Monte Carlo Simulation of High Energy Neutrino Event Transport in Cerenkov Detectors

By Mykalin Jones

A Thesis

submitted to the faculty of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Master of Science

in

Nuclear Science and Engineering

January 2020
Acknowledgments

My years at WPI would not have been marked by such an incredible time of intellectual and personal growth were it not for the numerous faculty and staff members of this remarkable institution. I fondly remember the six years I spent within the halls of Olin, cultivating not only physical intuition, but also invaluable skills working alongside other young scientists under the guidance of caring and passionate professors. At WPI, I truly believe that the opportunity I had to pursue my own personal research interests, to such an extent, was incredibly unique. I continue to be grateful for the opportunity and for the support I received as I embarked down my own pathway to hopefully making but a dent on our understanding of the wonders of our cosmos.

Such an opportunity would not have arrived without the generous offer for me to join the University of Wisconsin River Falls (UWRF) Summer Program in Particle Astrophysics the summer after my junior year at WPI. The training I received there under James Madsen and Surujhdeo Seunarine has been the catalyst for nearly every professional journey I have embarked on since. I would also like to extend my deepest gratitude to the National Science Foundation, who continues to fund that REU, so that other young minds may benefit from everything it has to offer.

Even more, this thesis, my undergraduate degree, numerous interdisciplinary research opportunities within the nuclear laboratory, attending the National Nuclear Physics Summer School at Yale, and my current placement studying abroad in France would not have been possible were it not for David Medich agreeing to advise first my MQP and then my Master’s degree. Dave enthusiastically agreed to take on my project in neutrino physics, not in spite of it being outside of his research interest, but instead in a way that integrated
it into his lab, drawing analogies to other particle transport
problems. In this way, I received an education that crossed disciplines
and created multiple avenues for me to retain and apply knowledge
within a broader realm of nuclear and particle physics. I am
indescribably grateful for his years of skillful pedagogy,
encouragement, uncountable hours spent writing recommendations
and editing writing for his students, and his ever-helpful upbeat
advice delivered in short burst during his open-office hours usually
followed by ”do good things with your life” echoing down the hallway
as he sent us off with renewed hope in our endeavors.

This, of course, would have never happened if it weren’t for Shaun
Marshall. First, Shaun encouraged me to take this project to Dave
when I brought it back as an idea for my MQP senior year. Shaun
continued to be a mentor to me throughout this work and to this day
remains available to help with a variety of topics as I continue my
education. He is responsible for everything I know about GEANT4
and was an integral part of the development of the codes used to
carry out the experiments described within this work. I also had the
wonderful opportunity to help Shaun a small amount with his
doctoral research and learn about the re-suspension of radioactive
particles. Shaun fosters a delightful environment within a lab, where
his colleagues benefit from his willingness to help pass on his
unbelievable range of skills as a researcher, and even more from his
infectious enthusiasm for his research and physics as a whole. My
years at WPI would have not as been nearly as enjoyable or
successful were it not for Shaun, starting at day one at the physics
lounge help desk my freshman year and continuing today through our
work and friendship.

I cannot start the discussion on our topic at hand without first
mentioning several other professors who guided me along the way. I
would like to extend my appreciation to Konstantine Lurie in the
math department, for whom I was a Teaching Assistant. Professor
Lurie was always available to help in any way he could and patiently explained concepts in a way that led his students to a fundamental understanding of the topic. Izabela Stroe, a member of the NSE program as well as the physics department was a constant mentor to me over the years and served as my undergraduate advisor. She taught numerous courses and independent studies which I was enrolled in. Her fun, informative teaching style ingrained bits of useful information I am still able to recall many years later with a smile on my face. She encouraged me to take on projects in nuclear astrophysics and benefit from coursework by applying it to my own interests. Marko Popovic took it upon himself, without any other attendees, to personally pass on ”The Fairy Tale of Particle Physics” in our weekly meetings over numerous independent studies that continued for a year and a half despite his busy schedule. My copy of Peskin and Schroeder remains riddled with his notes about concepts and anecdotes concerning the physicists behind them, making it a unique personal guide to Quantum Field Theory and a testament to Professor Popovic’s vast knowledge and passion about this topic.

Finally, I would like to extend a most heart-felt thank you to the numerous friends and family members that continue to support me through the many years my education drags on. My mother, father, and step-father have all offered support one way or another, all while nodding their heads as I ramble on about neutrinos. Amy Fenster, Meg and Melanie Cantwell, Alexandre Pauwels, and many other friends I am so lucky to have, have been stead-fast and encouraging despite the ups-and-downs and my recurrent inability to be there in-person due to my education. My dog Annie, perhaps, deserves her own degree, always happy to sit by me in the lab or elsewhere tapping away at the computer or running across campus to grab yet another coffee. I couldn’t have asked for a better partner-in-crime. So, from Annie and me, thank you to all for your part in this small chapter of ”The Fairy Tale of Particle Physics”.

4
In memory of my Van de Graaff Generator (2011-2019)
Abstract

This body of work was aimed at the development of an application for simulating high energy neutrino events (up to 100 GeV) in GEANT4. While various Monte Carlo techniques for neutrino transport exist, this was motivated by current interest in expanding GEANT4’s application to neutrino physics and handling of neutrino related processes. Multiple neutrino observatories detect astrophysical neutrinos by measuring optical Cerenkov light produced from interactions; this application may be modified for a variety of experiments. The need for the generation of the computationally inexpensive model discussed in the paper arose from neutrino oscillation predictions for high energy cosmic neutrinos. A predicted flavour ratio of approximately 1:1:1 for $\nu_e : \nu_\mu : \nu_\tau$ within the IceCube array motivates work towards differentiating neutrino flavour across detected events. Simulation techniques can be implemented to provide information that may be used when designing data analysis algorithms. Tau neutrino transport simulation was explored in detail utilising the developed application to probe properties of 100 GeV $\nu_\tau$ events as a preliminary investigation into its many uses. Results from 10,000 $\nu_\tau$ events showed agreement with preliminary calculations relating the time of flight to the contribution to the signal over time of the optical photons produced at the neutrino interaction and tau decay vertices.
Contents

Acknowledgments 2
Dedication 5
Abstract 6

1 Introduction 13

2 Background 16
2.1 Discovery of the Neutrino . . . . . . . . . . . . . . . . 16
2.2 Neutrino Properties . . . . . . . . . . . . . . . . . . . 17
  2.2.1 Neutrino Sources . . . . . . . . . . . . . . . . . . 18
  2.2.2 Neutrino Oscillations . . . . . . . . . . . . . . . . 20
  2.2.3 Neutrino interactions: Deep Inelastic Scattering 20
2.3 Cerenkov radiation . . . . . . . . . . . . . . . . . . . . 22
2.4 Instrumentation . . . . . . . . . . . . . . . . . . . . . 24
  2.4.1 PMTs . . . . . . . . . . . . . . . . . . . . . . . . 24
2.5 Neutrinos and the IceCube array . . . . . . . . . . . . 26
  2.5.1 DOMS . . . . . . . . . . . . . . . . . . . . . . . 29
  2.5.2 Event Topography . . . . . . . . . . . . . . . . . 30
  2.5.3 The Tau Neutrino Search . . . . . . . . . . . . . 33

3 Methods and Materials 35
3.1 Preliminary Calculations . . . . . . . . . . . . . . . . . . 35
3.2 The Suite of Programs . . . . . . . . . . . . . . . . . . . 35
3.3 Monte Carlo Techniques . . . . . . . . . . . . . . . . . . 37
  3.3.1 GENIE . . . . . . . . . . . . . . . . . . . . . . . 37
  3.3.2 GEANT4 . . . . . . . . . . . . . . . . . . . . . . 38
List of Tables

1  The distribution of particle from the GENIE neutrino event generator and their corresponding kinematic information. 200 events were generated for a 100 Gev incident neutrino interacting in ice. 151 CC Tau Neutrino events were selected and averaged.  . . . . . . . . . . . . . . . . 44
List of Figures

1 Simulated Tau Neutrino event from the IceCube collaboration, of energy on the order of PeVs, depicting signature “double bang” topography. While this event has not been observed, high energy events such as this are often not fully contained. Expansion of the IceCube array in the next generation of the detector may help to identify tau neutrino events. Distinguishing tau neutrino events from highly energetic muons with ”catastrophic losses” along their track is a difficult process and an event that is not fully contained may be missing vital information for event analysis. [3] 15

2 Feynmann diagram of $\beta$ – decay[24] 16

3 Feynman diagram for and electron neutrino undergoing a CC DIS off of a neutron. This diagram can be generalized to any flavour neutrino by replacing the lepton with consideration to lepton number conservation. . . . 21

4 Cerenkov light emission and wave-front angles. In a dispersive medium, $\theta_c + \eta \neq 90^0$[19] . . . . . . . . . . . 22

5 Schematic for a PMT[15] . . . . . . . . . . . . . . . . . . . . . 26
The IceCube Array is made up of 86 strings, including 8 DeepCore strings, set around 125 meters apart from each other for a total of 5,160 Dynamic Optical Modules (DOMs). DeepCore is situated 2100 m below the surface of the icecap and is more densely packed, with around 480 optical modules making it sensitive to lower energy ranges (as low as 10 GeV)[7][6]. A layer of dust (not depicted) exists at around 1970 m and 2100 m. Because of thousands of years of compression, the Antarctic ice is very pure otherwise. However, due to refilling of the holes where strings were deployed, efforts to define the optical difference in the hole ice from the bulk of the array are underway. [11]

Schematic of a Dynamic Optical Module[5]

Feynman Diagrams of each event with their corresponding topography in the Array [25].

