2 + 2KE}} \quad (11)$$

where the neutron velocity v can be determined by its kinetic energy KE , rest mass m_o , and speed of light c . The result is even though 14.1 MeV neutrons travel at 5.1 cm/ns γ arrive much earlier as they travel near 30 cm/ns [23].

The Advanced Portable Neutron Imaging System (APNIS) is one such custom designed systems. The APNIS utilizes 14.1 MeV neutrons generated by a Thermo-Fisher Scientific API-120 neutron generator (or any other DT neutron generator available) and detected by polyvinyl toluene (PVT) block detectors to reconstruct an image based upon fast neutron transmission combined with a delayed fission neutron map. The image reconstruction typically has three goals [24]:

1. Identify the materials and their location.
2. Perform a dimensional analysis of simple shapes (e.g., rectangles and circles).

3. Locate anomalous features within the reconstruction that are unexpected (if the item geometry is expected) which conform to complex shapes.

Another type of neutron identification that can occur in API devices is induced-fission tracking. In these events, incident 14.1 MeV neutrons cause fission events in high Z materials. If multiple coincident neutrons arrive at appropriate times of flight for fission-spectrum neutrons, then fission events can be correctly identified in API systems. Such induced-fission neutrons arrive after the arrival of transmission neutrons but prior to the arrival of elastic scatter neutrons [23].

This transmission profile can further be combined with small angle scattering results to yield more information about an interrogated material. Small angle scatter takes advantage of the scattering of neutrons from relatively high Z-value materials. The larger the Z-value the larger the scattered angle. When including small angle scatter, the number of neutrons that either transmit directly or scatter and transmit through an object is:

$$N_{tot} = N_0 e^{-(\mu_{tot}-\mu_{sa})x} = N_{trans} e^{\mu_{sa}x} \quad (12)$$

where the attenuation coefficient has been reduced by the small angle scatter (μ_{sa}). Thus, the small angle scatter can be deduced from $\ln(N_{tot}/N_{trans})$. This is useful in the determination of material composition of nuclear material which can be further supplemented with large angle scatter [21].

Large angles scatter relies on the elastic scattering of neutrons off an interaction object. In the case of the APNIS the source to detector distance, r is fixed and the angle ϕ , is deduced from the knowledge of the initial neutron trajectory – due to associated α -

particle – and detection position (see Figure 2.7). Image reconstruction is the determination of distance d

$$d(A, t) = \frac{\left(\frac{vt}{r}\right)^2 - 1 + \frac{1}{A} \cdot \left(\frac{2vt}{r} \cdot \cos(\varphi) - 1 - \left(\frac{vt}{r}\right)^2\right)}{2 \cdot \left(\frac{vt}{r} - \cos(\varphi)\right)} \quad (13)$$

to each scatter location utilizing conservation of energy and momentum for elastic scatter from a single nucleus of mass number A given an arrival time of t for 14.1 MeV neutrons having speed v [21].

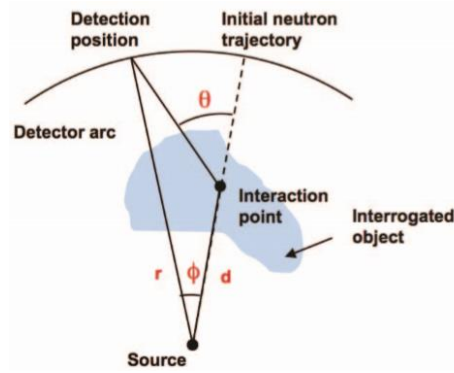


Figure 2.7. Elastic scatter diagram with variable identification and labeling

The resulting tomographic measurement is often smeared over the reconstruction area due to incorrectly reconstructed distances, d . As a result, the APNIS increments neutron counts in a two-dimensional histogram of detected angle, φ , vs detection time, t , and compares the result to established norms – based upon material make up and assigns a value of low, medium or high A to a material. Hydrogen is an ideal material to identify in this manner due to its importance in nuclear materials and its low A value of one makes it easily identifiable [21].

Mobile / Portable Systems

Portable neutron interrogation methods rely on employing one or more of the previously described imaging techniques in austere environments. Material characterization, geometry, and induced fission identification are three of the measurable standards for any portable system. Due to weight limitations of fieldable systems the number and size of detectors is limited. In the case of the Portable Nuclear Materials Identification System (PNMIS) eight PVT detectors each with dimensions of 2.54 x 2.54 x 15 cm were utilized representing a reasonable compromise between resolution and efficiency. This design utilized the associate particle technique to accumulate a transmission image (see Figure 2.8). This work will present a design which utilized 28 2.54 x 2.54 x 12cm PVT detectors. The gray scale indicates the local value of attenuation, $-\ln(N/N_0)$. Measurements with the eight-detector system take 1 to 5 minutes per detector position, with three positions taken in total, while exposure times are shielding dependent [25].

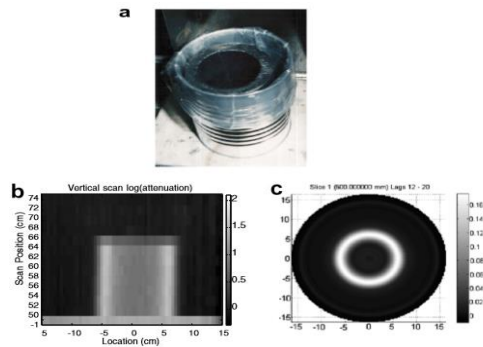


Figure 2.8. Images of a depleted uranium casting. The actual casting is shown in (a), while a transmission radiograph taken with the test system is shown in (b), and a tomographic reconstruction of the single projection assuming axial symmetry is shown in (c) [25].

Results yield a detailed understanding of the internal structure of the depleted uranium casting (Figure 2.8) – especially when compared to x-ray tomography which is unable to penetrate such high Z materials. The unfortunate shortcoming of such mobile systems is the repeated measurements that they require in order to generate complete images. Each of the detectors in the case of the APNIS only takes a 2.54 cm vertical slice image once every 3 – 15 minutes. The resulting image in (b) is the result of over 10 images, hardly an expedient system. In practical use only a few slices are necessary to reason the geometry of an object. This was the approach taken in this work. Only complex geometries with many objects would require so many images to arrive at an understanding of the overall makeup of the geometry.

Image Reconstruction and Post Processing

The current focus of deployable systems involves iterative computational analysis of both coincidences and multiplicity due to fission and transmission data to produce a detailed assessment. These systems include Filtered Back Projection (FBP), Maximum Likelihood Expectation Maximization (MLEM) Transmission Reconstruction, and Maximum Likelihood Expectation Maximization (MLEM) Fission Reconstruction [26]. Additional research is currently underway involving the use of machine learning algorithms – Discrete Tomography (DT) – to reconstruct an image composed of just a few known materials [27].

FBP takes advantage of the known geometry of the source, detectors and object's dimensions, x , to determine the attenuation coefficient, μ , based upon attenuation lengths $x\mu$. The more samples the more accurate this system as the system often relies on an axial

symmetry assumption to reconstruct an image. This can be seen in Figure 2.9 where a complete 360 degree scan is used to reconstruct a higher fidelity image.

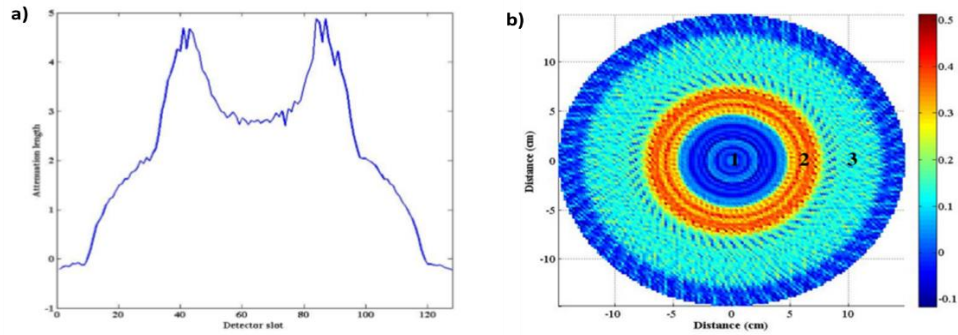


Figure 2.9. (a) Attenuation lengths (μx) of Object 1 plotted against detector slot (correlates to lateral position within the object) for a single rotation. (b) FBP radial tomography of Object 1 that plots the attenuation coefficient (cm^{-1}) as a function of distance (cm) [26].

MLEM reconstruction begins with an initial guess of 1 for the attenuation coefficient in each pixel of the reconstruction and a correction factor is applied to each forward projection pixel. The two processes yield similar results and take minimal computation time to evaluate. However, one cannot solely rely upon MLEM reconstruction as FBP is often used to determine overall dimensions of an object (Figure 2.9) [26]. Fission mapping MLEM applies the same process to the coincidence counts from collected neutron data. Fission mapping MLEM tomography reconstructs its images based on single and double neutron counts. This data is particularly useful in determining boundaries between HEU and DU. FBP and MLEM transmission imaging do not allow for delineation of this boundary due to the proximity of DU and HEU attenuation coefficients [26].

Assuming axial symmetry, a slice of the tomography image is taken. Utilizing the two doubles peaks one averages the distance between the peaks visible at 0.12 and the bases at near 0.0 to attain precise boundary measurements of materials [26]. Note this system does not rely on the distance between the peaks at .12. The analysis is conducted for each individual peak. If the sample was offset the peaks would be skewed in the image below – Figure 2.10 is the ideal scenario where imaging is focused on the center of mass of the object.

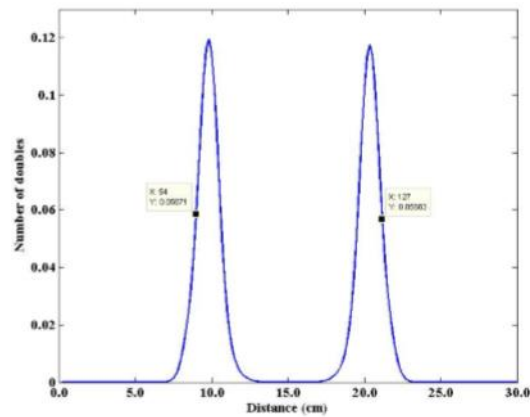


Figure 2.10. Centerline slice of the MLEM fission reconstruction doubles tomograph with the two peaks corresponding to the cylindrical region of HEU shielded on the outside by a region of DU [26].

Ultimately, any fieldable system must be able to conduct all this post processing quickly behind a GUI that does not require an advanced understanding of nuclear engineering in order to operate. Machine learning algorithms ideally fit this requirement as all calculations are done without the input of a user. While the field is relatively new, evolutionary algorithms show promise in the estimation of gray intensities in a discrete image and optimization tasks [27].

Scintillators and detectors

For many years the common tool in neutron radiography was the film method. Much as in x-ray or normal photography, a converter of a neutron absorbing material was used to produce a latent image on a silver halide film. Over time additional demands have arisen from scientist and industry alike. Three competing demands must now be optimized for each new system. They are spatial resolution, time resolution, and image quality [28].

Spatial resolution is a product of beam properties, detector performance, and detector plane. Beam properties are an inherent property of the neutron generator – they cannot be improved without entirely new systems developed. As a result, efforts focus on detector performance and proper alignment [28].

Time resolution is a function of two main components exposure time, t_{exp} , and readout time, t_{read} . Exposure time defined as the period over which neutrons are applied and converted into a detectable signal. Readout time accounts for the period required to process and convert an image to file, as well as transfer to a permanent storage medium. The real frame rate is thus defined as [28]:

$$FR = \frac{1}{(t_{exp} + t_{read})} \quad (14)$$

The image quality is the ratio of the signal to noise of the image, i.e.,

$$IQ = \frac{S}{\sqrt{N_{read}^2 + N_{dark}^2 + N_{shot}^2}} \quad (15)$$

where S is the total count for any pixel, N_{read} is the noise level from the readout process, N_{dark} is the noise level from background radiation or random effect, and N_{shot} is the noise level from the statistics of the shot process itself.

Table 2.1. Performance Parameters of Neutron Detection Systems for Imaging [30]

Detector system	X-ray film and transmission light scanner	Scintillator + CCD-camera	Imaging plates	Amorphous silicon flat panel	CMOS pixel detector
Max. spatial resolution (μm)	20–50	100–500	25–100	127–750	200
Typical exposure time for suitable image	5 min	10 s	20 s	1–10 s	0.1–50 s
Detector area (typical)	18 cm $\times$ 24 cm	25 cm $\times$ 25 cm	20 cm $\times$ 40 cm	30 cm $\times$ 40 cm	3.5 cm $\times$ 8 cm
Number of pixels per line	4000	1000	6000	1750	400
Dynamic range	10^2 (non-linear)	10^5 (linear)	10^5 (linear)	10^3 (non-linear)	10^5 (linear)
Digital format	8 bit	16 bit	16 bit	12 bit	16 bit
Readout time	20 min	2–100 s	5 min	0.03–1 s	0.2 s

Detectors for neutron imaging can be categorized into five unique systems: x-ray film and transmission light scanners, scintillator + CCD-Camera, imaging plates, amorphous silicon flat panels, and CMOS pixel detectors. Each system has unique characteristics making it applicable to differing mission sets (see Table 2.1). Captured below are some of their characteristics with three promising systems discussed.

Gaseous Neutron Detectors.

Gaseous neutron detectors rely on three main components – a neutron moderator, ionization chamber, and position encoding pickup aperture. In the neutron moderation section, a hydrogen-rich material converts fast-neutrons, via elastic scatter interactions, into detectable protons. H, Be⁹, and Li⁶ I are a few of the materials normally selected to

serve as moderators as they have relatively large neutron cross sections which subsequently promote the release of detectable protons or other charged particles such as α -particles. Once a heavy charged particle has been created the neutron is no longer of concern [29].

The focus of the detector then transitions to the identification of heavy charged particles within the ionization chamber. The ionization chamber consists of an inert noble gas. This gas can be ionized through interactions with the heavy charged particles. Ionization electrons are then collected and multiplied within this region using gaseous electron multipliers [30].

Finally, the electron counts are collected on a pickup electrode which is then converted to signal. This portion of the detector is known as the position encoding pickup aperture. Gaseous neutron imaging detectors suffer from low optical detection efficiency – 0.2% for 5 MeV neutrons within a 1mm thick converter [30]. Converters must remain thin as heavy charged particles exhibit extremely short mean free paths within matter due to high linear energy transfer to the transit medium [31]. Furthermore, using a broad-energy neutron beam multi-elemental target analysis is possible.

This analysis exploits the resonance energy regions within each element corresponding to differing neutron energies. By isolating each element's unique resonance energies images can effectively step through the periodic table for each material in question:

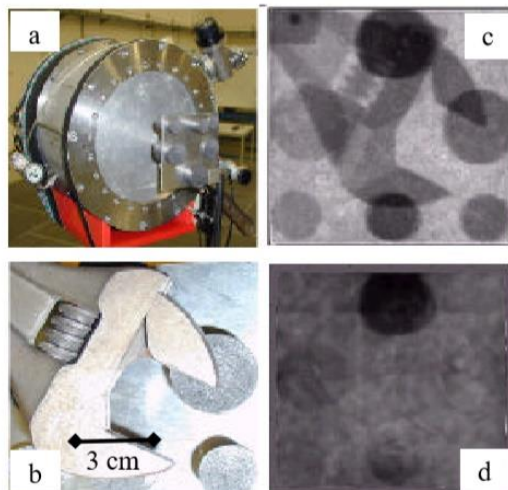


Figure 2.11: Imaging of carbon rods and a steel wrench with fast neutrons: a) detector and sample in the neutron beam, b) the sample which consists of a steel wrench and carbon rods of 30 and 20 mm in diameter and 20, 40 and 60 mm length, c) and d) combined neutron images [30].

Unfortunately, the interrogation area of such detectors remains limited. The largest gaseous neutron imaging detectors cover a 100 x 100 mm window size (see Figure 2.11). As a result, this method of examination remains limited in application.

Scintillators with Intensified CCD Cameras

Scintillators with Intensified CCD Cameras are the means of detection currently employed on the many versions of the NIS. Scintillators with intensified CCD Cameras convert high energy neutrons to light pulses. The CCD Camera is attached to the far end of the plastic scintillator. The scintillator is incased in an optically reflective casing which directs photons generated in the plastic matrix to the CCD. The CCD is only sensitive to a select energy window – normally a few hundred keV broad – out of the overall energy distribution generated by the neutrons within the scintillator [30]. As previously discussed, TOF data is extremely important for CCD Camera systems.

In order to attain accurate TOF data, scintillators with ns fall time constants are chosen despite their poor scintillation efficiency. These fast decay plastics result in reduced intrinsic detection efficiency. Typical intrinsic efficiency is on the order of a few percent (3-15%) for PVT [30]. As a work around, high neutron fluxes are utilized to overcome low detection efficiency.

These systems are often more limited by their detection arrays and neutron generator source than by any other component. A horde of plastic scintillators or many hundreds of mosaic images are required to image larger objects. Plastic scintillators also lack the spectroscopic capability to detect and identify γ -ray emitting materials [32]. For these reasons plastic scintillators are often reserved for critical areas of special interest during inspection while other systems conduct broad interrogation.

High Z Sensitized Scintillators / Doped Scintillators

High Z sensitized scintillators promise to overcome many of the short comings of unsensitized plastic scintillators. By increasing the effective atomic number of a plastic scintillator material, Z_{eff} , the ability to detect γ -rays increases. This is a direct result of the photoelectric effect which scales as a function of Z_{eff}^4 . Increasing Z_{eff} often results in lower light yield in large volumes due to low optical transparency of high Z materials. The challenge such scintillators face is how to combine the two retaining transparency and yield [32].

High-Z Sensitization by organometallics effectively adds a dye to a plastic scintillator to increase the materials light yield at higher energies. One of the primary obstacles any added dye must overcome is the intrinsic self-absorption properties of a dye.

Normally, photons emitted from a dye will just be reabsorbed by the dye itself. This self-absorption can be mitigated with a wavelength shifter incorporated into the dye. This increase in transmitted final emission is due to the slight decrease in total photon energy emission, k_f , of the wavelength shifter which allows the final photon emissions to transit any further dye relatively unimpeded (see Figure 2.12). Iridium complex fluor and triphenyl Bi-plastic (19-21 wt % Bi) are two promising examples of this development.

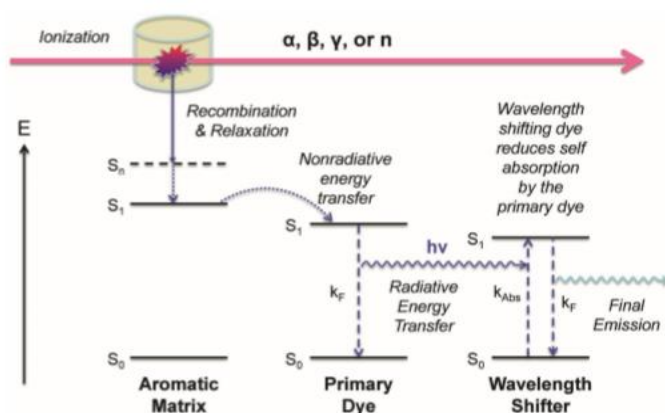


Figure 2.12: Summary of the major photophysical processes in the scintillation of organic materials [32].

High-Z Sensitization by nanoparticles is another method by which researchers are seeking to expand the capabilities of plastic scintillators. In this case attractive high-Z nanoparticles are suspended in the plastic (see Figure 2.13). The method of suspension is currently unsolved as direct mixing of nanoparticles and polymers is ineffective [32]. The exclusion zones of polymers, high specific surface area, and surface energy of nanoparticles have prevented successful suspension to date. Viscous acid matrixes hosting nanoparticles have also yielded similar results. Quantum dot loading via in situ polymerization shows the most promise currently. Many improvements remain to be

made including finding a material with a higher Z_{eff} , finer matrix polymers, and shorter wavelength of emitted photons [32].

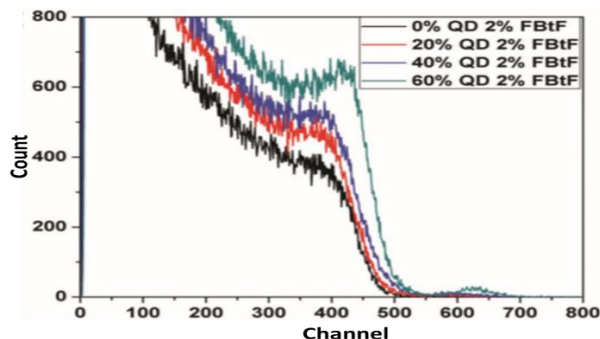


Figure 2.13: Pulse height spectra of 0–60% QD/2% FBtF/PVT nanocomposite scintillators [32].

Summary

Identification of nuclear materials using portable neutron interrogation is in its infancy with few systems deployed. The need exists for more fieldable systems such as the PINS Prompt Gamma-ray Neutron Activation Analysis system. Limited success has been demonstrated in the use of API techniques to rapidly identify potential nuclear materials. Incorporation of coded-aperture patterns to mobile neutron imaging [33], time-dependent neutron count rate analysis [34], scintillator efficiency enhancements using nanoparticle-based dyes [32], and advances in image reconstruction software [26] all show promise in moving neutron imaging from the lab to the field. Further energy must be directed to the incorporation of currently existing technology into a fieldable system. No one technology holds the key to nuclear material identification on its own. Only a device which enables the user via a graphical user interface to overlay each layer of

information as requested will effectively allow successful nuclear material identification.

To date this remains an untried course of research.

CHAPTER THREE

MATERIALS AND METHODS

Physical Materials and Methods

This section of the work first describes process taken in the physical construction of the detection system. After definition of the material components and the steps taken in construction of the fieldable system the electronics and calibration steps taken will be outlined. Although none of the electronics advance the field, they are pertinent to address in order to access future deployment and other factors such as timing resolution. An order of experiments is then presented before a description of the simulation method. Simulations were conducted utilizing MCNP6.2 to output a binary PTRAC file which is then parsed in python 3.7 to extract useful data. An in-depth account of neutron beam simulation is essential in order to arrive at subsequent radiograph images which are presented in the results section of this work.

Detector Arm

The detector arm itself was constructed of one solid piece of ASTM B209 Aluminum plate cut to match the beam size of the DT generator. Previous versions of a lightweight modular arm were constructed of 3D printed material - graphite infused carbon nylon (see Figure 3.1).



Figure 3.1. 3D printed graphite infused carbon nylon

However, this material suffered from material creep when subjected to loads for extended periods of time (24 hours plus). While 3D printing with standard plastics is generally successful, the aim of this material was to reduce weight while maintaining structural integrity. This process of printing with advanced materials is still Avant-garde and susceptible to setbacks. As a result, an aluminum arm was chosen as the direct mount for the detector boxes.

The arm was cut to radius of 1.151 meters and was of sufficient size to mount seven detector boxes without stability issues. This radius only physically allows seven detector boxes to fit within the beam width covered by the 13 of the 15 α -pixels of the ING-27 Russian D-T Generator (ING-27). Even then the last two detectors on the extremities of the arm will often reside out of the cone of the beam. Previous versions of the NMIS mounted 32 detectors on a similar sized arm [35] – yet, they were not mounted in modular units. Instead the scintillators were directly attached to the arm. The modular units would allow a user to quickly replace faulty detectors without the need to demount and mount each detector.

Modular Units

In order to securely mount the scintillators to the detector arm detector boxes were custom designed and fabricated in house. The design was created utilizing Autodesk Fusion 360 – a commonly employed 3D sculpting software and printed using available stock (see Figure 3.2).

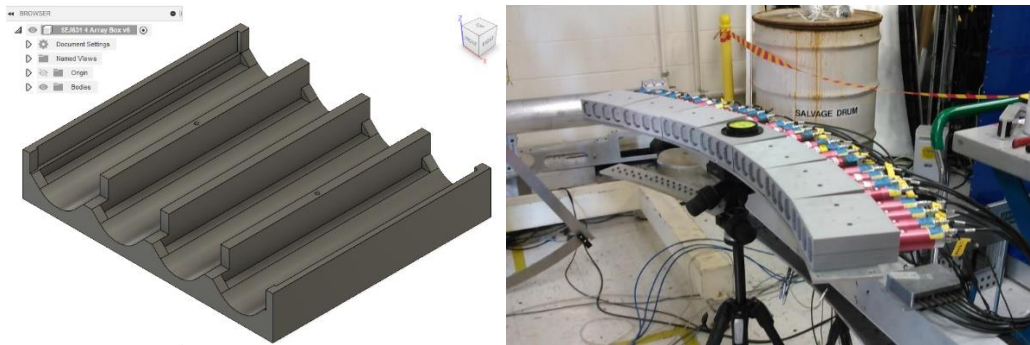


Figure 3.2. 3D printed detector casings

Each detector box consisted of four scintillators sandwiched between two housing frameworks. The two halves were secured together utilizing 1.5-inch screws drilled directly into the housing frames sandwiching the scintillators in place between them. The bottom frame was then secured directly into the aluminum arm utilizing $\frac{3}{4}$ -in screws.

Scintillators

Each SEJ631 Scintillator Detector consisted of a 30mm x 30mm x 10.16mm PVT scintillator block directly coupled with a Photomultiplier Tube (PMT). Each detector required two connections – one power source and one signal cable. These detectors were utilized as they were the most compact scintillators available at the time. Work on a replacement SiPM-based detector will most likely replace these detectors in the future. SiPM photodetectors are expected to be superior due to their reduced power requirements

arising from their lack of need for a photomultiplier tube. SiPM detectors would eliminate the need for the second tripod positioned behind the detector arm pictured below and highlighted in yellow (see Figure 3.3). This tripod's only purpose is to route power supply cabling to the detector arm.

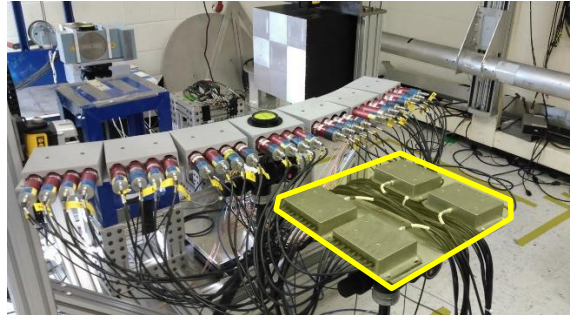


Figure 3.3. Portable fieldable assembly

All equipment in this pictured set up, to include the DT generator is mounted on lightweight Manfrotto 290 Dual camera tripods. Alignment devices are also mounted on similar camera tripods. In addition, the signal processing equipment is mounted in a wheelable VME crate. Set up is determined by one of two locations: laboratory or in the field. Set up methods are described in *Appendix: A – Laboratory Set Up* and *Appendix: B – Obstructed View Set Up*.

Readout Equipment

The signals from each detector pass through an Ortec CF8000 Octal Constant-Fraction Discriminator. Signal from eight detectors feeds each CFD. From there data from the four CFDs is passed to one Caen VME 8004B crate with 2 VME cards. The final counts are then passed to a laptop via a USB 3.0 cable. The α pixel data is passed similarly to the VME crate via two additional CFDs. The entire assembly is air cooled by

two assemblies of six consumer grade airflow cooling fans – one mounted on top of the VME crate and one directly below the CFDs. The entire assembly measures 22 x 22 x 20 in. Power is supplied to the detectors via one Caen 4 channel high voltage power supply. An auxiliary Ortec Dual Counter/Timer is mounted with the CFDs in order to conduct calibration and to allow on the spot adjustments once calibration measurements have been completed should detectors drift (see Figure 3.4).



Figure 3.4. Signal processing equipment crate

Software and Graphical User Interface (GUI)

The ING-27 is controlled using a simple GUI. Below is an image of the interface. In general operation the generator is initiated at 50% max power. Power is increased at a rate of 10% every minute until the desired level of neutron flux is attained (see Figure 3.5).

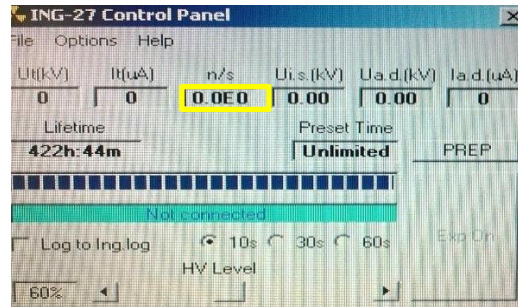


Figure 3.5. ING-27 Graphical user interface (GUI)

A benefit of the ING-27 is the neutron flux is listed clearly on the GUI in addition to the voltage and current. This is superior to the API-120 as no flux readout is displayed and one must rely on assumptions to estimate neutron flux with the API-120.

Unfortunately, the neutron flux can vary quite considerably despite relatively steady voltage and current. Fluctuations of 15 – 20% could be observed on the old ING-27 (420+ lifetime operational hours) and of 5 – 10% on the new ING-27 (100+ lifetime operational hours). The best way to avoid these fluctuations is to warm up the generators by allowing them to run for 3-5 minutes before taking measurements. A downside to the Russian design is there is no x-ray shielding attempt by the DT Generator itself. If ever fielded a new generator design with both neutron flux readout and x-ray shielding would be ideal.

Once the neutron flux has stabilized the correlation count software is executed. The correlation count script is executed by command line interface. Changes to parameters of the experiment are made in a C++ text file that is read into the script once executed. Length of experiment, timing resolution, correlation window, and name are some of the parameters that can be adjusted by altering the text file.

The output file is then read through another python parse program (developed by the author) which finds the peak correlation counts and sums the counts for the desired time window – in this case 5ns. This python program named Correlation_parser.py can be found in Appendix I: Correlation Parser Code. The resulting csv file allows for easy comparison to simulation results as both codes output in the same file format and most applications to include matlab and excel handle csv files with ease.

Calibration

Once all the equipment was connected and powered on a void measurement was conducted utilizing a 1 mCi Cs¹³⁷ source located at the focal point of the arm (see Figure 3.6) – in the case of this arm at 1.15 meters. This allowed all 28 detectors to be calibrated simultaneously. Each detector's individual threshold voltage was then set to read a count rate of 1650 counts per 10 second time interval. This previously determined, low count rate ensured no pile up effects with EJ200 (decay time of 2.1 ns).

In previous years, a much weaker μ Ci strength source was used when the mCi source was unavailable. Cs¹³⁷ was used in both cases due to its characteristic 662 keV peak caused by the 661.7 keV γ emission of its daughter nuclide Ba^{137m} as it deexcites to the ground state Ba¹³⁷ [36]. This weaker source would then be repositioned directly on the face of each detector for each calibration.

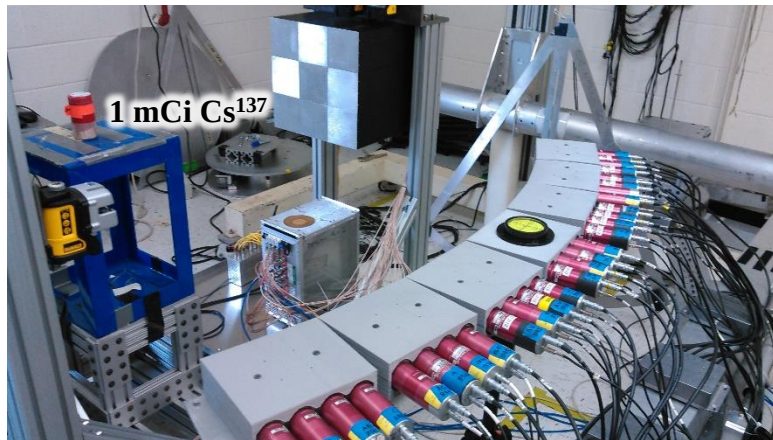


Figure 3.6. Cs137 Calibration set up

Similarly, when spot checks were conducted by the author this weaker $1.0 \mu\text{Ci Cs}^{137}$ source was utilized. Most frequently this was used to measure the response function on a specific region of the arm in order to ensure detectors within the vicinity were properly functioning.

In addition to calibration of measurements two non-functioning DT generators were restored to working order during this time. Both generators were Russian Federation ING-27s. Due to funding limitations and research priorities both generators had been sitting for over a year without functioning. As a rule, the longer the generators sit without functioning the less probably their chance of restoration to functioning status is. With the restoration of these generators the laboratory now has the capacity to generate 14.1 MeV neutron sources at fluxes of $1.7\text{E}7 \text{ n/s}$. However, the generators were generally operated at 50-60% of this flux due to the increased size of the radiological control area and posting requirements that is required to operate at full power.

Void Measurement

The first measurement executed on any day of measurement was the void measurement. This measurement served three purposes. First, it checked for drift in the thresholds of the detector arm and changes in the environment from one day of measurement to the next. Minimal drift was observed in these void measurements. Over a six-month period only one of 28 detectors observed a significant drift (see Figure 3.7).

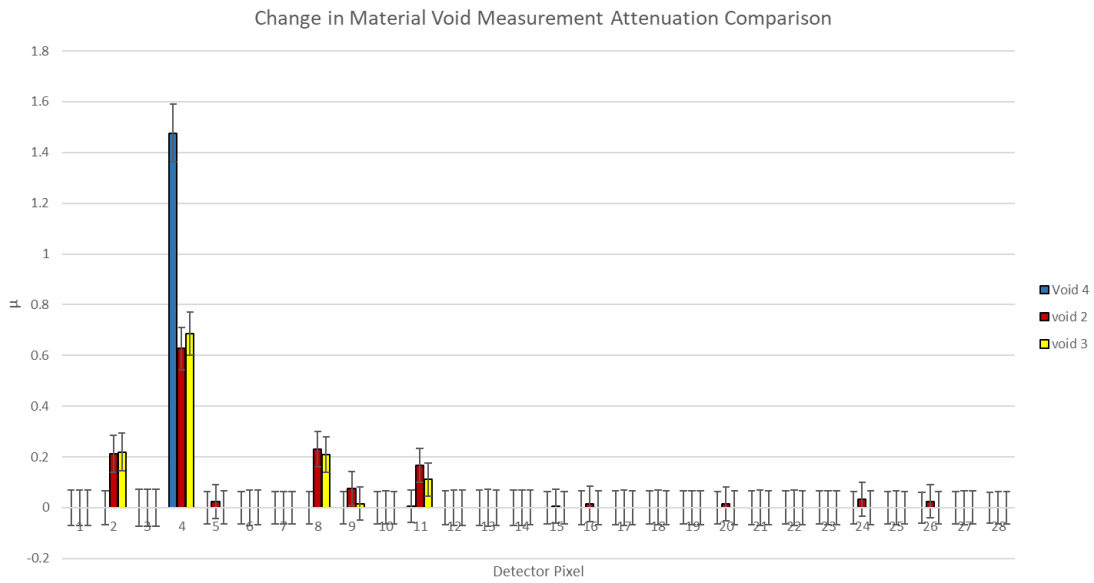


Figure 3.7. Detector drift observations

This material attenuation coefficient was calculated by comparison of the first void measurement counts, N_0 , to the 2nd, 3rd, and 4th void measurements, N_i , collected over the next two months according to:

$$\mu = -\ln \left(\frac{N_i}{N_0} \right) (1)$$

As can be seen above the only substantive drift that occurred was in detector number four. Minimal drift was observed in two other detectors on two separate occasions.

Trials – Order of Measurements

Data was collected over a period of five months. A total of 56 measurements were compiled. A steel/void measurement was the first object imaged. Over the next five months, aluminum, polyethylene, steel, depleted uranium and other advanced configurations were all imaged at various locations and offsets. Finally, combinations of materials and offsets and multiple materials were imaged. Figure 3.8 details all measurements collected during this study.

Detector Arm				Object Position			File Name	Notes
Date	X	Y	theta	X	Y			
10/11/2019	0	0	0	0	0	85.1	11OCT19_Troxell_steel.corr.csv	Centered Steel Cylinder 85.1 cm from face of detectors and 30 cm from face of DT generator
10/11/2019	0	0	0	0	0	0	11OCT19_Troxell_void.corr.csv	properly aligned void
10/23/2019	0	0	0	0	-10	57.55	Aluminum_10cm_left_23OCT19.corr_sd.csv	-10cm X axis Al Cylinder half way between detectors and DT generator (57.55 cm)
10/23/2019	0	0	0	0	5	57.55	Aluminum_5cm_right_23OCT19.corr_sd.csv	+5cm X axis Al Cylinder half way between detectors and DT generator (57.55 cm)
10/23/2019	0	0	0	0	0	57.55	Aluminum_center_23OCT19.corr_sd.csv	Centered Al Cylinder half way between detectors and DT generator (57.55 cm)
10/23/2019	0	0	0	0	-10	57.55	DU_10cm_left_23OCT19.corr_sd.csv	-10cm X axis DU Cylinder half way between detectors and DT generator (57.55 cm)
10/23/2019	0	0	0	0	5	57.55	DU_5cm_right_23OCT19.corr_sd.csv	+5cm X axis DU Cylinder half way between detectors and DT generator (57.55 cm)
10/23/2019	0	0	0	0	0	57.55	DU_center_23OCT19.corr_sd.csv	Centered DU Cylinder half way between detectors and DT generator (57.55 cm)
10/23/2019	0	0	0	0	-10	57.55	Poly_10cm_left_23OCT19.corr_sd.csv	-10cm X axis Poly Cylinder half way between detectors and DT generator (57.55 cm)
10/23/2019	0	0	0	0	5	57.55	Poly_5cm_right_23OCT19.corr_sd.csv	+5cm X axis Poly Cylinder half way between detectors and DT generator (57.55 cm)
10/23/2019	0	0	0	0	0	57.55	Poly_center_23OCT19.corr_sd.csv	Centered Poly Cylinder half way between detectors and DT generator (57.55 cm)
10/23/2019	0	0	0	0	-10	57.55	steel_10cm_left_23OCT19.corr_sd.csv	-10cm X axis Steel Cylinder half way between detectors and DT generator (57.55 cm)
10/23/2019	0	0	0	0	5	57.55	steel_5cm_right_23OCT19.corr_sd.csv	+5cm X axis Steel Cylinder half way between detectors and DT generator (57.55 cm)
10/23/2019	0	0	0	0	0	57.55	steel_center_23OCT19.corr_sd.csv	Centered Steel Cylinder half way between detectors and DT generator (57.55 cm)
10/23/2019	0	0	0	0	0	0	void_23OCT19.corr_sd.csv	properly aligned void
10/30/2019	1.27	0	0	0	0	0	void_1.27cm_right_30OCT19.corr_sd.csv	+1.25cm X axis offset void measurement
10/30/2019	0	0	-2.272	0	0	0	void_2.272_degreeleft_30OCT19.corr_sd.csv	2.2 degree counterclockwise rotation in XY plane void
10/30/2019	0	0	2.272	0	0	0	void_2.272_degreeright_30OCT19.corr_sd.csv	2.2 degree clockwise rotation in XY plane void
10/30/2019	0	0	0	0	0	0	void_30OCT19.corr_sd.csv	properly aligned void
10/30/2019	0	0	-4.537	0	0	0	void_4.537_degreeleft_30OCT19.corr_sd.csv	4.5 degree counterclockwise rotation in XY plane void
10/30/2019	0	0	4.537	0	0	0	void_4.537_degreeright_30OCT19.corr_sd.csv	4.5 degree clockwise rotation in XY plane void
10/30/2019	0	0	-6.789	0	0	0	void_6.789_degreeleft_30OCT19.corr_sd.csv	6.7 degree counterclockwise rotation in XY plane void
10/30/2019	0	0	6.789	0	0	0	void_6.789_degreeright_30OCT19.corr_sd.csv	6.7 degree clockwise rotation in XY plane void
11/8/2019	-5	0	0	0	0	0	void_5_left_08NOV19.corr.csv	-5cm X axis offset void measurement
11/8/2019	5	0	0	0	0	0	void_5_right_08NOV19.corr.csv	+5cm X axis offset void measurement
11/8/2019	0	0	0	0	0	0	void_08NOV19.corr.csv	properly aligned void
11/8/2019	-10	0	0	0	0	0	void_10_left_08NOV19.corr.csv	-10cm X axis offset void measurement
11/8/2019	10	0	0	0	0	0	void_10_right_08NOV19.corr.csv	+10cm X axis offset void measurement
11/8/2019	-15	0	0	0	0	0	void_15_left_08NOV19.corr.csv	-15cm X axis offset void measurement
11/8/2019	15	0	0	0	0	0	void_15_right_08NOV19.corr.csv	+15cm X axis offset void measurement
12/19/2019	0	0	0	0	0	0	centered_19DEC19.corr.csv	properly aligned void
12/19/2019	0	0	0	0	0	0	void_19DEC19.corr.csv	properly aligned void
12/20/2020	0	10	0	0	0	0	+10cm_Y_closer_20DEC19.corr.csv	+10cm Y axis offset void measurement
12/20/2020	0	5	0	0	0	0	+5cm_Y_closer_20DEC19.corr.csv	+5cm Y axis offset void measurement
12/20/2020	0	-10	0	0	0	0	-10cm_Y_away_20DEC19.corr.csv	-10cm Y axis offset void measurement
12/20/2020	0	-5	0	0	0	0	-5cm_Y_away_20DEC19.corr.csv	-5cm Y axis offset void measurement
12/20/2020	0	0	0	0	0	0	void_20DEC19.corr.csv	properly aligned void
1/27/2020	0	0	0	0	0	0	void_27JAN20.corr.csv	properly aligned void
1/27/2020	0	0	0	0	0	75.1	DU_Pb_centered_27JAN20.corr.csv	Centered DU Cylinder inside of Pb cylinder (75.1 cm)
1/27/2020	0	0	-2.272	0	0	75.1	2.272_degreeleftDU_Pb_27JAN20.corr.csv	2.2 degree counterclockwise rotation in XY plane DU and Pb
1/27/2020	0	0	-4.537	0	0	75.1	4.537_degreeleftDU_Pb_27JAN20.corr.csv	4.5 degree counterclockwise rotation in XY plane DU and Pb
1/27/2020	0	0	-6.789	0	0	0	6.789_degreeleftDU_Pb_27JAN20.corr.csv	6.7 degree counterclockwise rotation in XY plane void [forgot to raise target back up]
1/27/2020	0	0	-6.789	0	0	75.1	6.789_degreerightDU_Pb_27JAN20.corr.csv	6.7 degree clockwise rotation in XY plane DU and Pb
1/27/2020	0	0	2.272	0	0	75.1	2.272_degreeright_27JAN20.corr_sd.csv	2.2 degree clockwise rotation in XY plane DU and Pb
1/27/2020	0	0	4.537	0	0	75.1	4.537_degreeright_27JAN20.corr_sd.csv	4.5 degree clockwise rotation in XY plane DU and Pb
1/27/2020	0	0	6.789	0	0	75.1	6.789_degreeright_27JAN20.corr_sd.csv	6.7 degree clockwise rotation in XY plane DU and Pb
1/27/2020	0	5	0	0	0	75.1	+5cm_Y_closer_DU_Pb_27JAN20.corr.csv	+5cm Y axis offset DU+Pb measurement
1/27/2020	0	10	0	0	0	75.1	+10cm_Y_closer_DU_Pb_27JAN20.corr.csv	+10cm Y axis offset DU+Pb measurement
1/27/2020	0	-5	0	0	0	75.1	-5cm_Y_away_DU_Pb_27JAN20.corr.csv	-5cm Y axis offset DU+Pb measurement
1/27/2020	0	-10	0	0	0	94.1	-10cm_Y_away_DU_Pb_27JAN20.corr.csv	-10cm Y axis offset DU+Pb measurement
1/27/2020	5	0	0	0	0	75.1	+5cm_X_right_DU_Pb_27JAN20.corr.csv	+5cm X axis offset DU+Pb measurement
1/27/2020	10	0	0	0	0	75.1	+10cm_X_right_DU_Pb_27JAN20.corr.csv	+10cm X axis offset DU+Pb measurement
1/27/2020	15	0	0	0	0	75.1	+15cm_X_right_DU_Pb_27JAN20.corr.csv	+15cm X axis offset DU+Pb measurement
1/27/2020	-5	0	0	0	0	75.1	-5cm_X_left_DU_Pb_27JAN20.corr.csv	-5cm X axis offset DU+Pb measurement [channel 15 long cable]
1/27/2020	-10	0	0	0	0	75.1	-10cm_X_left_DU_Pb_27JAN20.corr.csv	-10cm X axis offset DU+Pb measurement
1/27/2020	-15	0	0	0	0	15	Steel_convert_for_gammas_27JAN20.corr.csv	

Figure 3.8. Experimental measurements chart

In synopsis, the measurements began with easily imaged objects in ideal locations and alignments. As our confidence in the system increased more difficult arrangements and geometries were imaged. This culminate with tests of more advanced geometries such as a DU cylinder encased in a 2.4 cm thick led cylinder and blind test.

Simulation Method

The process of simulation shared many similarities with the laboratory experimental process. In both cases the experiment had to be set up. In the case of the laboratory this entailed physical measurements, wiring, and safety checks. This associated to building the input deck for MCNP in the simulation space before execution and post-processing. Each scenario measured in the lab was slightly different than the last – which required slight adjustments to set ups. This was also true for the simulation process where variations in the input parameters needed adjustment. These adjustments were executed in a series of python scripts in order to minimize the human error component in creation of the MCNP input deck.

Each MCNP input deck consists of a minimum of three components: cell card, surface card, and data card. The cell card determines the boundaries of all objects, material identification, and densities. This card is relatively straight forward with only slight changes to material and densities which can easily be changed by direct edit to each file if necessary.

Second is the surface card. This card defines shape and location of objects. The defining of location is a particularly difficult process that is best left to a computer to calculate. Each time a new detector location is changed the detector geometry card script

is executed for the new position. In addition, material properties and interrogation object locations are adjusted in this card. These adjustments are done manually as they typically entail changing only two values. Finally, there is the data card. This card is the workhorse of MCNP defining the physics, materials, length of simulation, outputs to generate, and anything else required of the code.

Unlike most MCNP scenarios, the entire assembly requires the output of 15 separate decks to compile one complete scan image. This is a result of the tagged α pixels and their respective array profiles which will be discussed in length in the *Source Methodology* section. Just as in the experimental set up conducting a void measurement simulation was the first step of the simulation process. Following collection of a void measurement attenuation profilers were only then reconstructed for the total scan.

The primary workspace utilized during the process of simulation was Visual Studio Code. Visual Studio code was selected due to ability to run multiple the unique environmental constraints of the MCNP PTRAC module on windows platforms. Specifically, mcnptools is recommended and has only been tested on Visual Studio Code for python. All scripts were written in Python 3.7.3 unless otherwise specified. The flow diagram in Figure 3.9 and Figure 3.10 below lay out the paradigm of the process.

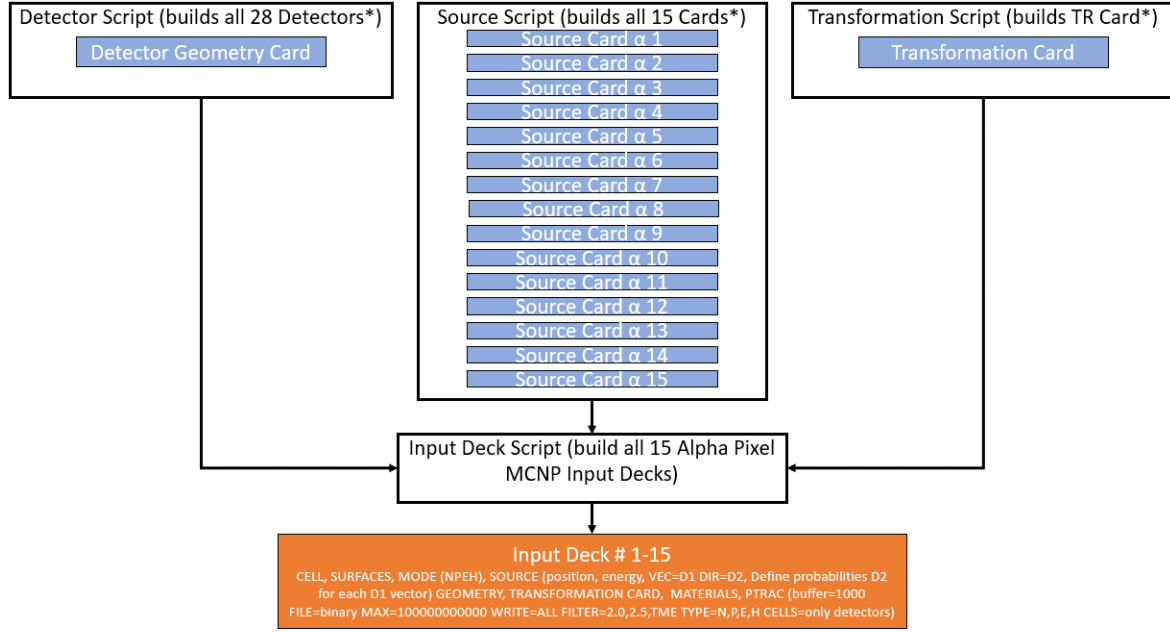


Figure 3.9. MCNP input deck generation flowchart

*Python 3.7.3 64-bit

First the Detector, Source, and Transformation Scripts are run for the simulation. These build the requisite files for the Input Deck Script (Figure 3.9). This deck builds the requisite directories and compiled deck for each of the 15 α pixels. Running MCNP generates an output PTRAC file. The post processor script finally tallies counts for each detector through each of the 15 PTAC files yielding the required data necessary to construct a radiograph (Figure 3.10). Only when a transmission measurement, N_i , and a void measurement, N_o , are post processed is the radiograph of the attenuation coefficient, μ , finally constructed for each detector pixel from:

$$\mu = -\ln \left(\frac{N_i}{N_o} \right) \quad (2)$$

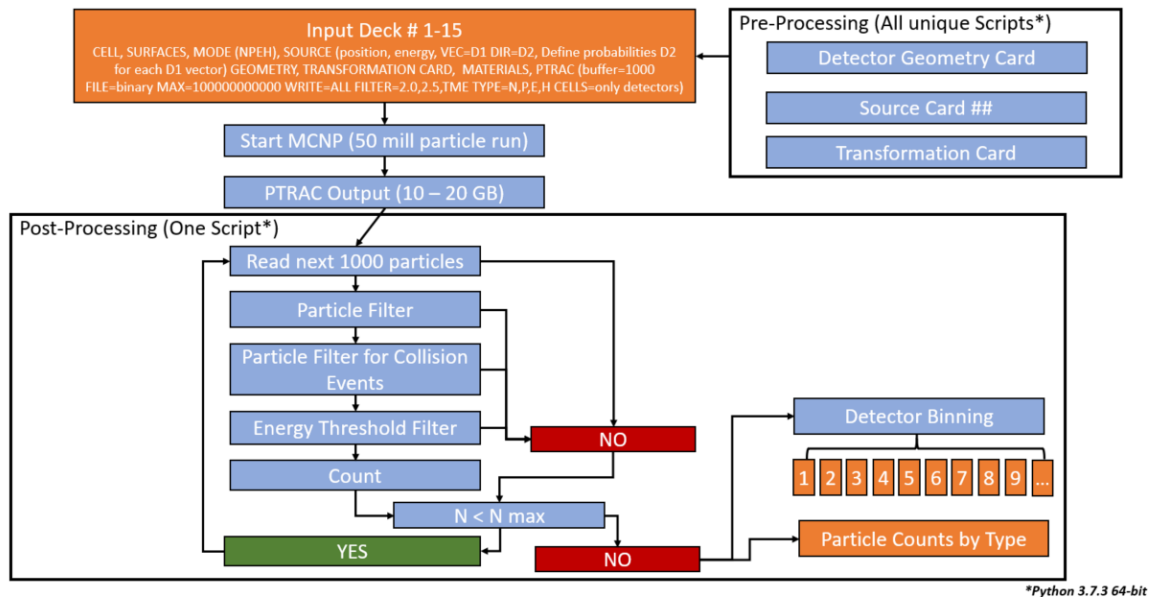


Figure 3.10. Ptrac post processing flowchart

World and Scintillator Generation

Environmental geometries are typically directly written into default decks for MCNP. In the case of the NMIS this included fission event detectors. The previous NMIS code placed considerable focus on generating 3D tomographic images. As such all detectors were always perfectly aligned in this code. The current code grants the user the ability to purposefully misalign these detectors and to include various other objects. For example, a steel box 5.0 mm in thickness was often added to scans as can be seen in Figure 3.11 represented by the maroon box.

FNIS code accomplishes this using multiple Transformation (TR) cards. Each TR card can be used to reposition and/or rotate previously positioned in the MCNP world geometry (see Figure 3.11). Each card relies on a transformation matrix to accomplish this. The transformation matrix is first calculated in the Transformation Script and a

resulting output card is created. This allows for complicated geometries such as curved arrays to be modeled relatively quickly and efficiently.

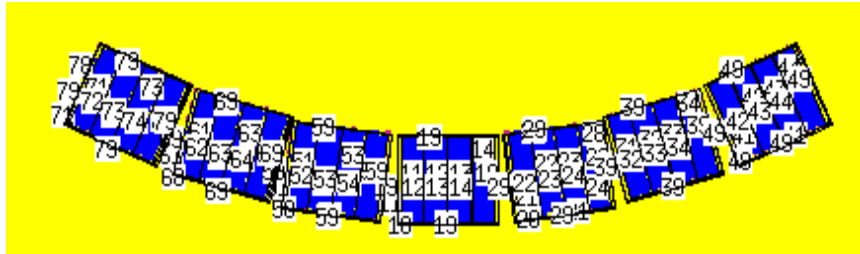


Figure 3.11. Detector geometry verification

In Figure 3.11 one can see the power of the TR card in use. From the central box (cells 11-19) 6 additional TR cards are applied each to create an additional detector box (cells 21-79). For rotation of the entire arm around a central point the universe card is utilized in MCNP. This involves a few additional notations for each cell defining to which universe it resides. This essentially results in a cocoon world (see Figure 3.12). The cocoon world is then rotated within the physical world resulting in the required misalignment.

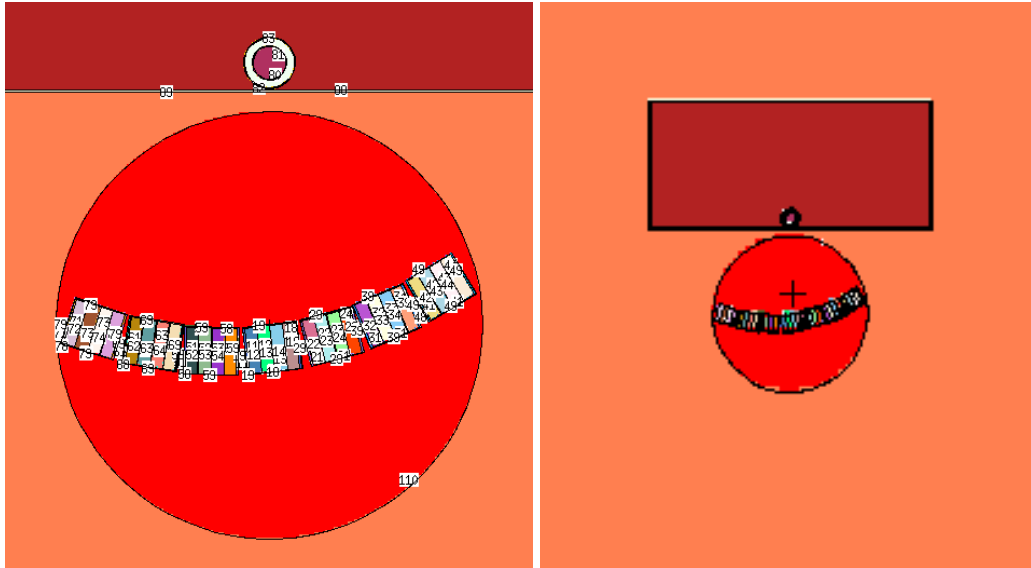


Figure 3.12. MCNP sub-level universes

In Figure 3.12 one can see this cocoon world represented as by the red disk. All objects within this red disk (physically a sphere) universe receive the same transformation. This allows for multiple levels of transformations to be applied to an object with differing origins for each transformation. There is no physical impact from the additional universe other than the transformation. Typically, this tool is utilized to fill a repeat structure within a reactor core or other object with a repeating pattern but was required in this work for rotational offsets.

Source Methodology

Extensive characterization has been collected for both the ING-27 and API-120 DT generators. Based upon the collection of data from these characterizations the best fit found to model their detection capabilities is a double gaussian probability distribution function (PDF). This PDF originally developed in 2008 [35] has been used extensively since and is the standard method used for NMIS imaging simulation. The same source

from NMIS code was recreated for this simulation. In the standard two-dimensional Cartesian version of the PDF, a double gaussian is formed from

$$P(x, y) = A_1 \exp\left(-\left(\frac{(x-x_\mu)^2}{2\sigma_{x,1}^2} + \frac{(y-y_\mu)^2}{2\sigma_{y,1}^2}\right)\right) + A_2 \exp\left(-\left(\frac{(x-x_\mu)^2}{2\sigma_{x,2}^2} + \frac{(y-y_\mu)^2}{2\sigma_{y,2}^2}\right)\right) \quad (2)$$

where $A_1, A_2, x_\mu, y_\mu, \sigma_{x,1}, \sigma_{x,2}, \sigma_{y,1},$ and $\sigma_{y,2}$ all have been characterized for the API-120.