A) does not exhibit any features that could be passable by the DPA. B) looks like a double pulse waveform but did not pass. C) passed the DPA as can be seen by the red DPA flag in the image.

Time in between the pulses for 2 PeV Tau neutrino. There exists multiple initial conditions such that the resolution in the instrumentation will not be able to differentiate between the light arriving from the neutrino interaction vertex and the tau decay vertex.

Time in between pulses for a 100 Gev Neutrino. The time of arrival for the light from the interaction and decay vertex is separated on the order of picoseconds. Instrumentation would not be sensitive enough to detect a double pulse signature.
12 The simulated environment in GEANT4. Here, the ice material is shown in the blue cube, and the simulated detector in the center.

13 A 1.98 GeV Proton produces Cerenkov photons in a cone.

14 Here the transport of a 51.6 GeV tau is depicted.

15 The vertex of the tau decay in the GEANT4 visualiser

16 Contribution to signal from each particle weighted by average yield

17 Contribution of initial cascade versus tau

18 Total Signal from 10,000 averaged events


1 Introduction

Neutrino oscillations predict the transition probability of one flavour state to another. This is dependent on the propagation distance ($L$) as well as the energy of the neutrino ($E$). Specifically, it is proportional to $(L/E)$. When averaged by propagation over astronomical distances, neutrino oscillations transform the flavor ratio according to the PMNS mixing matrix. The flux of all neutrino flavours through the IceCube Neutrino Telescope should have a 1:1:1 ratio between the flavours, electron, muon and tau ($e, \mu, \tau$). However, with showers and tracks serving as the only two identifiers for the three flavors in this analysis, there is an inherent degeneracy in the determination of astrophysical flavor ratios. This paper will discuss Monte Carlo methods for high energy neutrino transport simulation and explore how a tau neutrino event would present itself in the IceCube array, specifically, how a 100 GeV event would present itself in a waveform of Cerenkov radiation detected, by an individual optical sensor in the array[8].

Tau neutrino events were generated in GENIE, then the transports of the resultant particles were simulated in GEANT4. Outputs of histograms depicting optical photons hitting the simulated detector (a simplified ”Dynamic Optical Module” or ”DOM”) overtime were generated and analyzed looking for relationships between the distribution and the location of the event in the detector. The created application within GEANT4 may be useful for a variety of transport problems involving high energy neutrinos, and may be modified for usage with a variety of neutrino event generators. The defined material within the code is set as ice, which was not a native material within GEANT4, but was possible to add using material definitions. This could be modified for individual usage to support transport simulation in a variety of neutrino detectors.

The application of Monte Carlo simulation to the creation of tau
neutrino signatures is of a particularly interesting nature given current outstanding questions in regards to neutrino physics and our understanding of the astrophysical flux. Discrepancies between theoretical predictions and measured yields lead to the interest in developing such tools. They also pose the potential existence of beyond standard model physics as an explanation to ongoing problems within the neutrino physics community. As such, neutrino properties, oscillations, and experimental techniques are discussed in some detail throughout this work. Speculation of the possible existence of sterile neutrinos or pseudo-dirac neutrinos can add some theoretically interesting discussion to our understanding of potential implications of inconsistent flavour ratio results.
Figure 1: Simulated Tau Neutrino event from the IceCube collaboration, of energy on the order of PeVs, depicting signature "double bang" topography. While this event has not been observed, high energy events such as this are often not fully contained. Expansion of the IceCube array in the next generation of the detector may help to identify tau neutrino events. Distinguishing tau neutrino events from highly energetic muons with "catastrophic losses" along their track is a difficult process and an event that is not fully contained may be missing vital information for event analysis. [3]
2 Background

2.1 Discovery of the Neutrino

Neutrinos are only coupled to the weak and gravitational fields. Chargeless, they must be detected through indirect methods, which led to the long gap between their theorized existence by Wolfgang Pauli in 1938, to account for conservation of energy and momentum in $\beta$-decay.

\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{feynman.png}
\caption{Feynmann diagram of $\beta$ – decay [24]}\end{figure}

His prediction of a neutral, nearly massless particle lead to Enrico Fermi dubbing it the ”neutrino”, Italian for ”little neutral one”. Clyde Cowan and Fred Reines eventually discovered a particle that fit the description of the proposed neutrino by studying the particles created by a nuclear power plant. By doing this they discovered the electron neutrino. In 1962 the muon neutron was found by BNL physicists Leon Lederman, Mel Schwartz, and Jack Steinberger. The experiment used
a beam of the Alternating Gradient Synchrotron’s (AGS) energetic protons to produce a shower of pi mesons which decay into muons and muon neutrinos. A 5,000 ton steel wall was opaque to the muons, but remained transparent to the neutrino flux which instead traveled past it and into a spark chamber where their signature could be captured[17].

Perhaps one of the most renown neutrino experiments of the last century is the experiment conducted in the Homestake mine in South Dakota. Scientists used chemical detectors to measure the electron neutrino flux from the sun in 1968. Results disagreed with theoretical predictions based off of the nuclear models of the sun, only a third of the predicted interactions were measured, and thus the puzzle of the missing neutrinos became known as the Solar Neutrino Problem. Neutrino oscillations were at the time an existing theory, but as the standard model predicted massless neutrinos, this property which relied on mixing between the neutrino’s flavour and mass eigentates is not predicted by the standard model. However, the discrepancy in the experimental measurements appeared to be the first data that demonstrated the oscillation of neutrino flavours during propagation from source to detector, as the experiment was not sensitive to muon and tau neutrinos[2].

The existence of the third flavor of neutrino, the tau neutrino, was first inferred in 1978 with the discovery of the Tau particle at SLAC, the Stanford Linear Accelerator Center. By discovering the third generation of charged leptons, they inferred that there should also be a third generation of neutrinos. In 2000 the scientists at the DONUT collaboration observed a tau neutrino using protons accelerated by the Tevatron to produce tau neutrinos via decay of charmed mesons[18][4].

### 2.2 Neutrino Properties

There are three types of neutrinos: electron neutrino, muon neutrino, and tau neutrino. According to the standard model there exist
12 fundamental particles. Each "flavor" of neutrino has a correspond-
ing charged particle from which it gets its name. The Standard Model
consists of three generations and each generation has two quarks a
neutrino and a charged particle. The particles in the standard model
are separated into two types: quarks and leptons. The quarks interact
via the strong nuclear force while the leptons interact via the elec-
tromagnetic or the weak nuclear force. Neutrinos are nearly massless
and have no electric charge. Therefore, unlike the other particles, they
only interact via the weak nuclear force and gravitational force. The
weak nuclear force only acts at short ranges, thus neutrinos can pass
through massive objects without interacting with them.

2.2.1 Neutrino Sources

From core-collapsed supernovae to our very own sun, cosmic neu-
trinos are created by the weak interactions in a variety of astrophysical
phenomena. Even primordial neutrinos from the big bang still exist as
part of the remnants from the early universe.

2.2.1.1 Supernova

One of the most famous examples of supernovas as a neutrino source
is SN 1987a. Several detectors noticed an increase in noise and neu-
trino detection rates before the observation of this event. In the later
stages of stellar evolution, neutrinos may become trapped in what is
known as the neutrino-sphere. The exact mechanism for how the shock
is reignedghted is unknown, but it is believed that a "neutrino driven
wind" may play a role in this process. At some point, the system stops
being opaque to the neutrinos and they are allowed to escape the sys-
tem. This is why the neutrinos were detected before the supernova
was visibly observed. When the neutrinos leave, they also take energy
away from the star. Understanding their role in the energy budget of a
collapsing star is vital in obtaining 3d models of this phenomena and as such the detection of supernova neutrinos is of great interest. Furthermore, they may provide an early warning signal and allow traditional telescopes to be pointed in the right direction before the explosion is visible[14].