The ING-27 model originally reconstructed in 2015 during the beam characterization was more rudimentary. The characterization fitted a single gaussian to the correlation detector count peaks dropping the second exponential:

$$P(x, y) = A_1 \exp\left(-\left(\frac{(x-x_\mu)^2}{2\sigma_{x,1}^2} + \frac{(y-y_\mu)^2}{2\sigma_{y,1}^2}\right)\right) \quad (3)$$

This fit is accurate to approximately 5% of the maximum count. At 5% the gaussian diverges from the data (see Figure 3.13). This divergence is modeled in the second gaussian in the API-120 model but was not in the ING-27 original characterization. Below one can see the gaussian fit overlaid on α pixel 7.

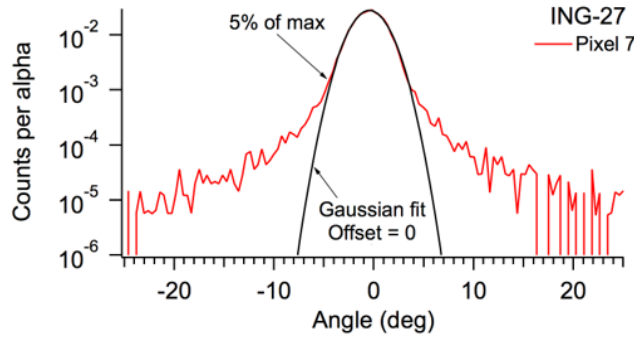


Figure 3.13. Single gaussian fit to the correlated count distribution for ING-27 α pixel 7. [37]

Seeking to improve upon this characterization, a second gaussian was incorporated to the fit for the simulations conducted in this study. The same data used for the initial beam characterization was still available – thanks to Dr. Seth McConchie. The characterization utilized an older NMIS setup of 32 detectors. The detectors were then translated in $\frac{1}{2}$ detector width translation (1.27 cm steps) four times for a total of 128 detector measurements. Unfortunately, only 13 of the 15 α pixels were captured during this characterization as the last two alpha’s profiles fell outside the collection area of the NMIS (see Figure 3.14).

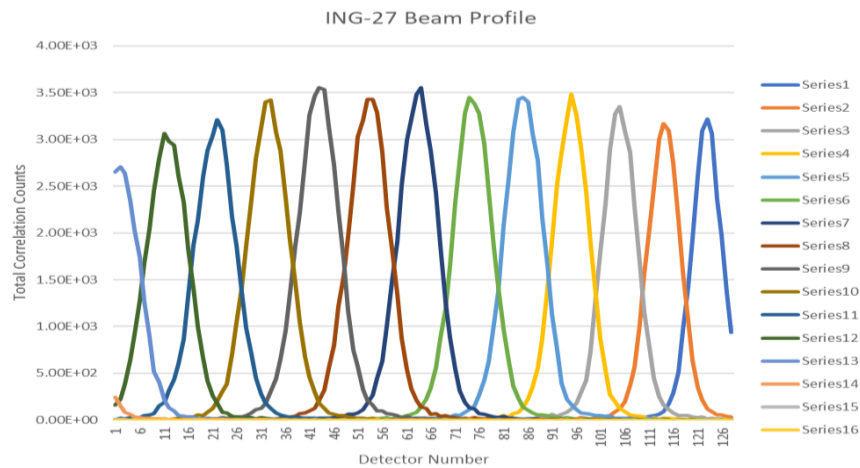


Figure 3.14. α pixel profiles for α pixels 1-16

As one can see in Figure 3.14, α pixel series 13 was only partially captured during the study. No meaningful data was collected on α pixels 14 or 15 and the signal read into α pixel 16 was only static as the ING-27 only has α 15 pixel port detectors. This did not present issue as the modular detector arm also did not typically collect counts from these two α pixels as they also fell outside the range of its detectors.

Another means of understanding the α pixel profile is to view correlation counts as a function of time bins per detector- α pixel combination. Each of the cells in the image below corresponds to a 100 ps time bin's counts (see Figure 3.15). Viewing the profiles in such a manner may allow future neural networks to determine misalignment. Arrival time of neutrons, peak profiles, scattered neutron counts, and overall intensity are some of the properties captured in this manner that are affected by alignment. These affects are beyond the scope of this work and should be considered for future research.

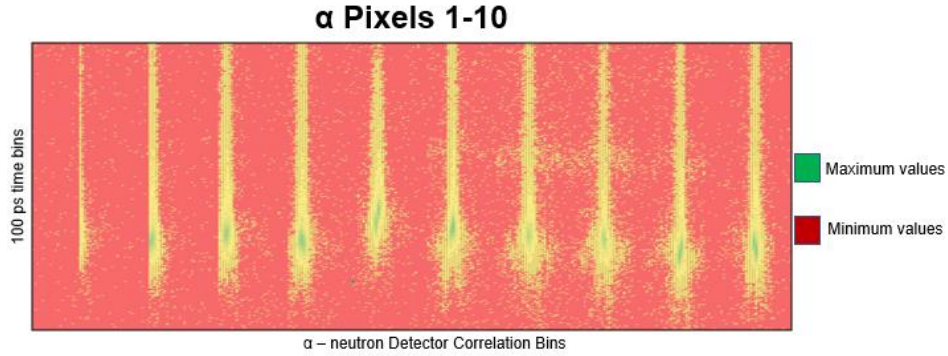


Figure 3.15. 100 ps arrival time bins for α pixel profiles 1-10

Each of the double gaussians was then fit using Tikhonov regularization for the minimizing the error vector in the l2 norm – otherwise known as least squares regression. This double gaussian followed the same form as the API-120 beam where $A_1, A_2, x_\mu, y_\mu, \sigma_{x,1}, \sigma_{x,2}, \sigma_{y,1},$ and $\sigma_{y,2}$ all were optimized to minimize the residual sum of squares (RSS):

$$P(x, y) = A_1 \exp\left(-\left(\frac{(x-x_\mu)^2}{2\sigma_{x,1}^2} + \frac{(y-y_\mu)^2}{2\sigma_{y,1}^2}\right)\right) + A_2 \exp\left(-\left(\frac{(x-x_\mu)^2}{2\sigma_{x,2}^2} + \frac{(y-y_\mu)^2}{2\sigma_{y,2}^2}\right)\right) \quad (4)$$

Below is the resulting fit of α pixel 7 in Figure 3.16. This pixel's peak and tails were fully captured during the characterization. One can see the impact the second gaussian has on the profile for data below 1% of the peak. This physically manifest itself in detector counts across all detectors during measurements – given a sufficiently large number of particles simulated.

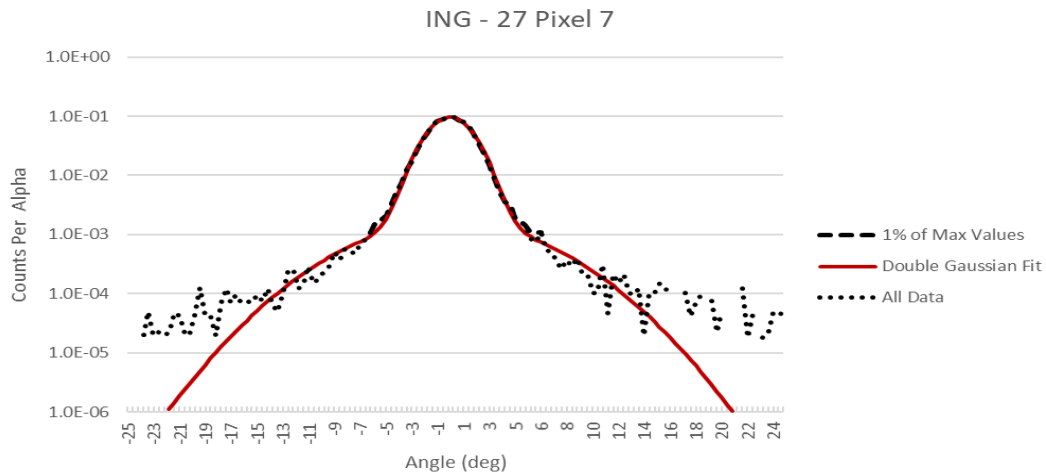


Figure 3.16. Double gaussian fit to the correlated count distribution for ING-27 α pixel 7

A Pixel beams 14 and 15 were unable to be fit in this manner due to a lack of data collection. A Pixel 13 was only partially captured – however an approximation was made from the data and used during simulations (see Figure 3.17). Thankfully much of the peak was captured during the characterization of the ING-27 as visible in the fit below. In all α pixel profiles, the gaps in data correspond to zero count tallies collected during the characterization.

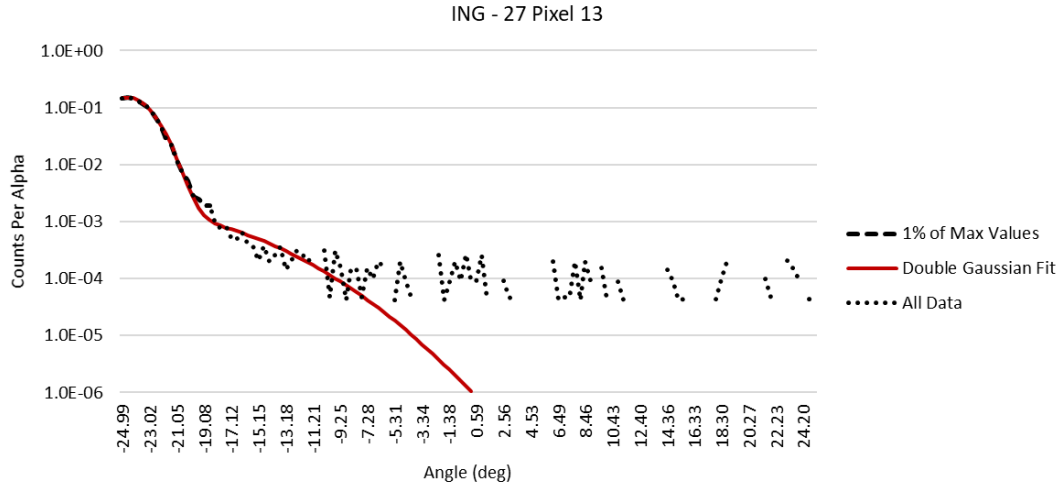


Figure 3.17. Double gaussian fit to the correlated count distribution for ING-27 α pixel 1

Each MCNP Input file models one of these α pixel profiles. Each α pixel profile generates a peak. Over the course of the 13 characterized profiles this peak translates along the orange path highlighted in Figure 3.18. The next step is collection and interpretation of transmission data. Scintillation light is collected within each of the 28 detectors according to Chou's expansion of Birks' Law.

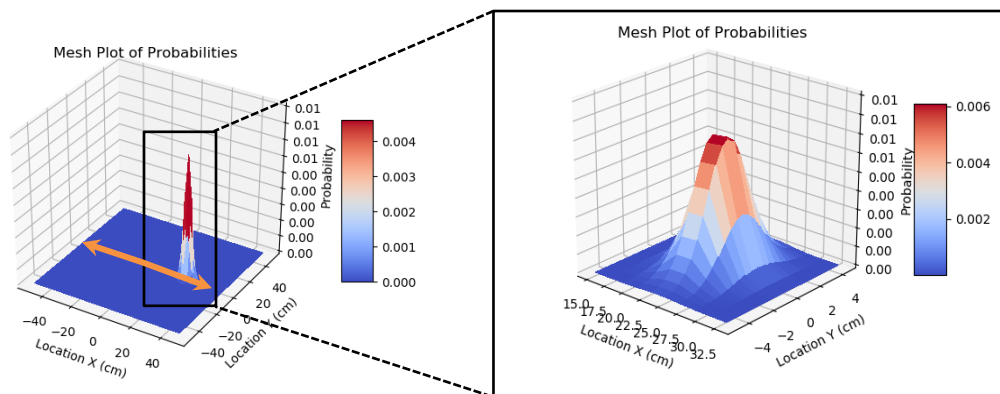


Figure 3.18. Double gaussian translation of normalized probabilities as a function of α pixel I

Light Yield Methodology

Since the probability of neutrons directly generating any photons is extremely low, neutrons are indirectly measured. This is in part due to the extremely small cross section for the (n,γ) reaction with hydrogen and other cross sections which may result in the release of γ's. Neutrons primarily impart kinetic energy to the charged particles through elastic scattering with hydrogen nuclei within the scintillator as previously discussed in the section *Neutron Interactions with Matter*. This is visually evident in the plots of the following cross sections of each reaction's cross sections (see Figure 3.19).

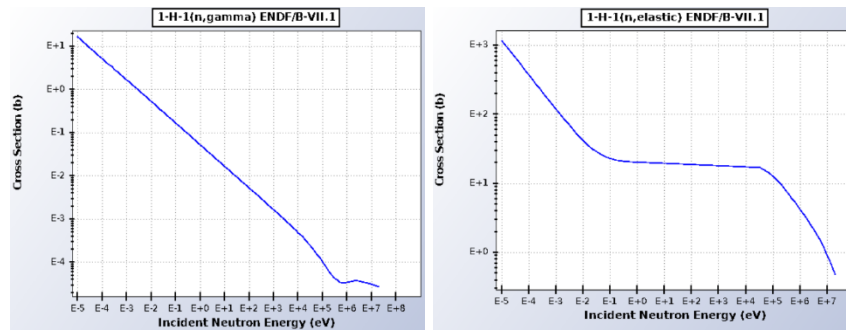


Figure 3.19. (Left) $^1\text{H}(n,\gamma)^1\text{H}$ reaction cross sections and (right) $^1\text{H}(n,\text{elastic})^1\text{H}$ cross sections [38]

The first to measure ionizing radiation and posit a relationship between the resulting fluorescence produced as a result in photo-multiplier tubes was J.B. Birks in 1951. In his paper he found:

$$\frac{dS}{dr} = \frac{A \frac{dE}{dr}}{1 + k_B \frac{dE}{dr}} \quad (5)$$

where $\frac{dS}{dr}$ is the light yield, A is the scintillation efficiency, $\frac{dE}{dr}$ is the energy loss of the particle per path length, k_B is Birks constant – material dependent. When integrated over

the entire energy deposition of the charged particle, one arrives at Birks' Semi-Empirical Formula:

$$L(E) = L_o \int_0^E \frac{dE}{1 + kB \frac{dE}{dx}} \quad (6)$$

where the number of excitons produced per unit path length is proportional to the specific energy loss $\frac{dE}{dx}$, E is the energy of the charged particle in MeV, L_o is the light yield constant in units of photons. As these excitons are produced local damaged molecules are left which reduce the light yield. The local concentration of damaged molecules at any point on the particle track in this case is proportional to the specific energy loss, $kB \frac{dE}{dx}$, where the values kB are computed from observations on the relative response for each material for each charged particle [39]. Chou expanded this description and included one additional expansion of the series, C to account for different modes of excitation, resulting in the following [40]:

$$L(E) = L_o \int_0^E \frac{dE}{1 + kB \frac{dE}{dx} + C(\frac{dE}{dx})^2} \quad (7)$$

Since then numerous studies have been conducted to find the values of C and kB for a host of materials given the differing stopping powers, $\frac{dE}{dx}$, of protons. The values for the EJ-200 detector used were $L_o = 1541 \text{ photons}$, $C = 9.29 \times 10^{-6} \text{ g/MeVcm}^2$, and $kB = 2.2 \times 10^{-2} \text{ g/MeVcm}^2$ [39]. With the differing stopping powers of protons one can determine threshold energies for each particle based upon the stopping power available from the NIST Pstar and Estar libraries [41].

Previous NMIS operators typically seek to set this threshold at 1MeV per proton and the author kept this threshold in this work. This threshold is an essential component

of PTRAC analysis and may be adjusted in the code should the future need occur. For instance, if one wished to calculate the equivalent light yield for electrons using Chou's Equation the threshold is 1.307 MeV.

PTRAC Analysis Methodology

The PTRAC post processor is a python script designed to parse through each of the 15 output files (PTRAC) generated by MCNP. The post processor examines each PTRAC file and only counts interactions where this threshold energy deposition is satisfied. The outputs are then summed for each detector and summed over all 15 output file decks to yield a final count per detector which is then used to create a radiograph. For a sense of scale each of these PTRAC files is approximately 1GB. Each simulation generates over 15GBs of data that must be processed and compared to the 15GBs of data from the void measurement before any insightful conclusions can be reached. PTRAC analysis typically takes 1 – 2 hours on an Intel i9-8950HK processor running at 2.90GHz.

The post processor accomplishes this threshold winnowing using the MCNP Tools package extension for python. This extension only properly installs and functions on windows using visual studio code according to the MCNP6.2 user guide [42]. In the post processor one can then call the requisite functions from MCNP tools to accomplish a desired task. In this case only count neutron-proton and neutron-carbon collision events where energy lost to the charged particle in the event exceed 1.000 MeV within a detector for protons and 8.3 MeV for carbon atoms. The threshold is set at 8.3 MeV as the frequency of carbon-alpha reactions become significant at this point – for 8.3 MeV is the threshold for the $^{12}\text{C}(n, n')3\alpha$ reaction.

The result is the exclusion of most $^{12}\text{C}(n, n')^{12}\text{C}$, $^{12}\text{C}(n, n')^{12}\text{C}(L1)$, and $^{12}\text{C}(n, n')^9\text{Be}$ reactions from possible counts (see Figure 3.20). Exclusion of all reactions not of interest only becomes possible with light and heavy ion transport are enabled in MCNP.

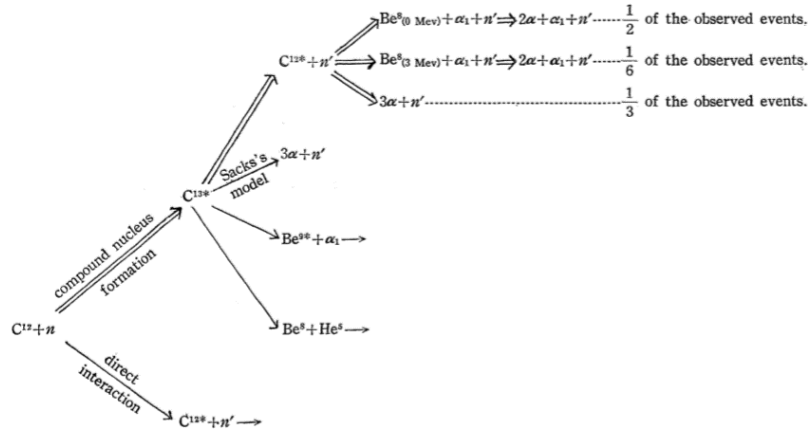


Figure 3.20. $^{12}\text{C}(n, n')$ induced Reactions by 14.1 MeV Neutrons [43]

Previous simulations have not modeled heavy ion transport and this work remained consistent only modeling neutron, proton, photon and electron interactions.

Computational time and PTRAC file size increase linearly in size as new particles are added to transport. Furthermore, when the magnitude of the light yield is included the dominant mechanism for detector remains elastically scattered protons ($> \sim 95\%$) [44].

Table 3.1 list the possible interactions for 14.1 MeV neutrons inside the detectors. Note the multiple α reactions possible given 14.1 MeV neutrons.

Table 3.1. Reactions of interest for a plastic scintillator at neutron energies below 15 MeV [45] [44]

Reaction	Q-val. (MeV)	E _{Th} (MeV)	Σ _{2,45} (cm ⁻¹)	P(Σ _{2,45}) (%)	Σ _{14,1} (cm ⁻¹)	P(Σ _{14,1}) (%)
¹ H(n,n) ¹ H	0.00	-----	2.58	64.3	0.683	38.1
¹² C(n,n) ¹² C	0.00	-----	1.59	35.7	0.827	41.6
¹² C(n,n') ¹² C (L1)	-4.40	4.81	-----	-----	0.210	10.6
¹² C(n,n')3α (L2)	-7.65	8.30	-----	-----	1.93e-2	1.0
¹² C(n,n') 3α (L3)	-9.64	10.4	-----	-----	6.66e-2	3.4
¹² C(n,n') 3α (L4)	-10.8	11.7	-----	-----	2.00e-2	1.0
¹² C(n,n') 3α (L5)	-11.8	12.8	-----	-----	1.20e-2	0.6
¹² C(n,n') 3α (L6)	-12.7	13.8	-----	-----	6.52e-3	0.3
¹² C(n,α) ⁹ Be	-5.70	6.18	-----	-----	6.99e-2	3.5

If there is more than one count recorded in each history a second function in the script deletes these additional counts. After all, only one detection is physically possible for each history in each detector.

An additional script was written which conducts a particle by type count for all counts. This check was added as a quality control step and is normally not called. However, during debugging it may be of use. If everything is functioning properly, one should see interactions with only: protons and carbon atoms in our detectors as MCNP6.2 does not model neutron decay.

$$n^0 \rightarrow p^+ + e^- + \bar{\nu}_e \quad (8)$$

Given the probabilities from Table 3.1 we expect to see 85.8% of interactions being ¹H(n,n') ¹H with the remaining 14.2% split across the remaining ¹²C(n,n')3α reactions and ¹²C(n,n')¹²C elastic scatter. The results of the check script are consistent with these expected values typically within ±2%. Figure 3.21 exhibits two examples with values of 16.6% and 15.2% of results classified as carbon recoil interactions [12]:

Line	Particle Type	Count
1	Neutron	0
2	Photon	0
3	Electron	0
4	Positron	0
5	Proton	223639
6	Carbon recoil	37777
7		

Line	Particle Type	Count
1	Neutron	0
2	Photon	0
3	Electron	0
4	Positron	0
5	Proton	157895
6	Carbon recoil	28199
7		

Figure 3.21. Ptrac Parser Total Count Tracker

This process is repeated for all 15 α pixel MCNP output PTRAC files if the API-120 generator is used and for all 13 α pixels when the ING-27 generator is used. A final summation script (Appendix: H – Detector Counts Summation Code) records the counts for each α pixel detector pair and outputs a .csv file of the results. This step is done to convert the resulting output format of the simulation matches the output format of the laboratory results.

Radiograph Reconstruction

Once the final detector counts have been collected only then can a radiographic image be reconstructed. First the detector counts from the scenario, I_1 , are compared to the previous void measurements, I_0 . From these two values the attenuation value, μ , is determined using the following:

$$\mu = -\ln \left(\frac{I_1}{I_0} \right) \quad (9)$$

The attenuation value, μ , is the combination of the total macroscopic cross section of the material, Σ_t and its thickness, x . This originates from the form of:

$$I(x) = I_0 e^{-\Sigma_t x} \quad (10)$$

$$I(x) = I_0 e^{-\mu} \quad (11)$$

The attenuation value is left in this relatively vague form as the material and thickness of the target are generally unknown. In this form a constant method of characterization remains available to the inspector. Should the material in question be known thickness can be easily determined via substitution of material properties for Σ_t . With the attenuation value for each of the 28 detectors calculated the radiograph is finally reconstructed (see Figure 3.22).

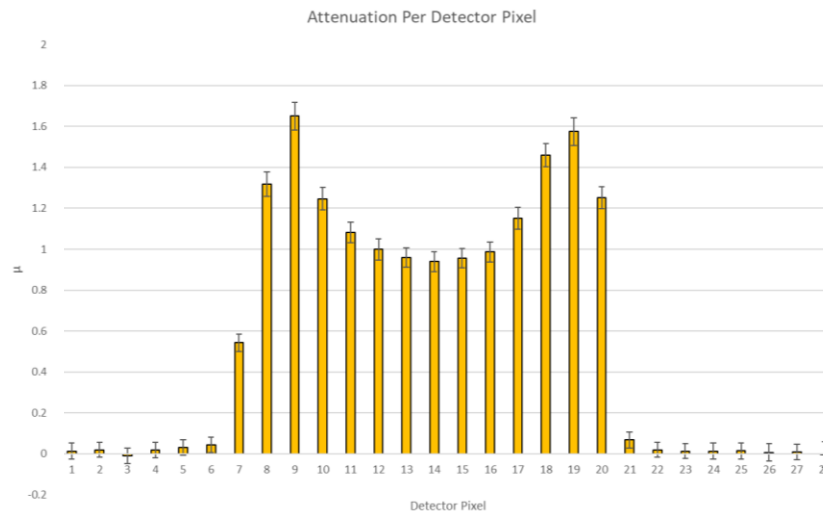


Figure 3.22. Simulated Steel Attenuation Profile

Detectors with a high attenuation value infer a region difficult for neutrons to penetrate. Typically, this is interpreted as a filled space. Conversely low attenuation values signify void space or materials with low absorption or scattering cross sections. With proper training and some cursory understanding of the interrogation object. If x-ray imaging is available to give dimensions geometry deductions may be accurately assessed

CHAPTER FOUR

RESULTS AND DISCUSSION

Prior to presenting the results of experiment and simulation the layout of this chapter must be presented. First the results of simple geometries will be exhibited before more complex objects. Each section of this report will focus on a tranche. In each of the five tranches the experimental results collected at ORNL will be presented first. Each experimental measurement will then be discussed. A number of experiments will be followed by the presentation of simulation results. The two results will then be compared in a corresponding discussion.

Five tranches of experiments were conducted over a period of five months. Each tranche examined the impact of changing one variable of alignment. Prior to all experiments one 10-minute void measurement was collected each day of experimentation to find N_0 . The canisters imaged in these experiments were constructed of a core ring 2.0 cm in thickness with an inner radius of 4.25 cm – composed of various materials listed. The core ring was encased in 2 mm thick steel walls on both the inner and outer radial walls of the canisters. All canisters were 16.5 cm in height except the DU canister which rose to a height of 18.3 cm (see Figure 4.1). All images were collected at a height of $\frac{1}{2}$ H of the canister.



Figure 4.1. Material canisters made of depleted uranium (DU), tungsten, lead, steel, aluminum, and polyethylene, pictured from left to right.

Initial Steel Canister

Prior to this initial measurement, software functionality tests were conducted. The initial measurement of a steel canister was conducted to better understand the ideal circumstances of the system. Resolution for the FNMIS, and all other API systems, is a direct function of the number of detectors an object covers. Thus, maximum resolution of an object would be achieved if the object covered all 28 detectors of the FNMIS.

However, this is not an experiment one would execute, as it does not allow for any error in misplacement of the object. In order to ensure the object is properly aligned and to observe the impact of air in void space, typically two or more detector paths on each side of the assembly are left unimpeded by the test object. There is also useful misalignment information stored in these void detectors, which is further discussed in the *Translational and Rotational Measurements* section of this work.

In this first measurement, the center of the steel canister was placed 30 cm from the face of the ING-27. This was then compared to a void measurement (N_o) in order to generate an attenuation profile. The attenuation profile generated from the experiment is

presented in green – note all error bars are 3σ unless otherwise annotated in this work (see Figure 4.2).

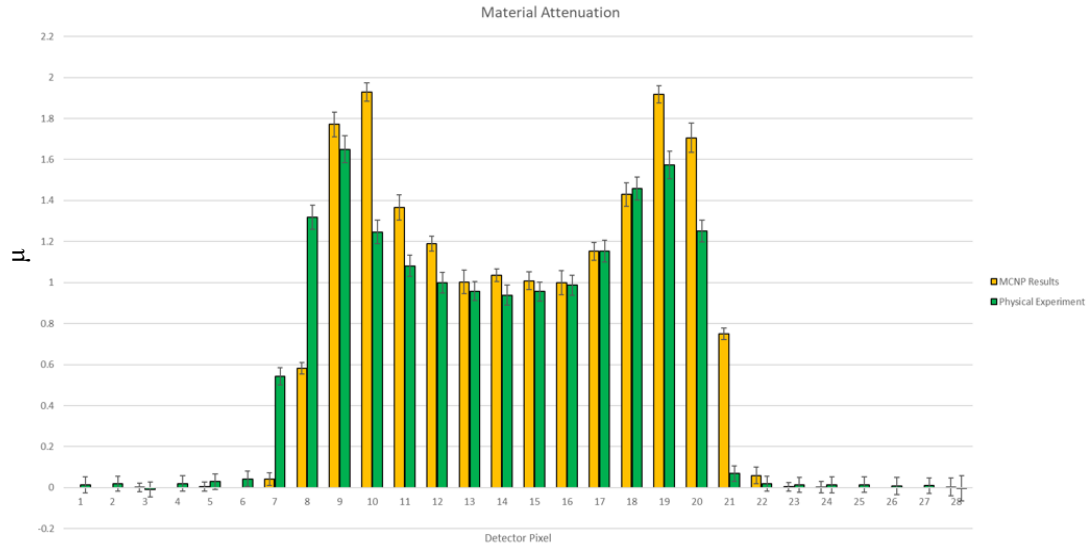


Figure 4.2. Initial steel attenuation profile

MCNP simulation results are plotted in orange. At the thickest part of the canister, the associated mean free path λ of steel (4.65 cm for 14.1 MeV neutrons given a density of steel of 7.82 g/cm³) would result in a value of 1.95 cm⁻¹ for μ . The MCNP simulation almost perfectly matches this result with a value of 1.93 $\mu \pm .04$ cm⁻¹. Note that $(\mu)(x)$ is unitless as λ has associated units of cm⁻¹ and distance x has units of cm¹.

The physical experiment did not achieve similar accuracy despite analogous precision. At maximum thickness, values of 1.65 $\mu \pm .07$ cm⁻¹ for the left and 1.57 $\mu \pm .07$ cm⁻¹ for the right were achieved – compared to the theoretical value of 1.95 cm⁻¹. The difference between the theoretical expected value and the physical experiment resulted from the impact of numerous variables including but not limited to environmental reflection, lack of a full characterization of the ING-27 beam profile,

difficulty in alignment of the DT generator, lack of atmospheric attenuation, and physical discrepancies in the construction of the arm as it is not perfectly symmetric.

Despite shortcomings, both images clearly identify a drop in μ for the canister. Both also correctly estimated the thickness of the walls of the canister at approximately 4.65 cm or 1λ of a 14.1 MeV neutron in 7.82 g cm^{-3} steel. When physical measurement and simulation are compared, only three detectors per side lay outside of 3σ overlap of each other.

Equidistant Measurements – Steel, Aluminum, Polyethylene, & Depleted Uranium

Following the success of first experiment the subsequent tranche of experiments measured less than ideal object locations. The interrogated objects were placed equidistant from the arm and the DT source and moved through 0 cm $\hat{x}$, +5 cm $\hat{x}$, and -10 cm $\hat{x}$ offsets. The coordinate reference frame is pictured below. The point source for 14.1 MeV neutrons is annotated by the red cross at 0 cm x, 115.1 cm y, 0 cm z. The interrogated object was placed midway along the y axis at 57.55 cm (teal ringlet) to allow for inclusion of future geometries that might encase the imaging target of interest such as a 5 mm steel wall pictured below. Figure 4.3 depicts these locations with transparent ringlets displaying offset measurement locations.

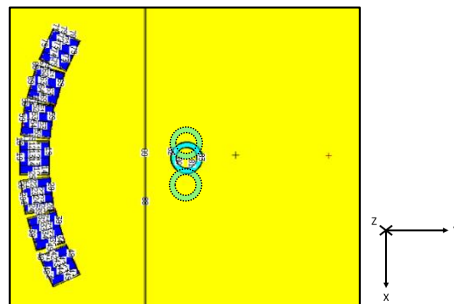


Figure 4.3. MCNP top down view of detectors and interrogation object

The result of this centered, steel canister measurement is reflected in Figure 4.5. All MCNP simulation results are presented in orange and physical results are annotated in green. Note the near perfect symmetry of the simulation results in comparison to the skew of the physical measurement. The skewness is the result of the α pixel beam profile offset of the DT generator as displayed below in the ING-27 manual (see Figure 4.4).

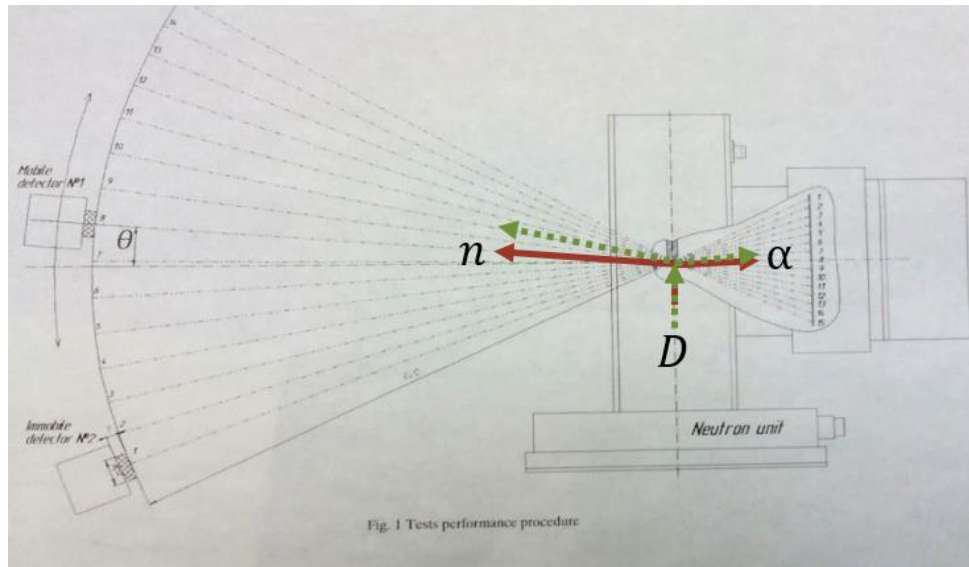


Figure 4.4. ING-27 DT generator offset of central alpha pixel

This is the direct result of the incident ion beam's forward momentum on the target inside the neutron generator incident onto a 45° target with a spot size of 5 mm. As the voltage of the ion beam is increased this forward momentum of the deuterium in the laboratory reference frame skews this offset further (dotted green vs. red path in Figure 4.4). Physically, the DT generator and detector arm are aligned; however, for imaging purposes a slight misalignment would be required for perfectly symmetric imaging as there is a slight offset of this central α pixel from the physical center of the neutron

source. This skew is most evident in detectors 12 and 13 where μ is first underestimated and then overestimated in the following detector (see Figure 4.5).

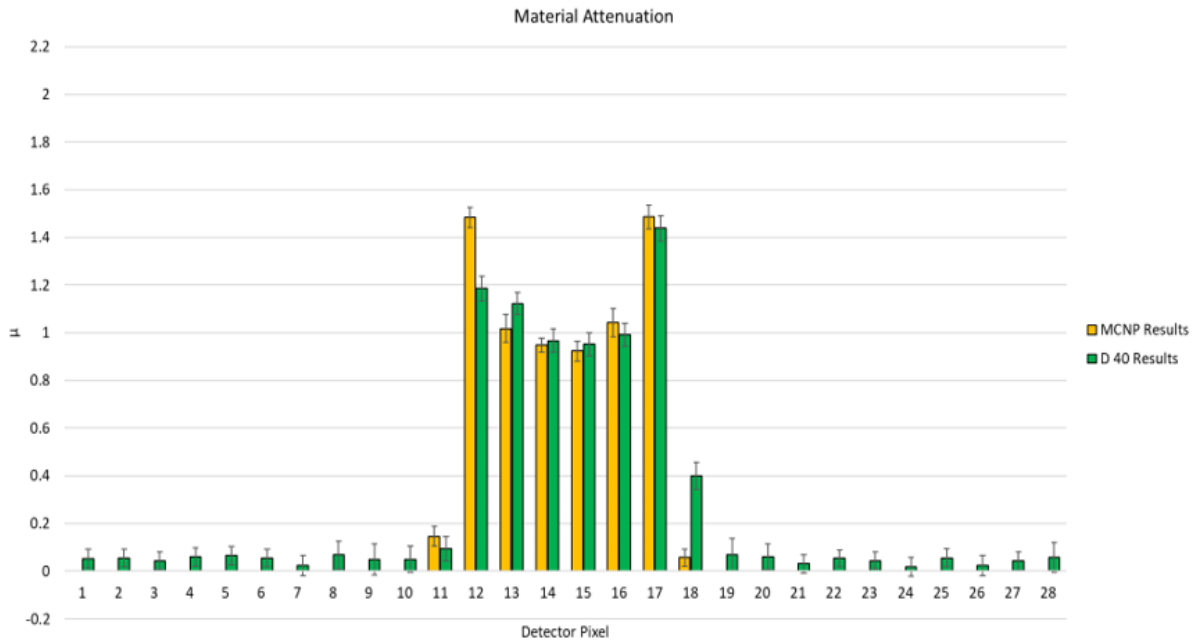


Figure 4.5. DT-detector equidistant, centered steel attenuation profile

Next are the results of the +5cm $\hat{x}$ and -10cm $\hat{x}$ target offsets for steel (see Figure 4.6). Each image continues to reproduce a slight skew with the clearest symmetrical void space evident in the +5cm offset. Lack of proper alignment is discernable through examination of which detectors spot the object. Proper alignment will result in a symmetrical split between detectors 14 and 15. Should the detections shift to higher numbered detectors this implies the object is too far to the right (+ $\hat{x}$ direction) out of alignment. Shifts to the left imply the object has been shifted left (- $\hat{x}$ direction) out of alignment. This pattern continues throughout all materials imaged in all tranches being the most useful insight gained in this tranche.



Figure 4.6. DT-detector equidistant, offset top (5cm) and offset bottom (10cm) steel attenuation profile

Aluminum Casting

Next to be imaged was an aluminum casting. The associated mean free path, λ , of aluminum is 9.547 cm for 14.1 MeV neutrons given a density of aluminum of 2.6989 g/cm³. This would result in a value of .995 cm⁻¹ for μ given a of ring thickness of 9.50 cm. The MCNP simulations match this result with values ranging from .992 $\mu \pm .007$ cm⁻¹ to .890 $\mu \pm .010$ cm⁻¹ (see Figure 4.7).

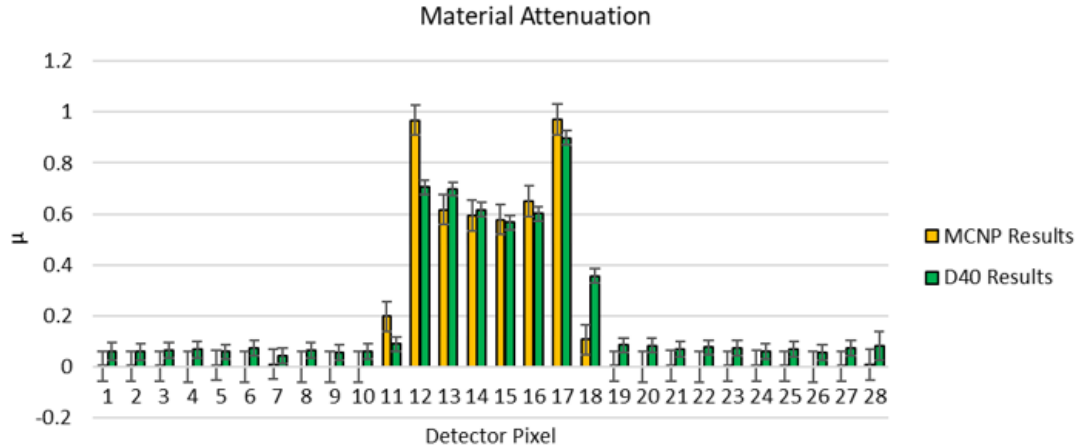


Figure 4.7. DT-detector equidistant, centered Al attenuation profile

The physical experiment achieves similar results to those of the MCNP model (value of $.960 \mu \pm .043 \text{ cm}^{-1}$) with disparity again at the left boundary of the canister ring (detector 12). This discrepancy was somewhat rectified in the +5cm $\hat{x}$ right offset and extenuated in the -10cm $\hat{x}$ left offset measurement (see Figure 4.8). Nevertheless, the overarching theme is consistent: the largest discrepancy between the two lays at edge location identification while void recognition is consistently accurate.

In addition, this offset position yielded consistently symmetric results throughout all trials leading to the extrapolation that the best alignment for imaging is not always a perfectly aligned system. Instead each arm will yield best results only after a similar series of measurements locates this ideal position. This position is dependent on the radius defining the curvature of the arm and the rotational alignment of the DT generators α pixels in relation to neutron detectors.

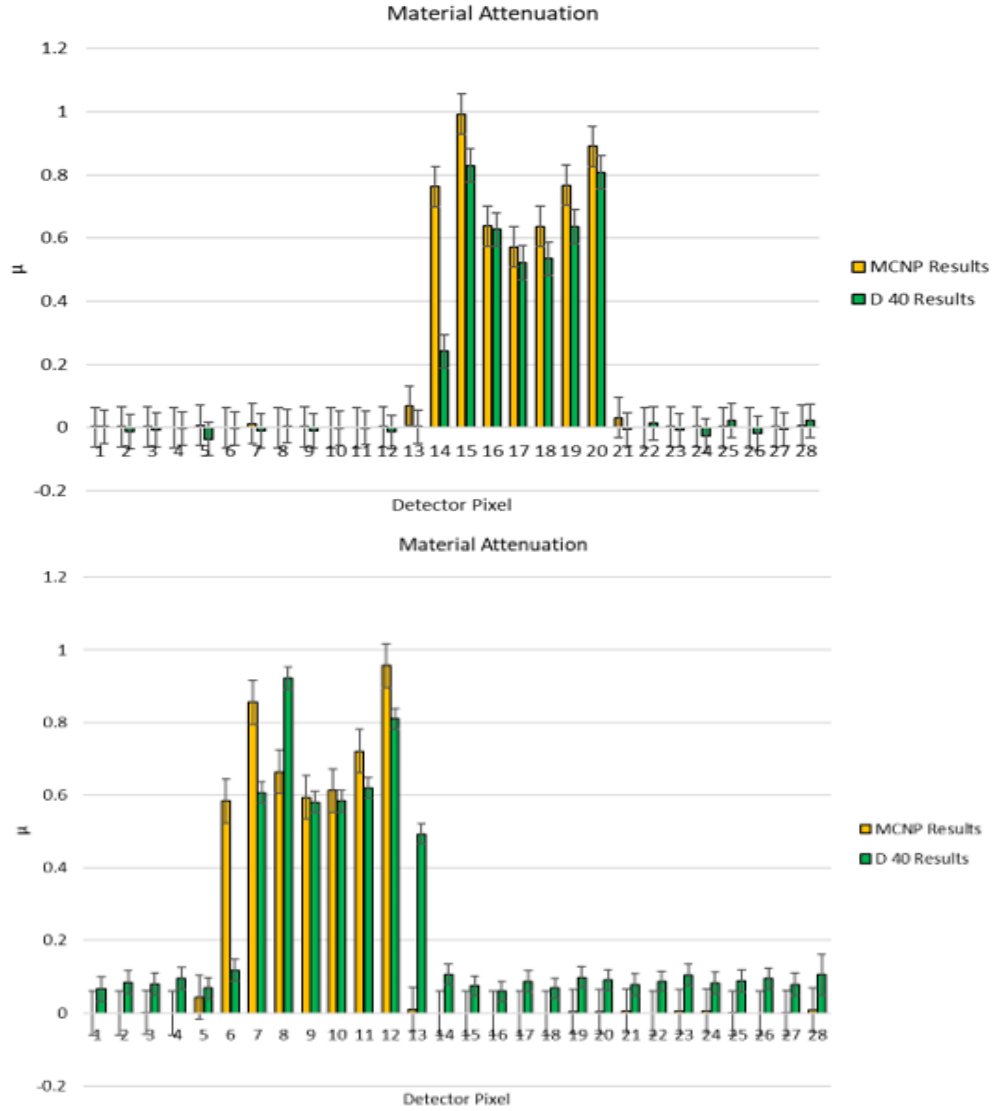


Figure 4.8. DT-detector equidistant, offset top (5cm) and offset bottom (10cm) Al attenuation profile

Polyethylene Casting

The subsequent tranche was a polyethylene casting. The associated mean free path, λ , of polyethylene is 8.411 cm for 14.1 MeV neutrons given a density of polyethylene of 1.032 g/cm³. This would result in a value of 1.129 cm⁻¹ for μ given a of ring 9.50 cm thick. The MCNP simulations fall short this result with values ranging from .986 $\mu \pm .007$ cm⁻¹ to .982 $\mu \pm .006$ cm⁻¹ (see Figure 4.9).

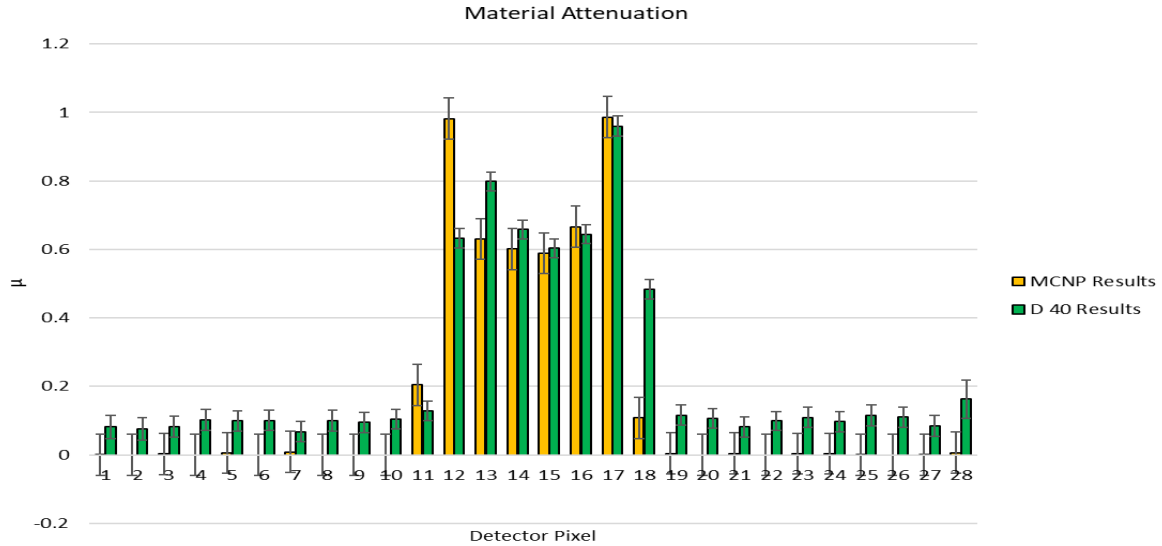


Figure 4.9. DT-Detector Equidistant, Centered Polyethylene Attenuation Profile

The physical experiment ($.960 \mu \pm .043 \text{ cm}^{-1}$ and $.802 \mu \pm .029 \text{ cm}^{-1}$) again achieved similar results to those of the MCNP model (values of $.986 \mu \pm .007 \text{ cm}^{-1}$ and $.982 \mu \pm .006 \text{ cm}^{-1}$) with disparity again at the left boundary of the canister ring. This discrepancy was rectified in Figure 4.10 for the +5cm $\hat{x}$ right offset and less obvious in the -10cm $\hat{x}$ left offset measurement. The theme is again consistent: the largest discrepancy between the two lays at edge location identification while void recognition within the target is consistently accurate. Generalized target misalignment can be inferred through which detectors are covered by the object.

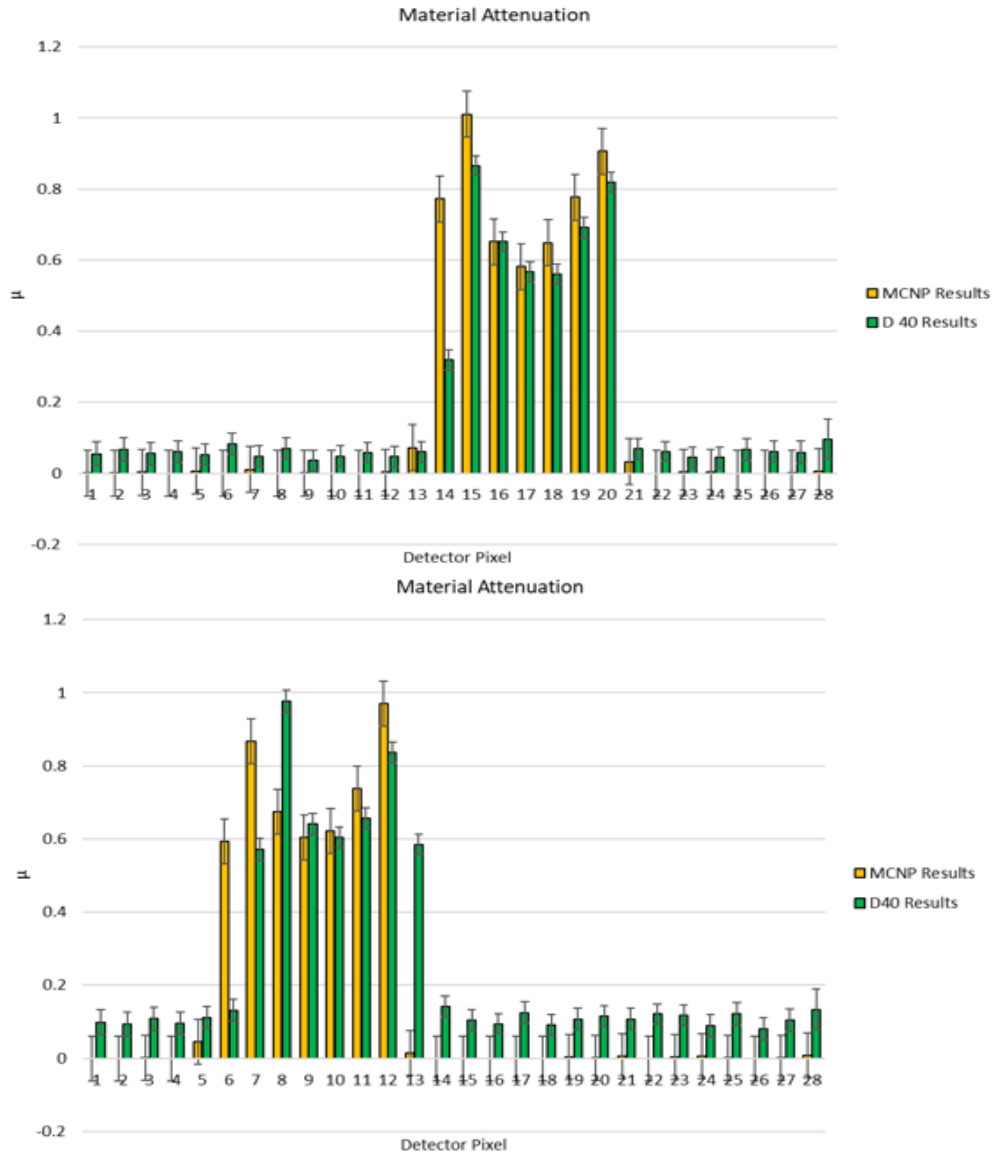


Figure 4.10. DT-detector equidistant, offset top (+5cm) and offset bottom (-10cm) polyethylene attenuation profile

Depleted Uranium (DU) Casting

The final single material imaged was a canister composed of DU encased in steel with the same dimensions of the before specified geometries with exception of the height of the object. The top and bottom on the canister were sealed with a caulking agent to prevent accidental chipping of the DU. For this reason, the steel casing extended 2 cm

above the height of the others. All canisters were imaged at $\frac{H}{2}$ so the impact of this additional steel was assumed minimal in its contribution to scatter neutrons arriving at the detectors inside the timing window and not modeled in MCNP.

The associated mean free path, λ , of DU is 3.57 cm for 14.1 MeV neutrons given a density of DU of 18.9 g/cm³. This would result in a value of 2.66 for μ given a ring 9.50 cm thick. At the center of the ring this would result in a value of 1.23 cm⁻¹ μ . The MCNP simulations fell below this theoretical value with results of $2.150 \mu \pm .007 \text{ cm}^{-1}$ and $2.071 \mu \pm .009 \text{ cm}^{-1}$ with physical experiment results of $(1.468 \mu \pm .052 \text{ cm}^{-1}$ and $1.228 \mu \pm .039 \text{ cm}^{-1})$. This result is the largest disagreement between the simulation and experimental data (see Figure 4.11).

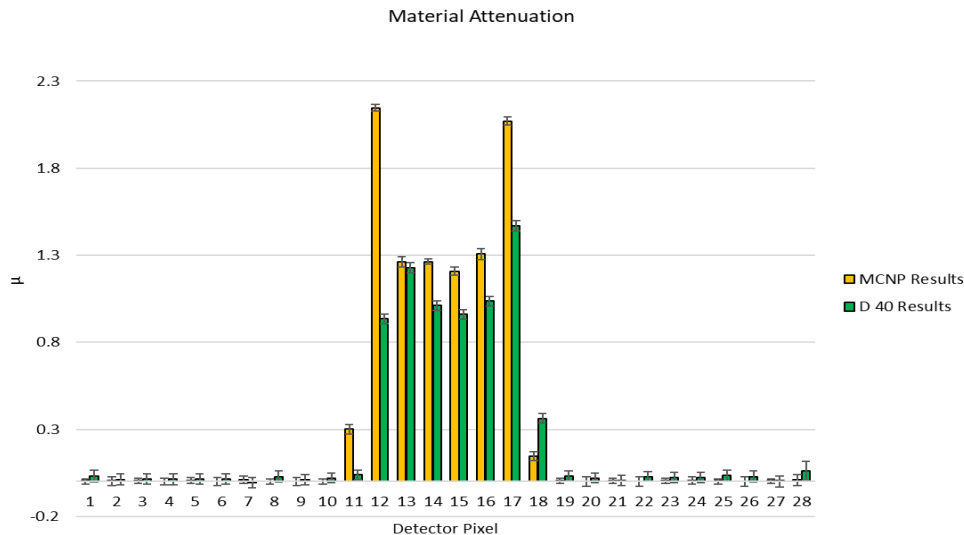


Figure 4.11. DT-detector equidistant, centered DU attenuation profile

The physical experiment underestimated μ with value of $1.468 \mu \pm .052 \text{ cm}^{-1}$ and $1.228 \mu \pm .039 \text{ cm}^{-1}$. It also displayed disparity again at the left boundary of the canister ring. This discrepancy was rectified in the +5cm $\hat{x}$ right offset and less obvious in the -

10cm $\hat{x}$ left offset measurement (see Figure 4.12). The theme continues: the largest discrepancy between the two lays at edge location identification while void recognition within the target is consistently accurate. Object identification is possible in all cases. A long exposure period or higher neutron flux should result in a more accurate estimation of μ . This was not attempted as 56 measurements already required substantial generator time on an aging device.

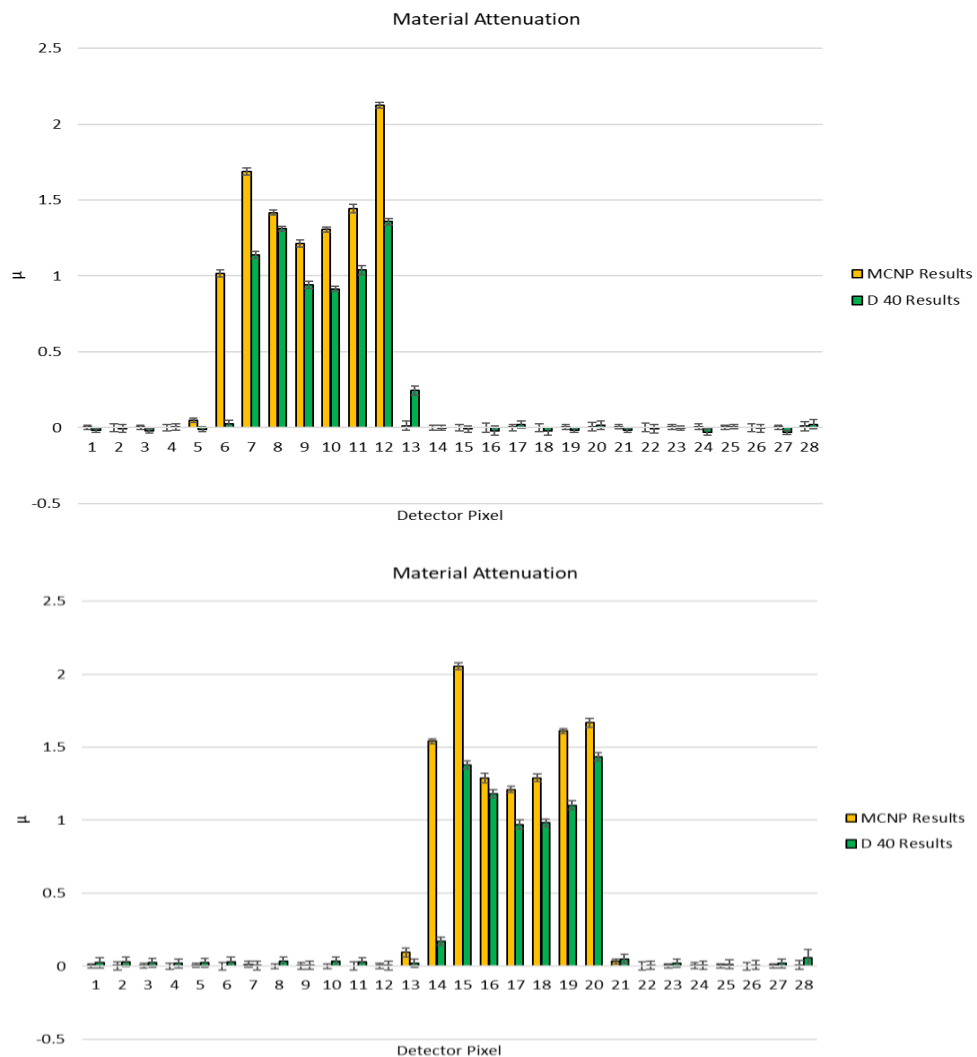


Figure 4.12. DT-detector equidistant, offset top (5cm) and offset bottom (10cm) DU attenuation profile

Translational Offset Void Measurements ($\pm X$ Axis)

The next tranche of measurements examined the impact of translational misalignment along the x axis in void measurements. In any fieldable measurement taking a perfectly aligned void measurement requires additional time in setup and exposure. Even after taking a void one will be unable to perfectly replicate the positioning of the system if repositioning is required to interrogate a target. X translation was the introduced as the second axis of freedom measured, in the laboratory, after rotational offset due to the additional difficulty in ensuring the correct offsets. The results are presented in this work first as they are more intuitive and allow additional insight to rotational offset measurements.