2.2.1.2 The Sun

Neutrinos are also created in the nuclear reactions that power the core of stars like our sun. Neutrinos are predominantly formed in the proton-proton chain but are created in other nuclear reactions in stars over their lifetime. The P-P chain is as follows:

\[ p + p \rightarrow H^+ + e^+ + \nu \]

where the deuteron is the nucleus of deuterium. In the sun, 4 hydrogen are being fused into Helium by means of the proton-proton chain. Neutrinos, unlike photons typically studied in astronomy allow measurement of processes occurring in the interior of the sun[14].

2.2.1.3 The Big Bang

The most abundant source of neutrinos was 15 billion years ago, in the nascent stages of the universe. At the order of around \(10^{-6}\) seconds after the big bang, neutrinos and other leptons were abundant, enabling weak processes. Within one second after the big bang \((T = 10^{10}K)\), the universe became transparent to the neutrino allowing them to travel freely through our universe. Their interactions were no longer important and thus began the era of neutrino decoupling, being virtually unaffected by nuclear interactions. There are approximately 330 million of these primordial neutrinos per \(m^3\); These neutrinos have very low energy and go undetected to date[16].
2.2.2 Neutrino Oscillations

Neutrino oscillations can be expressed through a unitary transformation between the mass eigenstates and flavor eigenstates of neutrinos. Neutrinos are produced in flavor eigenstates, which are linear superpositions of their mass eigenstates. Neutrino mass eigenstates are denoted as $j (j = 1, 2, 3)$ with Latin indices and flavor eigenstates as $\nu_{\alpha} (\alpha = e, \mu, \tau)$ with Greek indices. Two sets of eigenstates can be transformed into one another through a unitary matrix known as the PMNS (Pontecorvo-Maki-Nakagawa-Sakata) matrix. The flavour eigenstates $|\nu_{\alpha}\rangle$ can be expressed in terms of the mass eigenstates $|\nu_{j}\rangle$ and the PMNS matrix as follows:

$$
\begin{bmatrix}
\nu_e \\
\nu_\mu \\
\nu_\tau
\end{bmatrix} =
\begin{bmatrix}
U_{e1} & U_{e2} & U_{e3} \\
1U_{\mu1} & U_{\mu2} & U_{\mu3} \\
U_{\tau1} & U_{\tau2} & U_{\tau3}
\end{bmatrix}
\begin{bmatrix}
\nu_1 \\
\nu_2 \\
\nu_3
\end{bmatrix}.
$$

(1)

Evolution of the neutrino wave function $\psi$ is described by the scattering matrix, also known as the $S$-matrix, dependent on $U$ (the PMNS matrix), where the individual elements are probability transition amplitudes of the initial and final states of the neutrino. The neutrino wave function at time $\psi(t)$ may be expressed then, in terms of its initial wave function and the Scattering matrix such that

$$
\psi(t) = S^{-1}\psi(0)S
$$

(2)

We may then calculate the transition probability between flavours $\alpha$ and $\beta$ as

$$
P(\nu_\alpha \rightarrow \nu_\beta) = |S_{\alpha\beta}|^2
$$

(3)

[19][20]

2.2.3 Neutrino interactions: Deep Inelastic Scattering

For neutrinos above $O(\text{TeV})$ energies, deep inelastic scattering (DIS) is the dominant process for neutrino-nucleon interaction. All
flavors of neutrinos can undergo DIS with ice nuclei through charged-current (CC) and neutral current (NC) interactions.

**Figure 3:** Feynman diagram for and electron neutrino undergoing a CC DIS off of a neutron. This diagram can be generalized to any flavour neutrino by replacing the lepton with consideration to lepton number conservation.
2.3 Cerenkov radiation

Cerenkov radiation is a phenomenon by which a particle passing through a medium where its velocity \( v = \beta c \) is greater than the local phase velocity of light \( c/n \) where \( n \) is the index of refraction of the material. Such a phenomenon can be thought as analogous to a sonic boom resulting from a particle traveling faster than the speed of sound. The shock wave in this comparative physical system is instead a cone of radiation, in the form of optical photons. A variety of experiments utilize Cerenkov detectors, for this reason, to reconstruct information about particle classification and energy.

![Figure 4: Cerenkov light emission and wave-front angles. In a dispersive medium, \( \theta_c + \eta \neq 90^0 \)][19]

Cerenkov photons are emitted coherently at and angle of

\[
\theta_c = \cos^{-1}\left(\frac{1}{n(\lambda)\beta}\right)
\] (4)

22
This phenomena is demonstrated in our results, where the emitted photons can be seen the formed light cone along the direction of the charged particle. The number of optical photons produced per unit wavelength per unit distance from a particle with charge $ze$ is given as:

$$\frac{dN}{dx d\lambda} = \frac{2\pi \alpha z^2}{\lambda^2} (1 - \frac{1}{\beta^2 n^2(\lambda)})[19]$$  

(5)
2.4 Instrumentation

Neutrino detectors come in a wide variety of designs, but in general, they tend to be scintillation detectors or Cerenkov detectors. Both types of detectors require some form of PMT for detection of optical photons. The basic method for the detection of a neutrino must be indirect as with neutral particles. Instead, their interactions may produce charged particles that may be detected. In general, the goal of any detector is to eventually convert the detected event into an electron which may be used to produce a signal. To understand the setup used in the simulation, the basic operation of PMTs is a useful discussion.

2.4.1 PMTs

A photomultiplier tube is useful for light detection of very weak signals. It is a photoemissive device in which the absorption of a photon results in the emission of an electron. Photomultipliers acquire light through a glass or quartz window that covers a photosensitive surface, called a photocathode. The photocathode then releases electrons that are multiplied by electrodes known as metal channel dynodes. At the end of the dynode chain is an anode or collection electrode. Therefore, this system effectively amplifies the electrons generated by a photocathode when exposed to a photon flux. Over a very large range, the current flowing from the anode to ground is directly proportional to the photoelectron flux generated by the photocathode.

The spectral response, sensitivity, and dark current of a photomultiplier tube and quantum efficiency given as

\[ \alpha = \frac{\text{Number(photoelectrons}_{\text{absorbed}})}{\text{Number(photoelectrons}_{\text{emitted}})} \]  

are determined by the composition of the photocathode. Even the best photocathodes are less than 30 percent quantum efficient, meaning
that 70 percent of the photons impacting on the photocathode do not produce a photoelectron and are therefore not detected.

Photoelectrons are ejected from the front face of the photocathode and angled toward the first dynode. Electrons emitted by the photocathode are accelerated toward the dynode chain. Focusing electrodes may be present to ensure that photoelectrons emitted near the edges of the photocathode will tend to land on the first dynode. Upon contact with the first dynode, a photoelectron will invoke a chain reaction such that the release of additional electron occurs and they are accelerated toward the next dynode, and so on until the end of the chain is reached.

The gain can be calculated by the following relation

\[ Gain = \alpha \times \eta \times \delta^N \]  

(7)

where \( \alpha \) is the quantum efficiency of the photocathode, \( \eta \) is the collection efficiency, \( \delta \) is the multiplicative factor of each dynode, and \( N \) is the number of dynodes. Noise in the photomultiplier comes from a baseline signal none as the "dark current", arising from thermal emissions electrons from the several sources including: the photocathode, stray high energy radiation, leakage current between dynodes, as well as electronic noise[21].
2.5 Neutrinos and the IceCube array

Neutrinos are neutrally charged leptons that come to us from all around the cosmos travelling at nearly light speed. Unlike what is predicted by the standard model of particle physics, neutrinos are not massless, and as such, they can oscillate between their three flavours. The probability of oscillation to a different flavour is dependent on energy. At the energies observed in IceCube (above 35 TeV), we expect a 1:1:1 ratio between the flavours [8]. Neutrinos interact weakly via the Z boson (neutral current events), and the W boson (charged current events). For the purposes of this paper, we will only be discussing Charged Current interactions. Neutrinos cannot be detected directly; they must be detected indirectly (by Cerenkov radiation or scintillation light, in most cases). Essentially, the neutrino interacts with a nucleus in the ice and charged particles are produced. When charged particles
Figure 6: The IceCube Array is made up of 86 strings, including 8 DeepCore strings, set around 125 meters apart from each other for a total of 5,160 Dynamic Optical Modules (DOMs). DeepCore is situated 2100 m below the surface of the icecap and is more densely packed, with around 480 optical modules making it sensitive to lower energy ranges (as low as 10 GeV)[7][6]. A layer of dust (not depicted) exists at around 1970 m and 2100 m. Because of thousands of years of compression, the Antarctic ice is very pure otherwise. However, due to refilling of the holes where strings were deployed, efforts to define the optical difference in the hole ice from the bulk of the array are underway. [11]

travel faster than the speed of light in a medium, Cerenkov radiation is produced in a cone of blue light. This can be thought of as light’s equivalent to a sonic boom. The IceCube array uses ice as a medium, as its index of refraction [23] is high enough to for Cerenkov light to
be produced. The cubic kilometer of Antarctic ice which hosts it also has the added benefit of being quite clear as it’s been compressed from thousands of years, removing impurities. Within the array (depicted in figure 2), there are 5,160 Dynamic Optical Modules, or ”DOMs”, which have photo multipliers that can detect the optical Cerenkov light and output the signal. The model we have created using GEANT4 has been adjusted to emulate the environment within the IceCube Neutrino Observatory.
2.5.1 DOMS

The 5,160 Dynamic Optical Modules housed in the cubic kilometer of ice that makes up the array are sensitive to the cerenkov light produced by neutrino interactions[7].

Figure 7: Schematic of a Dynamic Optical Module[5]
2.5.2 Event Topography

There are 3 main topographies seen in the IceCube array: Charged current electron neutrino events and all neutral current events produce cascades in the array. Muon neutrinos as well as atmospheric muons produce track like events in the ice. In order to cut out the atmospheric muons, the earth is used like a large filter such that upward going events can be assumed to be astrophysical in origin (muons wouldn’t be able to make it all the way through the earth so downward going events are cut to reduce background). The expected topography for a tau event would be a signature ”double bang event” (Figure 1), which is a distinct pattern of an initial cascade, a track from the tau, and then a second cascade when the tau decays. The initial interaction of any flavour with a nucleus will generally produce at least a small initial cascade along the track. Stochastic energy losses (often referred to as ”catastrophic energy losses”) along the track of the muon can also create a second cascade, leading to even more complex analysis.
required to separate a tau signal from the muon background. The main
difference to distinguish these confounding events is the relationship
between the energy of the incident tau neutrino and the mean free
path of the tau produced at the interaction vertex until its subsequent
decay. The distance the tau particle may travel will be proportional
to the separation between the cascades and therefore the length of the
track. More explicitly, we can calculate the average distance the tau
propagates before decaying in lab frame with the following relations:

\[ \langle L \rangle = \gamma c \tau \]  

(8)

Where \( \gamma \) is the Lorentz factor and \( \tau \) is the tau mean lifetime. This
expression can be writing in terms of energy as

\[ \langle L \rangle \approx \frac{E_\tau}{[EeV]} \times 49km \]  

(9)

The inelasticity of the tau interaction along with the probability of the
branching ratios must be taken into account for this calculation[22].
Following this, we arrive at the general approximation for the tau en-
ergy such that

\[ E_\tau \approx .75E_{\nu_\tau} \]  

(10)

The average tau decay length roughly scales as 5cm/TeV at en-
ergies on the order of a few hundred TeV [9]. At ultra high energies
(PeV range) this translates to a decay length of about 50 m per PeV.
As such, the maximum event energy that may be contained within the
array would be about 20 PeV [10].

It is important to note that energy reconstruction within IceCube
is a part of a very complex group of event reconstruction techniques
that often poses many challenges. The reconstruction of cascades is
considerably well defined, but more complex topographies involving
tracks can present their own unique challenges that may be less well
defined. The event must also be completely contained in the array, as
stated previously, whilst having a high enough energy for the tau track to show distinct separation between the cascades. A densely packed array may increase resolution for relatively lower energy events, while expansion of the size of the array in the next generation of IceCube may prove to be part of the solution to the common occurrence of unconfined high energy events.
2.5.3 The Tau Neutrino Search

Despite the expected 1:1:1 ratio between the flavours, signatures of tau like events have not passed analysis with enough confidence to be labeled as a discovery after undergoing the cuts outlined in multiple promising tau sensitive algorithms sorting through 3 years of unblinded data. Lowering the energy range of searches was considered to optimise the amount of events that could be analysed. This poses a different set of problems such that it adds a cascade background. An event such as seen in figure 1 would need to be very high energy. However, the tau track length, which would determine the distinction between the two cascades, is dependent on the energy of the incident particle. At lowered energy levels, tau events would resolve as a cascade. Unable to resolve the characteristic topography, a possible solution was to analyse the wave-forms outputted from the individual DOMs, to search for double pulse wave-forms, where the peaks would correspond to the two cascades. The Double Pulse Algorithm (DPA), uses the wave-form derivatives in order to isolate wave-forms that have two periods of increase and decrease in the signal. When the unblinded data was analysed by the DPA, no tau neutrinos were found in 3 years [9]. It was proposed that the DPA may be missing wave-forms with double pulses and a method by which to improve the DPA’s recognition of double pulse wave-forms, such that it would pass more potential candidates, was suggested (see figure. 9).
Figure 9: A) does not exhibit any features that could be passable by the DPA. B) looks like a double pulse waveform but did not pass. C) passed the DPA as can be seen by the red DPA flag in the image.

The addition of machine learning algorithms to improve upon this method led to new efforts related to this technique. Tandem interest in lowering the energy ranges even further to GeV ranges has been suggested, though with this change, the subsequent relations such as transferred lepton energy as well as general neutrino oscillation relationships would need to be re-parameterized to develop algorithms sensitive to $\nu_\tau$ with $E_{\nu_\tau} \sim \mathcal{O}GeV$. 
3 Methods and Materials

3.1 Preliminary Calculations

Initial investigative calculations were carried out to probe the geometric effect of the incident neutrino interaction vertex location in relation to the DOM. A simple python script was created to looping over initial x, y, and theta values; this yielded an approximate relationship between the geometry of the event and the time between the pulses in a potential signal. The ultra high energy calculation assumed the general approximation for the tau decay length described by equation (8) and the following sections, utilizing data from the PDG. The Python script created for the purpose of this calculation can be found in Appendix A.

The calculations were repeated in a similar method for a 100 GeV neutrino (see Appendix B) but instead the decay length was calculated for a tau particle with the average energy taken from the GENIE neutrino event generator, the mean time of flight provided by the Particle Data Group and the path length equations provided in the background.

3.2 The Suite of Programs

GENIE was used to generate tau neutrino events. Two hundred events were generated yielding 151 CC events within the data set. An in-house Pearl script (Appendix C) preformed $\nu_{\tau}$ CC event ”cherry-picking” and averaging of these events in order to get an average flux of charged particles, their associated average momentum vectors, and energies. A general-purpose physics list for GEANT4 provided the initial set of interactions to be expanded upon with the appropriate transport processes and decay modes discussed in the following sections (Available on the Git Hub repository detailed in Appendix D). Particle transport simulation was then carried out by GEANT4 iterat-
through the particle distribution one by one, using the properties of the GENIE distribution 10,000 times for each particle in Table 1, given their specified kinematic information (see Appendix G). Within the geometrical definitions we specified in GEANT4, a 1 cubic meter of ice had been placed in the environment along with a simulated spherical detector that would count hits from optical photons (generated with the native G4Cerenkov process) and output a histogram of luminosity over time [13]. An average of these histograms for each particle were imported into Python to generate line plots of each waveform. The yield of each particle to the total event was then taken into account to construct a net waveform. GENIE is not meant for simulating neutrino events at above 100 GeV which sets an upper limit for the energies that can be probed with the methods outlined in this body of work, though the created application for GEANT4 may be applicable to particle distributions from higher energy neutrino event generators, so long as the particles and their kinematic properties do not exceed current limitations set by defined cross sections in GEANT4’s transport processes and decay modes [1][13]. The comparison of the peaks was shown by overlaying the waveform contribution from the tau with the constructed net initial cascade waveform.
3.3 Monte Carlo Techniques

Monte Carlo codes are stochastic problem solvers for complicated systems where deterministic answers may not exist. The code samples probabilities in order to produce a probability distribution from which conclusions may be drawn about the various properties of an event. GEANT4 is a highly validated toolkit for creating applications to solve particle transport problems. Current efforts in the physics community to apply GEANT4 as a tool in high energy neutrino physics appear to be sparse, while interesting and useful results may be found with such an application. Native neutrino processes in GEANT4, have for the most part been developed for the transport of daughter particles from processes producing a resultant neutrino. Incident high energy neutrino events must therefore be superficially dealt with by means of a neutrino event generator. The particle distribution from such a generator (such as GENIE), can provide the initial conditions for multiple transport simulations to be reconstructed into a net neutrino event.

3.3.1 GENIE

The GENIE neutrino event generator implements a modern framework for Monte Carlo event generators. GENIE calculates the double-differential cross-section \( \frac{d^2\sigma}{dx dy} \) for high-energy DIS events. It uses \( \frac{d^2\sigma}{dx dy} \) as the probability density function and, on an event-by-event basis, it generates \( x, y \), where \( y \) is the inelasticity of the interaction. This in turn, determines how the neutrino energy is split between the leptonic and hadronic constituents of the event.
3.3.1.1 Cherry picking

The "Cherry-Picking" options in GENIE allow the user to specifically select certain types of events out of the generated distributions. A suggestion for event selection options under the `gevpick` cherry-picking utility for the various decay modes of the tau lepton has been made, as an addition to the topologies supported by GENIE[1].