In each measurement the detector arm was moved either left or right ($-\hat{x}$ or $+\hat{x}$) relative to the DT generator by 5cm increments over 30cm total distance from -15cm to 15cm offset from center alignment (Figure 4.13).

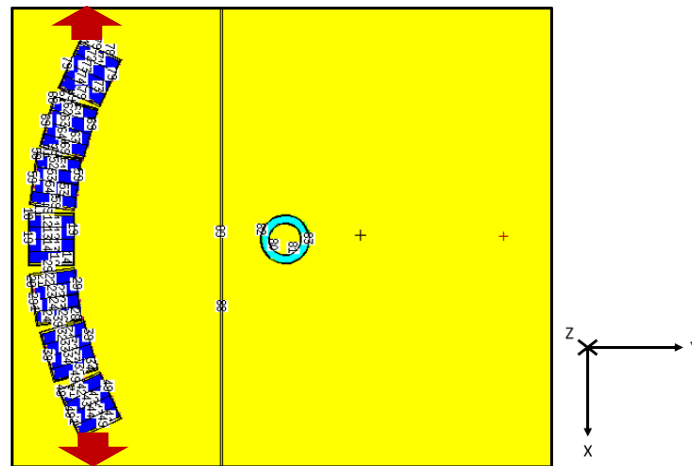


Figure 4.13. Translation X axis movement diagram. Detector arm in blue. Target in teal. Source in red cross.

The first measurement was taken centered at 0 cm x and has already been discussed in the void space comparison section of this report. Figure 4.14 depicts the 5 cm, 10 cm, and 15 cm offsets.

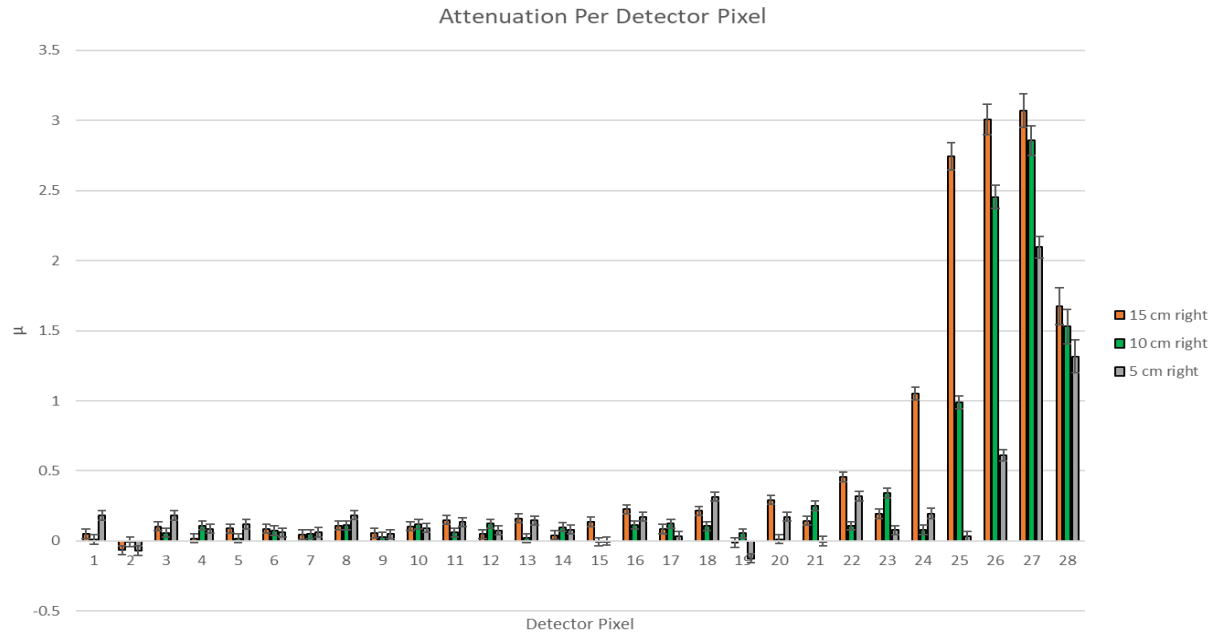


Figure 4.14. X detector translational offsets: 5 cm, 10 cm, and 15 cm

Note there is very little deviation across detectors 1 – 23 whereas detectors 24-28 experience significant reduction in counts. This reduction in counts can be attributed to the translation of the corresponding detectors out of the alpha pixel cone area of coverage. The expected number of detectors can be approximated by taking the detector width (2.54 cm) and dividing the distance offset to find dead detectors – yielding values of 6, 4, and 2 for each offset. The subsequent three left offsets -5 cm, -10 cm, and -15 cm do not follow the same pattern as the right offsets.

Instead the resulting counts and subsequent μ values remain relatively steady and consistent typically ranging from $+0.215 \mu \pm .034 \text{ cm}^{-1}$ to $-0.583 \mu \pm .056 \text{ cm}^{-1}$. This is

the result of α pixels 14 and 15 moving into range of the detector arm. Recall, detector 28 originally falls outside of α pixel 1's range. As it is moved left, it enters the range of α pixel 1 then α pixel 2 etc. This increase in counts vs. the void results in negative μ values in offsets -5 cm, -10 cm, and -15 cm. The results are presented below in Figure 4.15 for comparison.

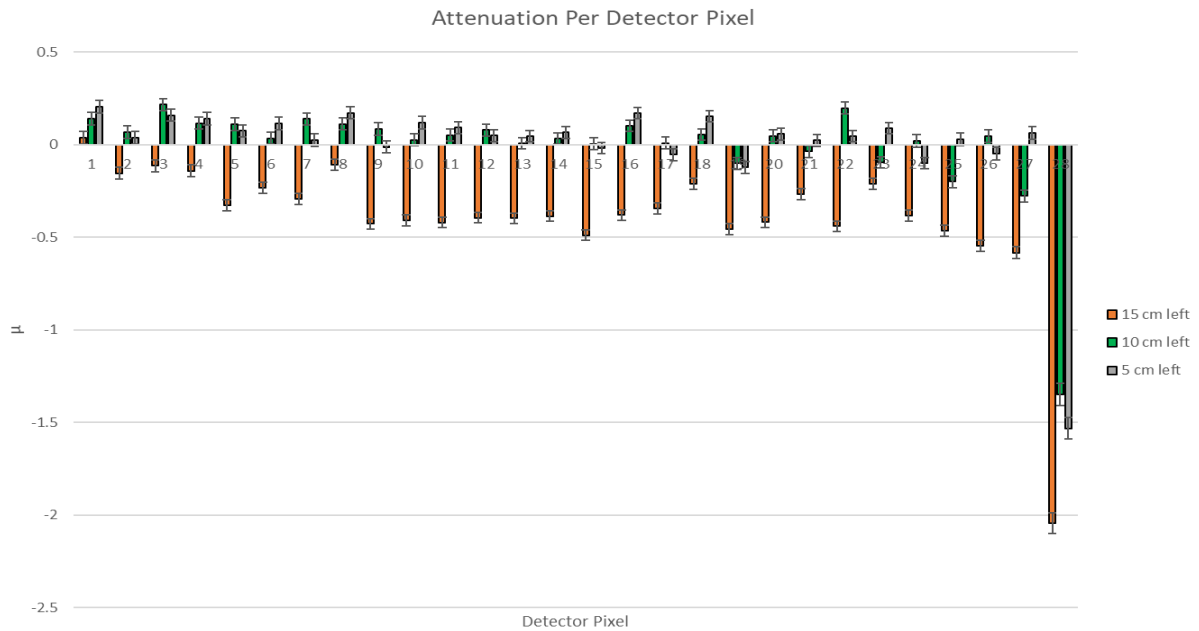


Figure 4.15. -X detector translational offsets: -5 cm, -10 cm, and -15 cm

Note the similarity of results from -5 and -10 cm offsets contrasted with the -15 cm offset measurement. This is most likely the result of typical increased neutron output fluctuations in the DT generator. The current generator in use is nearing the end of its lifecycle and variations in neutron flux outputs of 10 to 20% can be observed during an observation. The longer the measurement the lesser the impact of these fluctuations. Still the results from -5 and -10 cm are consistent with similar symmetrical offsets. MCNP simulations cannot accurately replicate these offsets as there is no beam characterization

data for α pixels 14 and 15, and only incomplete data for pixel 13. Note detector 28 consistently reports negative attenuation values as, at the origin, it is only partly covered. As it is translated left it becomes covered and records higher counts resulting in negative attenuation values.

Rotational Offset Void Measurements

This tranche furthered understanding of the impact of XY plane misalignment. This arrangement remains the easiest to properly align, both in the lab and fieldable methods due to the design of the tripod stands to which the detector arm is mounted. The cm offsets reference the center of the detector arm in relation to the face of the DT generator. Figure 4.16 illustrates this offset. The red lines indicate where the focal point of detector arm intersects the XZ plane to give ± 5 cm, ± 10 cm, and ± 15 cm offsets. The result is a translation of half of the detectors closer to the face of the DT generator while the second half of the arm moves away. Note the DT generator is never moved remaining at the location of the red cross in Figure 4.16. Seven measurements were taken in this tranche according to Table 4.1 below:

Table 4.1. Rotation offset void measurements with cm and corresponding degree offsets

Measurement	Offset (cm)	Offset (°)
Void – no offset	0	0
Counterclockwise 1	-5	2.272
Counterclockwise 2	-10	4.537
Counterclockwise 3	-15	6.789
Clockwise 4	+5	2.272
Clockwise 5	+10	4.537
Clockwise 6	+15	6.789

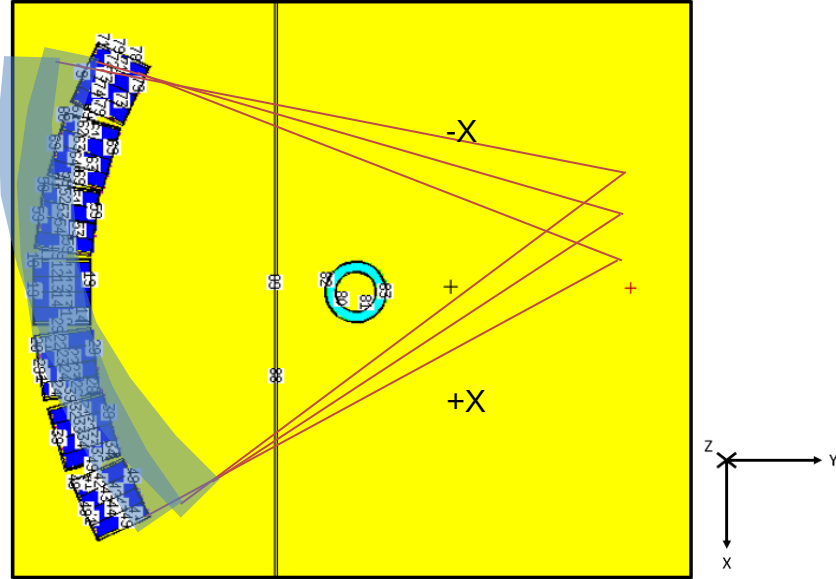


Figure 4.16. XY plane view of detector arm through $-\hat{x}$ Offsets: -5cm, -10cm, and -15cm

First, we will discuss the results of the three offsets where the left side ($+\hat{x}$) moves closer to the DT generator. As expected, counts increase in the left side of the arm. This results in negative μ values in detectors 1-14. Detectors 15-28 increase in distance from the DT generator. This results in decreased counts and positive μ values. Again, note detector 28 is an outlier as it resides at the extreme edge of α pixel 1 in Figure 4.17.

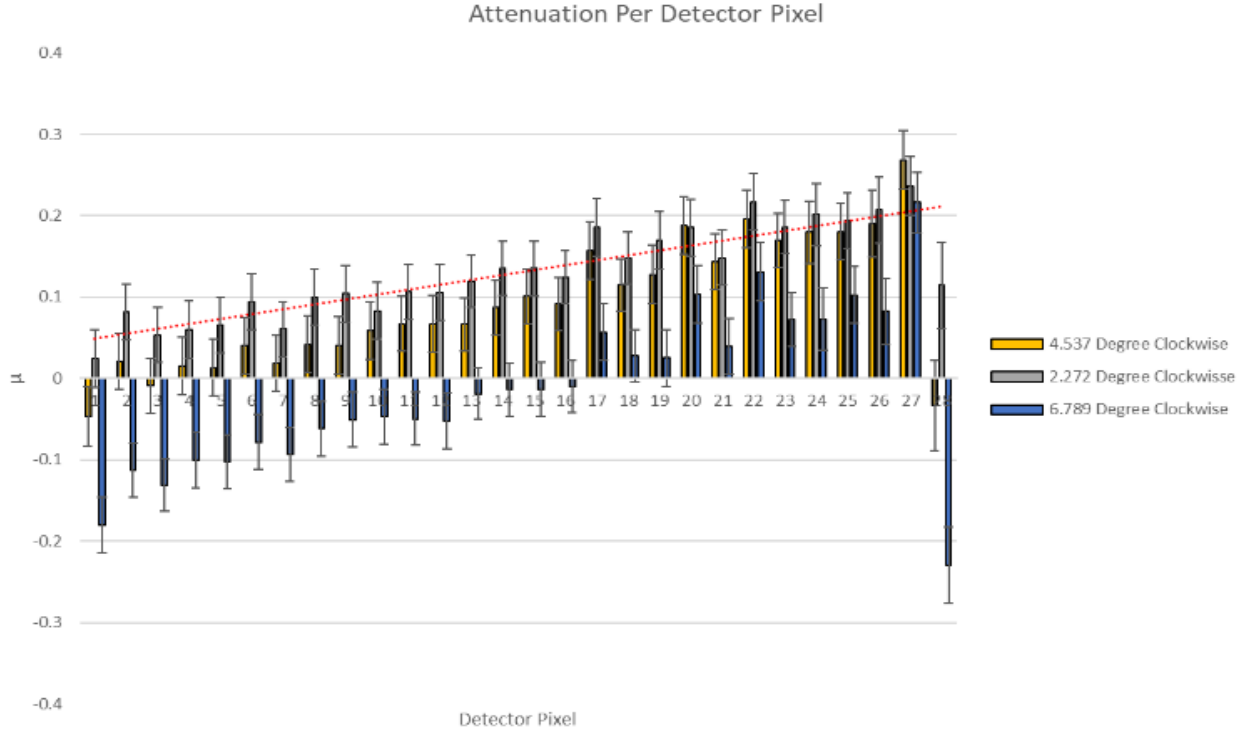


Figure 4.17. XY plane translational offsets: +5 cm, +10 cm, and +15 cm

Next the detector is translated in the $-\hat{x}$ direction (pictured in Figure 4.16.) In these measurements, detectors 15-28 move closer to the detector pictured in Figure 4.18. Again detector 28 is only partially covered in the fully aligned void measurement and remains only partially covered by α pixel 1 through the series of rotations.

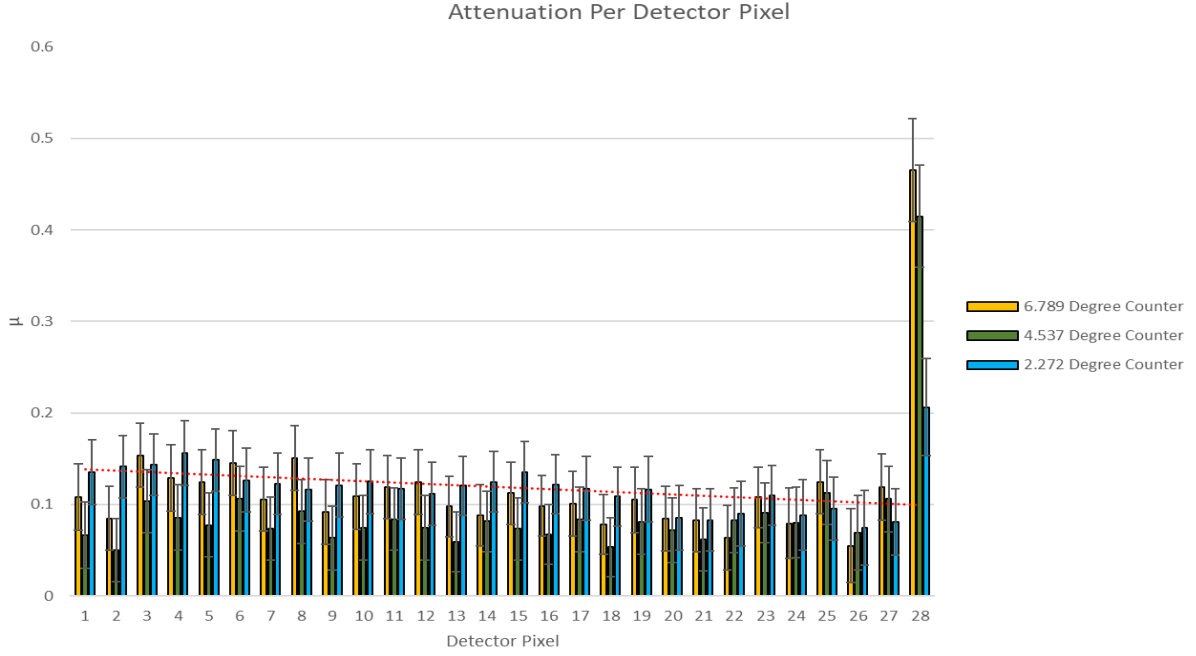


Figure 4.18. -XY plane translational offsets: -5 cm, -10 cm, and -15 cm

The pattern is clear when the detector arm is rotated $+\hat{x}$ in the xy plane with negative values of μ achieved by 6.789 degrees of rotation. Negative μ values are expected in all measurements where the detectors are moved closer to the DT source due to the reduction of R . In practice this is only achieved in conjunction with an increase in geometric efficiency. As the normal surface vectors of the detectors and beam lines of the α pixels approach parallel, the relative 4π surface area each detector covers increases – thus increasing geometric efficiency. In the $+\hat{x}$ rotation these two factors contribute to an overall increase in counts and negative μ values. (Note this only applies to detectors 1-14 as R increases for detectors 15-28 and surface vectors become orthogonal relative to beam lines reducing geometric efficiency). The two factors work against each other in $-\hat{x}$ rotation.

In the $-\hat{x}$ rotation surface vectors of the detectors diverge from parallel and move toward orthogonality with increased rotation. This leads to an overall decrease in the relative surface area of 4π each detector covers. This reduces geometric efficiency and alone would result in overall less counts. However, R is slightly reduced in detectors 15-28 in this case and the two factors work against each other. The result is the relative constant μ for all rotations in Figure 4.18 making identification of an offset difficult when rotated $-\hat{x}$.

Translational Offset Void Measurements ($\pm Y$ Axis)

As the need for more training data for neural networks became apparent additional data sets were collected. The first of these new tranches examined the impact of solely $\frac{1}{R^2}$ through translation the detector along the y axis closer to $+\hat{y}$ or further away $-\hat{y}$ from the source. Overall results matched expectation.

In the first measurement the detector arm was positioned +5 cm closer to the DT generator. As expected, negative μ values were observed on most detectors. Detectors 26 – 28 were no long covered by the α pixels and were branded outlier detectors. The recorded average value of $-0.058 \mu \pm 0.030 \text{ cm}^{-1}$ was below the expected value -0.085 cm^{-1} in Figure 4.19 below.

In the second measurement the detector arm was positioned +10 cm closer to the DT generator. Negative μ values were observed consistent with the results from the previous measurement. Again detectors 26-28 were no long covered by α pixels. The recorded average value of $-0.083 \mu \pm 0.030 \text{ cm}^{-1}$ was again below the expected value of -0.167 cm^{-1} . Below are the results of both $+\hat{y}$ measurements.

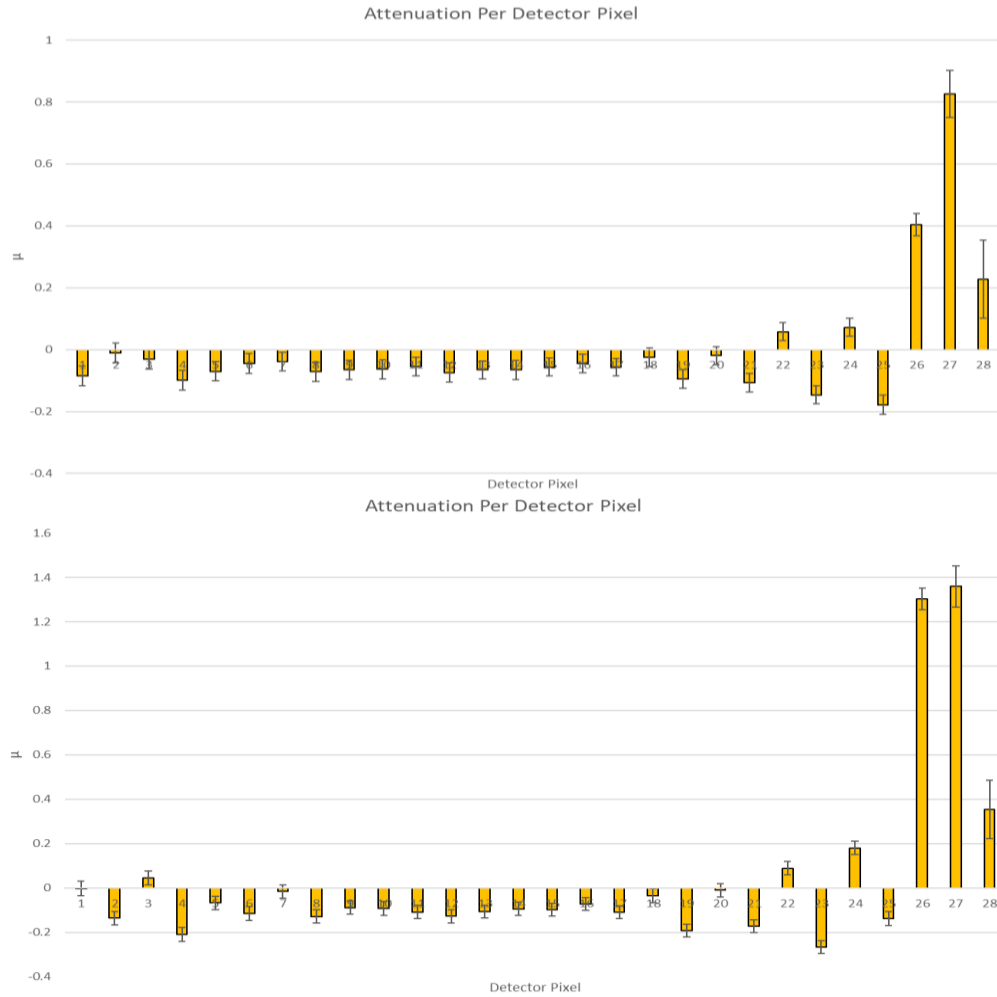


Figure 4.19. DT-detector (top) 110.1 cm distance (bottom) 105.1 cm distance

In the third measurement the detector arm was positioned- 5 cm away from the DT generator. Positive μ values were observed. However, the resulting μ values were larger than expected. The recorded average value of $0.137 \mu \pm 0.032 \text{ cm}^{-1} \mu$ was higher than the expected value of 0.085μ . In this measurement detectors 27 and 28 are covered by the beam – resulting in negative μ values.

In the final measurement of the tranche the detector arm was positioned -10 cm further away from the DT generator. The recorded average value of $0.220 \pm 0.033 \text{ cm}^{-1}$

was again higher than the expected value of 0.167 cm^{-1} with detectors 27 and 28 still covered by the beam. A pattern emerged from this tranche consistent with other tranches: the largest fluctuations reside at the extremities of the arm. In addition, the inverse response of detectors 27 and 28 complement the findings. Figure 4.20 depicts the results of both $-\hat{y}$ measurements.

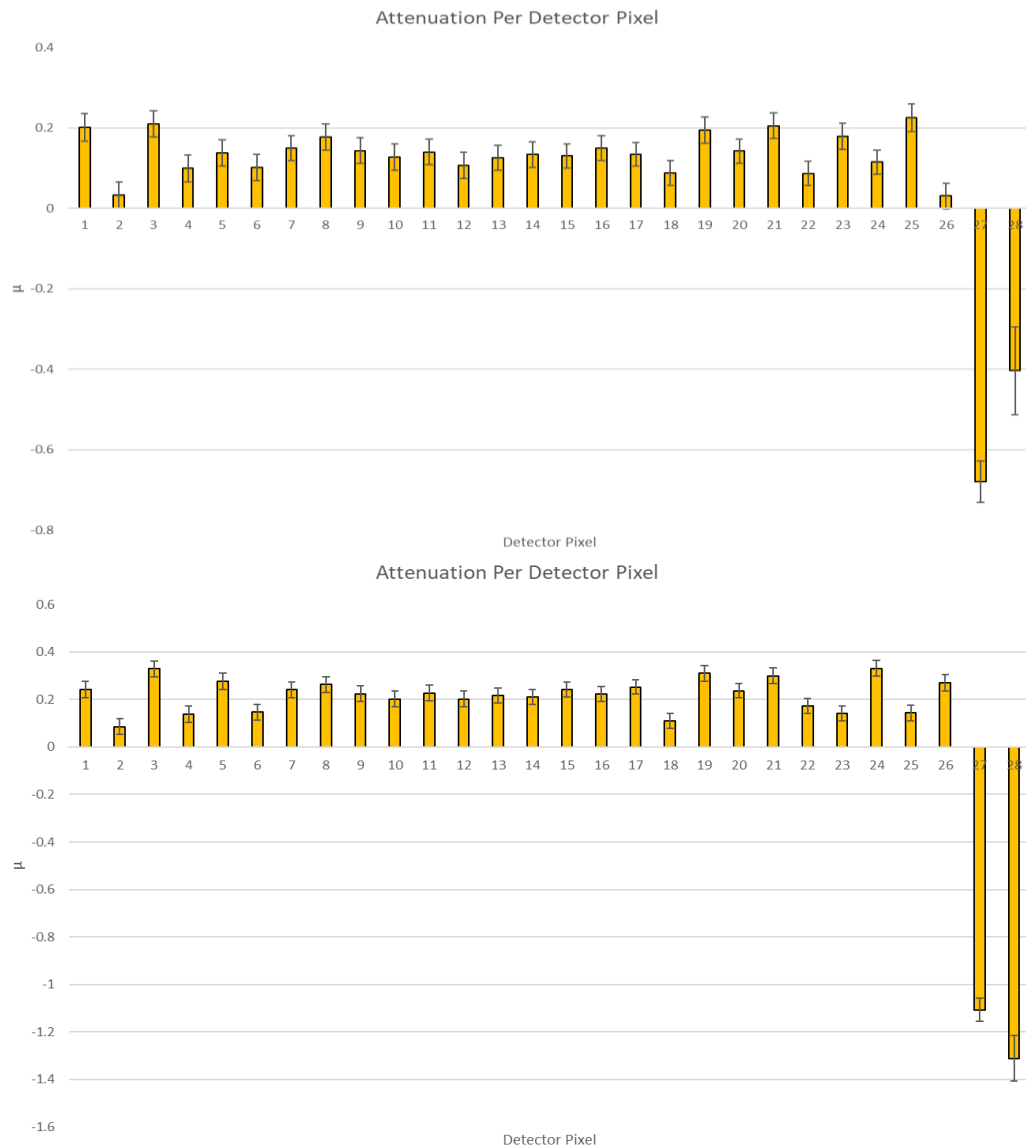


Figure 4.20. DT-detector (top) 120.1 cm distance (bottom) 125.1 cm distance

Advanced Configurations

The first advanced configuration measured two nested cylinders with perfect alignment. The inner cylinder used was the same DU canister from previous measurements. It was encased in a second cylinder composed of solid lead ringlets stacked on top of each other to form a cylinder. Two steel rods held the lead ringlets in place with zip ties ensuring no air gaps. The outer lead cylinder had an inner radius of 10.10 cm and thickness of 2.41 cm. This resulted in a 3.45 cm air gap between the two rings. Figure 4.21 is an image of the configuration.

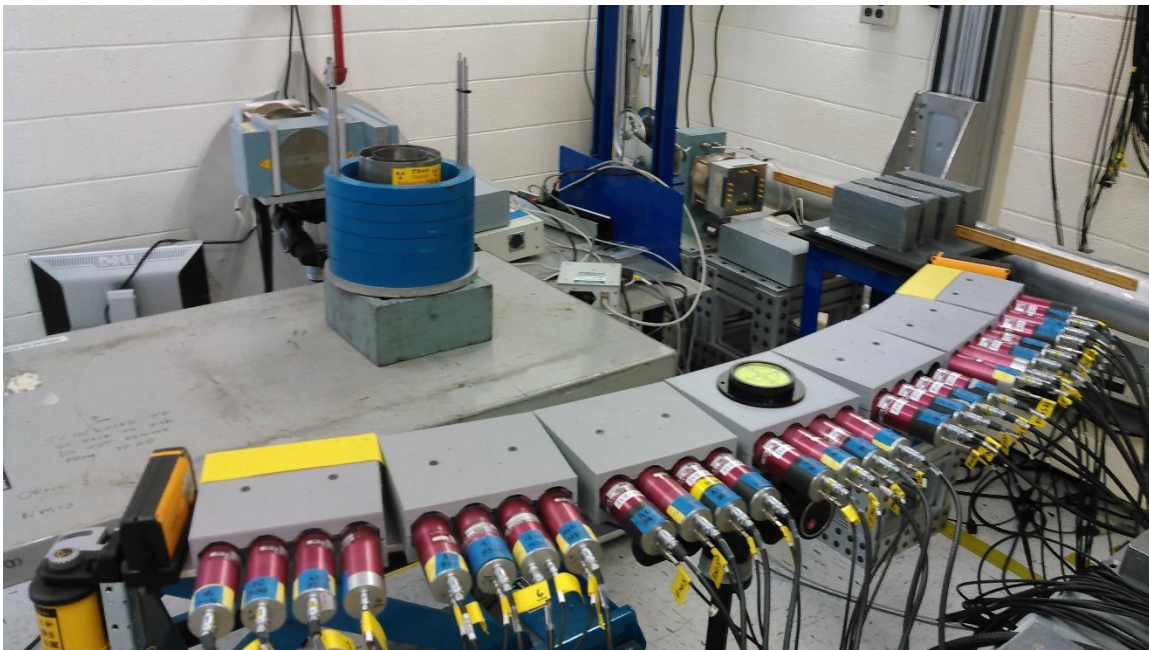


Figure 4.21. Lead, DU nested cylinder assembly

A void measurement was first taken with the configuration lowered to the floor as to leave the neutron flight path unobstructed. The configuration was then raised into position so that the API beam intersected the midpoint of the canister. The center of mass

of the assembly was located 40 cm from the face of the DT generator directly in line with center alignment. The measure was then taken, simulated in MCNP, and the ensuing attenuation profiles were generated in Figure 4.22.

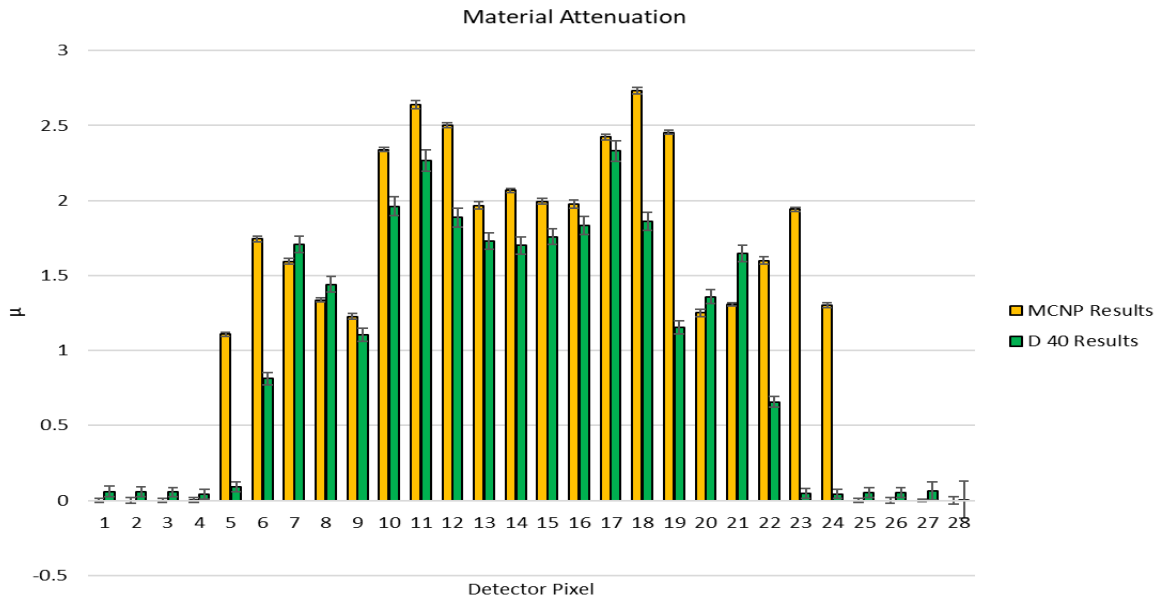


Figure 4.22. Lead, DU nested cylinder assembly attenuation profile

The associated mean free path, λ , of lead is 7.05 cm for 14.1 MeV neutrons given a density of lead of 11.35 g/cm³. This would result in a value of 1.98 cm⁻¹ for μ given a of ring 14.0 cm thick. The physical experiment achieved a relatively close value of μ at 1.706 $\mu \pm .056$ cm⁻¹ and 1.650 $\mu \pm .054$ cm⁻¹. The two air gaps are clearly visible in the drop of μ . The second peak clearly identified the inner casing of DU with values of μ at 2.330 $\mu \pm .070$ cm⁻¹ and 2.267 $\mu \pm .072$ cm⁻¹.

The MCNP simulation yielded results consistent with the experimental data. Again, the MCNP model produced results closer to theoretical values. In the case of the

14.0 cm thick outer lead ring the MCNP simulation exactly matched the theoretical value of μ at $1.98 \mu \pm .011 \text{ cm}^{-1}$. The MCNP model displays a magnification in detector values consistent throughout all simulation-data collection comparisons. During this experiment a drawback was identified for simulations with multiple layers of high Z material – computational time increases by a factor of 10 due to particle tracking requirements of the PTRAC file.

This small increase in the magnification of the target in the MCNP is the result of the geometry of the arm as the detector arm is slightly more compact in the MCNP model than in the physical assembly. The detector housing in MCNP is simply two rectangular parallelepipeds. Physically, the casing is much more complex (see Figure 3.2). As a result, there exists a slight magnification consistent throughout all simulations. The result in Figure 4.23 is akin to those of an experiment conducted by Dr. Paul Hausladen and the author using a modified version of x-ray imaging.

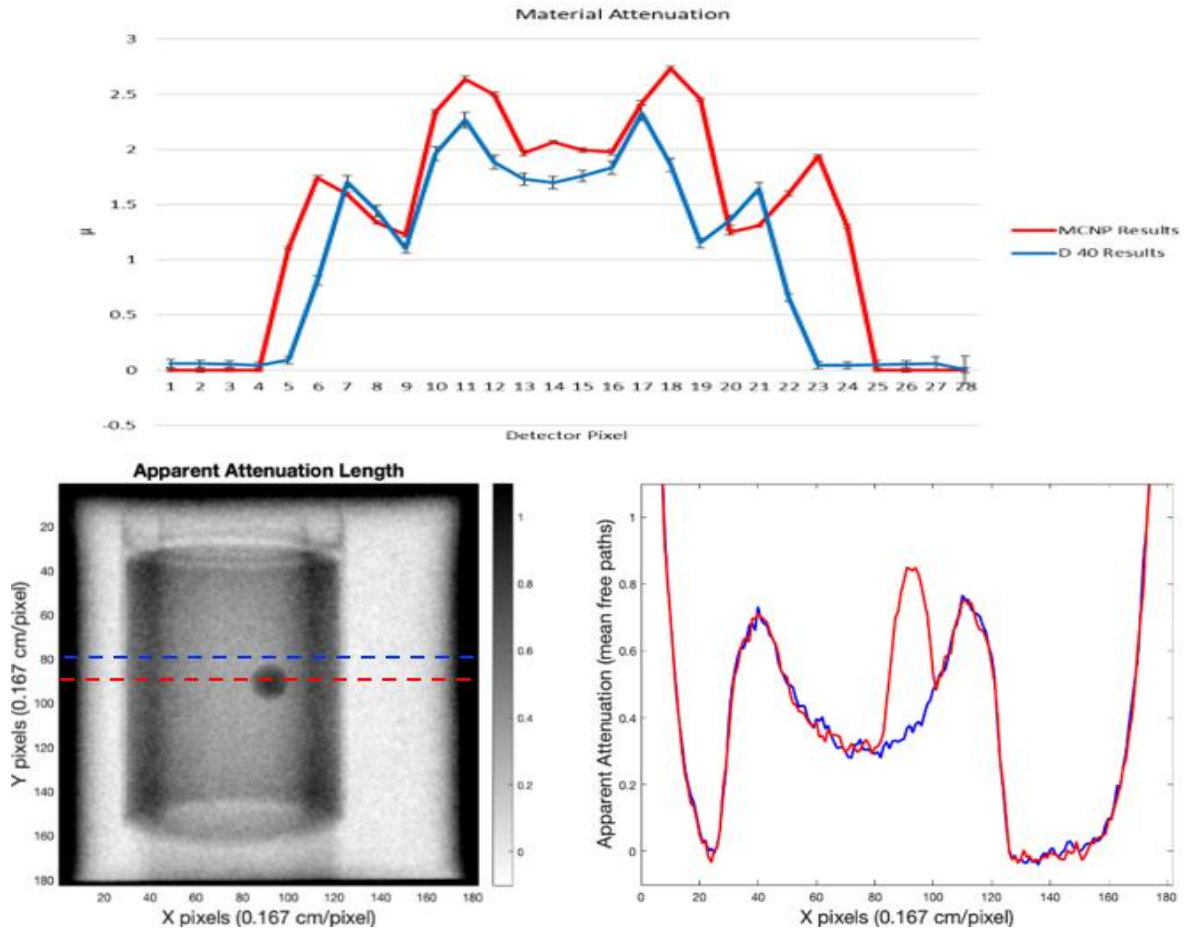


Figure 4.23. (top) API results converted to linear plots. (left) The apparent thickness (in mean free paths) is shown along with (right) two lineouts whose positions are shown by the dashed lines.

In this study, there was an opportunity to generate fast neutron images using the combination of a storage phosphor and a high-density polyethylene (HDPE) converter. The HDPE converter was approximately 0.95 cm thick. The downside to this method of imaging is the limited distance it is suitable for and increased measurement duration. In Figure 4.24, the exposure was taken over a one-hour period where the DT generator voltage was increased from 60% to 80% of maximum power over the initial 20 minutes. The average emitted neutron intensity over this time was 1.3×10^6 neutrons s^{-1} .

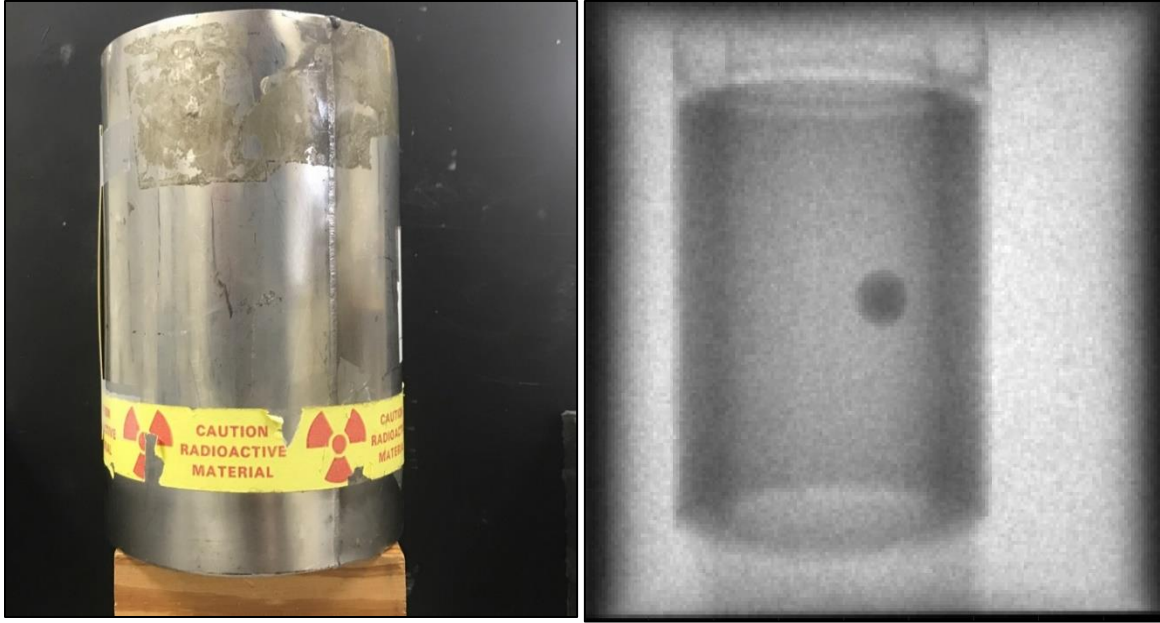


Figure 4.24. (Left) photograph of DU Casting and (right) raw fast-neutron radiograph of the DU casting

The pixel resolution is 0.169 cm^2 in the case of storage phosphors and 6.45 cm^2 for the API design. However, the HDPE exposure time is 6 times longer. In addition, the panel is positioned 49.4 cm from the neutron source vs. the API's 115.1 cm. Figure 4.25 portrays the layout of the measurement.

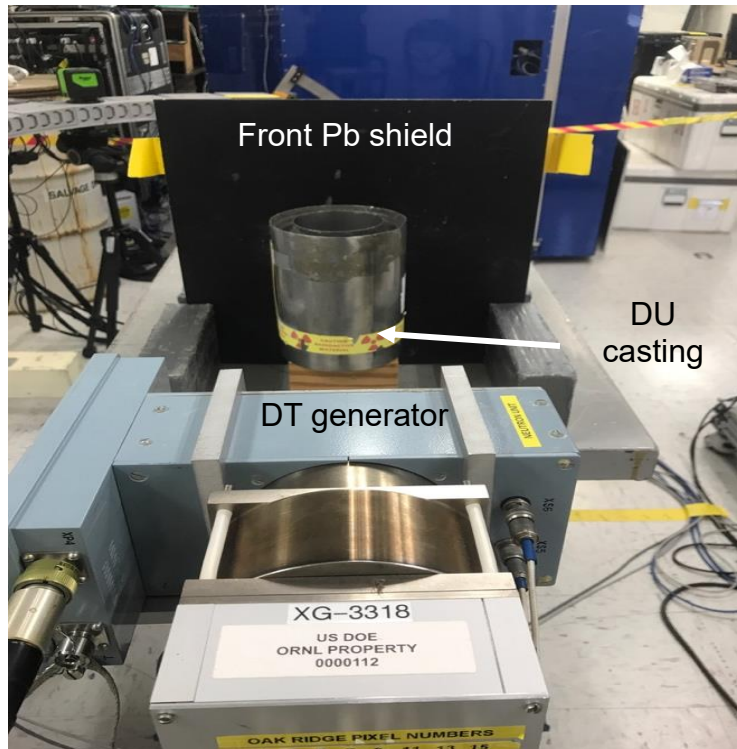


Figure 4.25. The experimental set up with the ING-27 DT neutron generator visible in the foreground and the near-side lead shield visible in the background. In this photograph, a DU casting is positioned in front of the detector.

In Figure 4.25, one can see the physical set up of this experiment. Note the proximity of the detector face to the DT generator. Also note the apparent hole in Figure 4.24. This is the result of a physical hole in the HDPE. No neutron conversion occurred in this space resulting in a visible circle in the resulting Figure 4.23 and 4.24. A 2D array of detectors would allow the reconstruction of similar images in API designs. Such a design is currently being studied at ORNL.

Additional measurements were then taken for a series of 15 offset positions. These offsets included six XY plane rotation measurements, four X plane offsets, four Y plane offsets, and a final steel panel measurement used to increase γ counts. While the primary purpose of these measurements was to supply more training data for machine

learning algorithms the results matched all previous experiments and predictions. Three of these measurements (Figures 4.26, 4.27, and 4.28) are presented to demonstrate the confirmation of patterns already identified. The DU and lead physical properties and dimensions remained the same consequently, theoretical μ values remained the same as previously stated in this work.

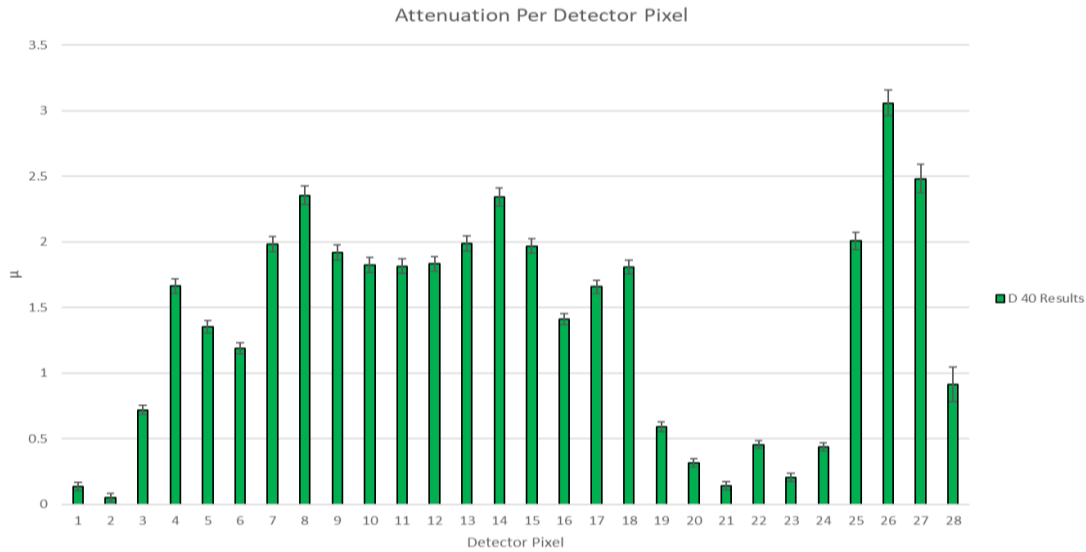


Figure 4.26 DU and lead canister casting with +10cm $\hat{x}$ offset

In this first experiment (Figure 4.26), the DU and lead were imaged with the detector arm offset +10 cm in the $\hat{x}$ direction. The first set of peaks clearly identified the outer casing of lead with values of μ at $1.664 \mu \pm .055 \text{ cm}^{-1}$ in detector 4 and $1.808 \mu \pm .053 \text{ cm}^{-1}$ in detector 18. The second set of peaks again clearly identified the inner DU canister with values of μ at $2.354 \mu \pm .072 \text{ cm}^{-1}$ in detector 8 and $2.340 \mu \pm .070 \text{ cm}^{-1}$ in detector 14. From the peaks in detectors 25-28 the author assessed an offset of 10.16 cm or a four-detector offset. Again, these peaks were the result of decreased counts in the

detectors as they exit the coverage of the tagged neutron beam.

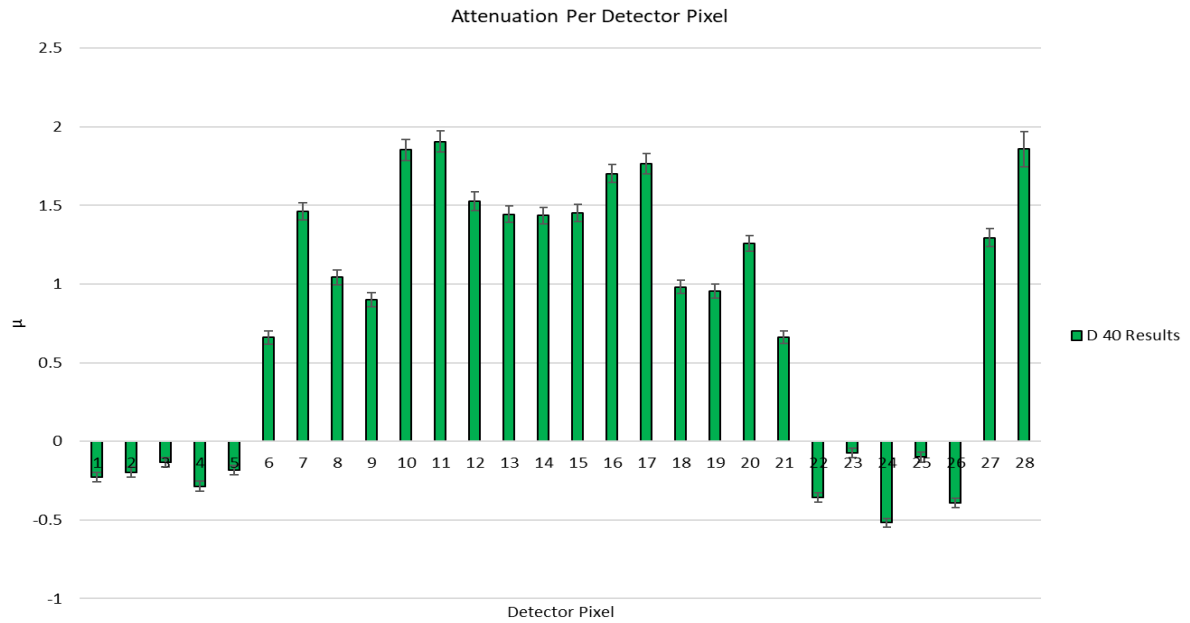


Figure 4.27 DU and lead canister casting with +5cm $\hat{y}$ offset

In this second experiment (Figure 4.27), the DU and lead were imaged with the detector arm offset +5 cm in the $\hat{y}$ direction – moving the arm closer to the DT generator. The first set of peaks clearly identified the outer casing of lead with values of μ at $1.426 \mu \pm .054 \text{ cm}^{-1}$ in detector 7 and $1.260 \mu \pm .050 \text{ cm}^{-1}$ in detector 20. The second set of peaks identified the inner DU canister with values of μ at $1.910 \mu \pm .065 \text{ cm}^{-1}$ in detector 11 and $1.766 \mu \pm .063 \text{ cm}^{-1}$ in detector 17. From the peaks in detectors 27-28 the author assessed an offset of 5.08 cm or a two-detector offset. This offset was then deduced to be in the $\hat{y}$ plane as the counts of detectors not covered by the objects increase (1-5 and 22-26). As the arm was moved closer to the DT generator detectors 27 and 28 exit beam coverage and overall counts increased. This movement closer to the DT generator slightly

decreased the apparent thickness of the object. However, the shape of the advanced configuration remained identifiable.

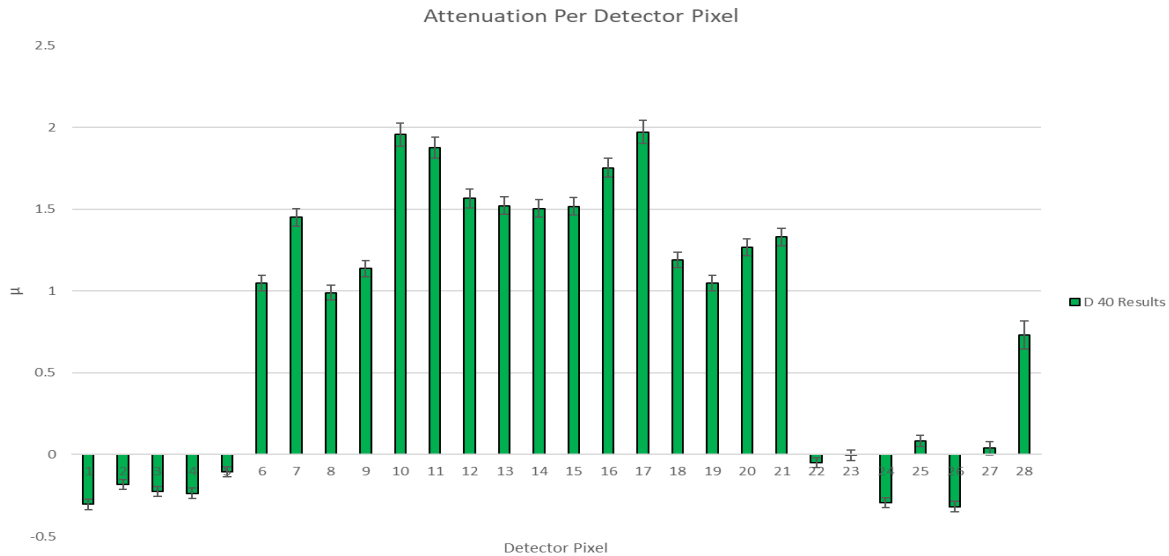


Figure 4.28 DU and lead canister casting with a clockwise rotation in the XY plane of 6.789°

In this final experiment (Figure 4.28) the DU and lead were imaged with the detector arm offset 6.789° in the XY plane – moving the left side of the detector arm closer to the DT generator and the right side away. The first set of peaks clearly identified the outer casing of lead with values of μ at $1.445 \mu \pm .055 \text{ cm}^{-1}$ in detector 7 and $1.330 \mu \pm .051 \text{ cm}^{-1}$ in detector 21. The second set of peaks identified the inner DU canister with values of μ at $1.957 \mu \pm .070 \text{ cm}^{-1}$ in detector 10 and $1.973 \mu \pm .069 \text{ cm}^{-1}$ in detector 17. From the lack of a clear pattern in the results of the unobstructed detectors rotation appeared the likely candidate. As the left side of the arm was moved closer to the DT generator detectors 1-5 exhibited increases in counts. The right side of the arm did not exhibit any pattern – with both increases and decreases. The lack of a uniform pattern led

the author to infer rotation – but no discernible value could be concluded. Nevertheless, the shape of the advanced configuration – two nested cylinders – remained clearly identifiable.

CHAPTER FIVE

CONCLUSIONS AND RECOMMENDATIONS

This work demonstrated the impact of misalignment on API measurements for a portable system. The current mass of the system is 100.09 kg consisting of four primary components: a 16.29 kg detector arm assembly, an 8.86 kg power supply assembly for the PMTs, a 20.64 kg DT generator assembly, and a 54.31 kg electronics assembly. The electronics assembly's weight is primarily due to two ORTEC Model 4002D Power Supply NIM bins each weighting 18.10 kg. Replacing PMTs with SiPMs would result in an 8.86 kg decrease. The decrease is purely a consequence of removing the power supply requirements to the PMTs. There would also be a significant decrease in the electronics assembly as NIM bins would be removed with SiPMs detectors. However, no hard figures are available on this reduction. Both the Eljen Tech EJ-204 with SiPMs array and the SEJ631 PMT detectors weigh 200.0 ± 0.5 g. As such, replacement of the physical detectors results in no weight reduction.

For the Fieldable Nuclear Material Identification System implementation of API, general offsets were identified with shifts as small as 5 cm in the $\hat{x}$ and $\hat{y}$ directions. Some rotational offsets are identifiable. Given the method of measurement collection and object type, identification remains possible despite misalignment in all cases. Object identification is rudimentary. All objects considered were cylinders symmetric around the z axis with uniform density.