3.3.2 GEANT4

Geant4 is a C++ toolkit from CERN for the simulation of the passage of particles through matter. Its areas of application include high energy, nuclear and accelerator physics. Geant4 works as a combination of two parts: the “physics list” and the geometrical modeling capabilities which are contained in multiple files which define the environment and the target.

3.3.2.1 The Physics List

The physics list is a file that contains all information about available processes that the simulation will take into account and/or include in the output. Geant4 has an relatively exhaustive list of processes and includes cross sections for a large amount of particles and ions, all of which are available for inclusion in the physics list information they provide. A generic physics list is also available for download.

Particle transport in Geant4 is the result of the combined actions of the Geant4 kernel’s Stepping Manager class and the actions of processes which it invokes. These are divided into the physics processes and the Transportation process which identifies the next volume boundary and also the geometrical volume within the volume boundary. To determine the expected length for the interaction to occur, all processes at each step that are applicable are polled. The simulation of particle transport is performed step-by-step, where a ”true step
length” for the next physics interaction is randomly sampled using either various step limitations or the mean free path which is calculated as

\[
\lambda(E) = \left(\sum_i [n_i \sigma(Z_i, E)]\right)^{-1}
\]

(11)

where

\[
n_i = \frac{N_p}{A}
\]

(12)

is the number of atoms per volume, \(N\) is Avagadro’s number, \(\rho\) is the density of the medium, and \(A\) is the mass of a mole. For a compound material, this is calculated for the number of atoms per volume of the \(i^{th}\) element.

The Tau decay process is defined under the G4 decay class where the mean free path for a decay in flight \(\lambda\) is calculated for each step by

\[
\lambda = \frac{\gamma \beta c \tau}{\gamma c} 
\]

(13)

where tau is the particle lifetime.

\[
\gamma = \frac{1}{\sqrt{1 - \beta^2}}
\]

(14)

and is calculated using the momentum at the beginning of the step. The leptonic tau decay defined in the \textit{G4TauLeptonicDecayChannel} simulated the tau decays according to V-A theory and is valid for the following modes:

\[
\tau^\pm \rightarrow e^\pm + \nu_\tau + \nu_e
\]

and

\[
\tau^\pm \rightarrow \mu^\pm + \nu_\tau + \nu_\mu
\]
In the physics list adapted for this body of work, the \textit{G4cerenkovprocess} was enabled to produce optical photons that would be registered as the signal. GEANT4 calculates the number of photons produced from a Poisson distribution with a mean of $<n> = \text{StepLength} \frac{dN}{dx}$.

3.3.2.2 Geometric Modeling-The World

Within this group of files the medium through which particles were being transported was defined by manually adding the optical properties of Ice (Appendix F). [12]

3.3.2.3 Geometric Modeling-The Detector

There are several pre-made geometries available on the Geant4 site that can be used to simulate a detector within the world. A simple sphere was placed in the volume with parameters defined to count 1 photon "hit" when it’s boundary was crossed by an optical photon. Photon tracks were killed once inside the detector to avoid double counting. The detector placed in the center is defined as a vacuum to avoid other interactions.

3.3.2.4 Trials and Data Collection

The neutrino event is fired from the particle gun which is placed head-on in the direction of the detector, meaning that the impact parameter for the trials was zero, defined in Appendix G.
4 Results

The results for this body of work demonstrate the up-to-date state of the chain of programs combined to enable neutrino transport simulation ultimately in a GEANT4 environment. Results of the general usage and functioning of the application will be discussed and followed by an experiment carried out utilising the created application.

4.1 Preliminary Calculations

Results from the preliminary calculations showed PeV energy ranges yielding a temporal separation in the signal from the photons produced at the interaction vertex and the decay vertex on the order of nanoseconds. The code provided in Appendix A can be altered for a variety of incident neutrino energies and orientations to show agreement with expected signal properties from 1 to 20 PeV (a possible double pulse signal that the detector may be sensitive to)
Figure 10: Time in between the pulses for 2 PeV Tau neutrino. There exists multiple initial conditions such that the resolution in the instrumentation will not be able to differentiate between the light arriving from the neutrino interaction vertex and the tau decay vertex.
Lowering the energy levels to $\mathcal{O}(GeV)$ showed a separation on the order of picoseconds, which experimental setups would not be sensitive to. This is in agreement with our simulation, where histogram binning was set at 0.25 ns.

![Figure 11: Time in between pulses for a 100 Gev Neutrino. The time of arrival for the light from the interaction and decay vertex is separated on the order of picoseconds. Instrumentation would not be sensitive enough to detect a double pulse signature.](image)
4.2 Charged Particle Flux from Neutrino Interaction

Using GENIE, the flux of charged particles from the neutrino interaction of a tau neutrino, with an energy of 100 GeV, was successfully acquired for 151 events and averaged. Table 1 includes the yield of each particle, its momentum vectors, and their energies[1].

<table>
<thead>
<tr>
<th>Particle</th>
<th>$P_x$</th>
<th>$P_y$</th>
<th>$P_z$</th>
<th>E [GeV]</th>
<th>Avg Yield/Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tau</td>
<td>0.0611 ± 1.7789</td>
<td>-3.873 ± 1.8716</td>
<td>55.2856 ± 27.685</td>
<td>55.4170 ± 27.6029</td>
<td>1 ± 0</td>
</tr>
<tr>
<td>Proton</td>
<td>-0.0038 ± 0.3496</td>
<td>0.0558 ± 0.3083</td>
<td>2.6619 ± 4.1188</td>
<td>3.2072 ± 3.9590</td>
<td>1.6358 ± 1.15867</td>
</tr>
<tr>
<td>Antiproton</td>
<td>-0.1833 ± 0.8121</td>
<td>0.2196 ± 0.5601</td>
<td>13.8963 ± 13.0768</td>
<td>13.9964 ± 13.0152</td>
<td>0.7285 ± 0.2598</td>
</tr>
<tr>
<td>Kaon+</td>
<td>0.038 ± 0.9107</td>
<td>0.1505 ± 0.7742</td>
<td>13.2971 ± 15.3758</td>
<td>13.4542 ± 15.2945</td>
<td>0.1722 ± 0.2945</td>
</tr>
<tr>
<td>Kaon-</td>
<td>0.1513 ± 0.4934</td>
<td>0.0497 ± 0.4704</td>
<td>6.7756 ± 6.6985</td>
<td>6.8571 ± 6.9584</td>
<td>0.1126 ± 0.3161</td>
</tr>
<tr>
<td>P+</td>
<td>-0.0009 ± 0.4266</td>
<td>0.0234 ± 0.4851</td>
<td>0.8723 ± 7.7021</td>
<td>0.9609 ± 7.6805</td>
<td>1.9338 ± 1.0337</td>
</tr>
<tr>
<td>Pi-</td>
<td>-0.0098 ± 0.3354</td>
<td>0.0156 ± 0.4327</td>
<td>5.0193 ± 7.0711</td>
<td>5.1027 ± 7.0507</td>
<td>1.3311 ± 0.9678</td>
</tr>
<tr>
<td>Lamda c+</td>
<td>0.456025 ± 1.4066</td>
<td>0.1074 ± 0.8707</td>
<td>26.1928 ± 14.6652</td>
<td>26.4011 ± 14.5698</td>
<td>0.0529 ± 0.2239</td>
</tr>
</tbody>
</table>

Table 1: The distribution of particle from the GENIE neutrino event generator and their corresponding kinematic information. 200 events were generated for a 100 Gev incident neutrino interacting in ice. 151 CC Tau Neutrino events were selected and averaged.
4.3 Monte Carlo Application Visualization

Figure 12 depicts the environment and the detector in the GEANT4 visualiser. Individual runs of both a proton (figure 13) and a tau (figure 14), using the information from GENIE, were performed to visually assess the validity of the simulation. The vertex of the tau decay is seen in figure 15.

![The simulated environment in GEANT4. Here, the ice material is shown in the blue cube, and the simulated detector in the center.](image)

**Figure 12:** The simulated environment in GEANT4. Here, the ice material is shown in the blue cube, and the simulated detector in the center.
Figure 13: A 1.98 GeV Proton produces Cerenkov photons in a cone.

Figure 14: Here the transport of a 51.6 GeV tau is depicted.
Figure 15: The vertex of the tau decay in the GEANT4 visualiser
4.4 Application Calculation of Tau Neutrino Signal

10,000 events of each individual particle were ran. Figures 16 through 18 depict the averaged signal luminosity from all of the charged particles detailed in the GENIE distribution.