Without cylindrical symmetry rotational misalignment geometry determination would not have been always possible. Translational (XY plane) misalignment geometry determination would still be possible for most modest shapes given complementary x-ray

images. However, geometry determination given rotational misalignment remains unreliable.

Similar results have been reproduced in simulation. The author developed a custom series of MCNP6.2 and python 3.7 scripts to simulate experiments. These simulations took advantage of a great volume of experimental data collected at ORNL over the last five years and numerous characterizations of DT generator sources.

Overall, both objectives of this work were accomplished. The first of these was to examine the impact of misalignment on measurements. The second objective was to simulate results in MCNP6.2 for the purpose of interrogating objects not available in the laboratory. The creation of a portable system proved to be a team effort with software support (Jim Mullens) and oversight (Paul Hausladen) provided to the author at various points along the journey from others on staff both at ORNL and UTK. While almost all physical construction was done by the author, guidance along the way, as well as all the requisite equipment, proved essential.

Coding the simulation workspace proved to be the most challenging and time-consuming effort in this undertaking. The result exceeded expectation as related simulation code to this point still operated on python 2.7. The current simulation workspace also enables misalignment studies, additional room/environmental variables, and detector variation options. The simulation workspace achieved validation, as it attained similar results to those of the physical experiment shown throughout *Chapter Four: Equidistant Measurements – Steel, Aluminum, Polyethylene, & Depleted Uranium* of this work. Appendices C-I contain the code developed in this effort.

Misalignment in the lateral $\hat{x}$ direction was evident at 5 cm offsets to within 3σ confidence. This confidence was a direct result of coverage by detectors of most of the tagged neutron beam profile. As the arm was moved left or right, large variations in the attenuation coefficient μ were observed in the outlying detectors compared with those which were more centrally located (covered by the tagged neutron beam). Those detectors covered by the tagged neutron beam throughout both measurements saw no appreciable change. These two corresponding factors allowed for identification of offsets. The theoretical maximum for this identification is 2.54 cm at 115.1 cm given physical size and resolution of each detector. Subsampling could increase this resolution but would bring with it its own disadvantages.

If the arm were to include a mechanical slide – either motorized or manually adjustable at fixed intervals – resolution could be increased. This is a proven method of increasing resolution. In previous versions of the NMIS, subsampling was executed at $\frac{1}{4}$ detector steps (0.635 cm) utilizing an integrated servo motor [46]. This integrated servo motor could be incorporated into a new detector arm adding a minimum of 1.55 kg to total system mass. More significantly, it would require additional time to complete scans at each subsampling step. If one did not desire to increase subsampling time, increases in resolution would only be possible if detector resolution were improved.

Increasing the number of detectors to cover the entirety of the tagged neutron beam would also aid in identification of offset as currently the left most two alpha pixels are beyond coverage of the detectors. In order to accomplish this with the current design 8 additional detectors would need to be added to the arm. This would increase total

weight by a minimum of 2.20 kg or 2.2%, 1.60 kg for additional detectors and 0.60 kg for additional mounting casings.

Misalignment in the vertical $\hat{y}$ direction is evident at 5 cm offsets to within 3σ confidence. This method of detection is contingent on two factors agreeing: outlier and covered detectors. First, covered detectors provide a mean μ value. As the arm is moved closer to the DT generator (keeping source strength constant), μ becomes increasingly negative. Conversely, as the arm is moved away from the DT generator (keeping source strength constant), μ becomes increasingly positive. Next, outlier detectors provide an inverse response. As the arm is moved closer to the DT generator, outlier detectors receive fewer counts, resulting in larger μ values. As the arm is moved away from the DT generator, outlier detectors record more counts as they become covered by the beam. The result is μ becomes negative.

The negative μ pattern is evident in Figure 4.20. DT-detector (Top) 120.1 cm (-5 cm $\hat{y}$). In this experiment, two detectors (27 and 28) enter coverage of the beam. As a result, they exhibit negative μ values due to increased counts. The remaining detectors see decreases in counts typically associated from moving a detector physically further away from the source. With both factors agreeing, one can assess a $-\hat{y}$ offset. The result is $\hat{y}$ offsets are differentiable from $\hat{x}$ offsets.

Outlier detectors are used to determine offsets in the case of $\hat{x}$ offsets. Positive μ values correspond to $+\hat{x}$ offsets in the rightmost detectors (24-28). Negative μ values correspond to $-\hat{x}$ offsets in leftmost detectors (1-4). Offsets do not correspond to changes in μ for detectors which remain covered by the beam as R remains relatively constant. If the number of detectors were increased by eight detectors the results of the leftmost

detectors (1-4) would mirror in magnitude with reverse sign the results of the rightmost detectors (32-36). These additional flanking detectors would continue with the current curvature of the arm at a radial distance of 115.1 cm.

If constructed, one would expect to see negative μ values in the left detectors (1-4), this would correspond to $+\hat{x}$ offsets, as the detectors would enter the beam's coverage. Positive μ values on left detectors (1-4) would correspond to $-\hat{x}$ offsets as the leftmost detectors would move out of the beam's coverage. Negative μ values in the right detectors (32-36) would correspond to $-\hat{x}$ offsets, as these detectors would enter the beam's coverage. Positive μ values on right detectors (32-36) would correspond to $+\hat{x}$ offsets as the right most detectors would move out of the beam's coverage. This pattern is clearly demonstrated in Figure 4.14 where $+\hat{x}$ offsets of 5cm, 10 cm, and 15 cm are visible in the positive values of right detectors (24-28).

Misalignment information is more difficult to ascertain in rotation measurements. In this case, one cannot rely on mean μ values. No misalignment was visible in the XY-plane of rotation clockwise around the $\hat{z}$ axis until 6.789° of rotation was achieved (3σ). At less than 6.789° , rotation is indiscernible. As the arm is rotated clockwise to 6.789° , R is reduced for detectors 1-14 and geometric efficiency is increased. In detectors 15-28, R is increased, and geometric efficiency is reduced. The resulting μ values yield a "bow-tie" with the outlier of detector 28. Detector 28 is not covered by the tagged neutron beam; hence, as it moves into beam coverage and collects more counts, it returns a negative μ value. This bowtie visibly evident in Figure 4.17 in the 6.789° rotation.

At less than 6.789° of rotation, no unique pattern is identifiable. Furthermore, no misalignment information is visible to within any certainty with counterclockwise

rotation due to the conflicting factors involved in geometric efficiency. The impact of geometric efficiency could potentially be reduced if the detector were centered on α pixel 8 and not 7. Centered on α pixel 8, the results would mirror Figure 4.17. Instead, results are nearly identical to $+\hat{x}$ behavior – see Figure 4.18 vs Figure 4.17.

While rotational misalignment remains challenging to identify given these methods, the impact of rotational misalignment remains minute on rotationally symmetric cylinders. The double wall peaks of the two nested cylinders are clearly visible at 6.789° rotation evidenced in Figure 4.28. As a result, rotational misalignment had no impact on identification of void spaces within cylindrically symmetric objects. This does not extend to all asymmetric shapes. With rotational misalignment, identification of greatly asymmetric shapes would not be possible. All experimental results collected allowed for clear identification of nested cylinders. The primary goal was thus achieved. One can clearly identify objects in question – assuming cylindrical symmetry.

Recommendations

Looking forward, one potential method that needs to be investigated is incorporation of time-of-flight information to identify the misalignment more accurately as arrival times in detectors could potentially allow for rotation identification. Previous works have utilized time-of-flight list mode data to successfully identify fissile material [46]. This application would examine the same data set with new intent. Instead of searching for late arrival neutrons, peak arrival times would be examined. Given that 100 ps timing resolution is now available (previously 1ns), it may be possible to identify delays in tagged-neutron arrival peaks resulting from rotational misalignment. This

increase timing resolution is a feature recently introduced in the software updates implemented by Jim Mullens in the fall of 2019. According to the Caen TDC user manual the options now are 800 ps, 200 ps, and 100 ps. This could enable the challenging task of identifying rotational misalignment.

The fact that outlier detectors – such as detector 28 in most trials – provided indispensable insights and complement the patterns exhibited by covered detectors highlights the importance of outlier detectors. Without outlier detectors many conclusions reached in this work would not have been possible. As such any future system must ensure it incorporates outlier detectors on both flanks of the arm. Even with additional outlier detectors, larger objects could present an issue. If the object occluded these outlier detectors additional detectors would be of no further benefit. This would make misalignment determinations impossible for large objects ($>1 \text{ m}^3$). The largest object measured in this work filled a volume of 0.00908 m^3 . The impact of occlusion is significant on misalignment as offsets were only verified when the covered detectors' pattern and the outlier detectors' results complemented each other.

Another issue with the current design is the center of the detector arm, in the current assembly, is not located at the center of the tagged neutron beam (see Figure 4.4. ING-27 DT Generator Offset of Central Alpha Pixel). An offset of $-5 \text{ cm } \hat{x}$ would center the detector arm in this beam at 115.1 cm from the DT generator given the current assembly. However, doing so results in no outlier detectors. This will remain a factor for consideration in any future systems employing the ING-27. This could easily be addressed in a future design incorporating at least eight more 2.54 cm detectors – four on each side of the arm.

While physically producing another arm was not feasible given material availability, simulation in MCNP was achievable. A detector assembly with 36 (2.54x2.54x10.16cm) detectors was simulated using the scripts developed in this work. A void measurement was compared to a +5 cm $\hat{x}$ offset of the detector arm to yield a simulated attenuation profile similar to the comparison profiles.

An offset of 5.08 cm was clearly identifiable in detectors 31 and 32. Poor statistics, resulting from low counts in outlier detectors, resulted in attenuation values as large as $\pm 8.215 \mu$ in outlier detectors (1-4 and 33-36). The result that is one is unable to identify detectors 3 and 4 entering coverage by the tagged neutron beam. The results of the simulation are promising. By adding eight additional detectors one would ensure the entire tagged neutron beam is covered by the detector assembly – even with an offset.

An accompanying issue in attaining many of the results in this work is the requirement of an unobstructed flight path for at least a portion of tagged neutrons to outlier detectors. There are two work arounds for this issue. First, it is possible to increase the size of the detector arm and tagged neutron beam such that it includes unobstructed detectors. This aforementioned solution works well for small (less than 0.027m³) and medium-sized containers (less than 1 m³ with a lateral dimension of less than 1 m). However, portability of the arm becomes an issue if the container is larger than 1 m³ (with a lateral dimension larger than 1 m) as the size of the array increases significantly.

A second potential work around would be to collect pre-characterized void measurements for common larger containers. One would first measure an empty large container prior to going to the field. In the field, the field measurement would be compared against the pre-characterized container void. For example, if the object of

interest resided inside a shipping container, the collected field measurement would be compared to a pre-characterized void of an empty shipping container. The feasibility of this option requires additional study.

Until such a time that software is intelligent enough to interpret resulting profiles for the operator, pattern recognition must be included in any user operator course. There are patterns clearly visible in the images. These patterns correspond to offsets. With sufficient system familiarity an operator would be able to infer general misalignment based upon these patterns.

Future Work

Future work should be focused in two areas. First, the physical detector size and weight must both be reduced to employ the system more effectively in the field. SiPM detectors should be investigated as they promise to significantly reduce power requirements and consequently weight. Resolution can be improved if individual detector size is reduced (1.61 cm^2 from 6.45 cm^2) and detectors (4 for 1) are added to cover the same solid angle of exposure. Subsampling could also be employed to increase resolution as demonstrated by the APNIS system. However, subsampling would need to be limited due to the increased measurement time associated with each additional subsample.

Dual X-ray and neutron sensitive materials may also prove manufacturable and achieve similar performance to previous API results while yielding additional information from the X-ray images. A dual sensitive material would not require time-of-flight information to generate a radiograph. Instead, only an amorphous silicon thin film transistor array would be required to generate a radiograph. This would significantly

reduce the weight due to the electronics associated with API. Additionally, a two-dimensional panel array would vastly improve product quality from a 1D “slice” to a full 2D radiograph, as briefly demonstrated in *Chapter Four: Advanced Configurations Figure 4.24*.

Second, neural networks or some form machine learning should be applied to the collected data from reported misalignment measurements and any future misalignment studies. If the human eye can infer a pattern in the data, it is only a matter of time before someone is able to apply a machine learning algorithm to the data to achieve better, quantitative misalignment estimates. Ideally, this would be included in the fieldable system, allowing for a “corrected image” as well as providing offset information to the operator. The difficulty in this analysis resides in which features of the data to assess. Only when the proper data is presented to the machine will it yield accurate and useful information. This will begin as supervised machine learning. Additional data collection will most likely be required in order to provide sufficient training sets to the algorithm.

LIST OF REFERENCES

- [1] J. C. o. Staff, Joint Publication 1-02 Department of Defense Dictionary of Military and Associated Terms, Washington, DC: Department of Defense, 2019.
- [2] G. Zentai, "X-ray Imaging for Homeland Security," *IEEE International Workshop on Imaging - Systems and Techniques*, pp. 978-984, 2008.
- [3] A. J. S. E. M. L. J. M. B. Jarrold T. Bushberg, The essential physics of medical imaging, Philadelphia, PA: Lippincott Williams and Wilkins, 2002.
- [4] A. J. B. John R. Lamarsh, Introduction to Nuclear Engineering 3rd Edition, Upper Saddle River: Prentice Hall, 2001.
- [5] N. N. D. Center, "Evaluated Nuclear Data File," Brookhaven National Laboratory, Brookhaven, 2019.
- [6] I. M. N. K. A. H. M. D. J. B. M. Stroble, "Advances in Neutron Radiography and Tomography," *Journal of Physics D: Applied Physics*, vol. 42, 2009.
- [7] J. Chadwick, "Possible Existence of a Neutron," *Nature*, vol. 129, no. DOI 10.1038/129312a0, pp. 327-29, 1932.
- [8] J. S. Brenizer, "A Review of Significant Advances in Neutron Imaging from Conception to the Present," *Physics Procedia*, vol. 43, no. DOI: 10.1016/j.phpro.2013.03.002., pp. 10-20, 2013.
- [9] J. Thewlis, "Neutron Radiography," *British Journal of Applied Physics*, vol. 7, no. DOI: 10.1088/0508-3443/7/10/301, pp. 345-350, 1956.
- [10] J. Barton, "Neutron Radiography – An Overview. In: Berger H, editor. Practical Applications of Neutron Radiography and Gaging," *American Society for Testing and Materials, STP*, vol. 586, pp. 5-19, 1976.
- [11] ThermoFisher Scientific, "Minigene Neutron Generator," ThermoFisher Scientific, Waltham, MA, 2018.
- [12] UT-Battelle, LLC, MPNC User's Manual Code Version 6.2, Oak Ridge: Oak Ridge National Laboratory, 2017.
- [13] T. S. a. H. P. Vankov, "Nature of the highest energy cosmic rays," *Physical Review*, vol. 55, no. DOI: 10.1103/PhysRevD.55.1365., p. 1365–1371, Feb. 1997.
- [14] C. R. Heimbach, "Cosmic coincidences: Investigations for neutron background suppression," *Journal of Research of the National Institute of Standards and Technology*, vol. 112, no. DOI: 10.6028/jres.112.007, p. 95, Mar. 2007.
- [15] J. W. e. al., "China Spallation Neutron Source: Design, R&D, and outlook," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 600, no. DOI: 10.1016/j.nima.2008.11.017, pp. 10-13, Feb. 2009.
- [16] T. S. a. H. P. Vankov, "Nature of the highest energy cosmic rays," *Physical Review*, vol. 55, no. DOI: 10.1103/PhysRevD.55.1365., pp. 1365-1371, Feb. 1997.
- [17] L. C. e. al., "The Macromolecular Neutron Diffractometer MaNDi at the Spallation Neutron Source," *Journal of Applied Crystallography*, vol. 48, no. DOI: 10.1107/S1600576715011243., p. 1302–1306, Aug. 2015.

- [18] S. R. R. J. S. B. D. S. a. J. R. T. J. E. Eberhardt, "Fast neutron radiography scanner for the detection of contraband in air cargo containers," *Applied Radiation and Isotopes*, vol. 63, no. DOI: 10.1016/j.apradiso.2005.05.012, pp. 179-188, Aug. 2005.
- [19] M. F. L'Annunziata, *Solid Scintillation Analysis*, 3rd ed., New York: Academic Press, 2012, pp. 191-231.
- [20] V. Valkovic, *Associated Alpha Particle-Tagged Neutron Beams 14 MeV Neutrons Physics and Applications*, Boca Raton: CRC Pres, 2016.
- [21] M. A. B. a. P. A. Hausladen, "Fast-neutron elastic-scatter imaging for material characterization," in *IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC)*, San Diego, CA, USA, 2015.
- [22] J. S. N. a. J. T. M. P. A. Hausladen, "An alpha particle detector for a portable neutron generator for the Nuclear Materials Identification System (NMIS)," *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, vol. 241, no. DOI:10.1016/j.nimb.2005.07.138, pp. 835-838, Dec. 2005.
- [23] Y. L. e. al., "Neutron spectroscopy and spectral unfolding with 4He fast neutron scintillators," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 922, no. DOI:10.1016/j.nima.2018.10.098, pp. 1-7, Apr. 2019.
- [24] M. C. W. S. M. M. D. E. A. a. B. A. P. J. T. Mihalcz, "Transportable, Low-Dose Active Fast-Neutron Imaging," ORNL/TM-2017/187, 1400208, Oak Ridge, TN, Aug. 2017.
- [25] P. R. B. J. S. N. J. A. M. a. J. T. M. P. A. Hausladen, "Portable fast-neutron radiography with the nuclear materials identification system for fissile material transfers," *Nuclear Instruments and Methods in Physics Research Section B; Beam Interactions with Materials and Atoms*, vol. 261, no. DOI:10.1016/j.nimb.2007.04.206, pp. 387-290, 2007.
- [26] A. L. Swift, "Materials and Configuration from NMIS-Type Neutron Imaging and Gamma Spectroscopy," ORNL/TM-2012/189, 1081926, Oak Ridge, May 2012.
- [27] T. S. T. a. P. B. M. Gara, "Machine Learning as a Preprocessing Phase in Discrete Tomograph," *Applications of Discrete Geometry and Mathematical Morphology*, no. DOI 10.1007/978-3-642-32313-3, pp. 109-124, 2012.
- [28] P. V. G. F. a. C. B. E. H. Lehmann, "Neutron imaging—detector options and practical results," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 531, pp. 228-237, SEP 2004.
- [29] V. M. G. F. [22] V. Dangendorf, "Detectors for Energy-Resolved Fast Neutron Imaging," *Physikal Technische Bundesanstalt*, pp. 1-4, 2004.
- [30] R. B. Murray, "Use of Li6I as a Scintillation Detector and Spectrometer for Fast Neutrons," *Nuclear Instruments*, vol. 2, pp. 237-248, 1957.

- [31] J. K.-E. a. Y. S. Horowitz, " Heavy charged particle thermoluminescence dosimetry: Track structure theory and experiments," *The International Journal of Applied Radiation and Isotopes*, vol. 33, no. 11, pp. 1085-1100, NOV 1982.
- [32] C. L. N. J. C. a. Q. P. T. J. Hajagos, "High-Z Sensitized Plastic Scintillators: A Review," *Advanced Materials*, vol. 30, no. DOI: 10.1002/adma.201706956, p. 1706956, JUL 2018.
- [33] C. L. e. al., "Demonstration of coded-aperture fast-neutron imaging based on Timepix detector," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, no. DOI: 10.1016/j.nima.2018.10.051, OCT 2018.
- [34] J. K. P. M. L. R. M. J. M. S. D. C. a. S. A. P. M. C. Hamel, "Active neutron and gamma-ray imaging of highly enriched uranium for treaty verification," *Scientific Reports*, vol. 7, no. DOI: 10.1038/s41598-017-08253-x, DEC 2017.
- [35] P. H. S. M. J. M. J. M. B.R. Grogan, "Alpha Detector Pixelization Effects on NMIS Imaging," in *INMM*, Nashville, TN, 2008.
- [36] A. Sonzogni, "NuDat 2.7," National Nuclear Data Center, Brookhaven National Laboratory, 2019. [Online]. Available: <https://www.nndc.bnl.gov/nudat2/reCenter.jsp?z=55&n=82>.
- [37] S. McConchie, "Preliminary Acceptance Testing of the ING-27," Oak Ridge National Labs, Oak Ridge, 2015.
- [38] "Natioinal Nuclear Data Center Evaluated Nuclear Data File Retrieval and Plotting," Natioinal Nuclear Data Center, Brookhaven National Laboratory, 2011. [Online]. Available: <https://www.nndc.bnl.gov>. [Accessed 4 September 2019].
- [39] J. Birks, "Scintillations from Organic Crystals: Specific Flourescence and Relative Response to Different Radiations," *Proc. Phys. Soc. London Sect. A*, vol. 64, no. 10, 1951.
- [40] C. Chou, "The Nature of the Saturation Effect of Flourescent Scintillators," *Physics Rev.*, vol. 87, no. 5, p. 904, 1952.
- [41] J. C. M. Z. a. J. C. M. Berger, "Stopping-power and range tables for electrons, protons, and helium ions," NIST, 2015. [Online]. Available: <http://physics.nist.gov>.
- [42] C. B. J. K. Clell J. Solomon, "The MCNPTools Package: Installation and Use," 30 March 2017. [Online]. Available: https://mcnp.lanl.gov/pdf_files/la-ur-17-21779.pdf. [Accessed 15 JUL 2019].
- [43] Y. Tsuneoka, "C12 Reaction Induced by 14.8 MeV Neutrons," *Journal of the Physical Society of Japan*, vol. 14, 1959.
- [44] J. D. Stryon, *Time and Energy Characterization of a Neutron Time of Flight Detector Using a Novel Coincidence Method for Constraining Neutron Yield, Ion Temperature, and Liner Density Measurements from MAGLIF Experiments*, Albuquerque: University of New Mexico, 2017.
- [45] G. F. Knoll, *Radiation Detection and Measurement*, J. Welter, Ed., Hobonken, NJ: John Wiley & Sons Inc., 2010.

- [46] K. E. Pena, "Template-Based Imaging Analysis of Arbitrary Nuclear Material Configurations Using Time and Directionally-Tagged Fast Neutrons," *Tennessee Research and Creative Exchange*, 2015.
- [47] H. M. K. Robert S. Norris, "Global Nuclear Stockpiles, 1945-2006," *Bulletin of the Atomic Scientists*, vol. 4, pp. 64-66, 2006.
- [48] A. Rising, World Nuclear Performance Report, vol. 4, London, England: World Nuclear Association, 2018.
- [49] K. Davenport, "Nuclear Weapons Fact Sheets & Briefs," Arms Control Association, Washington, DC, 2019.
- [50] W. a. E. S. Section, "Estimation of Global Inventories of Radioactive Waste and Other Radioactive Materials," International Atomic Energy Agency, Vienna, Austria, 2008.
- [51] O. o. t. Historian, "The Collaspe of the Soviet Union," United States Department of State, Washington, DC, 2019.
- [52] W. T. O. a. W. Bloem, "The Rise and Fall of ISIS From Evitability to Inevitability," The Hague Centre for Strategic Studies, The Hauge Netherlands, 2017.
- [53] A. Haynes, The Gunpowder Plot: Faith in Rebellion, London: Sutton Publications, 1994.

APPENDICES

APPENDIX: A – LABORATORY SET UP

Following assembly of all equipment in the field, the alignment procedures begin. Step one is to conduct a general alignment of the ING-27 with the detector arm utilizing the APNIS 115cm aluminum alignment arm. This lightweight alignment device was the primary alignment method for the APNIS. Once the general alignment is complete fine tune alignments can begin.

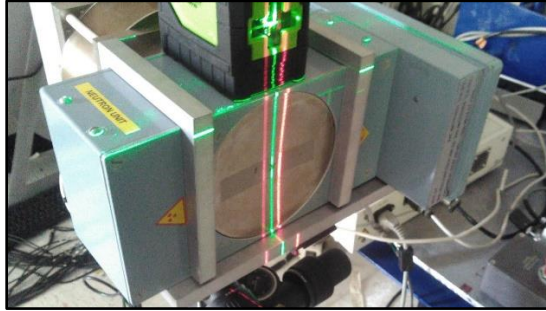


This second step properly positions two three-axis laser levels used for offset measurements. Each of these laser levels projects a cross perpendicular to each respective plane. Each laser is self-leveling and projects a flashing beam until the point it has self-leveled. Only after achieving a level position will the beam become solid red.

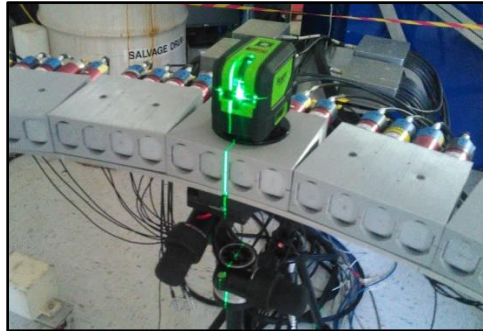


These beams project directly onto the floor and the face and the ING-27. Pre-measured tick marks on the floor are then used to offset the assembly or generator left or right as required for experiments. The alignment beams cross at a point approximately 2cm inside the generator – corresponding to the location of the beam point of origin. As

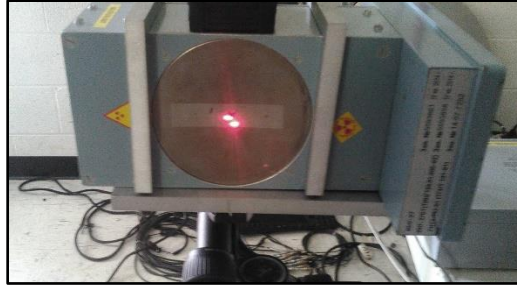
such they leave a small gap centered on the face of the ING-27. A third green laser projected from the center of the arm bisects this gap and the center of the ING-27.



Step four requires the centering of the ING-27 beam onto the center of the detector arm. The generator is positioned by rotating the camera platform it is mounted onto. The generator is centered once the projected green beam splits the detector banks. This is done while ensuring the beam dissection on the face of the generator has not move across the face of the generator.



Step five is distance verification utilizing two HT-60 laser range finders (accurate to $\pm 2\text{mm}$ at 50 meters). These range finders continuously read out the distance to the ING-27 and produce a steady auditory chirp whose frequency changes depending on the distance to the ING-27. They also project a red laser dot. Note one is slightly higher than the other due to the location of the beam projection being offset. The two dots are centered on the face of the ING-27.



Typical set up requires 5 to 10 minutes. Step one: general alignment is only required when one wishes to verify the self-leveling lasers have not drifted. As a rule, one should start with step one at the beginning of each day in the lab. Typically, no shifts occur on their own yet there have been instances of someone bumping the lasers and nullifying proper intersection and alignment.

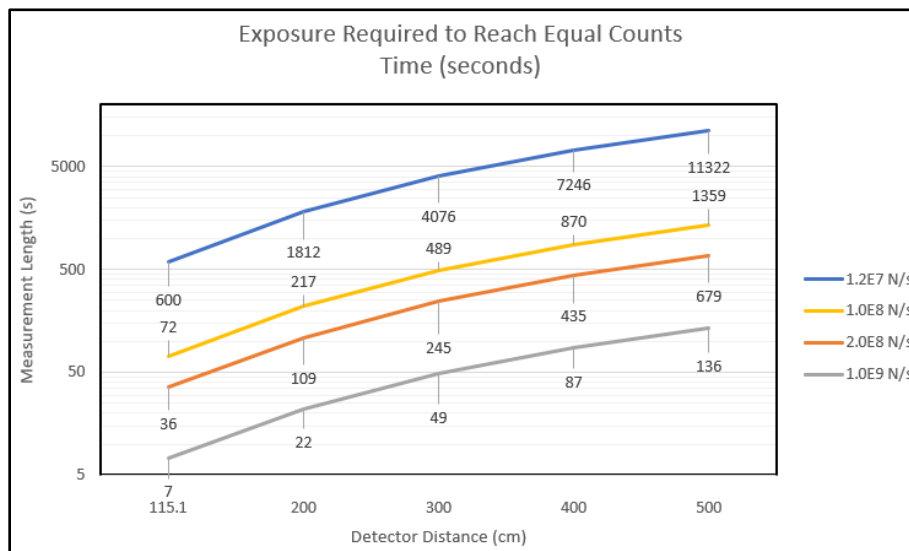
APPENDIX: B – OBSTRUCTED VIEW SET UP

The current system remains bound to the laboratory for several reasons. First it requires a power supply. Second the VME crate has not been ruggedized and exposed electronics could be ruined by any appreciable amount of moisture. Finally, it would require the regulatory authorization to use a mobile neutron generator on the ORNL campus along with proper postings. Assuming all these obstacles were to be overcome mobile imaging would follow these steps.

First the DT generator would be positioned in such a manner that a portion of the tagged neutron beam would extend beyond the bounds of each side of the interrogated object. Next the DT generator would be horizontally and vertically leveled centered on the location of interest. Two additional self-leveling lasers – mounted on tripods – would be positioned alongside and on top of the interrogated object. These would allow for $\hat{x}$ and $\hat{z}$ axis alignment of the detector arm.

Utilizing these two beams the detector arm would then be positioned on the far side of the interrogated object. After the detector arm is in position it would then be translated in the $\hat{y}$ plane utilizing a HT-60 laser range finder to determine distance to the DT Generator. Since the path would be obstructed by the interrogated object two HT-60 laser range finders would be employed. One would measure distance from the DT Generator to the interrogated object. The other would measure distance from the detector arm to the interrogated object. The correct distance would then be calculated by taking the two distances measured and adding in the size of the interrogated object to obtain the desired distance of separation.

Ideally, DT Generator to detector distance would be kept at 115.1 cm. However, larger objects may require greater distances. These larger objects will require larger detector arms – to cover the same solid angle – and lengthened measurement times – due to the decrease in neutron flux. For example, at 300 cm 73 detectors would be required, at 400 cm 97 detectors would be required, and at 500 cm 122 detectors would be required. A more powerful DT Generator would help offset this distance due to increased source strength. See below for highlights of these potential tradeoffs.



No void measurement would be collected in the field. Instead, the obstructed results would be compared to a void measurement collected prior to deployment. Detector drift does not appear to be an issue if the void is collected within five months of use. However, jostling and disconnection of components during transportation has not been tested and may result in significant detector drift.

APPENDIX: C – MCNP6.2 DETECTOR GENERATION CODE

```
#C [CENTER ONE] SCINTILLATOR BLOCK #1
#11 RPP 0 2.54 0.0 2.54 -10.16 0.0
    $ 2.54 cm x 2.54 cm x 10.16 cm Plastic Scintillator R1
#12 RPP 2.541 5.08 0.0 2.54 -10.16 0.0
    $ 2.54 cm x 2.54 cm x 10.16 cm Plastic Scintillator R2
#13 RPP -2.54 -.001 0.0 2.54 -10.16 0.0
    $ 2.54 cm x 2.54 cm x 10.16 cm Plastic Scintillator L1
#14 RPP -5.08 -2.541 0.0 2.54 -10.16 0.0
    $ 2.54 cm x 2.54 cm x 10.16 cm Plastic Scintillator L2
#18 RPP -5.081 5.081 -.01 3.81 -17.78 .02
    $ 10.16 x 3.81 cm x 17.80 cm Plastic holding box #1 [inside]
#19 RPP -5.082 5.082 -.02 3.82 -17.79 .03
    $ 1 mm thick plastic holding box #1 [outside]

RPP11 = [-6.5, -3.4, -11.9, 0, -1.5, 1.5]
RPP12 = [-3.35, -0.025, -11.9, 0, -1.5, 1.5]
RPP13 = [0.025, 3.35, -11.9, 0, -1.5, 1.5]
RPP14 = [3.4, 6.5, -11.9, 0, -1.5, 1.5]
RPP18 = [-6.6, 6.6, -12, 0.1, -1.51, 1.51]
RPP19 = [-6.8, 6.8, -12.1, 0.2, -1.52, 1.52]

# Make more rows of detector_locations directly next to SCINTILLATOR BLOCK #1

a=1.159
b=4.7095
c=10.8631
d=20.1034

XSHIFT2 = [0, 0, a, a, 0, 0]
RPP21 = [sum(x) for x in zip(RPP11,XSHIFT2)]
RPP22 = [sum(x) for x in zip(RPP12,XSHIFT2)]
RPP23 = [sum(x) for x in zip(RPP13,XSHIFT2)]
RPP24 = [sum(x) for x in zip(RPP14,XSHIFT2)]
RPP28 = [sum(x) for x in zip(RPP18,XSHIFT2)]
RPP29 = [sum(x) for x in zip(RPP19,XSHIFT2)]

XSHIFT3 = [0, 0, b, b, 0, 0]
RPP31 = [sum(x) for x in zip(RPP11,XSHIFT3)]
RPP32 = [sum(x) for x in zip(RPP12,XSHIFT3)]
RPP33 = [sum(x) for x in zip(RPP13,XSHIFT3)]
RPP34 = [sum(x) for x in zip(RPP14,XSHIFT3)]
RPP38 = [sum(x) for x in zip(RPP18,XSHIFT3)]
RPP39 = [sum(x) for x in zip(RPP19,XSHIFT3)]
```

XSHIFT4 = [0, 0, c, c, 0, 0]	# This box is right 3 x
RPP41 = [sum(x) for x in zip(RPP11,XSHIFT4)]	
RPP42 = [sum(x) for x in zip(RPP12,XSHIFT4)]	
RPP43 = [sum(x) for x in zip(RPP13,XSHIFT4)]	
RPP44 = [sum(x) for x in zip(RPP14,XSHIFT4)]	
RPP48 = [sum(x) for x in zip(RPP18,XSHIFT4)]	
RPP49 = [sum(x) for x in zip(RPP19,XSHIFT4)]	
XSHIFT5 = [0, 0, d, d, 0, 0]	# This box is right 4 ...x
RPP51 = [sum(x) for x in zip(RPP11,XSHIFT5)]	
RPP52 = [sum(x) for x in zip(RPP12,XSHIFT5)]	
RPP53 = [sum(x) for x in zip(RPP13,XSHIFT5)]	
RPP54 = [sum(x) for x in zip(RPP14,XSHIFT5)]	
RPP58 = [sum(x) for x in zip(RPP18,XSHIFT5)]	
RPP59 = [sum(x) for x in zip(RPP19,XSHIFT5)]	
XSHIFT6 = [0, 0, a, a, 0, 0]	# This box is left 1
RPP61 = [sum(x) for x in zip(RPP11,XSHIFT6)]	
RPP62 = [sum(x) for x in zip(RPP12,XSHIFT6)]	
RPP63 = [sum(x) for x in zip(RPP13,XSHIFT6)]	
RPP64 = [sum(x) for x in zip(RPP14,XSHIFT6)]	
RPP68 = [sum(x) for x in zip(RPP18,XSHIFT6)]	
RPP69 = [sum(x) for x in zip(RPP19,XSHIFT6)]	
XSHIFT7 = [0, 0, b, b, 0, 0]	# This box is left 2
RPP71 = [sum(x) for x in zip(RPP11,XSHIFT7)]	
RPP72 = [sum(x) for x in zip(RPP12,XSHIFT7)]	
RPP73 = [sum(x) for x in zip(RPP13,XSHIFT7)]	
RPP74 = [sum(x) for x in zip(RPP14,XSHIFT7)]	
RPP78 = [sum(x) for x in zip(RPP18,XSHIFT7)]	
RPP79 = [sum(x) for x in zip(RPP19,XSHIFT7)]	
XSHIFT8 = [0, 0, c, c, 0, 0]	# This box is left 3
RPP81 = [sum(x) for x in zip(RPP11,XSHIFT8)]	
RPP82 = [sum(x) for x in zip(RPP12,XSHIFT8)]	
RPP83 = [sum(x) for x in zip(RPP13,XSHIFT8)]	
RPP84 = [sum(x) for x in zip(RPP14,XSHIFT8)]	
RPP88 = [sum(x) for x in zip(RPP18,XSHIFT8)]	
RPP89 = [sum(x) for x in zip(RPP19,XSHIFT8)]	
XSHIFT9 = [0, 0, d, d, 0, 0]	# This box is left 4
RPP91 = [sum(x) for x in zip(RPP11,XSHIFT9)]	
RPP92 = [sum(x) for x in zip(RPP12,XSHIFT9)]	
RPP93 = [sum(x) for x in zip(RPP13,XSHIFT9)]	

```
RPP94 = [sum(x) for x in zip(RPP14,XSHIFT9)]
RPP98 = [sum(x) for x in zip(RPP18,XSHIFT9)]
RPP99 = [sum(x) for x in zip(RPP19,XSHIFT9)]
```

with open("detector_locations_rotated.txt","w") as f:

```
f.write('C Center Box 0\n')
f.write('11 1 RPP '+str(RPP11)+'\n')
f.write('12 1 RPP '+str(RPP12)+'\n')
```

```
f.write('13 1 RPP '+str(RPP13)+'\n')
f.write('14 1 RPP '+str(RPP14)+'\n')
f.write('18 1 RPP '+str(RPP18)+'\n')
f.write('19 1 RPP '+str(RPP19)+'\n')
f.write('C Right Box 1\n')
f.write('21 2 RPP '+str(RPP21)+'\n')
f.write('22 2 RPP '+str(RPP22)+'\n')
```

```
f.write('23 2 RPP '+str(RPP23)+'\n')
f.write('24 2 RPP '+str(RPP24)+'\n')
f.write('28 2 RPP '+str(RPP28)+'\n')
f.write('29 2 RPP '+str(RPP29)+'\n')
f.write('C Right Box 2\n')
f.write('31 3 RPP '+str(RPP31)+'\n')
```

```
f.write('32 3 RPP '+str(RPP32)+'\n')
f.write('33 3 RPP '+str(RPP33)+'\n')
f.write('34 3 RPP '+str(RPP34)+'\n')
f.write('38 3 RPP '+str(RPP38)+'\n')
f.write('39 3 RPP '+str(RPP39)+'\n')
f.write('C Right Box 3\n')
f.write('41 4 RPP '+str(RPP41)+'\n')
f.write('42 4 RPP '+str(RPP42)+'\n')
```

```
f.write('43 4 RPP '+str(RPP43)+'\n')
f.write('44 4 RPP '+str(RPP44)+'\n')
f.write('48 4 RPP '+str(RPP48)+'\n')
f.write('49 4 RPP '+str(RPP49)+'\n')
f.write('C Right Box 4\n')
f.write('51 5 RPP '+str(RPP51)+'\n')
```

```
f.write('52 5 RPP '+str(RPP52)+'\n')
f.write('53 5 RPP '+str(RPP53)+'\n')
f.write('54 5 RPP '+str(RPP54)+'\n')
f.write('58 5 RPP '+str(RPP58)+'\n')
f.write('59 5 RPP '+str(RPP59)+'\n')
```

```

f.write('C Left Box 1 \n')
f.write('61 6 RPP '+str(RPP61)+ '\n')

f.write('62 6 RPP '+str(RPP62)+ '\n')
f.write('63 6 RPP '+str(RPP63)+ '\n')
f.write('64 6 RPP '+str(RPP64)+ '\n')
f.write('68 6 RPP '+str(RPP68)+ '\n')
f.write('69 6 RPP '+str(RPP69)+ '\n')
f.write('C Left Box 2 \n')
f.write('71 7 RPP '+str(RPP71)+ '\n')

f.write('72 7 RPP '+str(RPP72)+ '\n')
f.write('73 7 RPP '+str(RPP73)+ '\n')
f.write('74 7 RPP '+str(RPP74)+ '\n')
f.write('78 7 RPP '+str(RPP78)+ '\n')
f.write('79 7 RPP '+str(RPP79)+ '\n')
f.write('C \n')
f.write('C Left Box 3 \n')
f.write('81 8 RPP '+str(RPP81)+ '\n')

f.write('82 8 RPP '+str(RPP82)+ '\n')
f.write('83 8 RPP '+str(RPP83)+ '\n')
f.write('84 8 RPP '+str(RPP84)+ '\n')
f.write('88 8 RPP '+str(RPP88)+ '\n')
f.write('89 8 RPP '+str(RPP89)+ '\n')
f.write('C \n')
f.write('C Left Box 4 \n')
f.write('91 9 RPP '+str(RPP91)+ '\n')

f.write('92 9 RPP '+str(RPP92)+ '\n')
f.write('93 9 RPP '+str(RPP93)+ '\n')
f.write('94 9 RPP '+str(RPP94)+ '\n')
f.write('98 9 RPP '+str(RPP98)+ '\n')
f.write('99 9 RPP '+str(RPP99)+ '\n')
f.write('C \n')
f.close()

s = open("detector_locations_rotated.txt").read()
s = s.replace('[', '')
sss = s.replace(',', '')
f = open("detector_locations_rotated.txt", 'w')
f.write(sss)
f.close()

s = open("detector_locations_rotated.txt").read()

```

```
s = s.replace(']', ' ')
f = open("detector_locations_rotated.txt", 'w')
f.write(s)
f.close()

s = open("detector_locations_rotated.txt").read()
print(s)
```

APPENDIX: D – MCNP6.2 TRANSFORMATION CARD

GENERATION CODE

```
#C [CENTER ONE] SCINTILLATOR BLOCK #1
#*TR2 0 0 0 4.6 85.4 90 94.6 4.6 90 90 90 0 $ transform # 2 - for 1.51 M
#*TR3 0 0 0 9.1 80.9 90 99.1 9.1 90 90 90 0 $ transform # 3 - "
#*TR4 0 0 0 13.8 76.2 90 103.8 13.8 90 90 90 0 $ transform # 4 - "
#*TR5 0 0 0 4.6 94.6 90 85.4 4.6 90 90 90 0 $ transform # 3 - "
#*TR6 0 0 0 9.1 99.1 90 80.9 9.1 90 90 90 0 $ transform # 3 - "
#*TR7 0 0 0 13.8 103.8 90 76.2 13.8 90 90 90 0 $ transform # 3 - "
```

```
TR1= [0, 90, 90, 90, 0, 90, 90, 90, 0]
TR2= [0, 90, 90, 90, 0, 90, 90, 90, 0]
TR3= [0, 90, 90, 90, 0, 90, 90, 90, 0]
TR4= [0, 90, 90, 90, 0, 90, 90, 90, 0]
TR5= [0, 90, 90, 90, 0, 90, 90, 90, 0]
TR6= [0, 90, 90, 90, 0, 90, 90, 90, 0]
TR7= [0, 90, 90, 90, 0, 90, 90, 90, 0]
```

```
# THESE ARE ONLY DEGREE CHANGES - NO CHANGE TO X,Y,Z
COORDINATES HERE
```

```
# THAT IS DONE ON detector_location_generation_rotated.py card
```

```
# For 3cm x 3 cm x 12cm detectors mounted at 16cm intervals
# 7.9|15.9|24.3
# For arm with old detector profile
# 4.6|9.1|13.8 for 1.51 meters
# 3.4|6.9|10.4 for 2.00 meters
# 2.3|4.6|6.9| for 3.00 meters
rotate = 0
```

```
a=7.9
b=15.9
c=24.3
```

```
#
SHIFT1 = [0,0,0,0,0,0,0,0,0]
SHIFT2 = [+a, -a, 0, +a, +a, 0, 0, 0, 0]
SHIFT3 = [+b, -b, 0, +b, +b, 0, 0, 0, 0]
SHIFT4 = [+c, -c, 0, +c, +c, 0, 0, 0, 0]
SHIFT5 = [-a, +a, 0, -a, -a, 0, 0, 0, 0]
SHIFT6 = [-b, +b, 0, -b, -b, 0, 0, 0, 0]
SHIFT7 = [-c, +c, 0, -c, -c, 0, 0, 0, 0]
#
```

```

Transform1 = [sum(x) for x in zip(TR1,SHIFT1)]
Transform2 = [sum(x) for x in zip(TR2,SHIFT2)]
Transform3 = [sum(x) for x in zip(TR3,SHIFT3)]
Transform4 = [sum(x) for x in zip(TR4,SHIFT4)]
Transform5 = [sum(x) for x in zip(TR5,SHIFT5)]
Transform6 = [sum(x) for x in zip(TR6,SHIFT6)]
Transform7 = [sum(x) for x in zip(TR7,SHIFT7)]
#

# Move left (-x) and right (+x)
x_axis_shift = 0
TF1X = 0 + x_axis_shift
TF2X = 15 + x_axis_shift # defaults 15
TF3X = 30 + x_axis_shift # 30
TF4X = 47 + x_axis_shift # 47
TF5X = -15 + x_axis_shift # -15
TF6X = -30 + x_axis_shift # -30
TF7X = -47 + x_axis_shift # -47

# Move towards DT Gen (-y) and away from DT Gen (+y)
# Set y_axis_shift = 0 if no rotation in this plane
y_axis_shift = 0

# bring right side forward: 1
# bring left side forward: 2
right_side_closer = 1

if right_side_closer == 1:
    TF1Y = 0
    TF2Y = 0 + 1.00*y_axis_shift
    TF3Y = 0 + 2.00*y_axis_shift
    TF4Y = 0 + 3.00*y_axis_shift
    TF5Y = 0 - 1.00*y_axis_shift
    TF6Y = 0 - 2.00*y_axis_shift
    TF7Y = 0 - 3.00*y_axis_shift
elif right_side_closer == 2:
    TF1Y = 0
    TF2Y = 0 - 1.00*y_axis_shift
    TF3Y = 0 - 2.00*y_axis_shift
    TF4Y = 0 - 3.00*y_axis_shift
    TF5Y = 0 + 1.00*y_axis_shift
    TF6Y = 0 + 2.00*y_axis_shift
    TF7Y = 0 + 3.00*y_axis_shift
else:
    print("Error Must set right_side_closer to 1 or 2")

```

```

# Write the Output File
with open("TR_card.txt","w") as f:
    f.write('C Transformation Cards\n')
    f.write('*TR1 ' +str(TF1X)+ ' ' +str(TF1Y)+ ' 0 ' +str(Transform1)+ '\n')
    f.write('*TR2 ' +str(TF2X)+ ' ' +str(TF2Y)+ ' 0 ' +str(Transform2)+ '\n')
    f.write('*TR3 ' +str(TF3X)+ ' ' +str(TF3Y)+ ' 0 ' +str(Transform3)+ '\n')
    f.write('*TR4 ' +str(TF4X)+ ' ' +str(TF4Y)+ ' 0 ' +str(Transform4)+ '\n')
    f.write('*TR5 ' +str(TF5X)+ ' ' +str(TF5Y)+ ' 0 ' +str(Transform5)+ '\n')
    f.write('*TR6 ' +str(TF6X)+ ' ' +str(TF6Y)+ ' 0 ' +str(Transform6)+ '\n')
    f.write('*TR7 ' +str(TF7X)+ ' ' +str(TF7Y)+ ' 0 ' +str(Transform7)+ '\n')
    f.write('C \n')
f.close()

s = open("TR_card.txt").read()
s = s.replace('[', ' ')
sss = s.replace(',', ' ')
f = open("TR_card.txt", 'w')
f.write(sss)
f.close()

s = open("TR_card.txt").read()
s = s.replace(']', ' ')
f = open("TR_card.txt", 'w')
f.write(s)
f.close()

s = open("TR_card.txt").read()
print(s)

```

APPENDIX: E – MCNP6.2 NEUTRON SOURCE GENERATION CODE

```
import numpy as np
import pandas as pd
import math
import csv
from itertools import islice
import os
from sklearn import preprocessing

# read in Alpha Pixel csv info and set DT to detector distance for this SDEF Deck:
Alpha_pixel_number = 1
DT_distance = 115.1
# Every in Alpha Pixel SDEF Card is created in this while loop:
while Alpha_pixel_number <=13:
    with open('sigma_values_for_Alphas_ING_27_Double_Gaussian.csv') as csvDataFile:
        csvReader = csv.reader(csvDataFile)
        data=list(csv.reader(csvDataFile))
        radians_to_degrees = math.pi/180.0
        xmu_value = float(data[Alpha_pixel_number][0])
        xmu = DT_distance*math.tan(radians_to_degrees*xmu_value)
        ymu = float(data[Alpha_pixel_number][1])
        A1 = float(data[Alpha_pixel_number][2])
        A2 = float(data[Alpha_pixel_number][3])
        xsig1 = float(data[Alpha_pixel_number][4])
        xsig2 = float(data[Alpha_pixel_number][5])
        ysig1 = float(data[Alpha_pixel_number][6])
        ysig2 = float(data[Alpha_pixel_number][7])

        # create PDF and write resulting intensity to PDF column of CSV

        data = pd.read_csv('Gaussian_Vector_Field.csv', delimiter=',')

        data["PDF"] = A1*np.exp(-(((data['X']-xmu)**2/(2*xsig1**2)+(data['Z']-ymu)**2/(2*ysig1**2))))+A2*np.exp(-(((data['X']-xmu)**2/(2*xsig2**2)+(data['Z']-ymu)**2/(2*ysig2**2))))
        df = pd.DataFrame(data)
        df["PDF Normalized"] = df["PDF"]/df["PDF"].sum()

    data.to_csv('Gaussian_Vector_Field_With_Intensities'+str(Alpha_pixel_number)+'.csv')

    # create SDEF source card for MCNP input deck
```

```

with open('SDEF_Card_'+ str(Alpha_pixel_number)+'.txt', 'w') as f2:
    with
open('Gaussian_Vector_Field_With_Intensities'+str(Alpha_pixel_number)+'.csv',
mode='r') as infile:
    reader = csv.reader(infile)
    header = next(reader) # store the headers and advance reader pointer
    linecount = 0
    f2.write('C Alpha Pixel:'+ str(Alpha_pixel_number) +'\n')
    f2.write('SDEF   POS=0 '+str(DT_distance)+' 0 ERG=14.1 PAR=1\n')
    f2.write('      DIR=D1 VEC=D2\n')
    f2.write('C location | energy | neutron | cos of the angle between VEC and the
particles direction of flight\n')
    f2.write('C in this case .5 degrees\n')
    f2.write('C VEC of flight\n')
    f2.write('SI1 0.999961923100 1.000000000000\n')
    f2.write('SP1 0.000000000000 1.000000000000\n')
    f2.write('SI2 L')
    for row in reader:
        f2.write(' '+row[5]+' '+row[6]+' '+row[7]+' \n')
    f2.write('SP2 D')
    infile.seek(0) # reset the CSV reader to the beginning of the page
    header = next(reader) # store the headers and advance reader pointer
    for row in reader:
        f2.write(' '+row[9]+' \n')
    # now move onto next alpha pixel
    Alpha_pixel_number +=1

```

APPENDIX: F – MCNP6.2 INPUT DECK GENERATION CODE

```
import os
import shutil
import subprocess
import itertools
from collections import deque
import re

# First Define Which Trial This Is:
trial = 49
# Create directories:
def main():
    dirName1 = '/Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a1'
    dirName2 = '/Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a2'
    dirName3 = '/Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a3'
    dirName4 = '/Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a4'
    dirName5 = '/Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a5'
    dirName6 = '/Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a6'
    dirName7 = '/Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a7'
    dirName8 = '/Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a8'
    dirName9 = '/Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a9'
    dirName10 = '/Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a10'
    dirName11 = '/Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a11'
    dirName12 = '/Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a12'
    dirName13 = '/Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a13'
    os.mkdir(dirName1)
    os.mkdir(dirName2)
    os.mkdir(dirName3)
    os.mkdir(dirName4)
    os.mkdir(dirName5)
    os.mkdir(dirName6)
    os.mkdir(dirName7)
    os.mkdir(dirName8)
    os.mkdir(dirName9)
    os.mkdir(dirName10)
    os.mkdir(dirName11)
    os.mkdir(dirName12)
    os.mkdir(dirName13)

# Call Function to Create directories:
if __name__ == '__main__':
    main()

# make copies of MCNP default input deck and SDEF Cards:
```

```

files = ["/Users/5bt/Desktop/Research Trials
3/Simulations/"+str(trial)+"/Default_"+str(trial)+".txt"]

for f in files:
    shutil.copy(f, '/Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a1/')
    shutil.copy(f, '/Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a2/')
    shutil.copy(f, '/Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a3/')
    shutil.copy(f, '/Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a4/')
    shutil.copy(f, '/Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a5/')
    shutil.copy(f, '/Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a6/')
    shutil.copy(f, '/Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a7/')
    shutil.copy(f, '/Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a8/')
    shutil.copy(f, '/Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a9/')
    shutil.copy(f, '/Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a10/')
    shutil.copy(f, '/Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a11/')
    shutil.copy(f, '/Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a12/')
    shutil.copy(f, '/Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a13/')

SDEF_Card1 = ["/Users/5bt/Desktop/Research Trials
3/Cards/0_SDEF_Generation_Card/SDEF_Card_1.txt"]
for g in SDEF_Card1:
    shutil.copy(g, '/Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a1/')

SDEF_Card2 = ["/Users/5bt/Desktop/Research Trials
3/Cards/0_SDEF_Generation_Card/SDEF_Card_2.txt"]
for g in SDEF_Card2:
    shutil.copy(g, '/Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a2/')

SDEF_Card3 = ["/Users/5bt/Desktop/Research Trials
3/Cards/0_SDEF_Generation_Card/SDEF_Card_3.txt"]
for g in SDEF_Card3:
    shutil.copy(g, '/Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a3/')

SDEF_Card4 = ["/Users/5bt/Desktop/Research Trials
3/Cards/0_SDEF_Generation_Card/SDEF_Card_4.txt"]
for g in SDEF_Card4:
    shutil.copy(g, '/Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a4/')

SDEF_Card5 = ["/Users/5bt/Desktop/Research Trials
3/Cards/0_SDEF_Generation_Card/SDEF_Card_5.txt"]
for g in SDEF_Card5:
    shutil.copy(g, '/Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a5/')

SDEF_Card6 = ["/Users/5bt/Desktop/Research Trials
3/Cards/0_SDEF_Generation_Card/SDEF_Card_6.txt"]

```

```

for g in SDEF_Card6:
    shutil.copy(g, 'Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a6/')

SDEF_Card7 = ["/Users/5bt/Desktop/Research Trials
3/Cards/0_SDEF_Generation_Card/SDEF_Card_7.txt"]
for g in SDEF_Card7:
    shutil.copy(g, 'Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a7/')

SDEF_Card8 = ["/Users/5bt/Desktop/Research Trials
3/Cards/0_SDEF_Generation_Card/SDEF_Card_8.txt"]
for g in SDEF_Card8:
    shutil.copy(g, 'Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a8/')

SDEF_Card9 = ["/Users/5bt/Desktop/Research Trials
3/Cards/0_SDEF_Generation_Card/SDEF_Card_9.txt"]
for g in SDEF_Card9:
    shutil.copy(g, 'Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a9/')

SDEF_Card10 = ["/Users/5bt/Desktop/Research Trials
3/Cards/0_SDEF_Generation_Card/SDEF_Card_10.txt"]
for g in SDEF_Card10:
    shutil.copy(g, 'Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a10/')

SDEF_Card11 = ["/Users/5bt/Desktop/Research Trials
3/Cards/0_SDEF_Generation_Card/SDEF_Card_11.txt"]
for g in SDEF_Card11:
    shutil.copy(g, 'Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a11/')

SDEF_Card12 = ["/Users/5bt/Desktop/Research Trials
3/Cards/0_SDEF_Generation_Card/SDEF_Card_12.txt"]
for g in SDEF_Card12:
    shutil.copy(g, 'Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a12/')

SDEF_Card13 = ["/Users/5bt/Desktop/Research Trials
3/Cards/0_SDEF_Generation_Card/SDEF_Card_13.txt"]
for g in SDEF_Card13:
    shutil.copy(g, 'Users/5bt/Desktop/Research Trials 3/Simulations/'+str(trial)+'a13/')

# Now Begin Creating READY_TO_RUN.txt for MCNP

# Now define the function that runs in each subfolder a1 to a15 and makes
READY_TO_RUN input decks
def main2():
    #First Copy lines from default to point of insertion
    search_str = "C INSERT_PIXEL_DATA"

```

```

search_file = open("/Users/5bt/Desktop/Research Trials
3/Simulations/"+str(trial)+"/a"+str(A_value)+"/Default_"+str(trial)+".txt", "r")
counter = 1

for line in search_file:
    counter += 1
    if line.strip() == search_str:
        Keyword = counter

search_file.close()

#loop that copies file line by line and terminates loop when i reaches the Keyword line
i = 0
f = open("/Users/5bt/Desktop/Research Trials
3/Simulations/"+str(trial)+"/a"+str(A_value)+"/Default_"+str(trial)+".txt", "r")
copy = open("/Users/5bt/Desktop/Research Trials
3/Simulations/"+str(trial)+"/a"+str(A_value)+"/READY_TO_RUN.txt", "wt")
while i < Keyword:
    line = f.readline()
    copy.write(str(line))
    i = i + 1
f.close()
copy.close()

# now insert SDEF for specific scenario
fin = open("/Users/5bt/Desktop/Research Trials
3/Simulations/"+str(trial)+"/a"+str(A_value)+"/SDEF_Card_"+str(A_value)+".txt", "r")
data2 = fin.read()
fin.close()
fout = open("/Users/5bt/Desktop/Research Trials
3/Simulations/"+str(trial)+"/a"+str(A_value)+"/READY_TO_RUN.txt", "a")
fout.write(data2)
fout.close()

copy = open("/Users/5bt/Desktop/Research Trials
3/Simulations/"+str(trial)+"/a"+str(A_value)+"/READY_TO_RUN.txt", "a")
f = open("/Users/5bt/Desktop/Research Trials
3/Simulations/"+str(trial)+"/a"+str(A_value)+"/Default_"+str(trial)+".txt", "r")
for line in f:
    while i > Keyword:
        line = f.readline()
        copy.write()
        i = i + 1
f.close()
copy.close()

```

```

# Now Determine how long the file is and subtract the location of the Keyword to
determine how many lines to copy from the end to the ready to run file
end_count = len(open("/Users/5bt/Desktop/Research Trials
3/Simulations/"+str(trial)+"/a"+str(A_value)+"/Default_"+str(trial)+".txt").readlines( ))
end_to_copy = end_count - Keyword

# Now Get the lines and write them
with open("/Users/5bt/Desktop/Research Trials
3/Simulations/"+str(trial)+"/a"+str(A_value)+"/Default_"+str(trial)+".txt",'r') as fin,
open("/Users/5bt/Desktop/Research Trials
3/Simulations/"+str(trial)+"/a"+str(A_value)+"/READY_TO_RUN.txt",'a') as fout:
    fout.writelines(deque(fin, end_to_copy))

# Now CALL MAIN2() and go through each subfolder and make READY_TO_RUN.txt
MCNP Input Deck
A_value = 1
A_max = 13

while A_value <= A_max:
    main2()
    A_value += 1

```

APPENDIX: G – PTRAC PARSER CODE

```
"""Main module."""
from mcnpertools import Ptrac, PtracEvent, PtracHistory, PtracNps
from sys import stdout
import os
import numpy as np
import pandas as pd
import collections
from pathlib import Path
import csv
import shutil
import math

# Trial #
Trail = 49
# alphas
A_value = 1
A_max = 13

def main():
    cells = []
    e0s = []
    e1s = []
    diff_e1s = [] # delta between initial energy and final energy e0s - e1s = diff_e1s
    times = []
    particle = []
    history_ID = []

    # explicitly open the file as a binary ptrac
    p = Ptrac("/Users/5bt/Desktop/Research Trials
3/Simulations/"+str(Trail)+"/a"+str(A_value)+"/ptrac", Ptrac.BIN_PTRAC)
    # read histories in batches of 10000
    hists = p.ReadHistories(10000)
    while hists:
        # loop over all histories
        for h in hists:
            # loop over all events in the history
            tmp = h.GetEvent(0) # not used
            e0 = 14.1

            for i in range(h.GetNumEvents()):
                ev = h.GetEvent(i)
                cell = ev.Get(Ptrac.CELL)
```

```

time = ev.Get(Ptrac.TIME)
collision_number = ev.Get(Ptrac.COLLISION_NUMBER)
surface = ev.Type() # == 3000 [not used]
collision = ev.Type() # == 4000
banked = ev.Type() # == 2000
terminated = ev.Type() # == 5000 [not used]
e1 = ev.Get(Ptrac.ENERGY)
particle_type = ev.Get(Ptrac.PARTICLE)
if ev.Has(Ptrac.ZAID):
    target = ev.Get(Ptrac.ZAID)
else:
    target = 0.00

in_detector = cell in range(11, 100)

# if the neutron hits a proton (1000)
if collision == 4000 and target == 1000 and particle_type == 1:
    cells.append(cell)
    e0s.append(e0)
    e1s.append(e1)
    times.append(time)
    particle.append(target)
    diff_e1s.append((e0-e1))
    history_ID.append(h)
    # Yes this technically means energy deposited is 14.1 MeV - the outgoing
energy of each neutron after collision.
    # This effectively is a summation count and is countered by the fact there is
only 1 count per history which removes any additional collisions.

# if the neutron hits a carbon atom (6000)
elif collision_number == 1 and target == 6000 and particle_type == 1:
    cells.append(cell)
    e0s.append(e0)
    e1s.append(e1)
    times.append(time)
    particle.append(target)
    diff_e1s.append((e0-e1))
    history_ID.append(h)

hists = p.ReadHistories(10000)
## save detector | initial energy | final energy | time in a CSV ##
## this is a event log of information. Not actually utilized further in the code##
a = np.asarray([ cells, e0s, e1s, diff_e1s, times, particle ])

```

```

np.savetxt("/Users/5bt/Desktop/Research Trials
3/Simulations/"+str(Trail)+"/a"+str(A_value)+"/parsed.csv", a, delimiter=",")

## remove counts where energy deposited is less than < 1 MeV in detector for
hydrogen and 8.30 MeV for carbon atom ##
## remove counts where particle is neutron particle = 1 ##
## NOTE: Time filter already implemented on PTRAC card of MCNP Input Deck
FILTER=2.0.2.5##
new_cells = []
new_diff_e1s = []
new_particle = []
new_history_ID = []

for cells, diff_e1s, particle, history_ID in zip(cells, diff_e1s, particle, history_ID):
    if diff_e1s >= 1.00 and particle == 1000:
        new_cells.append(cells)
        new_diff_e1s.append(diff_e1s)
        new_particle.append(particle)
        new_history_ID.append(history_ID)
    if diff_e1s >= 8.30 and particle == 6000:
        new_cells.append(cells)
        new_diff_e1s.append(diff_e1s)
        new_particle.append(particle)
        new_history_ID.append(history_ID)

particle_IDs_0 = pd.DataFrame([new_history_ID, new_cells, new_particle])
particle_IDs = particle_IDs_0.drop_duplicates()

pd.DataFrame(particle_IDs).to_csv("/Users/5bt/Desktop/Research Trials
3/Simulations/"+str(Trail)+"/a"+str(A_value)+"/particle_count_initial.csv",
header=False, index=False)
with open("/Users/5bt/Desktop/Research Trials
3/Simulations/"+str(Trail)+"/a"+str(A_value)+"/particle_count_initial.csv", 'r') as f:
    with open("/Users/5bt/Desktop/Research Trials
3/Simulations/"+str(Trail)+"/a"+str(A_value)+"/particle_count.csv", 'w') as f1:
        next(f) # skip header line
        next(f)
        for line in f:
            f1.write(line)
## read CSV and save counts for each particle
with open("/Users/5bt/Desktop/Research Trials
3/Simulations/"+str(Trail)+"/a"+str(A_value)+"/particle_count.csv", 'r') as f:
    contents = f.read()
    count_neutron = contents.count('1')
    count_photon = contents.count('2')

```

```

    count_electron = contents.count('3')
    count_carbon = contents.count('6000')
    count_positron = contents.count('8')
    count_proton = contents.count('1000')
f.close()

## Write the finalized particle counts in a TEXT FILE ##
with open("/Users/5bt/Desktop/Research Trials
3/Simulations/"+str(Trail)+"/a"+str(A_value)+"/particle_counts_final.txt","w+") as f:
    f.write('Neutron: '+str(count_neutron)+ '\n')
    f.write('Photon: '+str(count_photon)+ '\n')
    f.write('Electron: '+str(count_electron)+ '\n')
    f.write('Positron: '+str(count_positron)+ '\n')
    f.write('Proton: '+str(count_proton)+ '\n')
    f.write('Carbon recoil: '+str(count_carbon)+ '\n')
f.close()

## save cell array counts and history ID to csv ##
detection_array = pd.DataFrame([new_history_ID, new_cells])
unique_detections = detection_array.drop_duplicates()

pd.DataFrame(unique_detections).to_csv("/Users/5bt/Desktop/Research Trials
3/Simulations/"+str(Trail)+"/a"+str(A_value)+"/cell_array.csv")

# Delete first 2 rows and only keep "Cells"
with open("/Users/5bt/Desktop/Research Trials
3/Simulations/"+str(Trail)+"/a"+str(A_value)+"/cell_array.csv",'r') as f:
    with open("/Users/5bt/Desktop/Research Trials
3/Simulations/"+str(Trail)+"/a"+str(A_value)+"/cell_array_final.csv", 'w') as f1:
        next(f) # skip header line
        next(f)
        for line in f:
            f1.write(line)
## change "a component" save .csv as a .txt
ifile = open("/Users/5bt/Desktop/Research Trials
3/Simulations/"+str(Trail)+"/a"+str(A_value)+"/cell_array_final.csv", 'r')
ofile = open("/Users/5bt/Desktop/Research Trials
3/Simulations/"+str(Trail)+"/a"+str(A_value)+"/cell_array_final.txt", 'w')
findL = ['0']
replaceL = ['0']

rep = dict(zip(findL, replaceL))

def findReplace(find, replace):
    s = ifile.read()

```

```

    for item, replacement in zip(findL, replaceL):
        s = s.replace(item, replacement)
    ofile.write(s)

for item in findL:
    findReplace(item, rep[item])

ifile.close()
ofile.close()

## open the cell txt file and sum counts in each detector ##
with open("/Users/5bt/Desktop/Research Trials
3/Simulations/"+str(Trail)+"a"+str(A_value)+"/cell_array_final.txt") as f:
    contents = f.read()
    count11 = contents.count('11')
    count12 = contents.count('12')
    count13 = contents.count('13')
    count14 = contents.count('14')
    count21 = contents.count('21')
    count22 = contents.count('22')
    count23 = contents.count('23')
    count24 = contents.count('24')
    count31 = contents.count('31')
    count32 = contents.count('32')
    count33 = contents.count('33')
    count34 = contents.count('34')
    count41 = contents.count('41')
    count42 = contents.count('42')
    count43 = contents.count('43')
    count44 = contents.count('44')
    count51 = contents.count('51')
    count52 = contents.count('52')
    count53 = contents.count('53')
    count54 = contents.count('54')
    count61 = contents.count('61')
    count62 = contents.count('62')
    count63 = contents.count('63')
    count64 = contents.count('64')
    count71 = contents.count('71')
    count72 = contents.count('72')
    count73 = contents.count('73')
    count74 = contents.count('74')
f.close()
## Write the finalized detector counts in a TEXT FILE ##

```

```

with open("/Users/5bt/Desktop/Research Trials
3/Simulations/"+str(Trail)+"a"+str(A_value)+"/detector_counts.txt","w") as f:
    f.write(str(count71)+ '\n')
    f.write(str(count72)+ '\n')
    f.write(str(count73)+ '\n')
    f.write(str(count74)+ '\n')
    f.write(str(count61)+ '\n')
    f.write(str(count62)+ '\n')
    f.write(str(count63)+ '\n')
    f.write(str(count64)+ '\n')
    f.write(str(count51)+ '\n')
    f.write(str(count52)+ '\n')
    f.write(str(count53)+ '\n')
    f.write(str(count54)+ '\n')
    f.write(str(count11)+ '\n')
    f.write(str(count12)+ '\n')
    f.write(str(count13)+ '\n')
    f.write(str(count14)+ '\n')
    f.write(str(count21)+ '\n')
    f.write(str(count22)+ '\n')
    f.write(str(count23)+ '\n')
    f.write(str(count24)+ '\n')
    f.write(str(count31)+ '\n')
    f.write(str(count32)+ '\n')
    f.write(str(count33)+ '\n')
    f.write(str(count34)+ '\n')
    f.write(str(count41)+ '\n')
    f.write(str(count42)+ '\n')
    f.write(str(count43)+ '\n')
    f.write(str(count44)+ '\n')
f.close()

# remove unwanted files or keep for debugging purposes
os.remove("/Users/5bt/Desktop/Research Trials
3/Simulations/"+str(Trail)+"a"+str(A_value)+"/particle_count.csv')
os.remove("/Users/5bt/Desktop/Research Trials
3/Simulations/"+str(Trail)+"a"+str(A_value)+"/particle_count_initial.csv')
os.remove("/Users/5bt/Desktop/Research Trials
3/Simulations/"+str(Trail)+"a"+str(A_value)+"/cell_array.csv')
os.remove("/Users/5bt/Desktop/Research Trials
3/Simulations/"+str(Trail)+"a"+str(A_value)+"/cell_array_final.csv')
os.remove("/Users/5bt/Desktop/Research Trials
3/Simulations/"+str(Trail)+"a"+str(A_value)+"/cell_array_final.txt')
#os.remove("/Users/5bt/Desktop/Research Trials
3/Simulations/"+str(Trail)+"a"+str(A_value)+"/cell_array_and_energy_dep.csv')

```

```
os.remove("/Users/5bt/Desktop/Research Trials  
3/Simulations/"+str(Trail)+"/a"+str(A_value)+'/parsed.csv')  
#os.remove('cell_array_and_energy_dep.csv')  
  
# call main  
while A_value <= A_max:  
    main()  
    A_value += 1
```

APPENDIX: H – DETECTOR COUNTS SUMMATION CODE

```
"""Main module."""
from mcnpertools import Ptrac, PtracEvent, PtracHistory, PtracNps
from sys import stdout
import os
import numpy as np
import pandas as pd
import collections
from pathlib import Path
import csv
import shutil
import math

# Trial #
Trail = 50
# alphas
A_value = 1
A_max = 13
# placeholder rows for all alpha detectors
A1 = []
A2 = []
A3 = []
A4 = []
A5 = []
A6 = []
A7 = []
A8 = []
A9 = []
A10 = []
A11 = []
A12 = []
A13 = []

# define function to run
def main():
    if A_value <= A_max:
        filepath = '/Users/5bt/Desktop/Research Trials
3/Simulations/'+str(Trail)+'/'+'a'+str(A_value)+'/'+'detector_counts.txt'
        with open(filepath) as fp:
            if A_value == 1:
                line = fp.readline()
                while line:
                    A1.append(int(line))
                    line = fp.readline()
```

```

elif A_value == 2:
    line = fp.readline()
    while line:
        A2.append(int(line))
        line = fp.readline()
elif A_value == 3:
    line = fp.readline()
    while line:
        A3.append(int(line))
        line = fp.readline()
elif A_value == 4:
    line = fp.readline()
    while line:
        A4.append(int(line))
        line = fp.readline()
elif A_value == 5:
    line = fp.readline()
    while line:
        A5.append(int(line))
        line = fp.readline()
elif A_value == 6:
    line = fp.readline()
    while line:
        A6.append(int(line))
        line = fp.readline()
elif A_value == 7:
    line = fp.readline()
    while line:
        A7.append(int(line))
        line = fp.readline()
elif A_value == 8:
    line = fp.readline()
    while line:
        A8.append(int(line))
        line = fp.readline()
elif A_value == 9:
    line = fp.readline()
    while line:
        A9.append(int(line))
        line = fp.readline()
elif A_value == 10:
    line = fp.readline()
    while line:
        A10.append(int(line))
        line = fp.readline()

```

```

elif A_value == 11:
    line = fp.readline()
    while line:
        A11.append(int(line))
        line = fp.readline()
elif A_value == 12:
    line = fp.readline()
    while line:
        A12.append(int(line))
        line = fp.readline()
elif A_value == 13:
    line = fp.readline()
    while line:
        A13.append(int(line))
        line = fp.readline()
else:
    pass
# call main
while A_value <= A_max:
    main()
    A_value += 1

# write into file
a = np.asarray([ A1, A2, A3, A4, A5, A6, A7, A8, A9, A10, A11, A12, A13])
np.savetxt("All_detector_counts.csv", a, delimiter=",")
# now transpose it so it works with my results set up
pd.read_csv('All_detector_counts.csv',
header=None).T.to_csv('All_detector_counts_sums.csv', header=False, index=False)
#delete original csv
os.remove("All_detector_counts.csv")

```

APPENDIX: I – CORRELATION PARSER CODE

```
"""Main module."""
import os
import numpy as np
import pandas as pd
import collections
import csv
import shutil
import math
import heapq

##### SETUP PARAMETERS #####

# set up initial detector number
detector = 1
# how many nanoseconds wide is our correlation window?:
neutron_window = 5
# what is the filepath to the file you wish to parse?
filepath = "C:/Users/5bt/Desktop/D40 Results/20DEC19/-
5cm_Y_away_20DEC19.corr.csv"

# background window is 50 nanosecond or 50x100 picoseconds
background_window = 500

##### WORKHORSE #####
# since correlation windows are now 100E-12 seconds
one_hundred_ps_neutron_window = int(neutron_window/.1)

# define the dataframe
df = pd.read_csv(filepath, na_filter=False)

# Background for each detector pixel 1 ns window - sum first 10 values of each row
background = df.tail(background_window).mean()
df_3 = pd.DataFrame(background)
df_3.to_csv("background_counts.csv", index=False)

df_4 = pd.read_csv("background_counts.csv", na_filter=False)
print(float(df_4.iat[0,0]))
print(float(df_4.iat[1,0]))
print(str(df_4.iat[2,0]))
print(str(df_4.iat[3,0]))
print(str(df_4.iat[4,0]))
#sometimes one will need to adjust the starting position Q allows this
```

```

#all detectors essentially perform the same steps just in different cells
Q = 1
while detector <= 15:
if detector == 1:

    # D1 = Alpha Pixel 1 FOR detector pixels A1-A28 (1-28)
    D1_A1 = df.nlargest(one_hundred_ps_neutron_window, [' dtp01-
img01']).sum()
    D1_A1_total = int(float(D1_A1.iloc[[2,2]].max()) - float(df_4.iat[2-
Q,0])*one_hundred_ps_neutron_window)
    D1_A2 = df.nlargest(one_hundred_ps_neutron_window, [' dtp01-
img02']).sum()
    D1_A2_total = int(float(D1_A2.iloc[[2,3]].max()) - float(df_4.iat[3-
Q,0])*one_hundred_ps_neutron_window)
    D1_A3 = df.nlargest(one_hundred_ps_neutron_window, [' dtp01-
img03']).sum()
    D1_A3_total = int(float(D1_A3.iloc[[2,4]].max()) - float(df_4.iat[4-
Q,0])*one_hundred_ps_neutron_window)
    D1_A4 = df.nlargest(one_hundred_ps_neutron_window, [' dtp01-
img04']).sum()
    D1_A4_total = int(float(D1_A4.iloc[[2,5]].max()) - float(df_4.iat[5-
Q,0])*one_hundred_ps_neutron_window)
    D1_A5 = df.nlargest(one_hundred_ps_neutron_window, [' dtp01-
img05']).sum()
    D1_A5_total = int(float(D1_A5.iloc[[2,6]].max()) - float(df_4.iat[6-
Q,0])*one_hundred_ps_neutron_window)
    D1_A6 = df.nlargest(one_hundred_ps_neutron_window, [' dtp01-
img06']).sum()
    D1_A6_total = int(float(D1_A6.iloc[[2,7]].max()) - float(df_4.iat[7-
Q,0])*one_hundred_ps_neutron_window)
    D1_A7 = df.nlargest(one_hundred_ps_neutron_window, [' dtp01-
img07']).sum()
    D1_A7_total = int(float(D1_A7.iloc[[2,8]].max()) - float(df_4.iat[8-
Q,0])*one_hundred_ps_neutron_window)
    D1_A8 = df.nlargest(one_hundred_ps_neutron_window, [' dtp01-
img08']).sum()
    D1_A8_total = int(float(D1_A8.iloc[[2,9]].max()) - float(df_4.iat[9-
Q,0])*one_hundred_ps_neutron_window)
    D1_A9 = df.nlargest(one_hundred_ps_neutron_window, [' dtp01-
img09']).sum()
    D1_A9_total = int(float(D1_A9.iloc[[2,10]].max()) - float(df_4.iat[10-
Q,0])*one_hundred_ps_neutron_window)
    D1_A10 = df.nlargest(one_hundred_ps_neutron_window, [' dtp01-
img10']).sum()

```

```

D1_A10_total = int(float(D1_A10.iloc[[2,11]].max()) - float(df_4.iat[11-
Q,0])*one_hundred_ps_neutron_window)
D1_A11 = df.nlargest(one_hundred_ps_neutron_window,[' dtp01-
img11']).sum()
D1_A11_total = int(float(D1_A11.iloc[[2,12]].max()) - float(df_4.iat[12-
Q,0])*one_hundred_ps_neutron_window)
D1_A12 = df.nlargest(one_hundred_ps_neutron_window,[' dtp01-
img12']).sum()
D1_A12_total = int(float(D1_A12.iloc[[2,13]].max()) - float(df_4.iat[13-
Q,0])*one_hundred_ps_neutron_window)
D1_A13 = df.nlargest(one_hundred_ps_neutron_window,[' dtp01-
img13']).sum()
D1_A13_total = int(float(D1_A13.iloc[[2,14]].max()) - float(df_4.iat[14-
Q,0])*one_hundred_ps_neutron_window)
D1_A14 = df.nlargest(one_hundred_ps_neutron_window,[' dtp01-
img14']).sum()
D1_A14_total = int(float(D1_A14.iloc[[2,15]].max()) - float(df_4.iat[15-
Q,0])*one_hundred_ps_neutron_window)
D1_A15 = df.nlargest(one_hundred_ps_neutron_window,[' dtp01-
img15']).sum()
D1_A15_total = int(float(D1_A15.iloc[[2,16]].max()) - float(df_4.iat[16-
Q,0])*one_hundred_ps_neutron_window)
D1_A16 = df.nlargest(one_hundred_ps_neutron_window,[' dtp01-
img16']).sum()
D1_A16_total = int(float(D1_A16.iloc[[2,17]].max()) - float(df_4.iat[17-
Q,0])*one_hundred_ps_neutron_window)
D1_A17 = df.nlargest(one_hundred_ps_neutron_window,[' dtp01-
img17']).sum()
D1_A17_total = int(float(D1_A17.iloc[[2,18]].max()) - float(df_4.iat[18-
Q,0])*one_hundred_ps_neutron_window)
D1_A18 = df.nlargest(one_hundred_ps_neutron_window,[' dtp01-
img18']).sum()
D1_A18_total = int(float(D1_A18.iloc[[2,19]].max()) - float(df_4.iat[19-
Q,0])*one_hundred_ps_neutron_window)
D1_A19 = df.nlargest(one_hundred_ps_neutron_window,[' dtp01-
img19']).sum()
D1_A19_total = int(float(D1_A19.iloc[[2,20]].max()) - float(df_4.iat[20-
Q,0])*one_hundred_ps_neutron_window)
D1_A20 = df.nlargest(one_hundred_ps_neutron_window,[' dtp01-
img20']).sum()
D1_A20_total = int(float(D1_A20.iloc[[2,21]].max()) - float(df_4.iat[21-
Q,0])*one_hundred_ps_neutron_window)
D1_A21 = df.nlargest(one_hundred_ps_neutron_window,[' dtp01-
img21']).sum()

```

```

        D1_A21_total = int(float(D1_A21.iloc[[2,22]].max()) - float(df_4.iat[22-
Q,0])*one_hundred_ps_neutron_window)
        D1_A22 = df.nlargest(one_hundred_ps_neutron_window,[' dtp01-
img22']).sum()
        D1_A22_total = int(float(D1_A22.iloc[[2,23]].max()) - float(df_4.iat[23-
Q,0])*one_hundred_ps_neutron_window)
        D1_A23 = df.nlargest(one_hundred_ps_neutron_window,[' dtp01-
img23']).sum()
        D1_A23_total = int(float(D1_A23.iloc[[2,24]].max()) - float(df_4.iat[24-
Q,0])*one_hundred_ps_neutron_window)
        D1_A24 = df.nlargest(one_hundred_ps_neutron_window,[' dtp01-
img24']).sum()
        D1_A24_total = int(float(D1_A24.iloc[[2,25]].max()) - float(df_4.iat[25-
Q,0])*one_hundred_ps_neutron_window)
        D1_A25 = df.nlargest(one_hundred_ps_neutron_window,[' dtp01-
img25']).sum()
        D1_A25_total = int(float(D1_A25.iloc[[2,26]].max()) - float(df_4.iat[26-
Q,0])*one_hundred_ps_neutron_window)
        D1_A26 = df.nlargest(one_hundred_ps_neutron_window,[' dtp01-
img26']).sum()
        D1_A26_total = int(float(D1_A26.iloc[[2,27]].max()) - float(df_4.iat[27-
Q,0])*one_hundred_ps_neutron_window)
        D1_A27 = df.nlargest(one_hundred_ps_neutron_window,[' dtp01-
img27']).sum()
        D1_A27_total = int(float(D1_A27.iloc[[2,28]].max()) - float(df_4.iat[28-
Q,0])*one_hundred_ps_neutron_window)
        D1_A28 = df.nlargest(one_hundred_ps_neutron_window,[' dtp01-
img28']).sum()
        D1_A28_total = int(float(D1_A28.iloc[[2,29]].max()) - float(df_4.iat[29-
Q,0])*one_hundred_ps_neutron_window)
        detector += 1
    elif detector == 2:
        # D2 = Alpha Pixel 2 FOR detector pixels A1-A28 (1-28)
        D2_A1 = df.nlargest(one_hundred_ps_neutron_window,[' dtp02-
img01']).sum()
        D2_A1_total = int(float(D2_A1.iloc[[2,30]].max()) - float(df_4.iat[30-
Q,0])*one_hundred_ps_neutron_window)
        D2_A2 = df.nlargest(one_hundred_ps_neutron_window,[' dtp02-
img02']).sum()
        D2_A2_total = int(float(D2_A2.iloc[[2,31]].max()) - float(df_4.iat[31-
Q,0])*one_hundred_ps_neutron_window)
        D2_A3 = df.nlargest(one_hundred_ps_neutron_window,[' dtp02-
img03']).sum()
        D2_A3_total = int(float(D2_A3.iloc[[2,32]].max()) - float(df_4.iat[32-
Q,0])*one_hundred_ps_neutron_window)

```

```

D2_A4 = df.nlargest(one_hundred_ps_neutron_window,[' dtp02-
img04']).sum()
D2_A4_total = int(float(D2_A4.iloc[[2,33]].max()) - float(df_4.iat[33-
Q,0])*one_hundred_ps_neutron_window)
D2_A5 = df.nlargest(one_hundred_ps_neutron_window,[' dtp02-
img05']).sum()
D2_A5_total = int(float(D2_A5.iloc[[2,34]].max()) - float(df_4.iat[34-
Q,0])*one_hundred_ps_neutron_window)
D2_A6 = df.nlargest(one_hundred_ps_neutron_window,[' dtp02-
img06']).sum()
D2_A6_total = int(float(D2_A6.iloc[[2,35]].max()) - float(df_4.iat[35-
Q,0])*one_hundred_ps_neutron_window)
D2_A7 = df.nlargest(one_hundred_ps_neutron_window,[' dtp02-
img07']).sum()
D2_A7_total = int(float(D2_A7.iloc[[2,36]].max()) - float(df_4.iat[36-
Q,0])*one_hundred_ps_neutron_window)
D2_A8 = df.nlargest(one_hundred_ps_neutron_window,[' dtp02-
img08']).sum()
D2_A8_total = int(float(D2_A8.iloc[[2,37]].max()) - float(df_4.iat[37-
Q,0])*one_hundred_ps_neutron_window)
D2_A9 = df.nlargest(one_hundred_ps_neutron_window,[' dtp02-
img09']).sum()
D2_A9_total = int(float(D2_A9.iloc[[2,38]].max()) - float(df_4.iat[38-
Q,0])*one_hundred_ps_neutron_window)
D2_A10 = df.nlargest(one_hundred_ps_neutron_window,[' dtp02-
img10']).sum()
D2_A10_total = int(float(D2_A10.iloc[[2,39]].max()) - float(df_4.iat[39-
Q,0])*one_hundred_ps_neutron_window)
D2_A11 = df.nlargest(one_hundred_ps_neutron_window,[' dtp02-
img11']).sum()
D2_A11_total = int(float(D2_A11.iloc[[2,40]].max()) - float(df_4.iat[40-
Q,0])*one_hundred_ps_neutron_window)
D2_A12 = df.nlargest(one_hundred_ps_neutron_window,[' dtp02-
img12']).sum()
D2_A12_total = int(float(D2_A12.iloc[[2,41]].max()) - float(df_4.iat[41-
Q,0])*one_hundred_ps_neutron_window)
D2_A13 = df.nlargest(one_hundred_ps_neutron_window,[' dtp02-
img13']).sum()
D2_A13_total = int(float(D2_A13.iloc[[2,42]].max()) - float(df_4.iat[42-
Q,0])*one_hundred_ps_neutron_window)
D2_A14 = df.nlargest(one_hundred_ps_neutron_window,[' dtp02-
img14']).sum()
D2_A14_total = int(float(D2_A14.iloc[[2,43]].max()) - float(df_4.iat[43-
Q,0])*one_hundred_ps_neutron_window)

```

```

D2_A15 = df.nlargest(one_hundred_ps_neutron_window,[' dtp02-
img15']).sum()
D2_A15_total = int(float(D2_A15.iloc[[2,44]].max()) - float(df_4.iat[44-
Q,0])*one_hundred_ps_neutron_window)
D2_A16 = df.nlargest(one_hundred_ps_neutron_window,[' dtp02-
img16']).sum()
D2_A16_total = int(float(D2_A16.iloc[[2,45]].max()) - float(df_4.iat[45-
Q,0])*one_hundred_ps_neutron_window)
D2_A17 = df.nlargest(one_hundred_ps_neutron_window,[' dtp02-
img17']).sum()
D2_A17_total = int(float(D2_A17.iloc[[2,46]].max()) - float(df_4.iat[46-
Q,0])*one_hundred_ps_neutron_window)
D2_A18 = df.nlargest(one_hundred_ps_neutron_window,[' dtp02-
img18']).sum()
D2_A18_total = int(float(D2_A18.iloc[[2,47]].max()) - float(df_4.iat[47-
Q,0])*one_hundred_ps_neutron_window)
D2_A19 = df.nlargest(one_hundred_ps_neutron_window,[' dtp02-
img19']).sum()
D2_A19_total = int(float(D2_A19.iloc[[2,48]].max()) - float(df_4.iat[48-
Q,0])*one_hundred_ps_neutron_window)
D2_A20 = df.nlargest(one_hundred_ps_neutron_window,[' dtp02-
img20']).sum()
D2_A20_total = int(float(D2_A20.iloc[[2,49]].max()) - float(df_4.iat[49-
Q,0])*one_hundred_ps_neutron_window)
D2_A21 = df.nlargest(one_hundred_ps_neutron_window,[' dtp02-
img21']).sum()
D2_A21_total = int(float(D2_A21.iloc[[2,50]].max()) - float(df_4.iat[50-
Q,0])*one_hundred_ps_neutron_window)
D2_A22 = df.nlargest(one_hundred_ps_neutron_window,[' dtp02-
img22']).sum()
D2_A22_total = int(float(D2_A22.iloc[[2,51]].max()) - float(df_4.iat[51-
Q,0])*one_hundred_ps_neutron_window)
D2_A23 = df.nlargest(one_hundred_ps_neutron_window,[' dtp02-
img23']).sum()
D2_A23_total = int(float(D2_A23.iloc[[2,52]].max()) - float(df_4.iat[52-
Q,0])*one_hundred_ps_neutron_window)
D2_A24 = df.nlargest(one_hundred_ps_neutron_window,[' dtp02-
img24']).sum()
D2_A24_total = int(float(D2_A24.iloc[[2,53]].max()) - float(df_4.iat[53-
Q,0])*one_hundred_ps_neutron_window)
D2_A25 = df.nlargest(one_hundred_ps_neutron_window,[' dtp02-
img25']).sum()
D2_A25_total = int(float(D2_A25.iloc[[2,54]].max()) - float(df_4.iat[54-
Q,0])*one_hundred_ps_neutron_window)

```

```

        D2_A26 = df.nlargest(one_hundred_ps_neutron_window,[' dtp02-
img26']).sum()
        D2_A26_total = int(float(D2_A26.iloc[[2,55]].max()) - float(df_4.iat[55-
Q,0])*one_hundred_ps_neutron_window)
        D2_A27 = df.nlargest(one_hundred_ps_neutron_window,[' dtp02-
img27']).sum()
        D2_A27_total = int(float(D2_A27.iloc[[2,56]].max()) - float(df_4.iat[56-
Q,0])*one_hundred_ps_neutron_window)
        D2_A28 = df.nlargest(one_hundred_ps_neutron_window,[' dtp02-
img28']).sum()
        D2_A28_total = int(float(D2_A28.iloc[[2,57]].max()) - float(df_4.iat[57-
Q,0])*one_hundred_ps_neutron_window)
        detector += 1
    elif detector == 3:
        # D3 = Alpha Pixel 3 FOR detector pixels A1-A28 (1-28)
        D3_A1 = df.nlargest(one_hundred_ps_neutron_window,[' dtp03-
img01']).sum()
        D3_A1_total = int(float(D3_A1.iloc[[2,58]].max()) - float(df_4.iat[58-
Q,0])*one_hundred_ps_neutron_window)
        D3_A2 = df.nlargest(one_hundred_ps_neutron_window,[' dtp03-
img02']).sum()
        D3_A2_total = int(float(D3_A2.iloc[[2,59]].max()) - float(df_4.iat[59-
Q,0])*one_hundred_ps_neutron_window)
        D3_A3 = df.nlargest(one_hundred_ps_neutron_window,[' dtp03-
img03']).sum()
        D3_A3_total = int(float(D3_A3.iloc[[2,60]].max()) - float(df_4.iat[60-
Q,0])*one_hundred_ps_neutron_window)
        D3_A4 = df.nlargest(one_hundred_ps_neutron_window,[' dtp03-
img04']).sum()
        D3_A4_total = int(float(D3_A4.iloc[[2,61]].max()) - float(df_4.iat[61-
Q,0])*one_hundred_ps_neutron_window)
        D3_A5 = df.nlargest(one_hundred_ps_neutron_window,[' dtp03-
img05']).sum()
        D3_A5_total = int(float(D3_A5.iloc[[2,62]].max()) - float(df_4.iat[62-
Q,0])*one_hundred_ps_neutron_window)
        D3_A6 = df.nlargest(one_hundred_ps_neutron_window,[' dtp03-
img06']).sum()
        D3_A6_total = int(float(D3_A6.iloc[[2,63]].max()) - float(df_4.iat[63-
Q,0])*one_hundred_ps_neutron_window)
        D3_A7 = df.nlargest(one_hundred_ps_neutron_window,[' dtp03-
img07']).sum()
        D3_A7_total = int(float(D3_A7.iloc[[2,64]].max()) - float(df_4.iat[64-
Q,0])*one_hundred_ps_neutron_window)
        D3_A8 = df.nlargest(one_hundred_ps_neutron_window,[' dtp03-
img08']).sum()

```

```

        D3_A8_total = int(float(D3_A8.iloc[[2,65]].max()) - float(df_4.iat[65-
Q,0])*one_hundred_ps_neutron_window)
        D3_A9 = df.nlargest(one_hundred_ps_neutron_window,[' dtp03-
img09']).sum()
        D3_A9_total = int(float(D3_A9.iloc[[2,66]].max()) - float(df_4.iat[66-
Q,0])*one_hundred_ps_neutron_window)
        D3_A10 = df.nlargest(one_hundred_ps_neutron_window,[' dtp03-
img10']).sum()
        D3_A10_total = int(float(D3_A10.iloc[[2,67]].max()) - float(df_4.iat[67-
Q,0])*one_hundred_ps_neutron_window)
        D3_A11 = df.nlargest(one_hundred_ps_neutron_window,[' dtp03-
img11']).sum()
        D3_A11_total = int(float(D3_A11.iloc[[2,68]].max()) - float(df_4.iat[68-
Q,0])*one_hundred_ps_neutron_window)
        D3_A12 = df.nlargest(one_hundred_ps_neutron_window,[' dtp03-
img12']).sum()
        D3_A12_total = int(float(D3_A12.iloc[[2,69]].max()) - float(df_4.iat[69-
Q,0])*one_hundred_ps_neutron_window)
        D3_A13 = df.nlargest(one_hundred_ps_neutron_window,[' dtp03-
img13']).sum()
        D3_A13_total = int(float(D3_A13.iloc[[2,70]].max()) - float(df_4.iat[70-
Q,0])*one_hundred_ps_neutron_window)
        D3_A14 = df.nlargest(one_hundred_ps_neutron_window,[' dtp03-
img14']).sum()
        D3_A14_total = int(float(D3_A14.iloc[[2,71]].max()) - float(df_4.iat[71-
Q,0])*one_hundred_ps_neutron_window)
        D3_A15 = df.nlargest(one_hundred_ps_neutron_window,[' dtp03-
img15']).sum()
        D3_A15_total = int(float(D3_A15.iloc[[2,72]].max()) - float(df_4.iat[72-
Q,0])*one_hundred_ps_neutron_window)
        D3_A16 = df.nlargest(one_hundred_ps_neutron_window,[' dtp03-
img16']).sum()
        D3_A16_total = int(float(D3_A16.iloc[[2,73]].max()) - float(df_4.iat[73-
Q,0])*one_hundred_ps_neutron_window)
        D3_A17 = df.nlargest(one_hundred_ps_neutron_window,[' dtp03-
img17']).sum()
        D3_A17_total = int(float(D3_A17.iloc[[2,74]].max()) - float(df_4.iat[74-
Q,0])*one_hundred_ps_neutron_window)
        D3_A18 = df.nlargest(one_hundred_ps_neutron_window,[' dtp03-
img18']).sum()
        D3_A18_total = int(float(D3_A18.iloc[[2,75]].max()) - float(df_4.iat[75-
Q,0])*one_hundred_ps_neutron_window)
        D3_A19 = df.nlargest(one_hundred_ps_neutron_window,[' dtp03-
img19']).sum()

```

```

        D3_A19_total = int(float(D3_A19.iloc[[2,76]].max()) - float(df_4.iat[76-
Q,0])*one_hundred_ps_neutron_window)
        D3_A20 = df.nlargest(one_hundred_ps_neutron_window,[' dtp03-
img20']).sum()
        D3_A20_total = int(float(D3_A20.iloc[[2,77]].max()) - float(df_4.iat[77-
Q,0])*one_hundred_ps_neutron_window)
        D3_A21 = df.nlargest(one_hundred_ps_neutron_window,[' dtp03-
img21']).sum()
        D3_A21_total = int(float(D3_A21.iloc[[2,78]].max()) - float(df_4.iat[78-
Q,0])*one_hundred_ps_neutron_window)
        D3_A22 = df.nlargest(one_hundred_ps_neutron_window,[' dtp03-
img22']).sum()
        D3_A22_total = int(float(D3_A22.iloc[[2,79]].max()) - float(df_4.iat[79-
Q,0])*one_hundred_ps_neutron_window)
        D3_A23 = df.nlargest(one_hundred_ps_neutron_window,[' dtp03-
img23']).sum()
        D3_A23_total = int(float(D3_A23.iloc[[2,80]].max()) - float(df_4.iat[80-
Q,0])*one_hundred_ps_neutron_window)
        D3_A24 = df.nlargest(one_hundred_ps_neutron_window,[' dtp03-
img24']).sum()
        D3_A24_total = int(float(D3_A24.iloc[[2,81]].max()) - float(df_4.iat[81-
Q,0])*one_hundred_ps_neutron_window)
        D3_A25 = df.nlargest(one_hundred_ps_neutron_window,[' dtp03-
img25']).sum()
        D3_A25_total = int(float(D3_A25.iloc[[2,82]].max()) - float(df_4.iat[82-
Q,0])*one_hundred_ps_neutron_window)
        D3_A26 = df.nlargest(one_hundred_ps_neutron_window,[' dtp03-
img26']).sum()
        D3_A26_total = int(float(D3_A26.iloc[[2,83]].max()) - float(df_4.iat[83-
Q,0])*one_hundred_ps_neutron_window)
        D3_A27 = df.nlargest(one_hundred_ps_neutron_window,[' dtp03-
img27']).sum()
        D3_A27_total = int(float(D3_A27.iloc[[2,84]].max()) - float(df_4.iat[84-
Q,0])*one_hundred_ps_neutron_window)
        D3_A28 = df.nlargest(one_hundred_ps_neutron_window,[' dtp03-
img28']).sum()
        D3_A28_total = int(float(D3_A28.iloc[[2,85]].max()) - float(df_4.iat[85-
Q,0])*one_hundred_ps_neutron_window)
        detector += 1
    elif detector == 4:
        # D4 = Alpha Pixel 4 FOR detector pixels A1-A28 (1-28)
        D4_A1 = df.nlargest(one_hundred_ps_neutron_window,[' dtp04-
img01']).sum()
        D4_A1_total = int(float(D4_A1.iloc[[2,86]].max()) - float(df_4.iat[86-
Q,0])*one_hundred_ps_neutron_window)

```

```

D4_A2 = df.nlargest(one_hundred_ps_neutron_window,[' dtp04-
img02']).sum()
D4_A2_total = int(float(D4_A2.iloc[[2,87]].max()) - float(df_4.iat[87-
Q,0])*one_hundred_ps_neutron_window)
D4_A3 = df.nlargest(one_hundred_ps_neutron_window,[' dtp04-
img03']).sum()
D4_A3_total = int(float(D4_A3.iloc[[2,88]].max()) - float(df_4.iat[88-
Q,0])*one_hundred_ps_neutron_window)
D4_A4 = df.nlargest(one_hundred_ps_neutron_window,[' dtp04-
img04']).sum()
D4_A4_total = int(float(D4_A4.iloc[[2,89]].max()) - float(df_4.iat[89-
Q,0])*one_hundred_ps_neutron_window)
D4_A5 = df.nlargest(one_hundred_ps_neutron_window,[' dtp04-
img05']).sum()
D4_A5_total = int(float(D4_A5.iloc[[2,90]].max()) - float(df_4.iat[90-
Q,0])*one_hundred_ps_neutron_window)
D4_A6 = df.nlargest(one_hundred_ps_neutron_window,[' dtp04-
img06']).sum()
D4_A6_total = int(float(D4_A6.iloc[[2,91]].max()) - float(df_4.iat[91-
Q,0])*one_hundred_ps_neutron_window)
D4_A7 = df.nlargest(one_hundred_ps_neutron_window,[' dtp04-
img07']).sum()
D4_A7_total = int(float(D4_A7.iloc[[2,92]].max()) - float(df_4.iat[92-
Q,0])*one_hundred_ps_neutron_window)
D4_A8 = df.nlargest(one_hundred_ps_neutron_window,[' dtp04-
img08']).sum()
D4_A8_total = int(float(D4_A8.iloc[[2,93]].max()) - float(df_4.iat[93-
Q,0])*one_hundred_ps_neutron_window)
D4_A9 = df.nlargest(one_hundred_ps_neutron_window,[' dtp04-
img09']).sum()
D4_A9_total = int(float(D4_A9.iloc[[2,94]].max()) - float(df_4.iat[94-
Q,0])*one_hundred_ps_neutron_window)
D4_A10 = df.nlargest(one_hundred_ps_neutron_window,[' dtp04-
img10']).sum()
D4_A10_total = int(float(D4_A10.iloc[[2,95]].max()) - float(df_4.iat[95-
Q,0])*one_hundred_ps_neutron_window)
D4_A11 = df.nlargest(one_hundred_ps_neutron_window,[' dtp04-
img11']).sum()
D4_A11_total = int(float(D4_A11.iloc[[2,96]].max()) - float(df_4.iat[96-
Q,0])*one_hundred_ps_neutron_window)
D4_A12 = df.nlargest(one_hundred_ps_neutron_window,[' dtp04-
img12']).sum()
D4_A12_total = int(float(D4_A12.iloc[[2,97]].max()) - float(df_4.iat[97-
Q,0])*one_hundred_ps_neutron_window)

```

```

D4_A13 = df.nlargest(one_hundred_ps_neutron_window,[' dtp04-
img13']).sum()
D4_A13_total = int(float(D4_A13.iloc[[2,98]].max()) - float(df_4.iat[98-
Q,0])*one_hundred_ps_neutron_window)
D4_A14 = df.nlargest(one_hundred_ps_neutron_window,[' dtp04-
img14']).sum()
D4_A14_total = int(float(D4_A14.iloc[[2,99]].max()) - float(df_4.iat[99-
Q,0])*one_hundred_ps_neutron_window)
D4_A15 = df.nlargest(one_hundred_ps_neutron_window,[' dtp04-
img15']).sum()
D4_A15_total = int(float(D4_A15.iloc[[2,100]].max()) - float(df_4.iat[100-
Q,0])*one_hundred_ps_neutron_window)
D4_A16 = df.nlargest(one_hundred_ps_neutron_window,[' dtp04-
img16']).sum()
D4_A16_total = int(float(D4_A16.iloc[[2,101]].max()) - float(df_4.iat[101-
Q,0])*one_hundred_ps_neutron_window)
D4_A17 = df.nlargest(one_hundred_ps_neutron_window,[' dtp04-
img17']).sum()
D4_A17_total = int(float(D4_A17.iloc[[2,102]].max()) - float(df_4.iat[102-
Q,0])*one_hundred_ps_neutron_window)
D4_A18 = df.nlargest(one_hundred_ps_neutron_window,[' dtp04-
img18']).sum()
D4_A18_total = int(float(D4_A18.iloc[[2,103]].max()) - float(df_4.iat[103-
Q,0])*one_hundred_ps_neutron_window)
D4_A19 = df.nlargest(one_hundred_ps_neutron_window,[' dtp04-
img19']).sum()
D4_A19_total = int(float(D4_A19.iloc[[2,104]].max()) - float(df_4.iat[104-
Q,0])*one_hundred_ps_neutron_window)
D4_A20 = df.nlargest(one_hundred_ps_neutron_window,[' dtp04-
img20']).sum()
D4_A20_total = int(float(D4_A20.iloc[[2,105]].max()) - float(df_4.iat[105-
Q,0])*one_hundred_ps_neutron_window)
D4_A21 = df.nlargest(one_hundred_ps_neutron_window,[' dtp04-
img21']).sum()
D4_A21_total = int(float(D4_A21.iloc[[2,106]].max()) - float(df_4.iat[106-
Q,0])*one_hundred_ps_neutron_window)
D4_A22 = df.nlargest(one_hundred_ps_neutron_window,[' dtp04-
img22']).sum()
D4_A22_total = int(float(D4_A22.iloc[[2,107]].max()) - float(df_4.iat[107-
Q,0])*one_hundred_ps_neutron_window)
D4_A23 = df.nlargest(one_hundred_ps_neutron_window,[' dtp04-
img23']).sum()
D4_A23_total = int(float(D4_A23.iloc[[2,108]].max()) - float(df_4.iat[108-
Q,0])*one_hundred_ps_neutron_window)

```

```

        D4_A24 = df.nlargest(one_hundred_ps_neutron_window,[' dtp04-
img24']).sum()
        D4_A24_total = int(float(D4_A24.iloc[[2,109]].max()) - float(df_4.iat[109-
Q,0])*one_hundred_ps_neutron_window)
        D4_A25 = df.nlargest(one_hundred_ps_neutron_window,[' dtp04-
img25']).sum()
        D4_A25_total = int(float(D4_A25.iloc[[2,110]].max()) - float(df_4.iat[110-
Q,0])*one_hundred_ps_neutron_window)
        D4_A26 = df.nlargest(one_hundred_ps_neutron_window,[' dtp04-
img26']).sum()
        D4_A26_total = int(float(D4_A26.iloc[[2,111]].max()) - float(df_4.iat[111-
Q,0])*one_hundred_ps_neutron_window)
        D4_A27 = df.nlargest(one_hundred_ps_neutron_window,[' dtp04-
img27']).sum()
        D4_A27_total = int(float(D4_A27.iloc[[2,112]].max()) - float(df_4.iat[112-
Q,0])*one_hundred_ps_neutron_window)
        D4_A28 = df.nlargest(one_hundred_ps_neutron_window,[' dtp04-
img28']).sum()
        D4_A28_total = int(float(D4_A28.iloc[[2,113]].max()) - float(df_4.iat[113-
Q,0])*one_hundred_ps_neutron_window)
        detector += 1
    elif detector == 5:
        # D5 = Alpha Pixel 5 FOR detector pixels A1-A28 (1-28)
        D5_A1 = df.nlargest(one_hundred_ps_neutron_window,[' dtp05-
img01']).sum()
        D5_A1_total = int(float(D5_A1.iloc[[2,114]].max()) - float(df_4.iat[114-
Q,0])*one_hundred_ps_neutron_window)
        D5_A2 = df.nlargest(one_hundred_ps_neutron_window,[' dtp05-
img02']).sum()
        D5_A2_total = int(float(D5_A2.iloc[[2,115]].max()) - float(df_4.iat[115-
Q,0])*one_hundred_ps_neutron_window)
        D5_A3 = df.nlargest(one_hundred_ps_neutron_window,[' dtp05-
img03']).sum()
        D5_A3_total = int(float(D5_A3.iloc[[2,116]].max()) - float(df_4.iat[116-
Q,0])*one_hundred_ps_neutron_window)
        D5_A4 = df.nlargest(one_hundred_ps_neutron_window,[' dtp05-
img04']).sum()
        D5_A4_total = int(float(D5_A4.iloc[[2,117]].max()) - float(df_4.iat[117-
Q,0])*one_hundred_ps_neutron_window)
        D5_A5 = df.nlargest(one_hundred_ps_neutron_window,[' dtp05-
img05']).sum()
        D5_A5_total = int(float(D5_A5.iloc[[2,118]].max()) - float(df_4.iat[118-
Q,0])*one_hundred_ps_neutron_window)
        D5_A6 = df.nlargest(one_hundred_ps_neutron_window,[' dtp05-
img06']).sum()

```

```

D5_A6_total = int(float(D5_A6.iloc[[2,119]].max()) - float(df_4.iat[119-
Q,0])*one_hundred_ps_neutron_window)
D5_A7 = df.nlargest(one_hundred_ps_neutron_window,[' dtp05-
img07']).sum()
D5_A7_total = int(float(D5_A7.iloc[[2,120]].max()) - float(df_4.iat[120-
Q,0])*one_hundred_ps_neutron_window)
D5_A8 = df.nlargest(one_hundred_ps_neutron_window,[' dtp05-
img08']).sum()
D5_A8_total = int(float(D5_A8.iloc[[2,121]].max()) - float(df_4.iat[121-
Q,0])*one_hundred_ps_neutron_window)
D5_A9 = df.nlargest(one_hundred_ps_neutron_window,[' dtp05-
img09']).sum()
D5_A9_total = int(float(D5_A9.iloc[[2,122]].max()) - float(df_4.iat[122-
Q,0])*one_hundred_ps_neutron_window)
D5_A10 = df.nlargest(one_hundred_ps_neutron_window,[' dtp05-
img10']).sum()
D5_A10_total = int(float(D5_A10.iloc[[2,123]].max()) - float(df_4.iat[123-
Q,0])*one_hundred_ps_neutron_window)
D5_A11 = df.nlargest(one_hundred_ps_neutron_window,[' dtp05-
img11']).sum()
D5_A11_total = int(float(D5_A11.iloc[[2,124]].max()) - float(df_4.iat[124-
Q,0])*one_hundred_ps_neutron_window)
D5_A12 = df.nlargest(one_hundred_ps_neutron_window,[' dtp05-
img12']).sum()
D5_A12_total = int(float(D5_A12.iloc[[2,125]].max()) - float(df_4.iat[125-
Q,0])*one_hundred_ps_neutron_window)
D5_A13 = df.nlargest(one_hundred_ps_neutron_window,[' dtp05-
img13']).sum()
D5_A13_total = int(float(D5_A13.iloc[[2,126]].max()) - float(df_4.iat[126-
Q,0])*one_hundred_ps_neutron_window)
D5_A14 = df.nlargest(one_hundred_ps_neutron_window,[' dtp05-
img14']).sum()
D5_A14_total = int(float(D5_A14.iloc[[2,127]].max()) - float(df_4.iat[127-
Q,0])*one_hundred_ps_neutron_window)
D5_A15 = df.nlargest(one_hundred_ps_neutron_window,[' dtp05-
img15']).sum()
D5_A15_total = int(float(D5_A15.iloc[[2,128]].max()) - float(df_4.iat[128-
Q,0])*one_hundred_ps_neutron_window)
D5_A16 = df.nlargest(one_hundred_ps_neutron_window,[' dtp05-
img16']).sum()
D5_A16_total = int(float(D5_A16.iloc[[2,129]].max()) - float(df_4.iat[129-
Q,0])*one_hundred_ps_neutron_window)
D5_A17 = df.nlargest(one_hundred_ps_neutron_window,[' dtp05-
img17']).sum()

```

```

D5_A17_total = int(float(D5_A17.iloc[[2,130]].max()) - float(df_4.iat[130-
Q,0])*one_hundred_ps_neutron_window)
D5_A18 = df.nlargest(one_hundred_ps_neutron_window,[' dtp05-
img18']).sum()
D5_A18_total = int(float(D5_A18.iloc[[2,131]].max()) - float(df_4.iat[131-
Q,0])*one_hundred_ps_neutron_window)
D5_A19 = df.nlargest(one_hundred_ps_neutron_window,[' dtp05-
img19']).sum()
D5_A19_total = int(float(D5_A19.iloc[[2,132]].max()) - float(df_4.iat[132-
Q,0])*one_hundred_ps_neutron_window)
D5_A20 = df.nlargest(one_hundred_ps_neutron_window,[' dtp05-
img20']).sum()
D5_A20_total = int(float(D5_A20.iloc[[2,133]].max()) - float(df_4.iat[133-
Q,0])*one_hundred_ps_neutron_window)
D5_A21 = df.nlargest(one_hundred_ps_neutron_window,[' dtp05-
img21']).sum()
D5_A21_total = int(float(D5_A21.iloc[[2,134]].max()) - float(df_4.iat[134-
Q,0])*one_hundred_ps_neutron_window)
D5_A22 = df.nlargest(one_hundred_ps_neutron_window,[' dtp05-
img22']).sum()
D5_A22_total = int(float(D5_A22.iloc[[2,135]].max()) - float(df_4.iat[135-
Q,0])*one_hundred_ps_neutron_window)
D5_A23 = df.nlargest(one_hundred_ps_neutron_window,[' dtp05-
img23']).sum()
D5_A23_total = int(float(D5_A23.iloc[[2,136]].max()) - float(df_4.iat[136-
Q,0])*one_hundred_ps_neutron_window)
D5_A24 = df.nlargest(one_hundred_ps_neutron_window,[' dtp05-
img24']).sum()
D5_A24_total = int(float(D5_A24.iloc[[2,137]].max()) - float(df_4.iat[137-
Q,0])*one_hundred_ps_neutron_window)
D5_A25 = df.nlargest(one_hundred_ps_neutron_window,[' dtp05-
img25']).sum()
D5_A25_total = int(float(D5_A25.iloc[[2,138]].max()) - float(df_4.iat[138-
Q,0])*one_hundred_ps_neutron_window)
D5_A26 = df.nlargest(one_hundred_ps_neutron_window,[' dtp05-
img26']).sum()
D5_A26_total = int(float(D5_A26.iloc[[2,139]].max()) - float(df_4.iat[139-
Q,0])*one_hundred_ps_neutron_window)
D5_A27 = df.nlargest(one_hundred_ps_neutron_window,[' dtp05-
img27']).sum()
D5_A27_total = int(float(D5_A27.iloc[[2,140]].max()) - float(df_4.iat[140-
Q,0])*one_hundred_ps_neutron_window)
D5_A28 = df.nlargest(one_hundred_ps_neutron_window,[' dtp05-
img28']).sum()