Figure 16: Contribution to signal from each particle weighted by average yield

The yield of each particle was used to determine their contribution to the net signal in order to construct a waveform to represent experimental data where all of these particles would be undergoing transport processes simultaneously. It is not yet possible to enable the charmed lambda+ particle in GEANT4, but as it’s yield was very low (.0529), this is assumed to be of little consequence to the overall signal. To demonstrate the difference between the pulse from the initial cascade
and the tau decay, the signal from the net cascade and the tau are overlaid in figure 17.

**Figure 17:** Contribution of initial cascade versus tau
A double pulse was not produced at 100 GeV where the impact parameter was set to zero. Overlaying the signals as seen in figure. 17 shows the contributions of the cascade of optical photons produced at the neutrino interaction vertex, versus the contribution of photons associated to the tau decay vertex to the signal. The incident cascade resulted in a pulse with a maximum luminosity of 5112.244, while the second pulse had a maximum luminosity of 2935.648.
5 Discussion

The results of this experiment, demonstrating implementation of this technique, must be considered along with the multiple factors that may have contributed to resultant errors. Rounding errors when translating the GENIE output for implementation into GEANT4 are one relevant consideration. Currently, the GEANT4 collaboration is working on a GENIE-GEANT4 interface that will facilitate future work in this area. GEANT4 will also need the appropriate cross-sections for Lambda particles. Cross sections of higher energy tau events may be integrated into GEANT4 processes to consider events with tau energies above 1 TeV[13]. Suggested usage of this physics list may include transport simulation of other neutrino flavours and highly energetic muons for comparison to other established Monte Carlo methods intended for this application. Varying the impact parameter of the incident particle and altering its distance from the detector will considerably affect the event, with the latter coming at the cost of computational expense with a larger volume. A multiple detector scenario may aid in establishing the effect of the geometric location of the event on the waveform, when considering the total array of detectors. This is vital when considering the triggering parameters laid out by experimental collaborations and validation of the signal by inter-detector analysis.

Apart from the GEANT4 application created, being our main focus in this body of work, it is absolutely essential and relevant to discuss the multiple factors that may be influencing discrepancies in the experimental measurements that motivated this work. Disappearances and unexpected flavour ratios from in-situ measurements indeed motivate new simulation techniques. They further call for more sophisticated event isolating algorithms used in searches through large sets of data, and also motivate certain detector set-up modifications/expansions. The continued efforts of the neutrino community in measuring
the astrophysical flux of high energy neutrinos combined with the multiple neutrino oscillation experiments may even suggest potential new physics as supported by various theoretical models in beyond standard model constructions.
6 Conclusion

An application for high energy neutrino transport simulation in a Cerenkov detector via GEANT4 is a new tool for neutrino physicists and of some interest to the computational physics community. While the demonstrated experiment was applied to the high energy astrophysical tau neutrino problem, its use-cases and ability to be arbitrarily modified for different flavours and initial conditions or expanded upon for more accurate but increasingly computationally expensive applications are numerous. More sophisticated geometry may also be applied to the detector construction placed within the environment to refine the parameters involved in measurement and data collection. The programs described in this paper may be considered a baseline toy-model of an extremely complex problem in beyond standard model physics by which it may be possible to gain insight into potential experimental presentations of related physical events in laboratories. While various Monte Carlo codes exist for neutrino event generation, transport, and signal production, this particular setup aims to streamline the simulation of high energy neutrino transport problems within the highly validated environment of GEANT4, adding to the open discussion of future production of native neutrino transport simulation in Monte Carlo techniques. In regards to the experiment discussed to show the application of our developed technique, several observations were made which provide interesting topics for future discussions in the search for high energy astrophysical neutrinos, but do not necessarily provide enough information to yet be of use in tau neutrino search algorithms without further simulation experiments under different parameters and rigorous statistical analysis. From the given data, it is hard to make conclusions about expected waveforms in regards to average luminosity of each peak as these represent weighted sums. These indicate a large spread of potential waveforms, which would be dependent on the flux

53
for the specific event. Even so, for events that produce a tau, it seems that they tend to have a ”broader” peak, such that the slope of the descending edge doesn’t as drastically decrease as the other particles which contribute to the incident cascade. Qualitatively, this suggests potential agreement with other techniques implemented in higher energy analysis of tau neutrino waveforms[9]. Analysis of the derivative of the rising and falling edge of the hadronic cascade peak along with the rising edge of the tau cascade peak may provide insight into analogous properties of tau neutrino events at different energies generated by different computational techniques.
References


A  Python scripts for time in between pulses at Ultra-High Energies

The following python scripts in appendix A and B were created in Jupyter notebook and written in Python 2. Some parts that weren’t used to generate plots in this report are commented out, but are interesting to use for exploring the effect different event geometries.

```python
import numpy as np
from numpy import ma
from matplotlib import colors, ticker, cm
from matplotlib.mlab import bivariate_normal
import scipy as sp
import matplotlib.pyplot as plt
import math
k = 228849204.58  # this is c/n where c here is the speed of light in m/s
# and n = 1.31 in ice. Other values
# used for n in other places in icecube seem to be n=1.33.
c = 299792458.  # m/s
x_1 = 1.
y_1 = 1.
z_1 = 0.
theta = math.pi / 2.
phi = math.pi / 2.
Ev = 2.  # Ev is energy of original tau neutrino in PeV.
# Try different values from 1 to 20 for our purposes
delta_tlist = []
x_1list = []
y_1list = []
z_1list = []
for x_1 in range(0, 200):  # this represents the starting point where
# the tau neutrino interacts and the tau lepton is produced
    for y_1 in range(0, 200):
        for theta in range(0, 360):
            for phi in range(0, 360):
                for Ev in range(1, 21):  # this is the energy
                    # of the original tau neutrino.
```

58
l_tau = 50* .75 *Ev #this gives the approximate
#relationship between the energy and track length
#I set some of these values as constants
#depending on what constraints I want to apply
#some variables are assigned above.
   x_tautau = l_tau * np.sin(theta) * np.cos(phi)
   y_tautau = l_tau * np.sin(theta) * np.sin(phi)
   z_tautau = l_tau * np.cos(theta)
#print(x_tautau)
#print(y_tautau)
#print(z_tautau)
   x_2 = x_1 + x_tautau #represents the location of the tau
   #in regards to the DOM which is
   #at the origin in the non dash frame.
   y_2 = y_1 + y_tautau
   z_2 = z_1 + z_tautau
#print(x_2)
#print(y_2)
#print(x_2, y_2)
#print(x_2)
   t_1 = math.sqrt((x_1 ** 2) + (y_1 ** 2) + (z_1 ** 2)) / k
   #time that first pulse hits sensor
   t_2 = math.sqrt((x_2 ** 2) + (y_2 ** 2) + (z_2 ** 2)) / k
   t_tautau = l_tau / c #the time that the tau decays
   delta_t = t_2 + t_tautau - t_1 #time between when the first
   #and second pulse hits the sensors
   delta_tlist.append(delta_t)
x_1list.append(x_1)
y_1list.append(y_1)
z_1list.append(z_1)
#print(x_2, y_2, z_2)
#print(t_1 *10**9)
#print(t_2 *10**9)
#print(t_tautau *10**9)
   #print(delta_t *10**9) #prints delta_t in nanoseconds
   time_plot =plt.scatter(x_1list, y_1list, c=delta_tlist, edgecolor='face')
   cb = plt.colorbar(time_plot)
cb.set_label('delta t [s]
plt.xlabel('Initial distance away from detector in x direction')
plt.ylabel('Initial distance away from detector in x direction')
plt.show()
B  Python scripts for time in between pulses 
at 100 GeV

This changes out the energy calculation with a literal definition of the tau decay length calculated from the mean lifetime of the tau particle, the tau energy from the genie distribution, and the relations discussed in section 4.5.2

```python
import numpy as np
from numpy import ma
from matplotlib import colors, ticker, cm
from matplotlib.mlab import bivariate_normal
import scipy as sp
import matplotlib.pyplot as plt
import math

k = 228849204.58
C = 299792458. #m/s
x1 = 1.
y1 = 1.
z1 = 0.
theta = math.pi/2.
phi = math.pi/2.

delta_t_list = []
x_1list = []
y_1list = []
z_1list = []
for x_1 in range(0,200):
    for y_1 in range(0,200):
        #for theta in range(0,360):
            #for phi in range(0,360):

                l_t = 0.00234 #calculated distance traveled
                #before tau decays in mters
                x_tauprime = l_t * np.sin(theta) * np.cos(phi)
y_tauprime = l_t * np.sin(theta) * np.sin(phi)
z_tauprime = l_t * np.cos(theta)

        #print(x_tauprime)
        #print(y_tauprime)
```

61
```python
# print(z_tautau)
x_2 = x_1 + x_tautau
y_2 = y_1 + y_tautau
z_2 = z_1 + z_tautau
# print(x_2)
# print(y_2)
# print(x_2, y_2)
# print(z_2)
t_1 = math.sqrt((x_1 ** 2) + (y_1 ** 2) + (z_1 ** 2)) / k
t_2 = math.sqrt((x_2 ** 2) + (y_2 ** 2) + (z_2 ** 2)) / k
tau = t_tautau / c
delta_t = t_2 + tau - t_1
delta_tlist.append(delta_t)
x_1list.append(x_1)
y_1list.append(y_1)
z_1list.append(z_1)
# print(x_2, y_2, z_2)
# print(t_1 *10**9)
# print(t_2 *10**9)
# print(t_tautau *10**9)
# print(delta_t *10**9)
time_plot =plt.scatter(x_1list, y_1list, c=delta_tlist, edgecolor='face')
cb = plt.colorbar(time_plot)
cb.set_label('delta_t [s]')
plt.xlabel('Initial distance away from detector in x direction')
plt.ylabel('Initial distance away from detector in x direction')
plt.show()
```
C Pearl Script for selecting CC $\nu_\tau$ events from GENIE

use strict;
use warnings;
use Data::Dumper;

our $diagnostics = 0;

our %particles; #The big, important global variable

{ #The list of every type of particle, loaded as a hash %particles. Note that the inputs from GENIE will have to have "+" for "plus" and "-" for "minus". This is done in the main block
#These comprise only the possible first generation particles

foreach (qw/K0 Lambda0_bar K_L0 antiproton tauminus gamma Lambda0 piminus proton Kminus K0_bar antineutron pi0 Lambda_cplus pplus neutron Kplus/) {
    %{$particles{$_}} = (
        'px' => [],
        'py' => [],
        'pz' => [],
        'e' => [],
        'yield' => []
    );
}

sub updateHash { #NOTE: This function makes use of the global hashes listed above.
    #takes as an input, in order, the hash name, px, py, pz, e, yield

    my $particleName = shift;
    my $px = shift;
    my $py = shift;
    my $pz = shift;

    my $e = shift;

    my $yield = shift;

}
my $pz = shift;
my $e = shift;
my $yield = shift;

print "$particleName, $px, $py, $pz, $e, $yield\n" if $diagnostics == 1; #diagnostic tool

if ($yield == 0)
{
    push(@%{$_->{}}, $yield);
    #Where the point of this data is to generate a combined
    #average event, we want to figure out how likely it is
    #that a particle will occur
    #therefore, we want the combined yield, including zeros.
    #Though, it doesn’t make any sense for a non-existent
    #particle’s kinematics to be counted.
    #Hence, if yield is zero, we skip the other data
} else {
    push(@%{$_->{}}, $px);
    push(@%{$_->{}}, $py);
    push(@%{$_->{}}, $pz);
    push(@%{$_->{}}, $e);
    push(@%{$_->{}}, $yield);
}

sub generateReport { #NOTE: This function makes use of the global
    #hashes listed above, though without modifying them
    #writes to output file skimmer_report.txt
    #takes the total number of events and charge events as arguments

    my $numTotalEvents = shift;
    my $numCCEvents = shift;

    #print(Dumper \%particles);
    open(FINAL, ">skimmer_report.txt") or die "$! \n";
    print(FINAL "Greetings! \nThere were $numTotalEvents events
of which $numCCEvents were CC events.\n\n");

64
foreach (qw/K0 Lambda0_bar K_L0 antiproton tauminus gamma Lambda0 piminus proton Kminus K0_bar antineutron pi0 Lambda_cplus piplus neutron Kplus/){
    my $curParticle = $_;
    print(FINAL "\n*** Particle: $curParticle ***\n");
    foreach (qw/px py pz e yield/){
        my $curVariable = $_;
        print (FINAL "$curVariable average: ".
            average(@{${$particles{"$curParticle"}}{$curVariable}}) . " ,
            "\n");
    }
}
close(FINAL);
}

sub average{
    my @input = @_
    return "0" if @input == 0;
    my $sum = 0;
    foreach (@input){
        $sum += _;
    }
    my $avg = ($sum / scalar(@input));
    return ($avg);
}

sub stDeviation{
    my @input = @_
    return "0" if @input == 0;
    my $mean = average(@input);
    my $subTotal = 0;
    foreach (@input){
        $subTotal += (abs(_ - $mean)) ** 2;
    }
}
my $std = sqrt($subTotal / scalar(@input));
return $std;
}

sub analyzeEvent{

print "\n\n\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\nn\n
my @event = @_; # takes a complete event where each line is loaded into an array
# and counts, for each event, for each type of particle, the yield,
# the px, the py, pz, and the E
# NOTE: Doesn’t interact with any of the global hashes
# Takes a pass through the event for each possible particle
# It’d be quicker to go all in one pass, but the resulting data
# structures would be more complicated

foreach qw/K0 Lambda0_bar K_L0 antiproton tauminus gamma Lambda0 piminus proton Kminus K0_bar antineutron pi0 Lambda_cplus piplus neutron Kplus/ {

my $curParticle = @_; # takes a complete event where each line is loaded into an array
# and counts, for each event, for each type of particle, the yield,
# the px, the py, pz, and the E
# NOTE: Doesn’t interact with any of the global hashes
# Takes a pass through the event for each possible particle
# It’d be quicker to go all in one pass, but the resulting data
# structures would be more complicated

foreach (@event) {

my $curLine = @_; if ($curLine =~ m/$curParticle\s*\|\s*\|\s*\|\s*\|\s*\|\s*\|\s*\|

my $yield = 0;

foreach (@event) {

my $curLine = @_; if ($curLine =~ m/$curParticle\s*\|\s*\|\s*\|\s*\|\s*\|\s*\|\s*\|

$yield++;

66
push (@px , $1 );
push (@py , $2 );
push (@pz , $3 );
push (@e , $4 );
}
}
updateHash ($curParticle , average (@px) , average (@py) ,
average (@pz) , average (@e) , $yield );
}

{#Main part of the code
open (INPUT, "<nu_tau200.txt") or die "can’t open input $!";
open (OUTPUT, ">cc_only.txt") or die "can’t open output $!";

my $reading = 0; #determines whether or not to start
#recording lines. If set to 1, it means we should
#start writing into an array which may eventually
#be written to an output
my @event; #what will ultimately be written to the
#cc_only file
my $numTotalEvents; #Every event processed
my $numCCEvents; #Only CC events

MAIN: while(<INPUT>){
  my $curLine = $-
  if ( $curLine = m/GENIE GHEP Event/){ #this is the start
    #of an event, start compiling the event
    $reading = 1; #start reading lines into @event
    $numTotalEvents++;
  }

  if ($reading == 1){ #in the middle of an event
    $curLine = s/\+/plus/; #we can’t have pluses
    #or minuses in variable names, so we’re going to
# turn them alphanumeric. Relevant for hash
# names and in the analyzeEvent subroutine
$curLine =~ s/([a-zA-Z])−/$1minus/;
if ($curLine =~ m/(nu_\tau)\s*(\|)\s*1/){
  # Get rid of the event and start looking for the next one
  $reading = 0; # stop reading
  @event = ""; # discard the event
  next MAIN;
}
if ($curLine =~ m/Summary/){ # you’ve reached the end
  # of the event. Write it all to output
  foreach (@event){
    print (OUTPUT $_); # write the event
  }
analyzeEvent (@event);
  $numCCEvents++;
  @event = ""; # discard the event
  $reading = 0; # stop reading
  next MAIN;
}
push (@event, $curLine);
}
close (INPUT);
close (OUTPUT);
generateReport ($numTotalEvents, $numCCEvents);
D The Ultra Physics List

For all of the codes associated with the GEANT4 modeling discussed in this paper, interested parties may directly find everything needed to replicate our results in the Git Repository located at

https://github.com/WPIRadiationPhysics/geant4-icecube

// * The Geant4 software is copyright of the Copyright Holders of *
// * the Geant4 Collaboration. It is provided under the terms and *
// * conditions of the Geant4 Software License, included in the file *
// * LICENSE and available at http://cern.ch/geant4/license. These *
// * include a list of copyright holders.
// *

// *

// * Neither the authors of this software system, nor their employing *
// * institutes, nor the agencies providing financial support for this *
// * work make any representation or warranty, express or implied, *
// * regarding this software system or assume any liability for its *
// * use. Please see the license in the file LICENSE and URL above *
// * for the full disclaimer and the limitation of liability *
// *

// *

// * This code implementation is the result of the scientific and *
/* technical work of the GEANT4 collaboration. * 
* By using, copying, modifying or distributing the software (or * 
* any work based on the software) you agree to * 
* acknowledge its * 
* use in resulting scientific publications, and * 
* indicate your * 
* acceptance of all terms of the Geant4 Software license. * 
*/

*****************************************************************************

GEANT 4 – ULTRA experiment example

Code developed by:
B. Tome, M.C. Espirito–Santo, A. Trindade, P. Rodrigues

Ultra Physics List class; Standard and Low Energy EM processes are defined for 
the relevant particles. Optical processes are declared.