```

```

        D5_A28_total = int(float(D5_A28.iloc[[2,141]].max())) - float(df_4.iat[141-
Q,0])*one_hundred_ps_neutron_window)
        detector += 1
    elif detector == 6:
        # D6 = Alpha Pixel 6 FOR detector pixels A1-A28 (1-28)
        D6_A1 = df.nlargest(one_hundred_ps_neutron_window,[' dtp06-
img01']).sum()
        D6_A1_total = int(float(D6_A1.iloc[[2,142]].max())) - float(df_4.iat[142-
Q,0])*one_hundred_ps_neutron_window)
        D6_A2 = df.nlargest(one_hundred_ps_neutron_window,[' dtp06-
img02']).sum()
        D6_A2_total = int(float(D6_A2.iloc[[2,143]].max())) - float(df_4.iat[143-
Q,0])*one_hundred_ps_neutron_window)
        D6_A3 = df.nlargest(one_hundred_ps_neutron_window,[' dtp06-
img03']).sum()
        D6_A3_total = int(float(D6_A3.iloc[[2,144]].max())) - float(df_4.iat[144-
Q,0])*one_hundred_ps_neutron_window)
        D6_A4 = df.nlargest(one_hundred_ps_neutron_window,[' dtp06-
img04']).sum()
        D6_A4_total = int(float(D6_A4.iloc[[2,145]].max())) - float(df_4.iat[145-
Q,0])*one_hundred_ps_neutron_window)
        D6_A5 = df.nlargest(one_hundred_ps_neutron_window,[' dtp06-
img05']).sum()
        D6_A5_total = int(float(D6_A5.iloc[[2,146]].max())) - float(df_4.iat[146-
Q,0])*one_hundred_ps_neutron_window)
        D6_A6 = df.nlargest(one_hundred_ps_neutron_window,[' dtp06-
img06']).sum()
        D6_A6_total = int(float(D6_A6.iloc[[2,147]].max())) - float(df_4.iat[147-
Q,0])*one_hundred_ps_neutron_window)
        D6_A7 = df.nlargest(one_hundred_ps_neutron_window,[' dtp06-
img07']).sum()
        D6_A7_total = int(float(D6_A7.iloc[[2,148]].max())) - float(df_4.iat[148-
Q,0])*one_hundred_ps_neutron_window)
        D6_A8 = df.nlargest(one_hundred_ps_neutron_window,[' dtp06-
img08']).sum()
        D6_A8_total = int(float(D6_A8.iloc[[2,149]].max())) - float(df_4.iat[149-
Q,0])*one_hundred_ps_neutron_window)
        D6_A9 = df.nlargest(one_hundred_ps_neutron_window,[' dtp06-
img09']).sum()
        D6_A9_total = int(float(D6_A9.iloc[[2,150]].max())) - float(df_4.iat[150-
Q,0])*one_hundred_ps_neutron_window)
        D6_A10 = df.nlargest(one_hundred_ps_neutron_window,[' dtp06-
img10']).sum()
        D6_A10_total = int(float(D6_A10.iloc[[2,151]].max())) - float(df_4.iat[151-
Q,0])*one_hundred_ps_neutron_window)

```

```

D6_A11 = df.nlargest(one_hundred_ps_neutron_window,[' dtp06-
img11']).sum()
D6_A11_total = int(float(D6_A11.iloc[[2,152]].max()) - float(df_4.iat[152-
Q,0])*one_hundred_ps_neutron_window)
D6_A12 = df.nlargest(one_hundred_ps_neutron_window,[' dtp06-
img12']).sum()
D6_A12_total = int(float(D6_A12.iloc[[2,153]].max()) - float(df_4.iat[153-
Q,0])*one_hundred_ps_neutron_window)
D6_A13 = df.nlargest(one_hundred_ps_neutron_window,[' dtp06-
img13']).sum()
D6_A13_total = int(float(D6_A13.iloc[[2,154]].max()) - float(df_4.iat[154-
Q,0])*one_hundred_ps_neutron_window)
D6_A14 = df.nlargest(one_hundred_ps_neutron_window,[' dtp06-
img14']).sum()
D6_A14_total = int(float(D6_A14.iloc[[2,155]].max()) - float(df_4.iat[155-
Q,0])*one_hundred_ps_neutron_window)
D6_A15 = df.nlargest(one_hundred_ps_neutron_window,[' dtp06-
img15']).sum()
D6_A15_total = int(float(D6_A15.iloc[[2,156]].max()) - float(df_4.iat[156-
Q,0])*one_hundred_ps_neutron_window)
D6_A16 = df.nlargest(one_hundred_ps_neutron_window,[' dtp06-
img16']).sum()
D6_A16_total = int(float(D6_A16.iloc[[2,157]].max()) - float(df_4.iat[157-
Q,0])*one_hundred_ps_neutron_window)
D6_A17 = df.nlargest(one_hundred_ps_neutron_window,[' dtp06-
img17']).sum()
D6_A17_total = int(float(D6_A17.iloc[[2,158]].max()) - float(df_4.iat[158-
Q,0])*one_hundred_ps_neutron_window)
D6_A18 = df.nlargest(one_hundred_ps_neutron_window,[' dtp06-
img18']).sum()
D6_A18_total = int(float(D6_A18.iloc[[2,159]].max()) - float(df_4.iat[159-
Q,0])*one_hundred_ps_neutron_window)
D6_A19 = df.nlargest(one_hundred_ps_neutron_window,[' dtp06-
img19']).sum()
D6_A19_total = int(float(D6_A19.iloc[[2,160]].max()) - float(df_4.iat[160-
Q,0])*one_hundred_ps_neutron_window)
D6_A20 = df.nlargest(one_hundred_ps_neutron_window,[' dtp06-
img20']).sum()
D6_A20_total = int(float(D6_A20.iloc[[2,161]].max()) - float(df_4.iat[161-
Q,0])*one_hundred_ps_neutron_window)
D6_A21 = df.nlargest(one_hundred_ps_neutron_window,[' dtp06-
img21']).sum()
D6_A21_total = int(float(D6_A21.iloc[[2,162]].max()) - float(df_4.iat[162-
Q,0])*one_hundred_ps_neutron_window)

```

```

        D6_A22 = df.nlargest(one_hundred_ps_neutron_window,[' dtp06-
img22']).sum()
        D6_A22_total = int(float(D6_A22.iloc[[2,163]].max()) - float(df_4.iat[163-
Q,0])*one_hundred_ps_neutron_window)
        D6_A23 = df.nlargest(one_hundred_ps_neutron_window,[' dtp06-
img23']).sum()
        D6_A23_total = int(float(D6_A23.iloc[[2,164]].max()) - float(df_4.iat[164-
Q,0])*one_hundred_ps_neutron_window)
        D6_A24 = df.nlargest(one_hundred_ps_neutron_window,[' dtp06-
img24']).sum()
        D6_A24_total = int(float(D6_A24.iloc[[2,165]].max()) - float(df_4.iat[165-
Q,0])*one_hundred_ps_neutron_window)
        D6_A25 = df.nlargest(one_hundred_ps_neutron_window,[' dtp06-
img25']).sum()
        D6_A25_total = int(float(D6_A25.iloc[[2,166]].max()) - float(df_4.iat[166-
Q,0])*one_hundred_ps_neutron_window)
        D6_A26 = df.nlargest(one_hundred_ps_neutron_window,[' dtp06-
img26']).sum()
        D6_A26_total = int(float(D6_A26.iloc[[2,167]].max()) - float(df_4.iat[167-
Q,0])*one_hundred_ps_neutron_window)
        D6_A27 = df.nlargest(one_hundred_ps_neutron_window,[' dtp06-
img27']).sum()
        D6_A27_total = int(float(D6_A27.iloc[[2,168]].max()) - float(df_4.iat[168-
Q,0])*one_hundred_ps_neutron_window)
        D6_A28 = df.nlargest(one_hundred_ps_neutron_window,[' dtp06-
img28']).sum()
        D6_A28_total = int(float(D6_A28.iloc[[2,169]].max()) - float(df_4.iat[169-
Q,0])*one_hundred_ps_neutron_window)
        detector += 1
    elif detector == 7:
        # D7 = Alpha Pixel 7 FOR detector pixels A1-A28 (1-28)
        D7_A1 = df.nlargest(one_hundred_ps_neutron_window,[' dtp07-
img01']).sum()
        D7_A1_total = int(float(D7_A1.iloc[[2,170]].max()) - float(df_4.iat[170-
Q,0])*one_hundred_ps_neutron_window)
        D7_A2 = df.nlargest(one_hundred_ps_neutron_window,[' dtp07-
img02']).sum()
        D7_A2_total = int(float(D7_A2.iloc[[2,171]].max()) - float(df_4.iat[171-
Q,0])*one_hundred_ps_neutron_window)
        D7_A3 = df.nlargest(one_hundred_ps_neutron_window,[' dtp07-
img03']).sum()
        D7_A3_total = int(float(D7_A3.iloc[[2,172]].max()) - float(df_4.iat[172-
Q,0])*one_hundred_ps_neutron_window)
        D7_A4 = df.nlargest(one_hundred_ps_neutron_window,[' dtp07-
img04']).sum()

```

```

D7_A4_total = int(float(D7_A4.iloc[[2,173]].max()) - float(df_4.iat[173-
Q,0])*one_hundred_ps_neutron_window)
D7_A5 = df.nlargest(one_hundred_ps_neutron_window,[' dtp07-
img05']).sum()
D7_A5_total = int(float(D7_A5.iloc[[2,174]].max()) - float(df_4.iat[174-
Q,0])*one_hundred_ps_neutron_window)
D7_A6 = df.nlargest(one_hundred_ps_neutron_window,[' dtp07-
img06']).sum()
D7_A6_total = int(float(D7_A6.iloc[[2,175]].max()) - float(df_4.iat[175-
Q,0])*one_hundred_ps_neutron_window)
D7_A7 = df.nlargest(one_hundred_ps_neutron_window,[' dtp07-
img07']).sum()
D7_A7_total = int(float(D7_A7.iloc[[2,176]].max()) - float(df_4.iat[176-
Q,0])*one_hundred_ps_neutron_window)
D7_A8 = df.nlargest(one_hundred_ps_neutron_window,[' dtp07-
img08']).sum()
D7_A8_total = int(float(D7_A8.iloc[[2,177]].max()) - float(df_4.iat[177-
Q,0])*one_hundred_ps_neutron_window)
D7_A9 = df.nlargest(one_hundred_ps_neutron_window,[' dtp07-
img09']).sum()
D7_A9_total = int(float(D7_A9.iloc[[2,178]].max()) - float(df_4.iat[178-
Q,0])*one_hundred_ps_neutron_window)
D7_A10 = df.nlargest(one_hundred_ps_neutron_window,[' dtp07-
img10']).sum()
D7_A10_total = int(float(D7_A10.iloc[[2,179]].max()) - float(df_4.iat[179-
Q,0])*one_hundred_ps_neutron_window)
D7_A11 = df.nlargest(one_hundred_ps_neutron_window,[' dtp07-
img11']).sum()
D7_A11_total = int(float(D7_A11.iloc[[2,180]].max()) - float(df_4.iat[180-
Q,0])*one_hundred_ps_neutron_window)
D7_A12 = df.nlargest(one_hundred_ps_neutron_window,[' dtp07-
img12']).sum()
D7_A12_total = int(float(D7_A12.iloc[[2,181]].max()) - float(df_4.iat[181-
Q,0])*one_hundred_ps_neutron_window)
D7_A13 = df.nlargest(one_hundred_ps_neutron_window,[' dtp07-
img13']).sum()
D7_A13_total = int(float(D7_A13.iloc[[2,182]].max()) - float(df_4.iat[182-
Q,0])*one_hundred_ps_neutron_window)
D7_A14 = df.nlargest(one_hundred_ps_neutron_window,[' dtp07-
img14']).sum()
D7_A14_total = int(float(D7_A14.iloc[[2,183]].max()) - float(df_4.iat[183-
Q,0])*one_hundred_ps_neutron_window)
D7_A15 = df.nlargest(one_hundred_ps_neutron_window,[' dtp07-
img15']).sum()

```

```

D7_A15_total = int(float(D7_A15.iloc[[2,184]].max()) - float(df_4.iat[184-
Q,0])*one_hundred_ps_neutron_window)
D7_A16 = df.nlargest(one_hundred_ps_neutron_window,[' dtp07-
img16']).sum()
D7_A16_total = int(float(D7_A16.iloc[[2,185]].max()) - float(df_4.iat[185-
Q,0])*one_hundred_ps_neutron_window)
D7_A17 = df.nlargest(one_hundred_ps_neutron_window,[' dtp07-
img17']).sum()
D7_A17_total = int(float(D7_A17.iloc[[2,186]].max()) - float(df_4.iat[186-
Q,0])*one_hundred_ps_neutron_window)
D7_A18 = df.nlargest(one_hundred_ps_neutron_window,[' dtp07-
img18']).sum()
D7_A18_total = int(float(D7_A18.iloc[[2,187]].max()) - float(df_4.iat[187-
Q,0])*one_hundred_ps_neutron_window)
D7_A19 = df.nlargest(one_hundred_ps_neutron_window,[' dtp07-
img19']).sum()
D7_A19_total = int(float(D7_A19.iloc[[2,188]].max()) - float(df_4.iat[188-
Q,0])*one_hundred_ps_neutron_window)
D7_A20 = df.nlargest(one_hundred_ps_neutron_window,[' dtp07-
img20']).sum()
D7_A20_total = int(float(D7_A20.iloc[[2,189]].max()) - float(df_4.iat[189-
Q,0])*one_hundred_ps_neutron_window)
D7_A21 = df.nlargest(one_hundred_ps_neutron_window,[' dtp07-
img21']).sum()
D7_A21_total = int(float(D7_A21.iloc[[2,190]].max()) - float(df_4.iat[190-
Q,0])*one_hundred_ps_neutron_window)
D7_A22 = df.nlargest(one_hundred_ps_neutron_window,[' dtp07-
img22']).sum()
D7_A22_total = int(float(D7_A22.iloc[[2,191]].max()) - float(df_4.iat[191-
Q,0])*one_hundred_ps_neutron_window)
D7_A23 = df.nlargest(one_hundred_ps_neutron_window,[' dtp07-
img23']).sum()
D7_A23_total = int(float(D7_A23.iloc[[2,192]].max()) - float(df_4.iat[192-
Q,0])*one_hundred_ps_neutron_window)
D7_A24 = df.nlargest(one_hundred_ps_neutron_window,[' dtp07-
img24']).sum()
D7_A24_total = int(float(D7_A24.iloc[[2,193]].max()) - float(df_4.iat[193-
Q,0])*one_hundred_ps_neutron_window)
D7_A25 = df.nlargest(one_hundred_ps_neutron_window,[' dtp07-
img25']).sum()
D7_A25_total = int(float(D7_A25.iloc[[2,194]].max()) - float(df_4.iat[194-
Q,0])*one_hundred_ps_neutron_window)
D7_A26 = df.nlargest(one_hundred_ps_neutron_window,[' dtp07-
img26']).sum()

```

```

        D7_A26_total = int(float(D7_A26.iloc[[2,195]].max()) - float(df_4.iat[195-
Q,0])*one_hundred_ps_neutron_window)
        D7_A27 = df.nlargest(one_hundred_ps_neutron_window,[' dtp07-
img27']).sum()
        D7_A27_total = int(float(D7_A27.iloc[[2,196]].max()) - float(df_4.iat[196-
Q,0])*one_hundred_ps_neutron_window)
        D7_A28 = df.nlargest(one_hundred_ps_neutron_window,[' dtp07-
img28']).sum()
        D7_A28_total = int(float(D7_A28.iloc[[2,197]].max()) - float(df_4.iat[197-
Q,0])*one_hundred_ps_neutron_window)
        detector += 1
    elif detector == 8:
        # D8 = Alpha Pixel 8 FOR detector pixels A1-A28 (1-28)
        D8_A1 = df.nlargest(one_hundred_ps_neutron_window,[' dtp08-
img01']).sum()
        D8_A1_total = int(float(D8_A1.iloc[[2,198]].max()) - float(df_4.iat[198-
Q,0])*one_hundred_ps_neutron_window)
        D8_A2 = df.nlargest(one_hundred_ps_neutron_window,[' dtp08-
img02']).sum()
        D8_A2_total = int(float(D8_A2.iloc[[2,199]].max()) - float(df_4.iat[199-
Q,0])*one_hundred_ps_neutron_window)
        D8_A3 = df.nlargest(one_hundred_ps_neutron_window,[' dtp08-
img03']).sum()
        D8_A3_total = int(float(D8_A3.iloc[[2,200]].max()) - float(df_4.iat[200-
Q,0])*one_hundred_ps_neutron_window)
        D8_A4 = df.nlargest(one_hundred_ps_neutron_window,[' dtp08-
img04']).sum()
        D8_A4_total = int(float(D8_A4.iloc[[2,201]].max()) - float(df_4.iat[201-
Q,0])*one_hundred_ps_neutron_window)
        D8_A5 = df.nlargest(one_hundred_ps_neutron_window,[' dtp08-
img05']).sum()
        D8_A5_total = int(float(D8_A5.iloc[[2,202]].max()) - float(df_4.iat[202-
Q,0])*one_hundred_ps_neutron_window)
        D8_A6 = df.nlargest(one_hundred_ps_neutron_window,[' dtp08-
img06']).sum()
        D8_A6_total = int(float(D8_A6.iloc[[2,203]].max()) - float(df_4.iat[203-
Q,0])*one_hundred_ps_neutron_window)
        D8_A7 = df.nlargest(one_hundred_ps_neutron_window,[' dtp08-
img07']).sum()
        D8_A7_total = int(float(D8_A7.iloc[[2,204]].max()) - float(df_4.iat[204-
Q,0])*one_hundred_ps_neutron_window)
        D8_A8 = df.nlargest(one_hundred_ps_neutron_window,[' dtp08-
img08']).sum()
        D8_A8_total = int(float(D8_A8.iloc[[2,205]].max()) - float(df_4.iat[205-
Q,0])*one_hundred_ps_neutron_window)

```

```

D8_A9 = df.nlargest(one_hundred_ps_neutron_window,[' dtp08-
img09']).sum()
D8_A9_total = int(float(D8_A9.iloc[[2,206]].max()) - float(df_4.iat[206-
Q,0])*one_hundred_ps_neutron_window)
D8_A10 = df.nlargest(one_hundred_ps_neutron_window,[' dtp08-
img10']).sum()
D8_A10_total = int(float(D8_A10.iloc[[2,207]].max()) - float(df_4.iat[207-
Q,0])*one_hundred_ps_neutron_window)
D8_A11 = df.nlargest(one_hundred_ps_neutron_window,[' dtp08-
img11']).sum()
D8_A11_total = int(float(D8_A11.iloc[[2,208]].max()) - float(df_4.iat[208-
Q,0])*one_hundred_ps_neutron_window)
D8_A12 = df.nlargest(one_hundred_ps_neutron_window,[' dtp08-
img12']).sum()
D8_A12_total = int(float(D8_A12.iloc[[2,209]].max()) - float(df_4.iat[209-
Q,0])*one_hundred_ps_neutron_window)
D8_A13 = df.nlargest(one_hundred_ps_neutron_window,[' dtp08-
img13']).sum()
D8_A13_total = int(float(D8_A13.iloc[[2,210]].max()) - float(df_4.iat[210-
Q,0])*one_hundred_ps_neutron_window)
D8_A14 = df.nlargest(one_hundred_ps_neutron_window,[' dtp08-
img14']).sum()
D8_A14_total = int(float(D8_A14.iloc[[2,211]].max()) - float(df_4.iat[211-
Q,0])*one_hundred_ps_neutron_window)
D8_A15 = df.nlargest(one_hundred_ps_neutron_window,[' dtp08-
img15']).sum()
D8_A15_total = int(float(D8_A15.iloc[[2,212]].max()) - float(df_4.iat[212-
Q,0])*one_hundred_ps_neutron_window)
D8_A16 = df.nlargest(one_hundred_ps_neutron_window,[' dtp08-
img16']).sum()
D8_A16_total = int(float(D8_A16.iloc[[2,213]].max()) - float(df_4.iat[213-
Q,0])*one_hundred_ps_neutron_window)
D8_A17 = df.nlargest(one_hundred_ps_neutron_window,[' dtp08-
img17']).sum()
D8_A17_total = int(float(D8_A17.iloc[[2,214]].max()) - float(df_4.iat[214-
Q,0])*one_hundred_ps_neutron_window)
D8_A18 = df.nlargest(one_hundred_ps_neutron_window,[' dtp08-
img18']).sum()
D8_A18_total = int(float(D8_A18.iloc[[2,215]].max()) - float(df_4.iat[215-
Q,0])*one_hundred_ps_neutron_window)
D8_A19 = df.nlargest(one_hundred_ps_neutron_window,[' dtp08-
img19']).sum()
D8_A19_total = int(float(D8_A19.iloc[[2,216]].max()) - float(df_4.iat[216-
Q,0])*one_hundred_ps_neutron_window)

```

```

        D8_A20 = df.nlargest(one_hundred_ps_neutron_window,[' dtp08-
img20']).sum()
        D8_A20_total = int(float(D8_A20.iloc[[2,217]].max()) - float(df_4.iat[217-
Q,0])*one_hundred_ps_neutron_window)
        D8_A21 = df.nlargest(one_hundred_ps_neutron_window,[' dtp08-
img21']).sum()
        D8_A21_total = int(float(D8_A21.iloc[[2,218]].max()) - float(df_4.iat[218-
Q,0])*one_hundred_ps_neutron_window)
        D8_A22 = df.nlargest(one_hundred_ps_neutron_window,[' dtp08-
img22']).sum()
        D8_A22_total = int(float(D8_A22.iloc[[2,219]].max()) - float(df_4.iat[219-
Q,0])*one_hundred_ps_neutron_window)
        D8_A23 = df.nlargest(one_hundred_ps_neutron_window,[' dtp08-
img23']).sum()
        D8_A23_total = int(float(D8_A23.iloc[[2,220]].max()) - float(df_4.iat[220-
Q,0])*one_hundred_ps_neutron_window)
        D8_A24 = df.nlargest(one_hundred_ps_neutron_window,[' dtp08-
img24']).sum()
        D8_A24_total = int(float(D8_A24.iloc[[2,221]].max()) - float(df_4.iat[221-
Q,0])*one_hundred_ps_neutron_window)
        D8_A25 = df.nlargest(one_hundred_ps_neutron_window,[' dtp08-
img25']).sum()
        D8_A25_total = int(float(D8_A25.iloc[[2,222]].max()) - float(df_4.iat[222-
Q,0])*one_hundred_ps_neutron_window)
        D8_A26 = df.nlargest(one_hundred_ps_neutron_window,[' dtp08-
img26']).sum()
        D8_A26_total = int(float(D8_A26.iloc[[2,223]].max()) - float(df_4.iat[223-
Q,0])*one_hundred_ps_neutron_window)
        D8_A27 = df.nlargest(one_hundred_ps_neutron_window,[' dtp08-
img27']).sum()
        D8_A27_total = int(float(D8_A27.iloc[[2,224]].max()) - float(df_4.iat[224-
Q,0])*one_hundred_ps_neutron_window)
        D8_A28 = df.nlargest(one_hundred_ps_neutron_window,[' dtp08-
img28']).sum()
        D8_A28_total = int(float(D8_A28.iloc[[2,225]].max()) - float(df_4.iat[225-
Q,0])*one_hundred_ps_neutron_window)
        detector += 1
    elif detector == 9:
        # D9 = Alpha Pixel 9 FOR detector pixels A1-A28 (1-28)
        D9_A1 = df.nlargest(one_hundred_ps_neutron_window,[' dtp09-
img01']).sum()
        D9_A1_total = int(float(D9_A1.iloc[[2,226]].max()) - float(df_4.iat[226-
Q,0])*one_hundred_ps_neutron_window)
        D9_A2 = df.nlargest(one_hundred_ps_neutron_window,[' dtp09-
img02']).sum()

```

```

D9_A2_total = int(float(D9_A2.iloc[[2,227]].max()) - float(df_4.iat[227-
Q,0])*one_hundred_ps_neutron_window)
D9_A3 = df.nlargest(one_hundred_ps_neutron_window,[' dtp09-
img03']).sum()
D9_A3_total = int(float(D9_A3.iloc[[2,228]].max()) - float(df_4.iat[228-
Q,0])*one_hundred_ps_neutron_window)
D9_A4 = df.nlargest(one_hundred_ps_neutron_window,[' dtp09-
img04']).sum()
D9_A4_total = int(float(D9_A4.iloc[[2,229]].max()) - float(df_4.iat[229-
Q,0])*one_hundred_ps_neutron_window)
D9_A5 = df.nlargest(one_hundred_ps_neutron_window,[' dtp09-
img05']).sum()
D9_A5_total = int(float(D9_A5.iloc[[2,230]].max()) - float(df_4.iat[230-
Q,0])*one_hundred_ps_neutron_window)
D9_A6 = df.nlargest(one_hundred_ps_neutron_window,[' dtp09-
img06']).sum()
D9_A6_total = int(float(D9_A6.iloc[[2,231]].max()) - float(df_4.iat[231-
Q,0])*one_hundred_ps_neutron_window)
D9_A7 = df.nlargest(one_hundred_ps_neutron_window,[' dtp09-
img07']).sum()
D9_A7_total = int(float(D9_A7.iloc[[2,232]].max()) - float(df_4.iat[232-
Q,0])*one_hundred_ps_neutron_window)
D9_A8 = df.nlargest(one_hundred_ps_neutron_window,[' dtp09-
img08']).sum()
D9_A8_total = int(float(D9_A8.iloc[[2,233]].max()) - float(df_4.iat[233-
Q,0])*one_hundred_ps_neutron_window)
D9_A9 = df.nlargest(one_hundred_ps_neutron_window,[' dtp09-
img09']).sum()
D9_A9_total = int(float(D9_A9.iloc[[2,234]].max()) - float(df_4.iat[234-
Q,0])*one_hundred_ps_neutron_window)
D9_A10 = df.nlargest(one_hundred_ps_neutron_window,[' dtp09-
img10']).sum()
D9_A10_total = int(float(D9_A10.iloc[[2,235]].max()) - float(df_4.iat[235-
Q,0])*one_hundred_ps_neutron_window)
D9_A11 = df.nlargest(one_hundred_ps_neutron_window,[' dtp09-
img11']).sum()
D9_A11_total = int(float(D9_A11.iloc[[2,236]].max()) - float(df_4.iat[236-
Q,0])*one_hundred_ps_neutron_window)
D9_A12 = df.nlargest(one_hundred_ps_neutron_window,[' dtp09-
img12']).sum()
D9_A12_total = int(float(D9_A12.iloc[[2,237]].max()) - float(df_4.iat[237-
Q,0])*one_hundred_ps_neutron_window)
D9_A13 = df.nlargest(one_hundred_ps_neutron_window,[' dtp09-
img13']).sum()

```

```

D9_A13_total = int(float(D9_A13.iloc[[2,238]].max()) - float(df_4.iat[238-
Q,0])*one_hundred_ps_neutron_window)
D9_A14 = df.nlargest(one_hundred_ps_neutron_window,[' dtp09-
img14']).sum()
D9_A14_total = int(float(D9_A14.iloc[[2,239]].max()) - float(df_4.iat[239-
Q,0])*one_hundred_ps_neutron_window)
D9_A15 = df.nlargest(one_hundred_ps_neutron_window,[' dtp09-
img15']).sum()
D9_A15_total = int(float(D9_A15.iloc[[2,240]].max()) - float(df_4.iat[240-
Q,0])*one_hundred_ps_neutron_window)
D9_A16 = df.nlargest(one_hundred_ps_neutron_window,[' dtp09-
img16']).sum()
D9_A16_total = int(float(D9_A16.iloc[[2,241]].max()) - float(df_4.iat[241-
Q,0])*one_hundred_ps_neutron_window)
D9_A17 = df.nlargest(one_hundred_ps_neutron_window,[' dtp09-
img17']).sum()
D9_A17_total = int(float(D9_A17.iloc[[2,242]].max()) - float(df_4.iat[242-
Q,0])*one_hundred_ps_neutron_window)
D9_A18 = df.nlargest(one_hundred_ps_neutron_window,[' dtp09-
img18']).sum()
D9_A18_total = int(float(D9_A18.iloc[[2,243]].max()) - float(df_4.iat[243-
Q,0])*one_hundred_ps_neutron_window)
D9_A19 = df.nlargest(one_hundred_ps_neutron_window,[' dtp09-
img19']).sum()
D9_A19_total = int(float(D9_A19.iloc[[2,244]].max()) - float(df_4.iat[244-
Q,0])*one_hundred_ps_neutron_window)
D9_A20 = df.nlargest(one_hundred_ps_neutron_window,[' dtp09-
img20']).sum()
D9_A20_total = int(float(D9_A20.iloc[[2,245]].max()) - float(df_4.iat[245-
Q,0])*one_hundred_ps_neutron_window)
D9_A21 = df.nlargest(one_hundred_ps_neutron_window,[' dtp09-
img21']).sum()
D9_A21_total = int(float(D9_A21.iloc[[2,246]].max()) - float(df_4.iat[246-
Q,0])*one_hundred_ps_neutron_window)
D9_A22 = df.nlargest(one_hundred_ps_neutron_window,[' dtp09-
img22']).sum()
D9_A22_total = int(float(D9_A22.iloc[[2,247]].max()) - float(df_4.iat[247-
Q,0])*one_hundred_ps_neutron_window)
D9_A23 = df.nlargest(one_hundred_ps_neutron_window,[' dtp09-
img23']).sum()
D9_A23_total = int(float(D9_A23.iloc[[2,248]].max()) - float(df_4.iat[248-
Q,0])*one_hundred_ps_neutron_window)
D9_A24 = df.nlargest(one_hundred_ps_neutron_window,[' dtp09-
img24']).sum()

```

```

        D9_A24_total = int(float(D9_A24.iloc[[2,249]].max()) - float(df_4.iat[249-
Q,0])*one_hundred_ps_neutron_window)
        D9_A25 = df.nlargest(one_hundred_ps_neutron_window,[' dtp09-
img25']).sum()
        D9_A25_total = int(float(D9_A25.iloc[[2,250]].max()) - float(df_4.iat[250-
Q,0])*one_hundred_ps_neutron_window)
        D9_A26 = df.nlargest(one_hundred_ps_neutron_window,[' dtp09-
img26']).sum()
        D9_A26_total = int(float(D9_A26.iloc[[2,251]].max()) - float(df_4.iat[251-
Q,0])*one_hundred_ps_neutron_window)
        D9_A27 = df.nlargest(one_hundred_ps_neutron_window,[' dtp09-
img27']).sum()
        D9_A27_total = int(float(D9_A27.iloc[[2,252]].max()) - float(df_4.iat[252-
Q,0])*one_hundred_ps_neutron_window)
        D9_A28 = df.nlargest(one_hundred_ps_neutron_window,[' dtp09-
img28']).sum()
        D9_A28_total = int(float(D9_A28.iloc[[2,253]].max()) - float(df_4.iat[253-
Q,0])*one_hundred_ps_neutron_window)
        detector += 1
    elif detector == 10:
        # D10 = Alpha Pixel 10 FOR detector pixels A1-A28 (1-28)
        D10_A1 = df.nlargest(one_hundred_ps_neutron_window,[' dtp10-
img01']).sum()
        D10_A1_total = int(float(D10_A1.iloc[[2,254]].max()) - float(df_4.iat[254-
Q,0])*one_hundred_ps_neutron_window)
        D10_A2 = df.nlargest(one_hundred_ps_neutron_window,[' dtp10-
img02']).sum()
        D10_A2_total = int(float(D10_A2.iloc[[2,255]].max()) - float(df_4.iat[255-
Q,0])*one_hundred_ps_neutron_window)
        D10_A3 = df.nlargest(one_hundred_ps_neutron_window,[' dtp10-
img03']).sum()
        D10_A3_total = int(float(D10_A3.iloc[[2,256]].max()) - float(df_4.iat[256-
Q,0])*one_hundred_ps_neutron_window)
        D10_A4 = df.nlargest(one_hundred_ps_neutron_window,[' dtp10-
img04']).sum()
        D10_A4_total = int(float(D10_A4.iloc[[2,257]].max()) - float(df_4.iat[257-
Q,0])*one_hundred_ps_neutron_window)
        D10_A5 = df.nlargest(one_hundred_ps_neutron_window,[' dtp10-
img05']).sum()
        D10_A5_total = int(float(D10_A5.iloc[[2,258]].max()) - float(df_4.iat[258-
Q,0])*one_hundred_ps_neutron_window)
        D10_A6 = df.nlargest(one_hundred_ps_neutron_window,[' dtp10-
img06']).sum()
        D10_A6_total = int(float(D10_A6.iloc[[2,259]].max()) - float(df_4.iat[259-
Q,0])*one_hundred_ps_neutron_window)

```

```

D10_A7 = df.nlargest(one_hundred_ps_neutron_window,[' dtp10-
img07']).sum()
D10_A7_total = int(float(D10_A7.iloc[[2,260]].max()) - float(df_4.iat[260-
Q,0])*one_hundred_ps_neutron_window)
D10_A8 = df.nlargest(one_hundred_ps_neutron_window,[' dtp10-
img08']).sum()
D10_A8_total = int(float(D10_A8.iloc[[2,261]].max()) - float(df_4.iat[261-
Q,0])*one_hundred_ps_neutron_window)
D10_A9 = df.nlargest(one_hundred_ps_neutron_window,[' dtp10-
img09']).sum()
D10_A9_total = int(float(D10_A9.iloc[[2,262]].max()) - float(df_4.iat[262-
Q,0])*one_hundred_ps_neutron_window)
D10_A10 = df.nlargest(one_hundred_ps_neutron_window,[' dtp10-
img10']).sum()
D10_A10_total = int(float(D10_A10.iloc[[2,263]].max()) -
float(df_4.iat[263-Q,0])*one_hundred_ps_neutron_window)
D10_A11 = df.nlargest(one_hundred_ps_neutron_window,[' dtp10-
img11']).sum()
D10_A11_total = int(float(D10_A11.iloc[[2,264]].max()) -
float(df_4.iat[264-Q,0])*one_hundred_ps_neutron_window)
D10_A12 = df.nlargest(one_hundred_ps_neutron_window,[' dtp10-
img12']).sum()
D10_A12_total = int(float(D10_A12.iloc[[2,265]].max()) -
float(df_4.iat[265-Q,0])*one_hundred_ps_neutron_window)
D10_A13 = df.nlargest(one_hundred_ps_neutron_window,[' dtp10-
img13']).sum()
D10_A13_total = int(float(D10_A13.iloc[[2,266]].max()) -
float(df_4.iat[266-Q,0])*one_hundred_ps_neutron_window)
D10_A14 = df.nlargest(one_hundred_ps_neutron_window,[' dtp10-
img14']).sum()
D10_A14_total = int(float(D10_A14.iloc[[2,267]].max()) -
float(df_4.iat[267-Q,0])*one_hundred_ps_neutron_window)
D10_A15 = df.nlargest(one_hundred_ps_neutron_window,[' dtp10-
img15']).sum()
D10_A15_total = int(float(D10_A15.iloc[[2,268]].max()) -
float(df_4.iat[268-Q,0])*one_hundred_ps_neutron_window)
D10_A16 = df.nlargest(one_hundred_ps_neutron_window,[' dtp10-
img16']).sum()
D10_A16_total = int(float(D10_A16.iloc[[2,269]].max()) -
float(df_4.iat[269-Q,0])*one_hundred_ps_neutron_window)
D10_A17 = df.nlargest(one_hundred_ps_neutron_window,[' dtp10-
img17']).sum()
D10_A17_total = int(float(D10_A17.iloc[[2,270]].max()) -
float(df_4.iat[270-Q,0])*one_hundred_ps_neutron_window)

```

```

D10_A18 = df.nlargest(one_hundred_ps_neutron_window,[' dtp10-
img18']).sum()
D10_A18_total = int(float(D10_A18.iloc[[2,271]].max()) -
float(df_4.iat[271-Q,0])*one_hundred_ps_neutron_window)
D10_A19 = df.nlargest(one_hundred_ps_neutron_window,[' dtp10-
img19']).sum()
D10_A19_total = int(float(D10_A19.iloc[[2,272]].max()) -
float(df_4.iat[272-Q,0])*one_hundred_ps_neutron_window)
D10_A20 = df.nlargest(one_hundred_ps_neutron_window,[' dtp10-
img20']).sum()
D10_A20_total = int(float(D10_A20.iloc[[2,273]].max()) -
float(df_4.iat[273-Q,0])*one_hundred_ps_neutron_window)
D10_A21 = df.nlargest(one_hundred_ps_neutron_window,[' dtp10-
img21']).sum()
D10_A21_total = int(float(D10_A21.iloc[[2,274]].max()) -
float(df_4.iat[274-Q,0])*one_hundred_ps_neutron_window)
D10_A22 = df.nlargest(one_hundred_ps_neutron_window,[' dtp10-
img22']).sum()
D10_A22_total = int(float(D10_A22.iloc[[2,275]].max()) -
float(df_4.iat[275-Q,0])*one_hundred_ps_neutron_window)
D10_A23 = df.nlargest(one_hundred_ps_neutron_window,[' dtp10-
img23']).sum()
D10_A23_total = int(float(D10_A23.iloc[[2,276]].max()) -
float(df_4.iat[276-Q,0])*one_hundred_ps_neutron_window)
D10_A24 = df.nlargest(one_hundred_ps_neutron_window,[' dtp10-
img24']).sum()
D10_A24_total = int(float(D10_A24.iloc[[2,277]].max()) -
float(df_4.iat[277-Q,0])*one_hundred_ps_neutron_window)
D10_A25 = df.nlargest(one_hundred_ps_neutron_window,[' dtp10-
img25']).sum()
D10_A25_total = int(float(D10_A25.iloc[[2,278]].max()) -
float(df_4.iat[278-Q,0])*one_hundred_ps_neutron_window)
D10_A26 = df.nlargest(one_hundred_ps_neutron_window,[' dtp10-
img26']).sum()
D10_A26_total = int(float(D10_A26.iloc[[2,279]].max()) -
float(df_4.iat[279-Q,0])*one_hundred_ps_neutron_window)
D10_A27 = df.nlargest(one_hundred_ps_neutron_window,[' dtp10-
img27']).sum()
D10_A27_total = int(float(D10_A27.iloc[[2,280]].max()) -
float(df_4.iat[280-Q,0])*one_hundred_ps_neutron_window)
D10_A28 = df.nlargest(one_hundred_ps_neutron_window,[' dtp10-
img28']).sum()
D10_A28_total = int(float(D10_A28.iloc[[2,281]].max()) -
float(df_4.iat[281-Q,0])*one_hundred_ps_neutron_window)
detector += 1

```

```

elif detector == 11:
    # D11 = Alpha Pixel 11 FOR detector pixels A1-A28 (1-28)
    D11_A1 = df.nlargest(one_hundred_ps_neutron_window,[' dtp11-
img01']).sum()
    D11_A1_total = int(float(D11_A1.iloc[[2,282]].max()) - float(df_4.iat[282-
Q,0])*one_hundred_ps_neutron_window)
    D11_A2 = df.nlargest(one_hundred_ps_neutron_window,[' dtp11-
img02']).sum()
    D11_A2_total = int(float(D11_A2.iloc[[2,283]].max()) - float(df_4.iat[283-
Q,0])*one_hundred_ps_neutron_window)
    D11_A3 = df.nlargest(one_hundred_ps_neutron_window,[' dtp11-
img03']).sum()
    D11_A3_total = int(float(D11_A3.iloc[[2,284]].max()) - float(df_4.iat[284-
Q,0])*one_hundred_ps_neutron_window)
    D11_A4 = df.nlargest(one_hundred_ps_neutron_window,[' dtp11-
img04']).sum()
    D11_A4_total = int(float(D11_A4.iloc[[2,285]].max()) - float(df_4.iat[285-
Q,0])*one_hundred_ps_neutron_window)
    D11_A5 = df.nlargest(one_hundred_ps_neutron_window,[' dtp11-
img05']).sum()
    D11_A5_total = int(float(D11_A5.iloc[[2,286]].max()) - float(df_4.iat[286-
Q,0])*one_hundred_ps_neutron_window)
    D11_A6 = df.nlargest(one_hundred_ps_neutron_window,[' dtp11-
img06']).sum()
    D11_A6_total = int(float(D11_A6.iloc[[2,287]].max()) - float(df_4.iat[287-
Q,0])*one_hundred_ps_neutron_window)
    D11_A7 = df.nlargest(one_hundred_ps_neutron_window,[' dtp11-
img07']).sum()
    D11_A7_total = int(float(D11_A7.iloc[[2,288]].max()) - float(df_4.iat[288-
Q,0])*one_hundred_ps_neutron_window)
    D11_A8 = df.nlargest(one_hundred_ps_neutron_window,[' dtp11-
img08']).sum()
    D11_A8_total = int(float(D11_A8.iloc[[2,289]].max()) - float(df_4.iat[289-
Q,0])*one_hundred_ps_neutron_window)
    D11_A9 = df.nlargest(one_hundred_ps_neutron_window,[' dtp11-
img09']).sum()
    D11_A9_total = int(float(D11_A9.iloc[[2,290]].max()) - float(df_4.iat[290-
Q,0])*one_hundred_ps_neutron_window)
    D11_A10 = df.nlargest(one_hundred_ps_neutron_window,[' dtp11-
img10']).sum()
    D11_A10_total = int(float(D11_A10.iloc[[2,291]].max()) -
float(df_4.iat[291-Q,0])*one_hundred_ps_neutron_window)
    D11_A11 = df.nlargest(one_hundred_ps_neutron_window,[' dtp11-
img11']).sum()

```

```

D11_A11_total = int(float(D11_A11.iloc[[2,292]].max()) -
float(df_4.iat[292-Q,0])*one_hundred_ps_neutron_window)
D11_A12 = df.nlargest(one_hundred_ps_neutron_window,[' dtp11-
img12']).sum()
D11_A12_total = int(float(D11_A12.iloc[[2,293]].max()) -
float(df_4.iat[293-Q,0])*one_hundred_ps_neutron_window)
D11_A13 = df.nlargest(one_hundred_ps_neutron_window,[' dtp11-
img13']).sum()
D11_A13_total = int(float(D11_A13.iloc[[2,294]].max()) -
float(df_4.iat[294-Q,0])*one_hundred_ps_neutron_window)
D11_A14 = df.nlargest(one_hundred_ps_neutron_window,[' dtp11-
img14']).sum()
D11_A14_total = int(float(D11_A14.iloc[[2,295]].max()) -
float(df_4.iat[295-Q,0])*one_hundred_ps_neutron_window)
D11_A15 = df.nlargest(one_hundred_ps_neutron_window,[' dtp11-
img15']).sum()
D11_A15_total = int(float(D11_A15.iloc[[2,296]].max()) -
float(df_4.iat[296-Q,0])*one_hundred_ps_neutron_window)
D11_A16 = df.nlargest(one_hundred_ps_neutron_window,[' dtp11-
img16']).sum()
D11_A16_total = int(float(D11_A16.iloc[[2,297]].max()) -
float(df_4.iat[297-Q,0])*one_hundred_ps_neutron_window)
D11_A17 = df.nlargest(one_hundred_ps_neutron_window,[' dtp11-
img17']).sum()
D11_A17_total = int(float(D11_A17.iloc[[2,298]].max()) -
float(df_4.iat[298-Q,0])*one_hundred_ps_neutron_window)
D11_A18 = df.nlargest(one_hundred_ps_neutron_window,[' dtp11-
img18']).sum()
D11_A18_total = int(float(D11_A18.iloc[[2,299]].max()) -
float(df_4.iat[299-Q,0])*one_hundred_ps_neutron_window)
D11_A19 = df.nlargest(one_hundred_ps_neutron_window,[' dtp11-
img19']).sum()
D11_A19_total = int(float(D11_A19.iloc[[2,300]].max()) -
float(df_4.iat[300-Q,0])*one_hundred_ps_neutron_window)
D11_A20 = df.nlargest(one_hundred_ps_neutron_window,[' dtp11-
img20']).sum()
D11_A20_total = int(float(D11_A20.iloc[[2,301]].max()) -
float(df_4.iat[301-Q,0])*one_hundred_ps_neutron_window)
D11_A21 = df.nlargest(one_hundred_ps_neutron_window,[' dtp11-
img21']).sum()
D11_A21_total = int(float(D11_A21.iloc[[2,302]].max()) -
float(df_4.iat[302-Q,0])*one_hundred_ps_neutron_window)
D11_A22 = df.nlargest(one_hundred_ps_neutron_window,[' dtp11-
img22']).sum()

```

```

        D11_A22_total = int(float(D11_A22.iloc[[2,303]].max()) -
float(df_4.iat[303-Q,0])*one_hundred_ps_neutron_window)
        D11_A23 = df.nlargest(one_hundred_ps_neutron_window,[' dtp11-
img23']).sum()
        D11_A23_total = int(float(D11_A23.iloc[[2,304]].max()) -
float(df_4.iat[304-Q,0])*one_hundred_ps_neutron_window)
        D11_A24 = df.nlargest(one_hundred_ps_neutron_window,[' dtp11-
img24']).sum()
        D11_A24_total = int(float(D11_A24.iloc[[2,305]].max()) -
float(df_4.iat[305-Q,0])*one_hundred_ps_neutron_window)
        D11_A25 = df.nlargest(one_hundred_ps_neutron_window,[' dtp11-
img25']).sum()
        D11_A25_total = int(float(D11_A25.iloc[[2,306]].max()) -
float(df_4.iat[306-Q,0])*one_hundred_ps_neutron_window)
        D11_A26 = df.nlargest(one_hundred_ps_neutron_window,[' dtp11-
img26']).sum()
        D11_A26_total = int(float(D11_A26.iloc[[2,307]].max()) -
float(df_4.iat[307-Q,0])*one_hundred_ps_neutron_window)
        D11_A27 = df.nlargest(one_hundred_ps_neutron_window,[' dtp11-
img27']).sum()
        D11_A27_total = int(float(D11_A27.iloc[[2,308]].max()) -
float(df_4.iat[308-Q,0])*one_hundred_ps_neutron_window)
        D11_A28 = df.nlargest(one_hundred_ps_neutron_window,[' dtp11-
img28']).sum()
        D11_A28_total = int(float(D11_A28.iloc[[2,309]].max()) -
float(df_4.iat[309-Q,0])*one_hundred_ps_neutron_window)
        detector += 1
    elif detector == 12:
        # D12 = Alpha Pixel 12 FOR detector pixels A1-A28 (1-28)
        D12_A1 = df.nlargest(one_hundred_ps_neutron_window,[' dtp12-
img01']).sum()
        D12_A1_total = int(float(D12_A1.iloc[[2,310]].max()) - float(df_4.iat[310-
Q,0])*one_hundred_ps_neutron_window)
        D12_A2 = df.nlargest(one_hundred_ps_neutron_window,[' dtp12-
img02']).sum()
        D12_A2_total = int(float(D12_A2.iloc[[2,311]].max()) - float(df_4.iat[311-
Q,0])*one_hundred_ps_neutron_window)
        D12_A3 = df.nlargest(one_hundred_ps_neutron_window,[' dtp12-
img03']).sum()
        D12_A3_total = int(float(D12_A3.iloc[[2,312]].max()) - float(df_4.iat[312-
Q,0])*one_hundred_ps_neutron_window)
        D12_A4 = df.nlargest(one_hundred_ps_neutron_window,[' dtp12-
img04']).sum()
        D12_A4_total = int(float(D12_A4.iloc[[2,313]].max()) - float(df_4.iat[313-
Q,0])*one_hundred_ps_neutron_window)

```

```

D12_A5 = df.nlargest(one_hundred_ps_neutron_window,[' dtp12-
img05']).sum()
D12_A5_total = int(float(D12_A5.iloc[[2,314]].max()) - float(df_4.iat[314-
Q,0])*one_hundred_ps_neutron_window)
D12_A6 = df.nlargest(one_hundred_ps_neutron_window,[' dtp12-
img06']).sum()
D12_A6_total = int(float(D12_A6.iloc[[2,315]].max()) - float(df_4.iat[315-
Q,0])*one_hundred_ps_neutron_window)
D12_A7 = df.nlargest(one_hundred_ps_neutron_window,[' dtp12-
img07']).sum()
D12_A7_total = int(float(D12_A7.iloc[[2,316]].max()) - float(df_4.iat[316-
Q,0])*one_hundred_ps_neutron_window)
D12_A8 = df.nlargest(one_hundred_ps_neutron_window,[' dtp12-
img08']).sum()
D12_A8_total = int(float(D12_A8.iloc[[2,317]].max()) - float(df_4.iat[317-
Q,0])*one_hundred_ps_neutron_window)
D12_A9 = df.nlargest(one_hundred_ps_neutron_window,[' dtp12-
img09']).sum()
D12_A9_total = int(float(D12_A9.iloc[[2,318]].max()) - float(df_4.iat[318-
Q,0])*one_hundred_ps_neutron_window)
D12_A10 = df.nlargest(one_hundred_ps_neutron_window,[' dtp12-
img10']).sum()
D12_A10_total = int(float(D12_A10.iloc[[2,319]].max()) -
float(df_4.iat[319-Q,0])*one_hundred_ps_neutron_window)
D12_A11 = df.nlargest(one_hundred_ps_neutron_window,[' dtp12-
img11']).sum()
D12_A11_total = int(float(D12_A11.iloc[[2,320]].max()) -
float(df_4.iat[320-Q,0])*one_hundred_ps_neutron_window)
D12_A12 = df.nlargest(one_hundred_ps_neutron_window,[' dtp12-
img12']).sum()
D12_A12_total = int(float(D12_A12.iloc[[2,321]].max()) -
float(df_4.iat[321-Q,0])*one_hundred_ps_neutron_window)
D12_A13 = df.nlargest(one_hundred_ps_neutron_window,[' dtp12-
img13']).sum()
D12_A13_total = int(float(D12_A13.iloc[[2,322]].max()) -
float(df_4.iat[322-Q,0])*one_hundred_ps_neutron_window)
D12_A14 = df.nlargest(one_hundred_ps_neutron_window,[' dtp12-
img14']).sum()
D12_A14_total = int(float(D12_A14.iloc[[2,323]].max()) -
float(df_4.iat[323-Q,0])*one_hundred_ps_neutron_window)
D12_A15 = df.nlargest(one_hundred_ps_neutron_window,[' dtp12-
img15']).sum()
D12_A15_total = int(float(D12_A15.iloc[[2,324]].max()) -
float(df_4.iat[324-Q,0])*one_hundred_ps_neutron_window)

```

```

D12_A16 = df.nlargest(one_hundred_ps_neutron_window,[' dtp12-
img16']).sum()
D12_A16_total = int(float(D12_A16.iloc[[2,325]].max()) -
float(df_4.iat[325-Q,0])*one_hundred_ps_neutron_window)
D12_A17 = df.nlargest(one_hundred_ps_neutron_window,[' dtp12-
img17']).sum()
D12_A17_total = int(float(D12_A17.iloc[[2,326]].max()) -
float(df_4.iat[326-Q,0])*one_hundred_ps_neutron_window)
D12_A18 = df.nlargest(one_hundred_ps_neutron_window,[' dtp12-
img18']).sum()
D12_A18_total = int(float(D12_A18.iloc[[2,327]].max()) -
float(df_4.iat[327-Q,0])*one_hundred_ps_neutron_window)
D12_A19 = df.nlargest(one_hundred_ps_neutron_window,[' dtp12-
img19']).sum()
D12_A19_total = int(float(D12_A19.iloc[[2,328]].max()) -
float(df_4.iat[328-Q,0])*one_hundred_ps_neutron_window)
D12_A20 = df.nlargest(one_hundred_ps_neutron_window,[' dtp12-
img20']).sum()
D12_A20_total = int(float(D12_A20.iloc[[2,329]].max()) -
float(df_4.iat[329-Q,0])*one_hundred_ps_neutron_window)
D12_A21 = df.nlargest(one_hundred_ps_neutron_window,[' dtp12-
img21']).sum()
D12_A21_total = int(float(D12_A21.iloc[[2,330]].max()) -
float(df_4.iat[330-Q,0])*one_hundred_ps_neutron_window)
D12_A22 = df.nlargest(one_hundred_ps_neutron_window,[' dtp12-
img22']).sum()
D12_A22_total = int(float(D12_A22.iloc[[2,331]].max()) -
float(df_4.iat[331-Q,0])*one_hundred_ps_neutron_window)
D12_A23 = df.nlargest(one_hundred_ps_neutron_window,[' dtp12-
img23']).sum()
D12_A23_total = int(float(D12_A23.iloc[[2,332]].max()) -
float(df_4.iat[332-Q,0])*one_hundred_ps_neutron_window)
D12_A24 = df.nlargest(one_hundred_ps_neutron_window,[' dtp12-
img24']).sum()
D12_A24_total = int(float(D12_A24.iloc[[2,333]].max()) -
float(df_4.iat[333-Q,0])*one_hundred_ps_neutron_window)
D12_A25 = df.nlargest(one_hundred_ps_neutron_window,[' dtp12-
img25']).sum()
D12_A25_total = int(float(D12_A25.iloc[[2,334]].max()) -
float(df_4.iat[334-Q,0])*one_hundred_ps_neutron_window)
D12_A26 = df.nlargest(one_hundred_ps_neutron_window,[' dtp12-
img26']).sum()
D12_A26_total = int(float(D12_A26.iloc[[2,335]].max()) -
float(df_4.iat[335-Q,0])*one_hundred_ps_neutron_window)

```

```

        D12_A27 = df.nlargest(one_hundred_ps_neutron_window,[' dtp12-
img27']).sum()
        D12_A27_total = int(float(D12_A27.iloc[[2,336]].max()) -
float(df_4.iat[336-Q,0])*one_hundred_ps_neutron_window)
        D12_A28 = df.nlargest(one_hundred_ps_neutron_window,[' dtp12-
img28']).sum()
        D12_A28_total = int(float(D12_A28.iloc[[2,337]].max()) -
float(df_4.iat[337-Q,0])*one_hundred_ps_neutron_window)
        detector += 1
    elif detector == 13:
        # D13 = Alpha Pixel 13 FOR detector pixels A1-A28 (1-28)
        D13_A1 = df.nlargest(one_hundred_ps_neutron_window,[' dtp13-
img01']).sum()
        D13_A1_total = int(float(D13_A1.iloc[[2,338]].max()) - float(df_4.iat[338-
Q,0])*one_hundred_ps_neutron_window)
        D13_A2 = df.nlargest(one_hundred_ps_neutron_window,[' dtp13-
img02']).sum()
        D13_A2_total = int(float(D13_A2.iloc[[2,339]].max()) - float(df_4.iat[339-
Q,0])*one_hundred_ps_neutron_window)
        D13_A3 = df.nlargest(one_hundred_ps_neutron_window,[' dtp13-
img03']).sum()
        D13_A3_total = int(float(D13_A3.iloc[[2,340]].max()) - float(df_4.iat[340-
Q,0])*one_hundred_ps_neutron_window)
        D13_A4 = df.nlargest(one_hundred_ps_neutron_window,[' dtp13-
img04']).sum()
        D13_A4_total = int(float(D13_A4.iloc[[2,341]].max()) - float(df_4.iat[341-
Q,0])*one_hundred_ps_neutron_window)
        D13_A5 = df.nlargest(one_hundred_ps_neutron_window,[' dtp13-
img05']).sum()
        D13_A5_total = int(float(D13_A5.iloc[[2,342]].max()) - float(df_4.iat[342-
Q,0])*one_hundred_ps_neutron_window)
        D13_A6 = df.nlargest(one_hundred_ps_neutron_window,[' dtp13-
img06']).sum()
        D13_A6_total = int(float(D13_A6.iloc[[2,343]].max()) - float(df_4.iat[343-
Q,0])*one_hundred_ps_neutron_window)
        D13_A7 = df.nlargest(one_hundred_ps_neutron_window,[' dtp13-
img07']).sum()
        D13_A7_total = int(float(D13_A7.iloc[[2,344]].max()) - float(df_4.iat[344-
Q,0])*one_hundred_ps_neutron_window)
        D13_A8 = df.nlargest(one_hundred_ps_neutron_window,[' dtp13-
img08']).sum()
        D13_A8_total = int(float(D13_A8.iloc[[2,345]].max()) - float(df_4.iat[345-
Q,0])*one_hundred_ps_neutron_window)
        D13_A9 = df.nlargest(one_hundred_ps_neutron_window,[' dtp13-
img09']).sum()

```

```

D13_A9_total = int(float(D13_A9.iloc[[2,346]].max()) - float(df_4.iat[346-
Q,0])*one_hundred_ps_neutron_window)
D13_A10 = df.nlargest(one_hundred_ps_neutron_window,[' dtp13-
img10']).sum()
D13_A10_total = int(float(D13_A10.iloc[[2,347]].max()) -
float(df_4.iat[347-Q,0])*one_hundred_ps_neutron_window)
D13_A11 = df.nlargest(one_hundred_ps_neutron_window,[' dtp13-
img11']).sum()
D13_A11_total = int(float(D13_A11.iloc[[2,348]].max()) -
float(df_4.iat[348-Q,0])*one_hundred_ps_neutron_window)
D13_A12 = df.nlargest(one_hundred_ps_neutron_window,[' dtp13-
img12']).sum()
D13_A12_total = int(float(D13_A12.iloc[[2,349]].max()) -
float(df_4.iat[349-Q,0])*one_hundred_ps_neutron_window)
D13_A13 = df.nlargest(one_hundred_ps_neutron_window,[' dtp13-
img13']).sum()
D13_A13_total = int(float(D13_A13.iloc[[2,350]].max()) -
float(df_4.iat[350-Q,0])*one_hundred_ps_neutron_window)
D13_A14 = df.nlargest(one_hundred_ps_neutron_window,[' dtp13-
img14']).sum()
D13_A14_total = int(float(D13_A14.iloc[[2,351]].max()) -
float(df_4.iat[351-Q,0])*one_hundred_ps_neutron_window)
D13_A15 = df.nlargest(one_hundred_ps_neutron_window,[' dtp13-
img15']).sum()
D13_A15_total = int(float(D13_A15.iloc[[2,352]].max()) -
float(df_4.iat[352-Q,0])*one_hundred_ps_neutron_window)
D13_A16 = df.nlargest(one_hundred_ps_neutron_window,[' dtp13-
img16']).sum()
D13_A16_total = int(float(D13_A16.iloc[[2,353]].max()) -
float(df_4.iat[353-Q,0])*one_hundred_ps_neutron_window)
D13_A17 = df.nlargest(one_hundred_ps_neutron_window,[' dtp13-
img17']).sum()
D13_A17_total = int(float(D13_A17.iloc[[2,354]].max()) -
float(df_4.iat[354-Q,0])*one_hundred_ps_neutron_window)
D13_A18 = df.nlargest(one_hundred_ps_neutron_window,[' dtp13-
img18']).sum()
D13_A18_total = int(float(D13_A18.iloc[[2,355]].max()) -
float(df_4.iat[355-Q,0])*one_hundred_ps_neutron_window)
D13_A19 = df.nlargest(one_hundred_ps_neutron_window,[' dtp13-
img19']).sum()
D13_A19_total = int(float(D13_A19.iloc[[2,356]].max()) -
float(df_4.iat[356-Q,0])*one_hundred_ps_neutron_window)
D13_A20 = df.nlargest(one_hundred_ps_neutron_window,[' dtp13-
img20']).sum()

```

```

        D13_A20_total = int(float(D13_A20.iloc[[2,357]].max()) -
float(df_4.iat[357-Q,0])*one_hundred_ps_neutron_window)
        D13_A21 = df.nlargest(one_hundred_ps_neutron_window,[' dtp13-
img21']).sum()
        D13_A21_total = int(float(D13_A21.iloc[[2,358]].max()) -
float(df_4.iat[358-Q,0])*one_hundred_ps_neutron_window)
        D13_A22 = df.nlargest(one_hundred_ps_neutron_window,[' dtp13-
img22']).sum()
        D13_A22_total = int(float(D13_A22.iloc[[2,359]].max()) -
float(df_4.iat[359-Q,0])*one_hundred_ps_neutron_window)
        D13_A23 = df.nlargest(one_hundred_ps_neutron_window,[' dtp13-
img23']).sum()
        D13_A23_total = int(float(D13_A23.iloc[[2,360]].max()) -
float(df_4.iat[360-Q,0])*one_hundred_ps_neutron_window)
        D13_A24 = df.nlargest(one_hundred_ps_neutron_window,[' dtp13-
img24']).sum()
        D13_A24_total = int(float(D13_A24.iloc[[2,361]].max()) -
float(df_4.iat[361-Q,0])*one_hundred_ps_neutron_window)
        D13_A25 = df.nlargest(one_hundred_ps_neutron_window,[' dtp13-
img25']).sum()
        D13_A25_total = int(float(D13_A25.iloc[[2,362]].max()) -
float(df_4.iat[362-Q,0])*one_hundred_ps_neutron_window)
        D13_A26 = df.nlargest(one_hundred_ps_neutron_window,[' dtp13-
img26']).sum()
        D13_A26_total = int(float(D13_A26.iloc[[2,363]].max()) -
float(df_4.iat[363-Q,0])*one_hundred_ps_neutron_window)
        D13_A27 = df.nlargest(one_hundred_ps_neutron_window,[' dtp13-
img27']).sum()
        D13_A27_total = int(float(D13_A27.iloc[[2,364]].max()) -
float(df_4.iat[364-Q,0])*one_hundred_ps_neutron_window)
        D13_A28 = df.nlargest(one_hundred_ps_neutron_window,[' dtp13-
img28']).sum()
        D13_A28_total = int(float(D13_A28.iloc[[2,365]].max()) -
float(df_4.iat[365-Q,0])*one_hundred_ps_neutron_window)
        detector += 1
    elif detector == 14:
        # D14 = Alpha Pixel 14 FOR detector pixels A1-A28 (1-28)
        D14_A1 = df.nlargest(one_hundred_ps_neutron_window,[' dtp14-
img01']).sum()
        D14_A1_total = int(float(D14_A1.iloc[[2,366]].max()) - float(df_4.iat[366-
Q,0])*one_hundred_ps_neutron_window)
        D14_A2 = df.nlargest(one_hundred_ps_neutron_window,[' dtp14-
img02']).sum()
        D14_A2_total = int(float(D14_A2.iloc[[2,367]].max()) - float(df_4.iat[367-
Q,0])*one_hundred_ps_neutron_window)

```

```

D14_A3 = df.nlargest(one_hundred_ps_neutron_window,[' dtp14-
img03']).sum()
D14_A3_total = int(float(D14_A3.iloc[[2,368]].max()) - float(df_4.iat[368-
Q,0])*one_hundred_ps_neutron_window)
D14_A4 = df.nlargest(one_hundred_ps_neutron_window,[' dtp14-
img04']).sum()
D14_A4_total = int(float(D14_A4.iloc[[2,369]].max()) - float(df_4.iat[369-
Q,0])*one_hundred_ps_neutron_window)
D14_A5 = df.nlargest(one_hundred_ps_neutron_window,[' dtp14-
img05']).sum()
D14_A5_total = int(float(D14_A5.iloc[[2,370]].max()) - float(df_4.iat[370-
Q,0])*one_hundred_ps_neutron_window)
D14_A6 = df.nlargest(one_hundred_ps_neutron_window,[' dtp14-
img06']).sum()
D14_A6_total = int(float(D14_A6.iloc[[2,371]].max()) - float(df_4.iat[371-
Q,0])*one_hundred_ps_neutron_window)
D14_A7 = df.nlargest(one_hundred_ps_neutron_window,[' dtp14-
img07']).sum()
D14_A7_total = int(float(D14_A7.iloc[[2,372]].max()) - float(df_4.iat[372-
Q,0])*one_hundred_ps_neutron_window)
D14_A8 = df.nlargest(one_hundred_ps_neutron_window,[' dtp14-
img08']).sum()
D14_A8_total = int(float(D14_A8.iloc[[2,373]].max()) - float(df_4.iat[373-
Q,0])*one_hundred_ps_neutron_window)
D14_A9 = df.nlargest(one_hundred_ps_neutron_window,[' dtp14-
img09']).sum()
D14_A9_total = int(float(D14_A9.iloc[[2,374]].max()) - float(df_4.iat[374-
Q,0])*one_hundred_ps_neutron_window)
D14_A10 = df.nlargest(one_hundred_ps_neutron_window,[' dtp14-
img10']).sum()
D14_A10_total = int(float(D14_A10.iloc[[2,375]].max()) -
float(df_4.iat[375-Q,0])*one_hundred_ps_neutron_window)
D14_A11 = df.nlargest(one_hundred_ps_neutron_window,[' dtp14-
img11']).sum()
D14_A11_total = int(float(D14_A11.iloc[[2,376]].max()) -
float(df_4.iat[376-Q,0])*one_hundred_ps_neutron_window)
D14_A12 = df.nlargest(one_hundred_ps_neutron_window,[' dtp14-
img12']).sum()
D14_A12_total = int(float(D14_A12.iloc[[2,377]].max()) -
float(df_4.iat[377-Q,0])*one_hundred_ps_neutron_window)
D14_A13 = df.nlargest(one_hundred_ps_neutron_window,[' dtp14-
img13']).sum()
D14_A13_total = int(float(D14_A13.iloc[[2,378]].max()) -
float(df_4.iat[378-Q,0])*one_hundred_ps_neutron_window)

```

```

D14_A14 = df.nlargest(one_hundred_ps_neutron_window,[' dtp14-
img14']).sum()
D14_A14_total = int(float(D14_A14.iloc[[2,379]].max()) -
float(df_4.iat[379-Q,0])*one_hundred_ps_neutron_window)
D14_A15 = df.nlargest(one_hundred_ps_neutron_window,[' dtp14-
img15']).sum()
D14_A15_total = int(float(D14_A15.iloc[[2,380]].max()) -
float(df_4.iat[380-Q,0])*one_hundred_ps_neutron_window)
D14_A16 = df.nlargest(one_hundred_ps_neutron_window,[' dtp14-
img16']).sum()
D14_A16_total = int(float(D14_A16.iloc[[2,381]].max()) -
float(df_4.iat[381-Q,0])*one_hundred_ps_neutron_window)
D14_A17 = df.nlargest(one_hundred_ps_neutron_window,[' dtp14-
img17']).sum()
D14_A17_total = int(float(D14_A17.iloc[[2,382]].max()) -
float(df_4.iat[382-Q,0])*one_hundred_ps_neutron_window)
D14_A18 = df.nlargest(one_hundred_ps_neutron_window,[' dtp14-
img18']).sum()
D14_A18_total = int(float(D14_A18.iloc[[2,383]].max()) -
float(df_4.iat[383-Q,0])*one_hundred_ps_neutron_window)
D14_A19 = df.nlargest(one_hundred_ps_neutron_window,[' dtp14-
img19']).sum()
D14_A19_total = int(float(D14_A19.iloc[[2,384]].max()) -
float(df_4.iat[384-Q,0])*one_hundred_ps_neutron_window)
D14_A20 = df.nlargest(one_hundred_ps_neutron_window,[' dtp14-
img20']).sum()
D14_A20_total = int(float(D14_A20.iloc[[2,385]].max()) -
float(df_4.iat[385-Q,0])*one_hundred_ps_neutron_window)
D14_A21 = df.nlargest(one_hundred_ps_neutron_window,[' dtp14-
img21']).sum()
D14_A21_total = int(float(D14_A21.iloc[[2,386]].max()) -
float(df_4.iat[386-Q,0])*one_hundred_ps_neutron_window)
D14_A22 = df.nlargest(one_hundred_ps_neutron_window,[' dtp14-
img22']).sum()
D14_A22_total = int(float(D14_A22.iloc[[2,387]].max()) -
float(df_4.iat[387-Q,0])*one_hundred_ps_neutron_window)
D14_A23 = df.nlargest(one_hundred_ps_neutron_window,[' dtp14-
img23']).sum()
D14_A23_total = int(float(D14_A23.iloc[[2,388]].max()) -
float(df_4.iat[388-Q,0])*one_hundred_ps_neutron_window)
D14_A24 = df.nlargest(one_hundred_ps_neutron_window,[' dtp14-
img24']).sum()
D14_A24_total = int(float(D14_A24.iloc[[2,389]].max()) -
float(df_4.iat[389-Q,0])*one_hundred_ps_neutron_window)

```

```

        D14_A25 = df.nlargest(one_hundred_ps_neutron_window,[' dtp14-
img25']).sum()
        D14_A25_total = int(float(D14_A25.iloc[[2,390]].max()) -
float(df_4.iat[390-Q,0])*one_hundred_ps_neutron_window)
        D14_A26 = df.nlargest(one_hundred_ps_neutron_window,[' dtp14-
img26']).sum()
        D14_A26_total = int(float(D14_A26.iloc[[2,391]].max()) -
float(df_4.iat[391-Q,0])*one_hundred_ps_neutron_window)
        D14_A27 = df.nlargest(one_hundred_ps_neutron_window,[' dtp14-
img27']).sum()
        D14_A27_total = int(float(D14_A27.iloc[[2,392]].max()) -
float(df_4.iat[392-Q,0])*one_hundred_ps_neutron_window)
        D14_A28 = df.nlargest(one_hundred_ps_neutron_window,[' dtp14-
img28']).sum()
        D14_A28_total = int(float(D14_A28.iloc[[2,393]].max()) -
float(df_4.iat[393-Q,0])*one_hundred_ps_neutron_window)
        detector += 1
    elif detector == 15:
        # D15 = Alpha Pixel 15 FOR detector pixels A1-A28 (1-28)
        D15_A1 = df.nlargest(one_hundred_ps_neutron_window,[' dtp15-
img01']).sum()
        D15_A1_total = int(float(D15_A1.iloc[[2,394]].max()) - float(df_4.iat[393-
Q,0])*one_hundred_ps_neutron_window)
        D15_A2 = df.nlargest(one_hundred_ps_neutron_window,[' dtp15-
img02']).sum()
        D15_A2_total = int(float(D15_A2.iloc[[2,395]].max()) - float(df_4.iat[393-
Q,0])*one_hundred_ps_neutron_window)
        D15_A3 = df.nlargest(one_hundred_ps_neutron_window,[' dtp15-
img03']).sum()
        D15_A3_total = int(float(D15_A3.iloc[[2,396]].max()) - float(df_4.iat[393-
Q,0])*one_hundred_ps_neutron_window)
        D15_A4 = df.nlargest(one_hundred_ps_neutron_window,[' dtp15-
img04']).sum()
        D15_A4_total = int(float(D15_A4.iloc[[2,397]].max()) - float(df_4.iat[393-
Q,0])*one_hundred_ps_neutron_window)
        D15_A5 = df.nlargest(one_hundred_ps_neutron_window,[' dtp15-
img05']).sum()
        D15_A5_total = int(float(D15_A5.iloc[[2,398]].max()) - float(df_4.iat[393-
Q,0])*one_hundred_ps_neutron_window)
        D15_A6 = df.nlargest(one_hundred_ps_neutron_window,[' dtp15-
img06']).sum()
        D15_A6_total = int(float(D15_A6.iloc[[2,399]].max()) - float(df_4.iat[393-
Q,0])*one_hundred_ps_neutron_window)
        D15_A7 = df.nlargest(one_hundred_ps_neutron_window,[' dtp15-
img07']).sum()

```

```

D15_A7_total = int(float(D15_A7.iloc[[2,400]].max()) - float(df_4.iat[400-
Q,0])*one_hundred_ps_neutron_window)
D15_A8 = df.nlargest(one_hundred_ps_neutron_window,[' dtp15-
img08']).sum()
D15_A8_total = int(float(D15_A8.iloc[[2,401]].max()) - float(df_4.iat[401-
Q,0])*one_hundred_ps_neutron_window)
D15_A9 = df.nlargest(one_hundred_ps_neutron_window,[' dtp15-
img09']).sum()
D15_A9_total = int(float(D15_A9.iloc[[2,402]].max()) - float(df_4.iat[402-
Q,0])*one_hundred_ps_neutron_window)
D15_A10 = df.nlargest(one_hundred_ps_neutron_window,[' dtp15-
img10']).sum()
D15_A10_total = int(float(D15_A10.iloc[[2,403]].max()) -
float(df_4.iat[403-Q,0])*one_hundred_ps_neutron_window)
D15_A11 = df.nlargest(one_hundred_ps_neutron_window,[' dtp15-
img11']).sum()
D15_A11_total = int(float(D15_A11.iloc[[2,404]].max()) -
float(df_4.iat[404-Q,0])*one_hundred_ps_neutron_window)
D15_A12 = df.nlargest(one_hundred_ps_neutron_window,[' dtp15-
img12']).sum()
D15_A12_total = int(float(D15_A12.iloc[[2,405]].max()) -
float(df_4.iat[405-Q,0])*one_hundred_ps_neutron_window)
D15_A13 = df.nlargest(one_hundred_ps_neutron_window,[' dtp15-
img13']).sum()
D15_A13_total = int(float(D15_A13.iloc[[2,406]].max()) -
float(df_4.iat[406-Q,0])*one_hundred_ps_neutron_window)
D15_A14 = df.nlargest(one_hundred_ps_neutron_window,[' dtp15-
img14']).sum()
D15_A14_total = int(float(D15_A14.iloc[[2,407]].max()) -
float(df_4.iat[407-Q,0])*one_hundred_ps_neutron_window)
D15_A15 = df.nlargest(one_hundred_ps_neutron_window,[' dtp15-
img15']).sum()
D15_A15_total = int(float(D15_A15.iloc[[2,408]].max()) -
float(df_4.iat[408-Q,0])*one_hundred_ps_neutron_window)
D15_A16 = df.nlargest(one_hundred_ps_neutron_window,[' dtp15-
img16']).sum()
D15_A16_total = int(float(D15_A16.iloc[[2,409]].max()) -
float(df_4.iat[409-Q,0])*one_hundred_ps_neutron_window)
D15_A17 = df.nlargest(one_hundred_ps_neutron_window,[' dtp15-
img17']).sum()
D15_A17_total = int(float(D15_A17.iloc[[2,410]].max()) -
float(df_4.iat[410-Q,0])*one_hundred_ps_neutron_window)
D15_A18 = df.nlargest(one_hundred_ps_neutron_window,[' dtp15-
img18']).sum()

```

```

        D15_A18_total = int(float(D15_A18.iloc[[2,411]].max()) -
float(df_4.iat[411-Q,0])*one_hundred_ps_neutron_window)
        D15_A19 = df.nlargest(one_hundred_ps_neutron_window,[' dtp15-
img19']).sum()
        D15_A19_total = int(float(D15_A19.iloc[[2,412]].max()) -
float(df_4.iat[412-Q,0])*one_hundred_ps_neutron_window)
        D15_A20 = df.nlargest(one_hundred_ps_neutron_window,[' dtp15-
img20']).sum()
        D15_A20_total = int(float(D15_A20.iloc[[2,413]].max()) -
float(df_4.iat[413-Q,0])*one_hundred_ps_neutron_window)
        D15_A21 = df.nlargest(one_hundred_ps_neutron_window,[' dtp15-
img21']).sum()
        D15_A21_total = int(float(D15_A21.iloc[[2,414]].max()) -
float(df_4.iat[414-Q,0])*one_hundred_ps_neutron_window)
        D15_A22 = df.nlargest(one_hundred_ps_neutron_window,[' dtp15-
img22']).sum()
        D15_A22_total = int(float(D15_A22.iloc[[2,415]].max()) -
float(df_4.iat[415-Q,0])*one_hundred_ps_neutron_window)
        D15_A23 = df.nlargest(one_hundred_ps_neutron_window,[' dtp15-
img23']).sum()
        D15_A23_total = int(float(D15_A23.iloc[[2,416]].max()) -
float(df_4.iat[416-Q,0])*one_hundred_ps_neutron_window)
        D15_A24 = df.nlargest(one_hundred_ps_neutron_window,[' dtp15-
img24']).sum()
        D15_A24_total = int(float(D15_A24.iloc[[2,417]].max()) -
float(df_4.iat[417-Q,0])*one_hundred_ps_neutron_window)
        D15_A25 = df.nlargest(one_hundred_ps_neutron_window,[' dtp15-
img25']).sum()
        D15_A25_total = int(float(D15_A25.iloc[[2,418]].max()) -
float(df_4.iat[418-Q,0])*one_hundred_ps_neutron_window)
        D15_A26 = df.nlargest(one_hundred_ps_neutron_window,[' dtp15-
img26']).sum()
        D15_A26_total = int(float(D15_A26.iloc[[2,419]].max()) -
float(df_4.iat[419-Q,0])*one_hundred_ps_neutron_window)
        D15_A27 = df.nlargest(one_hundred_ps_neutron_window,[' dtp15-
img27']).sum()
        D15_A27_total = int(float(D15_A27.iloc[[2,420]].max()) -
float(df_4.iat[420-Q,0])*one_hundred_ps_neutron_window)
        D15_A28 = df.nlargest(one_hundred_ps_neutron_window,[' dtp15-
img28']).sum()
        D15_A28_total = int(float(D15_A28.iloc[[2,421]].max()) -
float(df_4.iat[421-Q,0])*one_hundred_ps_neutron_window)
        detector += 1

```

Take Correlations from timewindow and place in array for each alpha pixel for all 28 detectors

```
alpha_pixel_array_1 = np.asarray([ D1_A1_total, D1_A2_total, D1_A3_total,  
D1_A4_total, D1_A5_total, D1_A6_total, D1_A7_total, D1_A8_total, D1_A9_total,  
D1_A10_total, D1_A11_total, D1_A12_total, D1_A13_total, D1_A14_total,  
D1_A15_total, D1_A16_total, D1_A17_total, D1_A18_total, D1_A19_total,  
D1_A20_total, D1_A21_total, D1_A22_total, D1_A23_total, D1_A24_total,  
D1_A25_total, D1_A26_total, D1_A27_total, D1_A28_total])
```

```
alpha_pixel_array_2 = np.asarray([ D2_A1_total, D2_A2_total, D2_A3_total,  
D2_A4_total, D2_A5_total, D2_A6_total, D2_A7_total, D2_A8_total, D2_A9_total,  
D2_A10_total, D2_A11_total, D2_A12_total, D2_A13_total, D2_A14_total,  
D2_A15_total, D2_A16_total, D2_A17_total, D2_A18_total, D2_A19_total,  
D2_A20_total, D2_A21_total, D2_A22_total, D2_A23_total, D2_A24_total,  
D2_A25_total, D2_A26_total, D2_A27_total, D2_A28_total])
```

```
alpha_pixel_array_3 = np.asarray([ D3_A1_total, D3_A2_total, D3_A3_total,  
D3_A4_total, D3_A5_total, D3_A6_total, D3_A7_total, D3_A8_total, D3_A9_total,  
D3_A10_total, D3_A11_total, D3_A12_total, D3_A13_total, D3_A14_total,  
D3_A15_total, D3_A16_total, D3_A17_total, D3_A18_total, D3_A19_total,  
D3_A20_total, D3_A21_total, D3_A22_total, D3_A23_total, D3_A24_total,  
D3_A25_total, D3_A26_total, D3_A27_total, D3_A28_total])
```

```
alpha_pixel_array_4 = np.asarray([ D4_A1_total, D4_A2_total, D4_A3_total,  
D4_A4_total, D4_A5_total, D4_A6_total, D4_A7_total, D4_A8_total, D4_A9_total,  
D4_A10_total, D4_A11_total, D4_A12_total, D4_A13_total, D4_A14_total,  
D4_A15_total, D4_A16_total, D4_A17_total, D4_A18_total, D4_A19_total,  
D4_A20_total, D4_A21_total, D4_A22_total, D4_A23_total, D4_A24_total,  
D4_A25_total, D4_A26_total, D4_A27_total, D4_A28_total])
```

```
alpha_pixel_array_5 = np.asarray([ D5_A1_total, D5_A2_total, D5_A3_total,  
D5_A4_total, D5_A5_total, D5_A6_total, D5_A7_total, D5_A8_total, D5_A9_total,  
D5_A10_total, D5_A11_total, D5_A12_total, D5_A13_total, D5_A14_total,  
D5_A15_total, D5_A16_total, D5_A17_total, D5_A18_total, D5_A19_total,  
D5_A20_total, D5_A21_total, D5_A22_total, D5_A23_total, D5_A24_total,  
D5_A25_total, D5_A26_total, D5_A27_total, D5_A28_total])
```

```
alpha_pixel_array_6 = np.asarray([ D6_A1_total, D6_A2_total, D6_A3_total,  
D6_A4_total, D6_A5_total, D6_A6_total, D6_A7_total, D6_A8_total, D6_A9_total,  
D6_A10_total, D6_A11_total, D6_A12_total, D6_A13_total, D6_A14_total,  
D6_A15_total, D6_A16_total, D6_A17_total, D6_A18_total, D6_A19_total,  
D6_A20_total, D6_A21_total, D6_A22_total, D6_A23_total, D6_A24_total,  
D6_A25_total, D6_A26_total, D6_A27_total, D6_A28_total])
```

```
alpha_pixel_array_7 = np.asarray([ D7_A1_total, D7_A2_total, D7_A3_total,  
D7_A4_total, D7_A5_total, D7_A6_total, D7_A7_total, D7_A8_total, D7_A9_total,  
D7_A10_total, D7_A11_total, D7_A12_total, D7_A13_total, D7_A14_total,  
D7_A15_total, D7_A16_total, D7_A17_total, D7_A18_total, D7_A19_total,  
D7_A20_total, D7_A21_total, D7_A22_total, D7_A23_total, D7_A24_total,  
D7_A25_total, D7_A26_total, D7_A27_total, D7_A28_total])
```

```

alpha_pixel_array_8 = np.asarray([ D8_A1_total, D8_A2_total, D8_A3_total,
D8_A4_total, D8_A5_total, D8_A6_total, D8_A7_total, D8_A8_total, D8_A9_total,
D8_A10_total, D8_A11_total, D8_A12_total, D8_A13_total, D8_A14_total,
D8_A15_total, D8_A16_total, D8_A17_total, D8_A18_total, D8_A19_total,
D8_A20_total, D8_A21_total, D8_A22_total, D8_A23_total, D8_A24_total,
D8_A25_total, D8_A26_total, D8_A27_total, D8_A28_total])

alpha_pixel_array_9 = np.asarray([ D9_A1_total, D9_A2_total, D9_A3_total,
D9_A4_total, D9_A5_total, D9_A6_total, D9_A7_total, D9_A8_total, D9_A9_total,
D9_A10_total, D9_A11_total, D9_A12_total, D9_A13_total, D9_A14_total,
D9_A15_total, D9_A16_total, D9_A17_total, D9_A18_total, D9_A19_total,
D9_A20_total, D9_A21_total, D9_A22_total, D9_A23_total, D9_A24_total,
D9_A25_total, D9_A26_total, D9_A27_total, D9_A28_total])

alpha_pixel_array_10 = np.asarray([ D10_A1_total, D10_A2_total,
D10_A3_total, D10_A4_total, D10_A5_total, D10_A6_total, D10_A7_total,
D10_A8_total, D10_A9_total, D10_A10_total, D10_A11_total, D10_A12_total,
D10_A13_total, D10_A14_total, D10_A15_total, D10_A16_total, D10_A17_total,
D10_A18_total, D10_A19_total, D10_A20_total, D10_A21_total, D10_A22_total,
D10_A23_total, D10_A24_total, D10_A25_total, D10_A26_total, D10_A27_total,
D10_A28_total])

alpha_pixel_array_11 = np.asarray([ D11_A1_total, D11_A2_total,
D11_A3_total, D11_A4_total, D11_A5_total, D11_A6_total, D11_A7_total,
D11_A8_total, D11_A9_total, D11_A10_total, D11_A11_total, D11_A12_total,
D11_A13_total, D11_A14_total, D11_A15_total, D11_A16_total, D11_A17_total,
D11_A18_total, D11_A19_total, D11_A20_total, D11_A21_total, D11_A22_total,
D11_A23_total, D11_A24_total, D11_A25_total, D11_A26_total, D11_A27_total,
D11_A28_total])

alpha_pixel_array_12 = np.asarray([ D12_A1_total, D12_A2_total,
D12_A3_total, D12_A4_total, D12_A5_total, D12_A6_total, D12_A7_total,
D12_A8_total, D12_A9_total, D12_A10_total, D12_A11_total, D12_A12_total,
D12_A13_total, D12_A14_total, D12_A15_total, D12_A16_total, D12_A17_total,
D12_A18_total, D12_A19_total, D12_A20_total, D12_A21_total, D12_A22_total,
D12_A23_total, D12_A24_total, D12_A25_total, D12_A26_total, D12_A27_total,
D12_A28_total])

alpha_pixel_array_13 = np.asarray([ D13_A1_total, D13_A2_total,
D13_A3_total, D13_A4_total, D13_A5_total, D13_A6_total, D13_A7_total,
D13_A8_total, D13_A9_total, D13_A10_total, D13_A11_total, D13_A12_total,
D13_A13_total, D13_A14_total, D13_A15_total, D13_A16_total, D13_A17_total,
D13_A18_total, D13_A19_total, D13_A20_total, D13_A21_total, D13_A22_total,
D13_A23_total, D13_A24_total, D13_A25_total, D13_A26_total, D13_A27_total,
D13_A28_total])

alpha_pixel_array_14 = np.asarray([ D14_A1_total, D14_A2_total,
D14_A3_total, D14_A4_total, D14_A5_total, D14_A6_total, D14_A7_total,
D14_A8_total, D14_A9_total, D14_A10_total, D14_A11_total, D14_A12_total,
D14_A13_total, D14_A14_total, D14_A15_total, D14_A16_total, D14_A17_total,
D14_A18_total, D14_A19_total, D14_A20_total, D14_A21_total, D14_A22_total,

```

```

D14_A23_total, D14_A24_total, D14_A25_total, D14_A26_total, D14_A27_total,
D14_A28_total])
    alpha_pixel_array_15 = np.asarray([ D15_A1_total, D15_A2_total,
D15_A3_total, D15_A4_total, D15_A5_total, D15_A6_total, D15_A7_total,
D15_A8_total, D15_A9_total, D15_A10_total, D15_A11_total, D15_A12_total,
D15_A13_total, D15_A14_total, D15_A15_total, D15_A16_total, D15_A17_total,
D15_A18_total, D15_A19_total, D15_A20_total, D15_A21_total, D15_A22_total,
D15_A23_total, D15_A24_total, D15_A25_total, D15_A26_total, D15_A27_total,
D15_A28_total])

    # Create .CSV file where each alpha pixel is one row and each column is 1 of 28
detectors
    df_2 = pd.DataFrame({"1": alpha_pixel_array_1, "2": alpha_pixel_array_2, "3":
alpha_pixel_array_3, "4": alpha_pixel_array_4, "5": alpha_pixel_array_5, "6":
alpha_pixel_array_6, "7": alpha_pixel_array_7, "8": alpha_pixel_array_8, "9":
alpha_pixel_array_9, "10": alpha_pixel_array_10, "11": alpha_pixel_array_11, "12":
alpha_pixel_array_12, "13": alpha_pixel_array_13, "14": alpha_pixel_array_14, "15":
alpha_pixel_array_15})
    df_2.to_csv("correlation_counts.csv", index=False)

    os.remove("background_counts.csv")

```

APPENDIX: J – EXAMPLE MCNP INPUT DECK

Array_of_Detectors

C

C ***** Block 1 - Cell Cards *****

C CENTER DETECTOR CARDS

```

11 2 -1.032 -11                                IMP:N=1 IMP:P=1
IMP:E=1 IMP:H=1 $ inside of Plastic Scintillator R1, density 1.032g/cm^3
12 2 -1.032 -12                                IMP:N=1 IMP:P=1
IMP:E=1 IMP:H=1 $ inside of Plastic Scintillator R2, density 1.032g/cm^3
13 2 -1.032 -13                                IMP:N=1 IMP:P=1
IMP:E=1 IMP:H=1 $ inside of Plastic Scintillator L1, density 1.032g/cm^3
14 2 -1.032 -14                                IMP:N=1 IMP:P=1
IMP:E=1 IMP:H=1 $ inside of Plastic Scintillator L2, density 1.032g/cm^3
18 3 -.001205 11 12 13 14 -18                 IMP:N=1 IMP:P=1 IMP:E=1
IMP:H=1 $ air space inside of plastic holding case #1
19 1 -1.09 18 -19                             IMP:N=1 IMP:P=1 IMP:E=1
IMP:H=1 $ inside of polystyrene holding case
C
21 2 -1.032 -21                                IMP:N=1 IMP:P=1
IMP:E=1 IMP:H=1 $ inside of Plastic Scintillator R1, density 1.032g/cm^3
22 2 -1.032 -22                                IMP:N=1 IMP:P=1
IMP:E=1 IMP:H=1 $ inside of Plastic Scintillator R2, density 1.032g/cm^3
23 2 -1.032 -23                                IMP:N=1 IMP:P=1
IMP:E=1 IMP:H=1 $ inside of Plastic Scintillator L1, density 1.032g/cm^3
24 2 -1.032 -24                                IMP:N=1 IMP:P=1
IMP:E=1 IMP:H=1 $ inside of Plastic Scintillator L2, density 1.032g/cm^3
28 3 -.001205 21 22 23 24 -28                 IMP:N=1 IMP:P=1 IMP:E=1
IMP:H=1 $ air space inside of plastic holding case #1
29 1 -1.09 28 -29                             IMP:N=1 IMP:P=1 IMP:E=1
IMP:H=1 $ inside of polystyrene holding case
C
31 2 -1.032 -31                                IMP:N=1 IMP:P=1
IMP:E=1 IMP:H=1 $ inside of Plastic Scintillator R1, density 1.032g/cm^3
32 2 -1.032 -32                                IMP:N=1 IMP:P=1
IMP:E=1 IMP:H=1 $ inside of Plastic Scintillator R2, density 1.032g/cm^3
33 2 -1.032 -33                                IMP:N=1 IMP:P=1
IMP:E=1 IMP:H=1 $ inside of Plastic Scintillator L1, density 1.032g/cm^3
34 2 -1.032 -34                                IMP:N=1 IMP:P=1
IMP:E=1 IMP:H=1 $ inside of Plastic Scintillator L2, density 1.032g/cm^3
38 3 -.001205 31 32 33 34 -38                 IMP:N=1 IMP:P=1 IMP:E=1
IMP:H=1 $ air space inside of plastic holding case #1
39 1 -1.09 38 -39                             IMP:N=1 IMP:P=1 IMP:E=1
IMP:H=1 $ inside of polystyrene holding case
C

```

41 2 -1.032 -41 IMP:N=1 IMP:P=1
 IMP:E=1 IMP:H=1 \$ inside of Plastic Scintillator R1, density 1.032g/cm³
 42 2 -1.032 -42 IMP:N=1 IMP:P=1
 IMP:E=1 IMP:H=1 \$ inside of Plastic Scintillator R2, density 1.032g/cm³
 43 2 -1.032 -43 IMP:N=1 IMP:P=1
 IMP:E=1 IMP:H=1 \$ inside of Plastic Scintillator L1, density 1.032g/cm³
 44 2 -1.032 -44 IMP:N=1 IMP:P=1
 IMP:E=1 IMP:H=1 \$ inside of Plastic Scintillator L2, density 1.032g/cm³
 48 3 -.001205 41 42 43 44 -48 IMP:N=1 IMP:P=1 IMP:E=1
 IMP:H=1 \$ air space inside of plastic holding case #1
 49 1 -1.09 48 -49 IMP:N=1 IMP:P=1 IMP:E=1
 IMP:H=1 \$ inside of polystyrene holding case
 C
 51 2 -1.032 -51 IMP:N=1 IMP:P=1
 IMP:E=1 IMP:H=1 \$ inside of Plastic Scintillator R1, density 1.032g/cm³
 52 2 -1.032 -52 IMP:N=1 IMP:P=1
 IMP:E=1 IMP:H=1 \$ inside of Plastic Scintillator R2, density 1.032g/cm³
 53 2 -1.032 -53 IMP:N=1 IMP:P=1
 IMP:E=1 IMP:H=1 \$ inside of Plastic Scintillator L1, density 1.032g/cm³
 54 2 -1.032 -54 IMP:N=1 IMP:P=1
 IMP:E=1 IMP:H=1 \$ inside of Plastic Scintillator L2, density 1.032g/cm³
 58 3 -.001205 51 52 53 54 -58 IMP:N=1 IMP:P=1 IMP:E=1
 IMP:H=1 \$ air space inside of plastic holding case #1
 59 1 -1.09 58 -59 IMP:N=1 IMP:P=1 IMP:E=1
 IMP:H=1 \$ inside of polystyrene holding case
 C
 61 2 -1.032 -61 IMP:N=1 IMP:P=1
 IMP:E=1 IMP:H=1 \$ inside of Plastic Scintillator R1, density 1.032g/cm³
 62 2 -1.032 -62 IMP:N=1 IMP:P=1
 IMP:E=1 IMP:H=1 \$ inside of Plastic Scintillator R2, density 1.032g/cm³
 63 2 -1.032 -63 IMP:N=1 IMP:P=1
 IMP:E=1 IMP:H=1 \$ inside of Plastic Scintillator L1, density 1.032g/cm³
 64 2 -1.032 -64 IMP:N=1 IMP:P=1
 IMP:E=1 IMP:H=1 \$ inside of Plastic Scintillator L2, density 1.032g/cm³
 68 3 -.001205 61 62 63 64 -68 IMP:N=1 IMP:P=1 IMP:E=1
 IMP:H=1 \$ air space inside of plastic holding case #1
 69 1 -1.09 68 -69 IMP:N=1 IMP:P=1 IMP:E=1
 IMP:H=1 \$ inside of polystyrene holding case
 C
 71 2 -1.032 -71 IMP:N=1 IMP:P=1
 IMP:E=1 IMP:H=1 \$ inside of Plastic Scintillator R1, density 1.032g/cm³
 72 2 -1.032 -72 IMP:N=1 IMP:P=1
 IMP:E=1 IMP:H=1 \$ inside of Plastic Scintillator R2, density 1.032g/cm³
 73 2 -1.032 -73 IMP:N=1 IMP:P=1
 IMP:E=1 IMP:H=1 \$ inside of Plastic Scintillator L1, density 1.032g/cm³

74 2 -1.032 -74 IMP:N=1 IMP:P=1
 IMP:E=1 IMP:H=1 \$ inside of Plastic Scintillator L2, density 1.032g/cm³
 78 3 -.001205 71 72 73 74 -78 IMP:N=1 IMP:P=1 IMP:E=1
 IMP:H=1 \$ air space inside of plastic holding case #1
 79 1 -1.09 78 -79 IMP:N=1 IMP:P=1 IMP:E=1
 IMP:H=1 \$ inside of polystyrene holding case
 C WORLD CARDS
 801 3 -.001205 -80 IMP:N=1 IMP:P=1 IMP:E=1
 IMP:H=1 \$ core - hollow air [DU 18.9 mat 10]
 802 8 -7.82 80 -81 IMP:N=1 IMP:P=1 IMP:E=1
 IMP:H=1 \$ inner shell
 803 10 -18.9 81 -82 IMP:N=1 IMP:P=1 IMP:E=1
 IMP:H=1 \$ 2 cm mid
 804 8 -7.82 82 -83 IMP:N=1 IMP:P=1 IMP:E=1
 IMP:H=1 \$ outter shell
 805 3 -.001205 83 -84 IMP:N=1 IMP:P=1 IMP:E=1
 IMP:H=1 \$ air gap
 806 14 -11.35 84 -85 IMP:N=1 IMP:P=1 IMP:E=1
 IMP:H=1 \$ lead ring
 807 3 -.001205 85 -88 IMP:N=1 IMP:P=1 IMP:E=1 IMP:H=1 \$ air inside
 tricon
 808 3 -.001205 88 -89 IMP:N=1 IMP:P=1 IMP:E=1
 IMP:H=1 \$ Steel Tricon Shipping container or dumpster depending on size made of steel,
 density 7.82g/cm³ mat 8
 809 3 -.001205 19 29 39 49 59 69 79 89 -99 IMP:N=1 IMP:P=1 IMP:E=1 IMP:H=1 \$
 Air surrounding inside shipping container .00125g/cm³
 999 0 99 IMP:N=0
 IMP:P=0 IMP:E=0 IMP:H=0 \$ graveyard universe

C ***** Block 2 - Surface Cards *****

C Center Box 0

11 RPP -6.5 -3.4 -11.9 0 -1.5 1.5
 12 RPP -3.35 -0.025 -11.9 0 -1.5 1.5
 13 RPP 0.025 3.35 -11.9 0 -1.5 1.5
 14 RPP 3.4 6.5 -11.9 0 -1.5 1.5
 18 RPP -7 7 -12 0.1 -1.51 1.51
 19 RPP -7.1 7.1 -12.1 0.2 -1.52 1.52

C Right Box 1

21 2 RPP -6.5 -3.4 -10.741 1.159 -1.5 1.5
 22 2 RPP -3.35 -0.025 -10.741 1.159 -1.5 1.5
 23 2 RPP 0.025 3.35 -10.741 1.159 -1.5 1.5
 24 2 RPP 3.4 6.5 -10.741 1.159 -1.5 1.5
 28 2 RPP -7 7 -10.841 1.2590000000000001 -1.51 1.51
 29 2 RPP -7.1 7.1 -10.940999999999999 1.359 -1.52 1.52

C Right Box 2

31 3 RPP -6.5 -3.4 -7.1905 4.7095 -1.5 1.5
 32 3 RPP -3.35 -0.025 -7.1905 4.7095 -1.5 1.5
 33 3 RPP 0.025 3.35 -7.1905 4.7095 -1.5 1.5
 34 3 RPP 3.4 6.5 -7.1905 4.7095 -1.5 1.5
 38 3 RPP -7 7 -7.2905 4.8095 -1.51 1.51
 39 3 RPP -7.1 7.1 -7.390499999999999 4.9095 -1.52 1.52
 C Right Box 3
 41 4 RPP -6.5 -3.4 -1.0369000000000001 10.8631 -1.5 1.5
 42 4 RPP -3.35 -0.025 -1.0369000000000001 10.8631 -1.5 1.5
 43 4 RPP 0.025 3.35 -1.0369000000000001 10.8631 -1.5 1.5
 44 4 RPP 3.4 6.5 -1.0369000000000001 10.8631 -1.5 1.5
 48 4 RPP -7 7 -1.1369000000000007 10.963099999999999 -1.51 1.51
 49 4 RPP -7.1 7.1 -1.2369000000000003 11.063099999999999 -1.52 1.52
 C Left Box 1
 51 5 RPP -6.5 -3.4 -10.741 1.159 -1.5 1.5
 52 5 RPP -3.35 -0.025 -10.741 1.159 -1.5 1.5
 53 5 RPP 0.025 3.35 -10.741 1.159 -1.5 1.5
 54 5 RPP 3.4 6.5 -10.741 1.159 -1.5 1.5
 58 5 RPP -7 7 -10.841 1.2590000000000001 -1.51 1.51
 59 5 RPP -7.1 7.1 -10.940999999999999 1.359 -1.52 1.52
 C Left Box 2
 61 6 RPP -6.5 -3.4 -7.1905 4.7095 -1.5 1.5
 62 6 RPP -3.35 -0.025 -7.1905 4.7095 -1.5 1.5
 63 6 RPP 0.025 3.35 -7.1905 4.7095 -1.5 1.5
 64 6 RPP 3.4 6.5 -7.1905 4.7095 -1.5 1.5
 68 6 RPP -7 7 -7.2905 4.8095 -1.51 1.51
 69 6 RPP -7.1 7.1 -7.390499999999999 4.9095 -1.52 1.52
 C Left Box 3
 71 7 RPP -6.5 -3.4 -1.0369000000000001 10.8631 -1.5 1.5
 72 7 RPP -3.35 -0.025 -1.0369000000000001 10.8631 -1.5 1.5
 73 7 RPP 0.025 3.35 -1.0369000000000001 10.8631 -1.5 1.5
 74 7 RPP 3.4 6.5 -1.0369000000000001 10.8631 -1.5 1.5
 78 7 RPP -7 7 -1.1369000000000007 10.963099999999999 -1.51 1.51
 79 7 RPP -7.1 7.1 -1.2369000000000003 11.063099999999999 -1.52 1.52
 C
 C WORLD
 C DU Sphere
 80 RCC 0 74.1 -8.0 0 0 16.0 4.25 \$ center of base (5,57.55,-10) height is 20 cm and
 axis is || to the z-axis radius is 10 cm
 81 RCC 0 74.1 -8.1 0 0 16.2 4.45 \$ 2 mm thick steel wall
 82 RCC 0 74.1 -8.2 0 0 16.4 6.45 \$ 2 cm steel ring

 83 RCC 0 74.1 -8.3 0 0 16.6 6.65 \$ 2 mm thick steel wall
 84 RCC 0 74.1 -8.4 0 0 16.8 10.1 \$ inner lead ring
 85 RCC 0 74.1 -8.5 0 0 17.0 12.5 \$ outer lead ring

C Steel Tricon
 88 RPP -99.5 99.5 40.5 149.5 -99.5 99.5 \$ Inner wall - Smaller
 89 RPP -100 100 40 150 -100 100 \$
 outter wall - Larger
 C world air
 99 SO 300
 \$ 6 meter radius air sphere centered at the origin

C ***** Block 3 - data card *****
 C Physics Mode
 MODE N P E H \$ Type of particles to transport: neutrons, photons, protons,
 & electrons
 PHYS:E 100 0 0 0 0 1 1 1 1 0 0 J J 0.917 0.001 0
 CUT:N J 1.0e-3
 CUT:E J 2.0e-3
 CUT:H J 2.0e-5
 C ***** Alpha Detector Source Pixels *****
 C ***** Simple Exemple *****
 C SDEF POS=0 115.1 0 ERG=14.1 VEC=0 -1 0 DIR=D1 \$ Point Source Collimated into
 a 30 degree cone in -Y direction 14.1 MeV Nuutron located at 151cm on Y axis
 C SI1 -1 .933 1 \$
 histogram for cosine bin limits
 C SP1 0 0.933 0.067 \$ frac. solid
 angle for each bin: $\text{bin3} = \sin(x \text{ degree cone}/2)^2$, $\text{bin2} = 1 - \text{bin3}$
 C SB1 0. 0. 1. \$ source bias
 for each bin
 C ***** Actual Pixels *****
 C INSERT_PIXEL_DATA
 C ***** END OF ALPHA PIXELS

 C Alpha Pixel:9
 SDEF POS=0 115.1 0 ERG=14.1 PAR=1
 DIR=D1 VEC=D2
 C location | energy | neutron | cos of the angle between VEC and the particles direction of
 flight
 C in this case .5 degrees
 C VEC of flight
 SI1 0.999961923100 1.000000000000
 SP1 0.000000000000 1.000000000000
 SI2 L -0.570671914 -0.821054216 -0.014266798
 -0.565832174 -0.82439599 -0.014324865
 -0.560933234 -0.827736093 -0.014382903
 -0.5559746489999999 -0.831073793 -0.0144409
 -0.550955987 -0.8344083440000001 -0.014498842
 -0.545876829 -0.8377389740000001 -0.014556715

-0.5407367710000001 -0.841064897 -0.014614506999999999
-0.535535422 -0.8443853020000001 -0.014672203
-0.530272407 -0.8476993620000001 -0.014729788999999998
-0.524947368 -0.8510062270000001 -0.01478725
-0.519559965 -0.854305028 -0.01484457
-0.514109875 -0.857594878 -0.014901736
-0.508596794 -0.860874868 -0.014958728999999999
-0.50302044 -0.864144068 -0.015015536000000001
-0.497380549 -0.867401533 -0.015072138
-0.49167688299999995 -0.870646295 -0.015128518999999998
-0.48590922299999995 -0.8738773670000001 -0.015184663000000001
-0.480077376 -0.8770937459999999 -0.015240552
-0.474181174 -0.880294406 -0.015296167
-0.468220475 -0.8834783070000001 -0.015351491
-0.46219516299999996 -0.886644388 -0.015406505
-0.45610515 -0.889791573 -0.015461192
-0.449950378 -0.8929187679999999 -0.01551553
-0.44373081799999997 -0.8960248609999999 -0.015569501999999999
-0.437446471 -0.899108728 -0.015623088
-0.431097371 -0.902169225 -0.015676268
-0.424683585 -0.905205198 -0.015729022
-0.418205214 -0.908215475 -0.015781329
-0.41166239200000004 -0.911198873 -0.015833168999999998
-0.40505529100000004 -0.9141541959999999 -0.015884521000000002
-0.39838411700000004 -0.917080236 -0.015935365
-0.39164911399999996 -0.919975777 -0.015985678
-0.38485056700000003 -0.922839588 -0.01603544
-0.377988796 -0.925670435 -0.01608463
-0.371064165 -0.928467073 -0.016133225
-0.364077074 -0.93122825 -0.016181203
-0.357027969 -0.93395271 -0.016228543999999998
-0.349917335 -0.936639192 -0.016275225
-0.342745701 -0.9392864320000001 -0.016321224
-0.335513638 -0.941893166 -0.016366519
-0.32822176399999997 -0.944458125 -0.016411088
-0.320870736 -0.946980045 -0.01645491
-0.313461262 -0.949457663 -0.016497961000000002
-0.30599408899999997 -0.95188972 -0.016540221
-0.298470014 -0.95427496 -0.016581666999999998
-0.290889876 -0.956612136 -0.016622279
-0.283254564 -0.95890001 -0.016662033
-0.27556500899999997 -0.96113735 -0.01670091
-0.26782219 -0.9633229390000001 -0.016738887
-0.26002713 -0.9654555709999999 -0.016775943999999997
-0.252180901 -0.967534055 -0.01681206

-0.244284616 -0.9695572170000001 -0.016847215
-0.23633943699999999 -0.9715239 -0.016881388
-0.22834656899999997 -0.973432967 -0.016914561
-0.220307262 -0.975283302 -0.016946712
-0.21222280899999998 -0.9770738109999999 -0.016977825
-0.20409454600000002 -0.978803427 -0.017007879
-0.195923853 -0.980471106 -0.017036857
-0.187712149 -0.9820758340000001 -0.017064741
-0.179460896 -0.983616626 -0.017091514
-0.171171595 -0.985092528 -0.017117159
-0.162845784 -0.9865026170000001 -0.017141661000000002
-0.15448504 -0.987846008 -0.017165003999999998
-0.14609097599999998 -0.989121847 -0.017187174
-0.13766524 -0.99032932 -0.017208155
-0.129209511 -0.99146765 -0.017227935
-0.12072550300000001 -0.9925361009999999 -0.0172465
-0.112214958 -0.993533977 -0.01726384
-0.10367964800000001 -0.994460625 -0.017279941
-0.095121371 -0.995315435 -0.017294795
-0.08654195 -0.9960978420000001 -0.01730839
-0.077943231 -0.996807326 -0.017320718
-0.069327084 -0.997443415 -0.017331771
-0.060695393 -0.998005681 -0.017341541000000002
-0.052050065 -0.9984937490000001 -0.017350022
-0.043393019000000005 -0.99890729 -0.017357207
-0.034726187 -0.999246023 -0.017363093
-0.026051513 -0.9995097209999999 -0.017367675
-0.017370951000000003 -0.999698205 -0.017370951000000003
-0.008686458000000001 -0.999811346 -0.017372916999999998
0.0 -0.999849068 -0.017373572
0.008686458000000001 -0.999811346 -0.017372916999999998
0.017370951000000003 -0.999698205 -0.017370951000000003
0.026051513 -0.9995097209999999 -0.017367675
0.034726187 -0.999246023 -0.017363093
0.043393019000000005 -0.99890729 -0.017357207
0.052050065 -0.9984937490000001 -0.017350022
0.060695393 -0.998005681 -0.017341541000000002
0.069327084 -0.997443415 -0.017331771
0.077943231 -0.996807326 -0.017320718
0.08654195 -0.9960978420000001 -0.01730839
0.095121371 -0.995315435 -0.017294795
0.10367964800000001 -0.994460625 -0.017279941
0.112214958 -0.993533977 -0.01726384
0.12072550300000001 -0.9925361009999999 -0.0172465
0.129209511 -0.99146765 -0.017227935

0.13766524 -0.99032932 -0.017208155
0.14609097599999998 -0.989121847 -0.017187174
0.15448504 -0.987846008 -0.017165003999999998
0.162845784 -0.9865026170000001 -0.017141661000000002
0.171171595 -0.985092528 -0.017117159
0.179460896 -0.983616626 -0.017091514
0.187712149 -0.9820758340000001 -0.017064741
0.195923853 -0.980471106 -0.017036857
0.20409454600000002 -0.978803427 -0.017007879
0.21222280899999998 -0.9770738109999999 -0.016977825
0.220307262 -0.975283302 -0.016946712
0.22834656899999997 -0.973432967 -0.016914561
0.23633943699999999 -0.9715239 -0.016881388
0.244284616 -0.9695572170000001 -0.016847215
0.252180901 -0.967534055 -0.01681206
0.26002713 -0.9654555709999999 -0.016775943999999997
0.26782219 -0.9633229390000001 -0.016738887
0.27556500899999997 -0.96113735 -0.01670091
0.283254564 -0.95890001 -0.016662033
0.290889876 -0.956612136 -0.016622279
0.298470014 -0.95427496 -0.016581666999999998
0.30599408899999997 -0.95188972 -0.016540221
0.313461262 -0.949457663 -0.016497961000000002
0.320870736 -0.946980045 -0.01645491
0.32822176399999997 -0.944458125 -0.016411088
0.335513638 -0.941893166 -0.016366519
0.342745701 -0.9392864320000001 -0.016321224
0.349917335 -0.936639192 -0.016275225
0.357027969 -0.93395271 -0.016228543999999998
0.364077074 -0.93122825 -0.016181203
0.371064165 -0.928467073 -0.016133225
0.377988796 -0.925670435 -0.01608463
0.38485056700000003 -0.922839588 -0.01603544
0.39164911399999996 -0.919975777 -0.015985678
0.39838411700000004 -0.917080236 -0.015935365
0.40505529100000004 -0.9141541959999999 -0.015884521000000002
0.41166239200000004 -0.911198873 -0.015833168999999998
0.418205214 -0.908215475 -0.015781329
0.424683585 -0.905205198 -0.015729022
0.431097371 -0.902169225 -0.015676268
0.437446471 -0.899108728 -0.015623088
0.44373081799999997 -0.8960248609999999 -0.015569501999999999
0.449950378 -0.8929187679999999 -0.01551553
0.45610515 -0.889791573 -0.015461192
0.46219516299999996 -0.886644388 -0.015406505

0.468220475 -0.8834783070000001 -0.015351491
0.474181174 -0.880294406 -0.015296167
0.480077376 -0.8770937459999999 -0.015240552
0.48590922299999995 -0.8738773670000001 -0.015184663000000001
0.49167688299999995 -0.870646295 -0.015128518999999998
0.497380549 -0.867401533 -0.015072138
0.50302044 -0.864144068 -0.015015536000000001
0.508596794 -0.860874868 -0.014958728999999999
0.514109875 -0.857594878 -0.014901736
0.519559965 -0.854305028 -0.01484457
0.524947368 -0.8510062270000001 -0.01478725
0.530272407 -0.8476993620000001 -0.014729788999999998
0.535535422 -0.8443853020000001 -0.014672203
0.5407367710000001 -0.841064897 -0.014614506999999999
0.545876829 -0.8377389740000001 -0.014556715
0.550955987 -0.8344083440000001 -0.014498842
0.5559746489999999 -0.831073793 -0.0144409
0.560933234 -0.827736093 -0.014382903
0.565832174 -0.82439599 -0.014324865
0.570671914 -0.821054216 -0.014266798
-0.570715477 -0.821116893 -0.007133943
-0.56587572 -0.8244594359999999 -0.007162984
-0.560976754 -0.827800312 -0.00719201
-0.556018133 -0.8311387929999999 -0.007221015
-0.550999425 -0.8344741290000001 -0.007249991999999999
-0.545920211 -0.83780555 -0.007278936
-0.540780086 -0.841132269 -0.007307839
-0.53557866 -0.844453476 -0.007336694
-0.530315557 -0.847768341 -0.007365494
-0.524990419 -0.8510760159999999 -0.007394230999999999
-0.519602904 -0.854375633 -0.007422899
-0.514152692 -0.857666302 -0.007451488
-0.508639476 -0.860947114 -0.0074799919999999995
-0.503062975 -0.8642171409999999 -0.007508403
-0.497422926 -0.867475435 -0.007536711
-0.49171908700000005 -0.87072103 -0.007564909
-0.485951242 -0.873952937 -0.007592988
-0.48011919799999997 -0.877170153 -0.00762094
-0.474222784 -0.880371653 -0.007648755
-0.46826186 -0.883556395 -0.007676424
-0.462236309 -0.886723319 -0.0077039380000000005
-0.45614604299999995 -0.889871347 -0.0077312890000000006
-0.44999100299999995 -0.892999386 -0.0077584659999999995
-0.44377116 -0.896106324 -0.0077854590000000001
-0.437486516 -0.899191035 -0.007812259

-0.431137104 -0.902252376 -0.007838856
-0.424722991 -0.9052891909999999 -0.007865241
-0.41824427799999997 -0.908300308 -0.007891401
-0.411701098 -0.911284545 -0.007917329
-0.40509362200000004 -0.914240704 -0.007943012
-0.39842205799999997 -0.917167579 -0.007968441
-0.391686651 -0.9200639490000001 -0.007993605
-0.384887682 -0.922928587 -0.008018493
-0.37802547299999995 -0.925760255 -0.008043095
-0.37110038799999995 -0.928557709 -0.008067399999999999
-0.36411282700000003 -0.9313196970000001 -0.008091395999999999
-0.357063235 -0.934044962 -0.008115074
-0.349952098 -0.936732243 -0.008138421
-0.342779944 -0.939380275 -0.008161426999999999
-0.335547345 -0.941987792 -0.008184081999999999
-0.328254918 -0.944553526 -0.008206373000000001
-0.320903322 -0.947076213 -0.00822829
-0.313493261 -0.9495545879999999 -0.008249823
-0.306025486 -0.951987391 -0.008270959
-0.298500793 -0.9543733679999999 -0.008291689
-0.29092002100000003 -0.956711269 -0.008312001
-0.283284058 -0.958999855 -0.008331884
-0.27559383600000004 -0.961237896 -0.008351328
-0.267850335 -0.9634241720000001 -0.008370323
-0.260054577 -0.965557478 -0.008388857
-0.25220763399999996 -0.967636622 -0.008406921
-0.244310621 -0.969660429 -0.008424504000000001
-0.236364698 -0.971627742 -0.008441595999999999
-0.22837107199999998 -0.973537422 -0.008458188
-0.220330992 -0.975388354 -0.008474269
-0.21224575199999998 -0.9771794429999999 -0.00848983
-0.204116689 -0.97890962 -0.008504862
-0.195945182 -0.980577843 -0.008519356
-0.18773265100000003 -0.982183096 -0.008533302
-0.179480558 -0.9837243940000001 -0.008546693000000001
-0.171190405 -0.985200781 -0.00855952
-0.16286373099999998 -0.986611337 -0.008571775
-0.154502112 -0.987955172 -0.008583451
-0.14610716199999998 -0.989231435 -0.008594539
-0.13768053 -0.9904393090000001 -0.008605033
-0.129223895 -0.9915780190000001 -0.008614926
-0.120738971 -0.9926468270000001 -0.008624212
-0.11222750199999999 -0.9936450379999999 -0.008632885
-0.10369125900000001 -0.994571997 -0.008640938
-0.09513204199999999 -0.9954270940000001 -0.008648366999999999

-0.08655167400000001 -0.996209765 -0.008655167
-0.077952002 -0.996919489 -0.008661334
-0.06933489400000001 -0.997555792 -0.008666862
-0.060702239000000005 -0.998118249 -0.008671748
-0.052055942 -0.998606482 -0.00867599
-0.043397922 -0.999020163 -0.008679584
-0.034730113 -0.9993590109999999 -0.008682528
-0.02605446 -0.999622798 -0.00868482
-0.017372916999999998 -0.999811346 -0.008686458000000001
-0.008687442 -0.9999245259999999 -0.008687442
0.0 -0.999962261 -0.008687769
0.008687442 -0.9999245259999999 -0.008687442
0.017372916999999998 -0.999811346 -0.008686458000000001
0.02605446 -0.999622798 -0.00868482
0.034730113 -0.9993590109999999 -0.008682528
0.043397922 -0.999020163 -0.008679584
0.052055942 -0.998606482 -0.00867599
0.060702239000000005 -0.998118249 -0.008671748
0.06933489400000001 -0.997555792 -0.008666862
0.077952002 -0.996919489 -0.008661334
0.08655167400000001 -0.996209765 -0.008655167
0.09513204199999999 -0.9954270940000001 -0.008648366999999999
0.10369125900000001 -0.994571997 -0.008640938
0.11222750199999999 -0.9936450379999999 -0.008632885
0.120738971 -0.9926468270000001 -0.008624212
0.129223895 -0.9915780190000001 -0.008614926
0.13768053 -0.9904393090000001 -0.008605033
0.14610716199999998 -0.989231435 -0.008594539
0.154502112 -0.987955172 -0.008583451
0.16286373099999998 -0.986611337 -0.008571775
0.171190405 -0.985200781 -0.00855952
0.179480558 -0.9837243940000001 -0.008546693000000001
0.18773265100000003 -0.982183096 -0.008533302
0.195945182 -0.980577843 -0.008519356
0.204116689 -0.97890962 -0.008504862
0.21224575199999998 -0.9771794429999999 -0.00848983
0.220330992 -0.975388354 -0.008474269
0.22837107199999998 -0.973537422 -0.008458188
0.236364698 -0.971627742 -0.008441595999999999
0.244310621 -0.969660429 -0.008424504000000001
0.25220763399999996 -0.967636622 -0.008406921
0.260054577 -0.965557478 -0.008388857
0.267850335 -0.9634241720000001 -0.008370323
0.27559383600000004 -0.961237896 -0.008351328
0.283284058 -0.958999855 -0.008331884

0.29092002100000003 -0.956711269 -0.008312001
0.298500793 -0.9543733679999999 -0.008291689
0.306025486 -0.951987391 -0.008270959
0.313493261 -0.9495545879999999 -0.008249823
0.320903322 -0.947076213 -0.00822829
0.328254918 -0.944553526 -0.008206373000000001
0.335547345 -0.941987792 -0.008184081999999999
0.342779944 -0.939380275 -0.008161426999999999
0.349952098 -0.936732243 -0.008138421
0.357063235 -0.934044962 -0.008115074
0.36411282700000003 -0.9313196970000001 -0.008091395999999999
0.37110038799999995 -0.928557709 -0.008067399999999999
0.37802547299999995 -0.925760255 -0.008043095
0.384887682 -0.922928587 -0.008018493
0.391686651 -0.9200639490000001 -0.007993605
0.39842205799999997 -0.917167579 -0.007968441
0.40509362200000004 -0.914240704 -0.007943012
0.411701098 -0.911284545 -0.007917329
0.41824427799999997 -0.908300308 -0.007891401
0.424722991 -0.9052891909999999 -0.007865241
0.431137104 -0.902252376 -0.007838856
0.437486516 -0.899191035 -0.007812259
0.44377116 -0.896106324 -0.007785459000000001
0.44999100299999995 -0.892999386 -0.007758465999999999
0.45614604299999995 -0.889871347 -0.0077312890000000006
0.462236309 -0.886723319 -0.0077039380000000005
0.46826186 -0.883556395 -0.007676424
0.474222784 -0.880371653 -0.007648755
0.48011919799999997 -0.877170153 -0.00762094
0.485951242 -0.873952937 -0.007592988
0.49171908700000005 -0.87072103 -0.007564909
0.497422926 -0.867475435 -0.007536711
0.503062975 -0.8642171409999999 -0.007508403
0.508639476 -0.860947114 -0.0074799919999999995
0.514152692 -0.857666302 -0.007451488
0.519602904 -0.854375633 -0.007422899
0.524990419 -0.8510760159999999 -0.007394230999999999
0.530315557 -0.847768341 -0.007365494
0.53557866 -0.844453476 -0.007336694
0.540780086 -0.841132269 -0.007307839
0.545920211 -0.83780555 -0.007278936
0.550999425 -0.8344741290000001 -0.007249991999999999
0.556018133 -0.8311387929999999 -0.007221015
0.560976754 -0.827800312 -0.00719201
0.56587572 -0.8244594359999999 -0.007162984

0.570715477 -0.821116893 -0.007133943
-0.5707300000000001 -0.8211377879999999 0.0
-0.5658902379999999 -0.824480587 0.0
-0.560991263 -0.827821722 0.0
-0.556032629 -0.831160463 0.0
-0.551013906 -0.834496061 0.0
-0.545934674 -0.837827746 0.0
-0.5407945270000001 -0.84115473 0.0
-0.535593074 -0.844476204 0.0
-0.530329942 -0.847791338 0.0
-0.525004771 -0.851099284 0.0
-0.51961722 -0.854399172 0.0
-0.514166966 -0.857690113 0.0
-0.508653706 -0.8609711999999999 0.0
-0.503077156 -0.864241503 0.0
-0.497437054 -0.8675000740000001 0.0
-0.49173315799999995 -0.870745945 0.0
-0.48596525100000004 -0.8739781320000001 0.0
-0.48013314100000004 -0.877195627 0.0
-0.474236657 -0.8803974059999999 0.0
-0.468275657 -0.8835824290000001 0.0
-0.46225002600000004 -0.8867496340000001 0.0
-0.456159676 -0.8898979440000001 0.0
-0.45000454700000003 -0.8930262640000001 0.0
-0.44378461 -0.896133484 0.0
-0.437499866 -0.8992184759999999 0.0
-0.431150351 -0.902280098 0.0
-0.42473612899999996 -0.905317193 0.0
-0.418257301 -0.908328591 0.0
-0.411714002 -0.911313108 0.0
-0.40510640200000003 -0.914269546 0.0
-0.39843470799999997 -0.9171966979999999 0.0
-0.391699165 -0.920093345 0.0
-0.384900056 -0.9229582590000001 0.0
-0.37803770200000003 -0.925790201 0.0
-0.371112464 -0.928587927 0.0
-0.364124747 -0.931350186 0.0
-0.35707499299999995 -0.9340757190000001 0.0
-0.349963688 -0.9367632659999999 0.0
-0.342791361 -0.939411562 0.0
-0.33555858299999997 -0.94201934 0.0
-0.328265972 -0.9445853329999999 0.0
-0.320914185 -0.9471082759999999 0.0
-0.31350393 -0.9495869029999999 0.0
-0.306035954 -0.952019955 0.0

-0.298511054 -0.9544061770000001 0.0
-0.290930071 -0.95674432 0.0
-0.283293891 -0.9590331440000001 0.0
-0.275603447 -0.9612714179999999 0.0
-0.267859718 -0.9634579240000001 0.0
-0.260063728 -0.965591455 0.0
-0.252216547 -0.9676708190000001 0.0
-0.24431929100000002 -0.9696948409999999 0.0
-0.23637312 -0.971662363 0.0
-0.228379241 -0.973572248 0.0
-0.22033890399999997 -0.9754233790000001 0.0
-0.21225340199999998 -0.977214661 0.0
-0.20412407100000002 -0.9789450259999999 0.0
-0.195952293 -0.98061343 0.0
-0.187739486 -0.982218858 0.0
-0.17948711399999998 -0.983760324 0.0
-0.171196677 -0.985236874 0.0
-0.162869714 -0.986647585 0.0
-0.154507804 -0.987991568 0.0
-0.146112559 -0.989267972 0.0
-0.137685627 -0.9904759809999999 0.0
-0.12922869 -0.991614817 0.0
-0.120743462 -0.992683744 0.0
-0.112231684 -0.993682066 0.0
-0.103695131 -0.9946091290000001 0.0
-0.09513560000000001 -0.995464323 0.0
-0.086554916 -0.9962470809999999 0.0
-0.077954926 -0.996956884 0.0
-0.069337498 -0.9975932590000001 0.0
-0.060704522000000004 -0.99815578 0.0
-0.052057901 -0.9986440679999999 0.0
-0.043399557000000005 -0.999057795 0.0
-0.034731422000000005 -0.999396682 0.0
-0.026055442999999998 -0.9996604990000001 0.0
-0.017373572 -0.999849068 0.0
-0.008687769 -0.999962261 0.0
0.0 -1.0 0.0
0.008687769 -0.999962261 0.0
0.017373572 -0.999849068 0.0
0.026055442999999998 -0.9996604990000001 0.0
0.034731422000000005 -0.999396682 0.0
0.043399557000000005 -0.999057795 0.0
0.052057901 -0.9986440679999999 0.0
0.060704522000000004 -0.99815578 0.0
0.069337498 -0.9975932590000001 0.0

0.077954926 -0.996956884 0.0
0.086554916 -0.9962470809999999 0.0
0.0951356000000001 -0.995464323 0.0
0.103695131 -0.9946091290000001 0.0
0.112231684 -0.993682066 0.0
0.120743462 -0.992683744 0.0
0.12922869 -0.991614817 0.0
0.137685627 -0.9904759809999999 0.0
0.146112559 -0.989267972 0.0
0.154507804 -0.987991568 0.0
0.162869714 -0.986647585 0.0
0.171196677 -0.985236874 0.0
0.17948711399999998 -0.983760324 0.0
0.187739486 -0.982218858 0.0
0.195952293 -0.98061343 0.0
0.20412407100000002 -0.9789450259999999 0.0
0.21225340199999998 -0.977214661 0.0
0.22033890399999997 -0.9754233790000001 0.0
0.228379241 -0.973572248 0.0
0.23637312 -0.971662363 0.0
0.24431929100000002 -0.9696948409999999 0.0
0.252216547 -0.9676708190000001 0.0
0.260063728 -0.965591455 0.0
0.267859718 -0.9634579240000001 0.0
0.275603447 -0.9612714179999999 0.0
0.283293891 -0.9590331440000001 0.0
0.290930071 -0.95674432 0.0
0.298511054 -0.9544061770000001 0.0
0.306035954 -0.952019955 0.0
0.31350393 -0.9495869029999999 0.0
0.320914185 -0.9471082759999999 0.0
0.328265972 -0.9445853329999999 0.0
0.33555858299999997 -0.94201934 0.0
0.342791361 -0.939411562 0.0
0.349963688 -0.9367632659999999 0.0
0.35707499299999995 -0.9340757190000001 0.0
0.364124747 -0.931350186 0.0
0.371112464 -0.928587927 0.0
0.37803770200000003 -0.925790201 0.0
0.384900056 -0.9229582590000001 0.0
0.391699165 -0.920093345 0.0
0.39843470799999997 -0.9171966979999999 0.0
0.40510640200000003 -0.914269546 0.0
0.411714002 -0.911313108 0.0
0.418257301 -0.908328591 0.0

0.42473612899999996 -0.905317193 0.0
0.431150351 -0.902280098 0.0
0.437499866 -0.8992184759999999 0.0
0.44378461 -0.896133484 0.0
0.45000454700000003 -0.8930262640000001 0.0
0.456159676 -0.8898979440000001 0.0
0.46225002600000004 -0.8867496340000001 0.0
0.468275657 -0.8835824290000001 0.0
0.474236657 -0.8803974059999999 0.0
0.48013314100000004 -0.877195627 0.0
0.48596525100000004 -0.8739781320000001 0.0
0.49173315799999995 -0.870745945 0.0
0.497437054 -0.8675000740000001 0.0
0.503077156 -0.864241503 0.0
0.508653706 -0.8609711999999999 0.0
0.514166966 -0.857690113 0.0
0.51961722 -0.854399172 0.0
0.525004771 -0.851099284 0.0
0.530329942 -0.847791338 0.0
0.535593074 -0.844476204 0.0
0.5407945270000001 -0.84115473 0.0
0.545934674 -0.837827746 0.0
0.551013906 -0.834496061 0.0
0.556032629 -0.831160463 0.0
0.560991263 -0.827821722 0.0
0.5658902379999999 -0.824480587 0.0
0.5707300000000001 -0.8211377879999999 0.0
-0.570715477 -0.821116893 0.007133943
-0.56587572 -0.8244594359999999 0.007162984
-0.560976754 -0.827800312 0.00719201
-0.556018133 -0.8311387929999999 0.007221015
-0.550999425 -0.8344741290000001 0.007249991999999999
-0.545920211 -0.83780555 0.007278936
-0.540780086 -0.841132269 0.007307839
-0.53557866 -0.844453476 0.007336694
-0.530315557 -0.847768341 0.007365494
-0.524990419 -0.8510760159999999 0.007394230999999999
-0.519602904 -0.854375633 0.007422899
-0.514152692 -0.857666302 0.007451488
-0.508639476 -0.860947114 0.0074799919999999995
-0.503062975 -0.8642171409999999 0.007508403
-0.497422926 -0.867475435 0.007536711
-0.49171908700000005 -0.87072103 0.007564909
-0.485951242 -0.873952937 0.007592988
-0.48011919799999997 -0.877170153 0.00762094

-0.474222784 -0.880371653 0.007648755
-0.46826186 -0.883556395 0.007676424
-0.462236309 -0.886723319 0.0077039380000000005
-0.45614604299999995 -0.889871347 0.0077312890000000006
-0.44999100299999995 -0.892999386 0.0077584659999999995
-0.44377116 -0.896106324 0.0077854590000000001
-0.437486516 -0.899191035 0.007812259
-0.431137104 -0.902252376 0.007838856
-0.424722991 -0.90528919099999999 0.007865241
-0.41824427799999997 -0.908300308 0.007891401
-0.411701098 -0.911284545 0.007917329
-0.40509362200000004 -0.914240704 0.007943012
-0.39842205799999997 -0.917167579 0.007968441
-0.391686651 -0.92006394900000001 0.007993605
-0.384887682 -0.922928587 0.008018493
-0.37802547299999995 -0.925760255 0.008043095
-0.37110038799999995 -0.928557709 0.008067399999999999
-0.36411282700000003 -0.93131969700000001 0.008091395999999999
-0.357063235 -0.934044962 0.008115074
-0.349952098 -0.936732243 0.008138421
-0.342779944 -0.939380275 0.008161426999999999
-0.335547345 -0.941987792 0.008184081999999999
-0.328254918 -0.944553526 0.0082063730000000001
-0.320903322 -0.947076213 0.00822829
-0.313493261 -0.94955458799999999 0.008249823
-0.306025486 -0.951987391 0.008270959
-0.298500793 -0.95437336799999999 0.008291689
-0.29092002100000003 -0.956711269 0.008312001
-0.283284058 -0.958999855 0.008331884
-0.27559383600000004 -0.961237896 0.008351328
-0.267850335 -0.96342417200000001 0.008370323
-0.260054577 -0.965557478 0.008388857
-0.25220763399999996 -0.967636622 0.008406921
-0.244310621 -0.969660429 0.0084245040000000001
-0.236364698 -0.971627742 0.008441595999999999
-0.22837107199999998 -0.973537422 0.008458188
-0.220330992 -0.975388354 0.008474269
-0.21224575199999998 -0.97717944299999999 0.00848983
-0.204116689 -0.97890962 0.008504862
-0.195945182 -0.980577843 0.008519356
-0.18773265100000003 -0.982183096 0.008533302
-0.179480558 -0.98372439400000001 0.0085466930000000001
-0.171190405 -0.985200781 0.00855952
-0.16286373099999998 -0.986611337 0.008571775
-0.154502112 -0.987955172 0.008583451

-0.14610716199999998 -0.989231435 0.008594539
-0.13768053 -0.9904393090000001 0.008605033
-0.129223895 -0.9915780190000001 0.008614926
-0.120738971 -0.9926468270000001 0.008624212
-0.11222750199999999 -0.9936450379999999 0.008632885
-0.10369125900000001 -0.994571997 0.008640938
-0.09513204199999999 -0.9954270940000001 0.008648366999999999
-0.08655167400000001 -0.996209765 0.008655167
-0.077952002 -0.996919489 0.008661334
-0.06933489400000001 -0.997555792 0.008666862
-0.060702239000000005 -0.998118249 0.008671748
-0.052055942 -0.998606482 0.00867599
-0.043397922 -0.999020163 0.008679584
-0.034730113 -0.9993590109999999 0.008682528
-0.02605446 -0.999622798 0.00868482
-0.017372916999999998 -0.999811346 0.008686458000000001
-0.008687442 -0.9999245259999999 0.008687442
0.0 -0.999962261 0.008687769
0.008687442 -0.9999245259999999 0.008687442
0.017372916999999998 -0.999811346 0.008686458000000001
0.02605446 -0.999622798 0.00868482
0.034730113 -0.9993590109999999 0.008682528
0.043397922 -0.999020163 0.008679584
0.052055942 -0.998606482 0.00867599
0.060702239000000005 -0.998118249 0.008671748
0.06933489400000001 -0.997555792 0.008666862
0.077952002 -0.996919489 0.008661334
0.08655167400000001 -0.996209765 0.008655167
0.09513204199999999 -0.9954270940000001 0.008648366999999999
0.10369125900000001 -0.994571997 0.008640938
0.11222750199999999 -0.9936450379999999 0.008632885
0.120738971 -0.9926468270000001 0.008624212
0.129223895 -0.9915780190000001 0.008614926
0.13768053 -0.9904393090000001 0.008605033
0.14610716199999998 -0.989231435 0.008594539
0.154502112 -0.987955172 0.008583451
0.16286373099999998 -0.986611337 0.008571775
0.171190405 -0.985200781 0.00855952
0.179480558 -0.9837243940000001 0.008546693000000001
0.18773265100000003 -0.982183096 0.008533302
0.195945182 -0.980577843 0.008519356
0.204116689 -0.97890962 0.008504862
0.21224575199999998 -0.9771794429999999 0.00848983
0.220330992 -0.975388354 0.008474269
0.22837107199999998 -0.973537422 0.008458188

0.236364698 -0.971627742 0.008441595999999999
0.244310621 -0.969660429 0.008424504000000001
0.25220763399999996 -0.967636622 0.008406921
0.260054577 -0.965557478 0.008388857
0.267850335 -0.9634241720000001 0.008370323
0.27559383600000004 -0.961237896 0.008351328
0.283284058 -0.958999855 0.008331884
0.29092002100000003 -0.956711269 0.008312001
0.298500793 -0.9543733679999999 0.008291689
0.306025486 -0.951987391 0.008270959
0.313493261 -0.9495545879999999 0.008249823
0.320903322 -0.947076213 0.00822829
0.328254918 -0.944553526 0.008206373000000001
0.335547345 -0.941987792 0.008184081999999999
0.342779944 -0.939380275 0.008161426999999999
0.349952098 -0.936732243 0.008138421
0.357063235 -0.934044962 0.008115074
0.36411282700000003 -0.9313196970000001 0.008091395999999999
0.37110038799999995 -0.928557709 0.008067399999999999
0.37802547299999995 -0.925760255 0.008043095
0.384887682 -0.922928587 0.008018493
0.391686651 -0.9200639490000001 0.007993605
0.39842205799999997 -0.917167579 0.007968441
0.40509362200000004 -0.914240704 0.007943012
0.411701098 -0.911284545 0.007917329
0.41824427799999997 -0.908300308 0.007891401
0.424722991 -0.9052891909999999 0.007865241
0.431137104 -0.902252376 0.007838856
0.437486516 -0.899191035 0.007812259
0.44377116 -0.896106324 0.007785459000000001
0.44999100299999995 -0.892999386 0.0077584659999999995
0.45614604299999995 -0.889871347 0.0077312890000000006
0.462236309 -0.886723319 0.0077039380000000005
0.46826186 -0.883556395 0.007676424
0.474222784 -0.880371653 0.007648755
0.48011919799999997 -0.877170153 0.00762094
0.485951242 -0.873952937 0.007592988
0.49171908700000005 -0.87072103 0.007564909
0.497422926 -0.867475435 0.007536711
0.503062975 -0.8642171409999999 0.007508403
0.508639476 -0.860947114 0.0074799919999999995
0.514152692 -0.857666302 0.007451488
0.519602904 -0.854375633 0.007422899
0.524990419 -0.8510760159999999 0.007394230999999999
0.530315557 -0.847768341 0.007365494

0.53557866 -0.844453476 0.007336694
0.540780086 -0.841132269 0.007307839
0.545920211 -0.83780555 0.007278936
0.550999425 -0.8344741290000001 0.007249991999999999
0.556018133 -0.8311387929999999 0.007221015
0.560976754 -0.827800312 0.00719201
0.56587572 -0.8244594359999999 0.007162984
0.570715477 -0.821116893 0.007133943
-0.570671914 -0.821054216 0.014266798
-0.565832174 -0.82439599 0.014324865
-0.560933234 -0.827736093 0.014382903
-0.5559746489999999 -0.831073793 0.0144409
-0.550955987 -0.8344083440000001 0.014498842
-0.545876829 -0.8377389740000001 0.014556715
-0.5407367710000001 -0.841064897 0.014614506999999999
-0.535535422 -0.8443853020000001 0.014672203
-0.530272407 -0.8476993620000001 0.014729788999999998
-0.524947368 -0.8510062270000001 0.01478725
-0.519559965 -0.854305028 0.01484457
-0.514109875 -0.857594878 0.014901736
-0.508596794 -0.860874868 0.014958728999999999
-0.50302044 -0.864144068 0.015015536000000001
-0.497380549 -0.867401533 0.015072138
-0.49167688299999995 -0.870646295 0.015128518999999998
-0.48590922299999995 -0.8738773670000001 0.015184663000000001
-0.480077376 -0.8770937459999999 0.015240552
-0.474181174 -0.880294406 0.015296167
-0.468220475 -0.8834783070000001 0.015351491
-0.46219516299999996 -0.886644388 0.015406505
-0.45610515 -0.889791573 0.015461192
-0.449950378 -0.8929187679999999 0.01551553
-0.44373081799999997 -0.8960248609999999 0.015569501999999999
-0.437446471 -0.899108728 0.015623088
-0.431097371 -0.902169225 0.015676268
-0.424683585 -0.905205198 0.015729022
-0.418205214 -0.908215475 0.015781329
-0.41166239200000004 -0.911198873 0.015833168999999998
-0.40505529100000004 -0.9141541959999999 0.015884521000000002
-0.39838411700000004 -0.917080236 0.015935365
-0.39164911399999996 -0.919975777 0.015985678
-0.38485056700000003 -0.922839588 0.01603544
-0.377988796 -0.925670435 0.01608463
-0.371064165 -0.928467073 0.016133225
-0.364077074 -0.93122825 0.016181203
-0.357027969 -0.93395271 0.016228543999999998

-0.349917335 -0.936639192 0.016275225
-0.342745701 -0.9392864320000001 0.016321224
-0.335513638 -0.941893166 0.016366519
-0.32822176399999997 -0.944458125 0.016411088
-0.320870736 -0.946980045 0.01645491
-0.313461262 -0.949457663 0.016497961000000002
-0.30599408899999997 -0.95188972 0.016540221
-0.298470014 -0.95427496 0.016581666999999998
-0.290889876 -0.956612136 0.016622279
-0.283254564 -0.95890001 0.016662033
-0.27556500899999997 -0.96113735 0.01670091
-0.26782219 -0.9633229390000001 0.016738887
-0.26002713 -0.9654555709999999 0.016775943999999997
-0.252180901 -0.967534055 0.01681206
-0.244284616 -0.9695572170000001 0.016847215
-0.23633943699999999 -0.9715239 0.016881388
-0.22834656899999997 -0.973432967 0.016914561
-0.220307262 -0.975283302 0.016946712
-0.21222280899999998 -0.9770738109999999 0.016977825
-0.20409454600000002 -0.978803427 0.017007879
-0.195923853 -0.980471106 0.017036857
-0.187712149 -0.9820758340000001 0.017064741
-0.179460896 -0.983616626 0.017091514
-0.171171595 -0.985092528 0.017117159
-0.162845784 -0.9865026170000001 0.017141661000000002
-0.15448504 -0.987846008 0.017165003999999998
-0.14609097599999998 -0.989121847 0.017187174
-0.13766524 -0.99032932 0.017208155
-0.129209511 -0.99146765 0.017227935
-0.12072550300000001 -0.9925361009999999 0.0172465
-0.112214958 -0.993533977 0.01726384
-0.10367964800000001 -0.994460625 0.017279941
-0.095121371 -0.995315435 0.017294795
-0.08654195 -0.9960978420000001 0.01730839
-0.077943231 -0.996807326 0.017320718
-0.069327084 -0.997443415 0.017331771
-0.060695393 -0.998005681 0.017341541000000002
-0.052050065 -0.9984937490000001 0.017350022
-0.043393019000000005 -0.99890729 0.017357207
-0.034726187 -0.999246023 0.017363093
-0.026051513 -0.9995097209999999 0.017367675
-0.017370951000000003 -0.999698205 0.017370951000000003
-0.008686458000000001 -0.999811346 0.017372916999999998
0.0 -0.999849068 0.017373572
0.008686458000000001 -0.999811346 0.017372916999999998

0.017370951000000003 -0.999698205 0.017370951000000003
0.026051513 -0.9995097209999999 0.017367675
0.034726187 -0.999246023 0.017363093
0.043393019000000005 -0.99890729 0.017357207
0.052050065 -0.9984937490000001 0.017350022
0.060695393 -0.998005681 0.017341541000000002
0.069327084 -0.997443415 0.017331771
0.077943231 -0.996807326 0.017320718
0.08654195 -0.9960978420000001 0.01730839
0.095121371 -0.995315435 0.017294795
0.10367964800000001 -0.994460625 0.017279941
0.112214958 -0.993533977 0.01726384
0.12072550300000001 -0.9925361009999999 0.0172465
0.129209511 -0.99146765 0.017227935
0.13766524 -0.99032932 0.017208155
0.14609097599999998 -0.989121847 0.017187174
0.15448504 -0.987846008 0.017165003999999998
0.162845784 -0.9865026170000001 0.017141661000000002
0.171171595 -0.985092528 0.017117159
0.179460896 -0.983616626 0.017091514
0.187712149 -0.9820758340000001 0.017064741
0.195923853 -0.980471106 0.017036857
0.20409454600000002 -0.978803427 0.017007879
0.21222280899999998 -0.9770738109999999 0.016977825
0.220307262 -0.975283302 0.016946712
0.22834656899999997 -0.973432967 0.016914561
0.23633943699999999 -0.9715239 0.016881388
0.244284616 -0.9695572170000001 0.016847215
0.252180901 -0.967534055 0.01681206
0.26002713 -0.9654557099999999 0.016775943999999997
0.26782219 -0.9633229390000001 0.016738887
0.27556500899999997 -0.96113735 0.01670091
0.283254564 -0.95890001 0.016662033
0.290889876 -0.956612136 0.016622279
0.298470014 -0.95427496 0.016581666999999998
0.30599408899999997 -0.95188972 0.016540221
0.313461262 -0.949457663 0.016497961000000002
0.320870736 -0.946980045 0.01645491
0.32822176399999997 -0.944458125 0.016411088
0.335513638 -0.941893166 0.016366519
0.342745701 -0.9392864320000001 0.016321224
0.349917335 -0.936639192 0.016275225
0.357027969 -0.93395271 0.016228543999999998
0.364077074 -0.93122825 0.016181203
0.371064165 -0.928467073 0.016133225

0.377988796 -0.925670435 0.01608463
 0.38485056700000003 -0.922839588 0.01603544
 0.39164911399999996 -0.919975777 0.015985678
 0.39838411700000004 -0.917080236 0.015935365
 0.40505529100000004 -0.9141541959999999 0.015884521000000002
 0.41166239200000004 -0.911198873 0.015833168999999998
 0.418205214 -0.908215475 0.015781329
 0.424683585 -0.905205198 0.015729022
 0.431097371 -0.902169225 0.015676268
 0.437446471 -0.899108728 0.015623088
 0.44373081799999997 -0.8960248609999999 0.015569501999999999
 0.449950378 -0.8929187679999999 0.01551553
 0.45610515 -0.889791573 0.015461192
 0.46219516299999996 -0.886644388 0.015406505
 0.468220475 -0.8834783070000001 0.015351491
 0.474181174 -0.880294406 0.015296167
 0.480077376 -0.8770937459999999 0.015240552
 0.48590922299999995 -0.8738773670000001 0.015184663000000001
 0.49167688299999995 -0.870646295 0.015128518999999998
 0.497380549 -0.867401533 0.015072138
 0.50302044 -0.864144068 0.015015536000000001
 0.508596794 -0.860874868 0.014958728999999999
 0.514109875 -0.857594878 0.014901736
 0.519559965 -0.854305028 0.01484457
 0.524947368 -0.8510062270000001 0.01478725
 0.530272407 -0.8476993620000001 0.014729788999999998
 0.535535422 -0.8443853020000001 0.014672203
 0.5407367710000001 -0.841064897 0.014614506999999999
 0.545876829 -0.8377389740000001 0.014556715
 0.550955987 -0.8344083440000001 0.014498842
 0.5559746489999999 -0.831073793 0.0144409
 0.560933234 -0.827736093 0.014382903
 0.565832174 -0.82439599 0.014324865
 0.570671914 -0.821054216 0.014266798
 SP2 D 3.0710824788522597e-45
 6.429262383035399e-44
 1.2818100252882073e-42
 2.4337671346834454e-41
 4.400755339148425e-40
 7.578238535569257e-39
 1.2428026254087237e-37
 1.9410150246254868e-36
 2.8870112511608924e-35
 4.0894123095500484e-34
 5.5165323292146736e-33

7.08703013824402e-32
8.670721475033053e-31
1.0102736742063546e-29
1.121025988584086e-28
1.1846366640608088e-27
1.192195567595175e-26
1.1426222590842327e-25
1.0429192933494775e-24
9.065495934364026e-24
7.504560747457798e-23
5.916322393853881e-22
4.4419250889643266e-21
3.17602178552081e-20
2.1626615532483508e-19
1.4024471064447486e-18
8.661184022477883e-18
5.094022813591314e-17
2.853233729541175e-16
1.5219719813742351e-15
7.731589892143221e-15
3.74044958942242e-14
1.7233426916464925e-13
7.561576175991955e-13
3.1597000770066335e-12
1.2573964622335325e-11
4.7653132194150854e-11
1.719901255247762e-10
5.911645995145548e-10
1.9351122835055263e-09
6.0324921324377806e-09
1.790936562763068e-08
5.0635661247405895e-08
1.363407293483725e-07
3.4961300902295097e-07
8.537729674337903e-07
1.985592207010922e-06
4.3977502673692875e-06
9.276067749660356e-06
1.8633314079669037e-05
3.564586361625768e-05
6.49413066196717e-05
0.00011267453211901027
0.0001861760672081099
0.0002929721649045437
0.00043924709524126207

0.0006302262486407868
0.0008937668199263352
0.0014320611112772171
0.003106517717331789
0.007871282028896037
0.0170538385993888
0.027072763108326393
0.030043709382343884
0.023139008817694175
0.01262126219533721
0.005288287838433926
0.0021412774787875673
0.001134358191652654
0.0007658112323593874
0.0005417398754263088
0.00037095508644123824
0.00024233765547952086
0.0001507898517321791
8.935508912097078e-05
5.0426570334065783e-05
2.710144316767512e-05
1.3871334628625156e-05
6.761404582890313e-06
3.1386901381552037e-06
1.3875633053935144e-06
5.841844786272465e-07
2.3422867720492373e-07
8.943818180797044e-08
3.2523606815990593e-08
1.1263342401111646e-08
3.7147425077919354e-09
1.1667639133571643e-09
3.4900375982257305e-10
9.941915488256251e-11
2.697136167213301e-11
6.968327002637102e-12
1.7145376119778256e-12
4.017522845504081e-13
8.965252746200508e-14
1.9052833154795774e-14
3.856110188303502e-15
7.432452640911329e-16
1.364293185986158e-16
2.3849326200366594e-17
3.970428335500329e-18

6.2949379521832635e-19
9.504699484051924e-20
1.3667156599346194e-20
1.8715903419521777e-21
2.4408228729740073e-22
3.031479094794136e-23
3.585632165596045e-24
4.0389617174983964e-25
4.332779448716876e-26
4.426457235995716e-27
4.306642354278097e-28
3.990379363236714e-29
3.521132830358188e-30
2.9589898513037016e-31
2.3680855452180202e-32
1.8048625862264646e-33
1.3100375436881075e-34
9.055578214111134e-36
5.961306809033335e-37
3.737314161377562e-38
2.231364827409339e-39
1.2687451757617451e-40
6.870226716553663e-42
3.542913736828754e-43
1.7399746928598115e-44
8.138008318188783e-46
3.624817712752604e-47
1.5376130544341142e-48
6.2115614775787105e-50
2.3897218942221153e-51
8.755617514933829e-53
3.055055078686513e-54
1.0151825678740352e-55
3.2126403019922794e-57
9.682174351743385e-59
2.778923163957721e-60
7.595791289454337e-62
1.9772534931316052e-63
4.901675131159546e-65
1.1572296194300951e-66
2.601880833953485e-68
5.571191762643729e-70
1.1360610148767514e-71
2.2062161051570384e-73
4.0802548392634586e-75

7.18653211578285e-77
1.2054363878655637e-78
1.92558222261003e-80
2.929359565716662e-82
4.244007490439668e-84
5.855614061906085e-86
7.694166721036154e-88
9.628166331798757e-90
1.1474094445498732e-91
1.3022252155670662e-93
1.4074942358868761e-95
1.4487718322431303e-97
1.4201892026554074e-99
1.3258221294293369e-101
1.1787376754833642e-103
3.3044730473833555e-45
6.917861830671533e-44
1.3792227039156722e-42
2.618724165029419e-41
4.735195979425191e-40
8.154155793558945e-39
1.3372508902513152e-37
2.0885247718381515e-36
3.106413107641128e-35
4.400192065696696e-34
5.9357677699758595e-33
7.625617429387024e-32
9.329663274404407e-31
1.0870506246199883e-29
1.2062196929588228e-28
1.2746645374350403e-27
1.2827978888409653e-26
1.2294571977419589e-25
1.1221771864484121e-24
9.754439088677579e-24
8.074878774240295e-23
6.365940356452806e-22
4.779494473383257e-21
3.4173873415728486e-20
2.32701559223255e-19
1.5090277436506425e-18
9.319401011789749e-18
5.481149140793914e-17
3.070068623060061e-16
1.6376360537224475e-15

8.319161269011867e-15
4.024709508277926e-14
1.8543101708161044e-13
8.136227158132079e-13
3.399825245392164e-12
1.3529538030768557e-11
5.12745886974082e-11
1.8506071983618645e-10
6.360908569258218e-10
2.082173444880657e-09
6.490938552598756e-09
1.92704091862608e-08
5.448377859620982e-08
1.467021053634024e-07
3.76182265792708e-07
9.186564603513454e-07
2.1364896502597754e-06
4.731962533639196e-06
9.981013559723765e-06
2.0049374962639135e-05
3.835481345183652e-05
6.987659855390001e-05
0.00012123737030978371
0.00020032490363387114
0.0003152430426260999
0.0004727818467381744
0.0006807391443826536
0.0009919164607815351
0.0017773403845278873
0.004595796692484097
0.012969762035051164
0.029299549740997133
0.04718119443501092
0.05248921654711569
0.040156190384646455
0.021404526747276272
0.00840842922201944
0.002943865039709494
0.001322616106205192
0.0008350377850544904
0.0005837174642547931
0.0003991863142324973
0.00026075574035873807
0.0001622493366614273
9.614573595236082e-05

5.425879757497254e-05
2.9161049600449237e-05
1.4925503214963507e-05
7.275245572357702e-06
3.3772186312293134e-06
1.4930128304221234e-06
6.285802734431511e-07
2.5202916433465065e-07
9.623514289393557e-08
3.499527702925759e-08
1.2119313513791787e-08
3.997048786379456e-09
1.2554335257674906e-09
3.7552671597414987e-10
1.0697463189781731e-10
2.902108240678057e-11
7.497893307695669e-12
1.8448359386372127e-12
4.3228392762595296e-13
9.646577800108692e-14
2.050077588923751e-14
4.149159872043341e-15
7.99729072630552e-16
1.4679742705916669e-16
2.5661784133136716e-17
4.2721657629108454e-18
6.773329254808614e-19
1.0227020434913891e-19
1.4705808433313553e-20
2.013824077767952e-21
2.626316112549912e-22
3.261859957013295e-23
3.858126550046636e-24
4.345907420850062e-25
4.66205418028768e-26
4.762851122516703e-27
4.6339307662456935e-28
4.293632992748033e-29
3.788725511052954e-30
3.183861807178884e-31
2.5480510249908906e-32
1.9420252668189202e-33
1.409595406175967e-34
9.743767659468871e-36
6.414343415809376e-37

4.021335799646687e-38
2.4009400534918322e-39
1.3651649755981805e-40
7.392337773601285e-42
3.812161683436856e-43
1.8722061407589372e-44
8.756465947110132e-46
3.900290049502441e-47
1.6544657887473028e-48
6.683616485772351e-50
2.5713316540915888e-51
9.421011089910337e-53
3.287227625864383e-54
1.0923325755050664e-55
3.4567887257910783e-57
1.0417982716426284e-58
2.990110737592208e-60
8.173042489868682e-62
2.127517225893498e-63
5.274183767281679e-65
1.245174661824625e-66
2.7996138649834396e-68
5.994581112110127e-70
1.2223973239709757e-71
2.373880124157072e-73
4.390338662555355e-75
7.732681178147348e-77
1.2970449610085096e-78
2.071919135663629e-80
3.151979732769287e-82
4.566535891374677e-84
6.3006184225566056e-86
8.278894079471158e-88
1.0359870292719344e-89
1.2346081910652945e-91
1.401189370880228e-93
1.5144584357023687e-95
1.5588729721269932e-97
1.5281181715814638e-99
1.426579560299185e-101
1.2683172481989894e-103
3.3861468750809754e-45
7.088844691506671e-44
1.4133117692101985e-42
2.683448925430042e-41

4.8522317594096206e-40
8.355695072516004e-39
1.3703025742059958e-37
2.1401450483274372e-36
3.183191657586334e-35
4.508947841112906e-34
6.082477053769831e-33
7.814093278663174e-32
9.560256564113303e-31
1.113918323082261e-29
1.2360327911307705e-28
1.3061693281565663e-27
1.3145037046372326e-26
1.2598446374002997e-25
1.1499130780286937e-24
9.995531198067435e-24
8.274458631067603e-23
6.523281847320149e-22
4.897625141269907e-21
3.501851975089125e-20
2.3845304418937303e-19
1.5463250888426221e-18
9.549740657950475e-18
5.61662200563229e-17
3.1459488775346023e-16
1.6781121002706461e-15
8.524778846862022e-15
4.124184803188384e-14
1.900141565782126e-13
8.337323310375858e-13
3.4838558116314915e-12
1.386393602467217e-11
5.254189875335044e-11
1.896347070131946e-10
6.518125693754775e-10
2.1336367410659377e-09
6.651369516731041e-09
1.974669937140478e-08
5.5830407447943506e-08
1.503280155477261e-07
3.854800403905632e-07
9.413621046039024e-07
2.189295433533226e-06
4.848918395315874e-06
1.0227705716093477e-05

2.0544918177144452e-05
3.9302796498092495e-05
7.160367855937667e-05
0.00012423389189490086
0.00020527624082019598
0.00032303753967201967
0.0004845402490686576
0.0006988093009024952
0.0010308056746946204
0.0019337007374756086
0.005305283039872379
0.01543021915749908
0.03523030994894752
0.05693060634066611
0.06337374576144936
0.04840422948775947
0.025653931668709745
0.009908785155357824
0.0033208795918867333
0.0014038314568618092
0.000860920438478245
0.0005985286386777965
0.00040907170611384155
0.0002672012566358758
0.0001662595290132761
9.852208791613134e-05
5.559986576601115e-05
2.9881798266409785e-05
1.5294404083695025e-05
7.45506158078458e-06
3.4606904491649026e-06
1.5299143487319198e-06
6.441163532389839e-07
2.5825835314856634e-07
9.861370442749084e-08
3.5860225293423867e-08
1.241885619716238e-08
4.095840414937688e-09
1.2864630000585754e-09
3.8480828790908115e-10
1.0961863217512578e-10
2.973837162357931e-11
7.683212309341901e-12
1.890433167127839e-12
4.429683189086523e-13

9.885003994488384e-14
2.1007475993501096e-14
4.251711099915372e-15
8.194952905861005e-16
1.5042569322813422e-16
2.6296044454116777e-17
4.37775722194603e-18
6.940739827860357e-19
1.0479792932339043e-19
1.506927929445062e-20
2.0635980412357973e-21
2.691228516609536e-22
3.34248055348444e-23
3.953484556774408e-24
4.453321489232728e-25
4.777282177120427e-26
4.880570430966768e-27
4.748463660761127e-28
4.399755038901262e-29
3.882368191790914e-30
3.262554590241065e-31
2.611029017970631e-32
1.990024640622919e-33
1.4444351675165944e-34
9.98459601229059e-36
6.572881243603437e-37
4.1207277094928533e-38
2.4602820306934568e-39
1.3989065880722544e-40
7.575047849594128e-42
3.9063836159579986e-43
1.9184798550734943e-44
8.97289200982856e-46
3.996690175303452e-47
1.6953578014295535e-48
6.848809705214158e-50
2.6348850544245125e-51
9.65386214527119e-53
3.3684752132611994e-54
1.1193308234196449e-55
3.5422272095459683e-57
1.0675475064869538e-58
3.064014741552019e-60
8.37504857510864e-62
2.1801012454450986e-63

5.40454124639492e-65
1.2759505765695903e-66
2.8688095210380668e-68
6.142744034866475e-70
1.2526102707810864e-71
2.4325532842812673e-73
4.498850899852519e-75
7.923803230325903e-77
1.329102909474865e-78
2.1231289848780418e-80
3.229884513927795e-82
4.679403044539224e-84
6.456345407178198e-86
8.483516407714687e-88
1.061592632620016e-89
1.2651229434101453e-91
1.4358213674521558e-93
1.5518900066545686e-95
1.5974022991035878e-97
1.565887358516178e-99
1.4618391044183552e-101
1.2996651584134033e-103
3.3044730473833555e-45
6.917861830671533e-44
1.3792227039156722e-42
2.618724165029419e-41
4.735195979425191e-40
8.154155793558945e-39
1.3372508902513152e-37
2.0885247718381515e-36
3.106413107641128e-35
4.400192065696696e-34
5.9357677699758595e-33
7.625617429387024e-32
9.329663274404407e-31
1.0870506246199883e-29
1.2062196929588228e-28
1.2746645374350403e-27
1.2827978888409653e-26
1.2294571977419589e-25
1.1221771864484121e-24
9.754439088677579e-24
8.074878774240295e-23
6.365940356452806e-22
4.779494473383257e-21

3.4173873415728486e-20
2.32701559223255e-19
1.5090277436506425e-18
9.319401011789749e-18
5.481149140793914e-17
3.070068623060061e-16
1.6376360537224475e-15
8.319161269011867e-15
4.024709508277926e-14
1.8543101708161044e-13
8.136227158132079e-13
3.399825245392164e-12
1.3529538030768557e-11
5.12745886974082e-11
1.8506071983618645e-10
6.360908569258218e-10
2.082173444880657e-09
6.490938552598756e-09
1.92704091862608e-08
5.448377859620982e-08
1.467021053634024e-07
3.76182265792708e-07
9.186564603513454e-07
2.1364896502597754e-06
4.731962533639196e-06
9.981013559723765e-06
2.0049374962639135e-05
3.835481345183652e-05
6.987659855390001e-05
0.00012123737030978371
0.00020032490363387114
0.0003152430426260999
0.0004727818467381744
0.0006807391443826536
0.0009919164607815351
0.0017773403845278873
0.004595796692484097
0.012969762035051164
0.029299549740997133
0.04718119443501092
0.05248921654711569
0.040156190384646455
0.021404526747276272
0.00840842922201944
0.002943865039709494

0.001322616106205192
0.0008350377850544904
0.0005837174642547931
0.0003991863142324973
0.00026075574035873807
0.0001622493366614273
9.614573595236082e-05
5.425879757497254e-05
2.9161049600449237e-05
1.4925503214963507e-05
7.275245572357702e-06
3.3772186312293134e-06
1.4930128304221234e-06
6.285802734431511e-07
2.5202916433465065e-07
9.623514289393557e-08
3.499527702925759e-08
1.2119313513791787e-08
3.997048786379456e-09
1.2554335257674906e-09
3.7552671597414987e-10
1.0697463189781731e-10
2.902108240678057e-11
7.497893307695669e-12
1.8448359386372127e-12
4.3228392762595296e-13
9.646577800108692e-14
2.050077588923751e-14
4.149159872043341e-15
7.99729072630552e-16
1.4679742705916669e-16
2.5661784133136716e-17
4.2721657629108454e-18
6.773329254808614e-19
1.0227020434913891e-19
1.4705808433313553e-20
2.013824077767952e-21
2.626316112549912e-22
3.261859957013295e-23
3.858126550046636e-24
4.345907420850062e-25
4.66205418028768e-26
4.762851122516703e-27
4.6339307662456935e-28
4.293632992748033e-29

3.788725511052954e-30
3.183861807178884e-31
2.5480510249908906e-32
1.9420252668189202e-33
1.409595406175967e-34
9.743767659468871e-36
6.414343415809376e-37
4.021335799646687e-38
2.4009400534918322e-39
1.3651649755981805e-40
7.392337773601285e-42
3.812161683436856e-43
1.8722061407589372e-44
8.756465947110132e-46
3.900290049502441e-47
1.6544657887473028e-48
6.683616485772351e-50
2.5713316540915888e-51
9.421011089910337e-53
3.287227625864383e-54
1.0923325755050664e-55
3.4567887257910783e-57
1.0417982716426284e-58
2.990110737592208e-60
8.173042489868682e-62
2.127517225893498e-63
5.274183767281679e-65
1.245174661824625e-66
2.7996138649834396e-68
5.994581112110127e-70
1.2223973239709757e-71
2.373880124157072e-73
4.390338662555355e-75
7.732681178147348e-77
1.2970449610085096e-78
2.071919135663629e-80
3.151979732769287e-82
4.566535891374677e-84
6.3006184225566056e-86
8.278894079471158e-88
1.0359870292719344e-89
1.2346081910652945e-91
1.401189370880228e-93
1.5144584357023687e-95
1.5588729721269932e-97

1.5281181715814638e-99
1.426579560299185e-101
1.2683172481989894e-103
3.0710824788522597e-45
6.429262383035399e-44
1.2818100252882073e-42
2.4337671346834454e-41
4.400755339148425e-40
7.578238535569257e-39
1.2428026254087237e-37
1.9410150246254868e-36
2.8870112511608924e-35
4.0894123095500484e-34
5.5165323292146736e-33
7.08703013824402e-32
8.670721475033053e-31
1.0102736742063546e-29
1.121025988584086e-28
1.1846366640608088e-27
1.192195567595175e-26
1.1426222590842327e-25
1.0429192933494775e-24
9.065495934364026e-24
7.504560747457798e-23
5.916322393853881e-22
4.4419250889643266e-21
3.17602178552081e-20
2.1626615532483508e-19
1.4024471064447486e-18
8.661184022477883e-18
5.094022813591314e-17
2.853233729541175e-16
1.5219719813742351e-15
7.731589892143221e-15
3.74044958942242e-14
1.7233426916464925e-13
7.561576175991955e-13
3.1597000770066335e-12
1.2573964622335325e-11
4.7653132194150854e-11
1.719901255247762e-10
5.911645995145548e-10
1.9351122835055263e-09
6.0324921324377806e-09
1.790936562763068e-08

5.0635661247405895e-08
1.363407293483725e-07
3.4961300902295097e-07
8.537729674337903e-07
1.985592207010922e-06
4.3977502673692875e-06
9.276067749660356e-06
1.8633314079669037e-05
3.564586361625768e-05
6.49413066196717e-05
0.00011267453211901027
0.0001861760672081099
0.0002929721649045437
0.00043924709524126207
0.0006302262486407868
0.0008937668199263352
0.0014320611112772171
0.003106517717331789
0.007871282028896037
0.0170538385993888
0.027072763108326393
0.030043709382343884
0.023139008817694175
0.01262126219533721
0.005288287838433926
0.0021412774787875673
0.001134358191652654
0.0007658112323593874
0.0005417398754263088
0.00037095508644123824
0.00024233765547952086
0.0001507898517321791
8.935508912097078e-05
5.0426570334065783e-05
2.710144316767512e-05
1.3871334628625156e-05
6.761404582890313e-06
3.1386901381552037e-06
1.3875633053935144e-06
5.841844786272465e-07
2.3422867720492373e-07
8.943818180797044e-08
3.2523606815990593e-08
1.1263342401111646e-08
3.7147425077919354e-09

1.1667639133571643e-09
3.4900375982257305e-10
9.941915488256251e-11
2.697136167213301e-11
6.968327002637102e-12
1.7145376119778256e-12
4.017522845504081e-13
8.965252746200508e-14
1.9052833154795774e-14
3.856110188303502e-15
7.432452640911329e-16
1.364293185986158e-16
2.3849326200366594e-17
3.970428335500329e-18
6.2949379521832635e-19
9.504699484051924e-20
1.3667156599346194e-20
1.8715903419521777e-21
2.4408228729740073e-22
3.031479094794136e-23
3.585632165596045e-24
4.0389617174983964e-25
4.332779448716876e-26
4.426457235995716e-27
4.306642354278097e-28
3.990379363236714e-29
3.521132830358188e-30
2.9589898513037016e-31
2.3680855452180202e-32
1.8048625862264646e-33
1.3100375436881075e-34
9.055578214111134e-36
5.961306809033335e-37
3.737314161377562e-38
2.231364827409339e-39
1.2687451757617451e-40
6.870226716553663e-42
3.542913736828754e-43
1.7399746928598115e-44
8.138008318188783e-46
3.624817712752604e-47
1.5376130544341142e-48
6.2115614775787105e-50
2.3897218942221153e-51
8.755617514933829e-53

3.055055078686513e-54
 1.0151825678740352e-55
 3.2126403019922794e-57
 9.682174351743385e-59
 2.778923163957721e-60
 7.595791289454337e-62
 1.9772534931316052e-63
 4.901675131159546e-65
 1.1572296194300951e-66
 2.601880833953485e-68
 5.571191762643729e-70
 1.1360610148767514e-71
 2.2062161051570384e-73
 4.0802548392634586e-75
 7.18653211578285e-77
 1.2054363878655637e-78
 1.92558222261003e-80
 2.929359565716662e-82
 4.244007490439668e-84
 5.855614061906085e-86
 7.694166721036154e-88
 9.628166331798757e-90
 1.1474094445498732e-91
 1.3022252155670662e-93
 1.4074942358868761e-95
 1.4487718322431303e-97
 1.4201892026554074e-99
 1.3258221294293369e-101
 1.1787376754833642e-103
 TRCL=1
 C Transformation Cards
 *TR2 15 0 0 7.9 82.1 90 97.9 7.9 90 90 90 0
 *TR3 30 0 0 15.9 74.1 90 105.9 15.9 90 90 90 0
 *TR4 47 0 0 24.3 65.7 90 114.3 24.3 90 90 90 0
 *TR5 -15 0 0 -7.9 97.9 90 82.1 -7.9 90 90 90 0
 *TR6 -30 0 0 -15.9 105.9 90 74.1 -15.9 90 90 90 0
 *TR7 -47 0 0 -24.3 114.3 90 65.7 -24.3 90 90 90 0
 C
 C ***** END OF ALPHA PIXELS

 C Plastic Casing - Polyvinyl Toluene
 M1 1000 -0.085000 6000 -0.915000
 \$ H and C
 C
 C Plastic Scintillator

M2 1000 -0.085000 6000 -0.915000
\$ Hydrogen H-1 and mass fraction Carbon 12 and mass fraction
C
C Air for space surrounding detector
M3 6012 -0.000124 7014 -0.755268 8016 -0.231781 18000 -0.012827 \$ C, N, O, Ar
libraries referenced from: https://mcnp.lanl.gov/pdf_files/la-ur-03-1987g_n.pdf
C
C RDX (HE Explosive)
M7 1001 -0.02723 6012 -0.16222 7014 -0.37836 8016 -0.432190 \$ H,C,N,O
C
C Standard Fricke Steel
M8 6012 -0.005 26000 -0.995
\$ Carbon, Fe - Iron
C
C Lithium 6 Deuteride
M9 3006 -.50 1002 -.50
\$ Lithium 6 50% & Deutirium 50%
C
C DU Ball
M10 92234 -0.000005 92235 -0.002500 92238 -0.997500 \$ U-
234,235,238
C HUE Ball
M11 92234 -0.000005 92235 -0.964995 92238 -0.035 \$ U-
234,235,238
C Aluminum
M12 13027 -1.000000
\$ 2.6989 g/cm^3
C Normal Polyethylene
M13 1001 -0.143716 6012 -0.856284
\$ H and C 0.9300g/cm^3
C lead
M14 82000 -1.000000
\$ lead (Pb density 11.35g/cm^3)
C Tallies:
C ***** TALLIES

C Not in use since PTRAC is created and counts are collected in PTRAC file
C

NPS 6.92E5 \$ Number of Particles Simulated
PTRAC BUFFER=1000 FILE=BIN MAX=100000000000 WRITE=ALL
FILTER=2,0,2,5,TME TYPE=N,P,E,H CELL=11,12,13,14,21,22,23,24
31,32,33,34,41,42,43,44,51,52,53,54,61,62,63,64,71,72,73,74

C Only count events between 20 and 25 nanoseconds that are neutrons, photons,
electrons, or protons that occur in

C cells

PRINT

generated

\$ Only print table 162 [Tally Table] if desired. PTRAC already

VITA

Benjamin Charles Troxell was born in Philadelphia, Pennsylvania on March 14, 1989. The eldest of four boys, he was raised in Harrisburg, Pennsylvania and graduated from Central Dauphin East High School in June 2007. In July 2007 he reported to the United States Military Academy at West Point to begin his undergraduate education. While at West Point he completed Airborne and Air Assault Schools in addition to earning a Bachelor of Science in Physics in 2011 – graduating in the top 2% of his class. He was then commissioned into the Infantry as a Second Lieutenant. In 2011 he completed the Infantry Basic Officer Leader's Course and in 2012 he graduated Ranger School. From 2012 – 2013 he deployed to Kandahar province Afghanistan where he served as a Platoon Leader. There he earned a Bronze Star and the Combat Infantryman's Badge. Upon returning to Tacoma Washington in 2013, he met his to be wife Lauren Myers – a native of Puyallup, Washington. Over the next four years he served as a Company Executive Officer and Company Commander at Fort Benning, Georgia and Fort Jackson, South Carolina. Along the way he would earn the Expert Infantryman's Badge and graduate from the Maneuver Center of Excellence Captain's Career Course. From 2017-2018 he deployed to Iraq in the fight against ISIS serving as a Brigade Plans Officer. In August 2018, 4 months after returning from Iraq, he reported to the University of Tennessee at Knoxville to complete his Master of Science in Nuclear Engineering. Upon completion his will report to Aberdeen Proving Grounds, Maryland to continue his service to the nation as a member of a Nuclear Disablement Team at the 20th Chemical, Biological, Radiological, Nuclear and Explosives Command.