#include "G4ios.hh"
#include "iomanip.h"
#include "globals.hh"
#include "UltraPhysicsList.hh"

#include "G4ParticleDefinition.hh"
#include "G4ParticleTypes.hh"
#include "G4ParticleWithCuts.hh"
#include "G4ParticleTable.hh"
#include "G4Material.hh"
#include "G4MaterialTable.hh"
#include "G4ProcessManager.hh"
#include "G4ProcessVector.hh"

UltraPhysicsList::UltraPhysicsList() : G4VUserPhysicsList()
{
}

UltraPhysicsList::~UltraPhysicsList()
{
}

void UltraPhysicsList::ConstructParticle()
{
    // In this method, static member functions should be called
    // for all particles which you want to use.
    // This ensures that objects of these particle types will be
    // created in the program.

    ConstructBosons();
    ConstructLeptons();
    ConstructMesons();
ConstructBaryons();

}

void UltraPhysicsList::ConstructBosons()
{
    // pseudo-particles
    G4Geantino::GeantinoDefinition();
    G4ChargedGeantino::ChargedGeantinoDefinition();

    // gamma
    G4Gamma::GammaDefinition();

    // optical photon
    G4OpticalPhoton::OpticalPhotonDefinition();
}

void UltraPhysicsList::ConstructLeptons()
{
    // leptons
    G4Electron::ElectronDefinition();
    G4Positron::PositronDefinition();
    G4NeutrinoE::NeutrinoEDefinition();
    G4AntiNeutrinoE::AntiNeutrinoEDefinition();
    G4MuonPlus::MuonPlusDefinition();
    G4MuonMinus::MuonMinusDefinition();
    G4NeutrinoMu::NeutrinoMuDefinition();
    G4AntiNeutrinoMu::AntiNeutrinoMuDefinition();
    G4TauPlus::TauPlusDefinition();  //
    G4TauMinus::TauMinusDefinition();  //
    G4NeutrinoTau::NeutrinoTauDefinition();

    72
G4AntiNeutrinoTau::AntiNeutrinoTauDefinition();
}

void UltraPhysicsList::ConstructMesons()
{
    // mesons
    G4PionPlus::PionPlusDefinition();
    G4PionMinus::PionMinusDefinition();
    G4PionZero::PionZeroDefinition();
    G4KaonPlus::KaonPlusDefinition();  //
    G4KaonMinus::KaonMinusDefinition();  //
    G4KaonZero::KaonZeroDefinition();  //
}

void UltraPhysicsList::ConstructBaryons()
{
    // barions
    G4Proton::ProtonDefinition();
    G4AntiProton::AntiProtonDefinition();
    G4Neutron::NeutronDefinition();
    G4AntiNeutron::AntiNeutronDefinition();
}

void UltraPhysicsList::ConstructProcess()
{
    AddTransportation();
    ConstructGeneral();
    ConstructEM();
    ConstructOp();

    73
```cpp
#include "G4Decay.hh"

void UltraPhysicsList::ConstructGeneral()
{
    G4Decay* theDecayProcess = new G4Decay();
    theParticleIterator->reset();
    while( (*theParticleIterator)() ){
        G4ParticleDefinition* particle = theParticleIterator->value();
        G4ProcessManager* pmanager = particle->GetProcessManager();
        if (theDecayProcess->IsApplicable(*particle)) {
            pmanager->AddDiscreteProcess(theDecayProcess);
        }
    }
}

#include "G4ComptonScattering.hh"
#include "G4GammaConversion.hh"
#include "G4PhotoElectricEffect.hh"
#include "G4eMultipleScattering.hh"
#include "G4MuMultipleScattering.hh"
#include "G4hMultipleScattering.hh"
#include "G4eIonisation.hh"
#include "G4Bremsstrahlung.hh"
#include "G4eplusAnnihilation.hh"
#include "G4MuIonisation.hh"
```
```cpp
#include "G4MuBremsstrahlung.hh"
#include "G4MuPairProduction.hh"
#include "G4hIonisation.hh"

// ....oooOO0OOoo  ....... oooOO0OO0oo  ....... oooOO0OO0oo
    ....... oooOO0OO0oo ....

void UltraPhysicsList::ConstructEM()
{
    theParticleIterator->reset();
    while( (*theParticleIterator)() ){
        G4ParticleDefinition* particle = theParticleIterator->value();
        G4ProcessManager* pmanager = particle->GetProcessManager();
        G4String particleName = particle->GetParticleName();

        if (particleName == "gamma") {
            // gamma
            // Construct processes for gamma
            pmanager->AddDiscreteProcess(new G4GammaConversion());
            pmanager->AddDiscreteProcess(new G4ComptonScattering());
            pmanager->AddDiscreteProcess(new G4PhotoElectricEffect());
        }
        else if (particleName == "e-") {
            // electron
            // Construct processes for electron
            pmanager->AddProcess(new G4eMultipleScattering(), -1,1,1);
            pmanager->AddProcess(new G4eIonisation(), -1,2,2);
            pmanager->AddProcess(new G4eBremsstrahlung(), -1,-1,3);
        }
        else if (particleName == "e+") {
            // positron
            // Construct processes for positron
            pmanager->AddProcess(new G4eMultipleScattering(), -1,1,1);
        }
    }
}
```
pmanager->AddProcess(new G4eIonisation(), -1, 2, 2);
pmanager->AddProcess(new G4eBremsstrahlung(), -1, -1, 3);
pmanager->AddProcess(new G4eplusAnnihilation(), 0, -1, 4);

} else if (particleName == "mu+" ||
           particleName == "mu-") {
    // muon
    // Construct processes for muon
    pmanager->AddProcess(new G4MuMultipleScattering(), -1, 1, 1);
pmanager->AddProcess(new G4MuIonisation(), -1, 2, 2);
pmanager->AddProcess(new G4MuBremsstrahlung(), -1, -1, 3);
pmanager->AddProcess(new G4MuPairProduction(), -1, -1, 4);

} else {
    if ((particle->GetPDGCharge() != 0.0) &&
        (particle->GetParticleName() != "chargedgeantino")
    ) {
        // all others charged particles except geantino (including tau+-)
        pmanager->AddProcess(new G4hMultipleScattering(), -1, 1, 1);
pmanager->AddProcess(new G4hIonisation(), -1, 2, 2);
    }
}

// ....oooOO00O00o ....... ooooOO00O00o ....... ooooOO00O00o
// ....... ooooOO00O00o .......

#include "G4Cerenkov.hh"
#include "G4Scintillation.hh"
#include "G4OpAbsorption.hh"
#include "G4OpRayleigh.hh"
#include "G4OpBoundaryProcess.hh"
void UltraPhysicsList::ConstructOp()
{
    // this Cerenkov Process
    G4Cerenkov* theCerenkovProcess = new G4Cerenkov("Cerenkov");
    // this absorption process inside optical media
    G4OpAbsorption* theAbsorptionProcess = new G4OpAbsorption();
    // Rayleigh scattering for optical photons (aerogel radiators)
    G4OpRayleigh* theRayleighScatteringProcess = new G4OpRayleigh();
    // Boundary process definition Class

    // Chose level 0 (no verbose)
    theCerenkovProcess->SetVerboseLevel(0);
    theAbsorptionProcess->SetVerboseLevel(0);
    theRayleighScatteringProcess->SetVerboseLevel(0);
    theBoundaryProcess->SetVerboseLevel(0);

    // Chose MaxNumPhotons that can be generated. Lets ignore this for now
    G4int MaxNumPhotons = 300;
    theCerenkovProcess->setMaxNumPhotonsPerStep(MaxNumPhotons);
    theCerenkovProcess->setMaxBetaChangePerStep(10.0);
    theCerenkovProcess->setTrackSecondariesFirst(true);

    theParticleIterator->reset();
    while ( (*theParticleIterator)() ){
        G4ParticleDefinition* particle = theParticleIterator->value();
        G4ProcessManager* pmanager = particle->GetProcessManager();
        G4String particleName = particle->GetParticleName();
if (theCerenkovProcess->IsApplicable(*particle)) {
    pmanager->AddProcess(theCerenkovProcess);
    pmanager->SetProcessOrdering(theCerenkovProcess,
                                idxPostStep);
}

if (particleName == "opticalphoton") {
    G4cout << ">>>>>>>>>>>> AddDiscreteProcess to OpticalPhoton " << G4endl;
    pmanager->AddDiscreteProcess(theAbsorptionProcess);
    pmanager->AddDiscreteProcess(theRayleighScatteringProcess);
    pmanager->AddDiscreteProcess(theBoundaryProcess);
}
}

void UltraPhysicsList::SetCuts()
{
    if (verboseLevel >1){
        G4cout << " UltraPhysicsList::SetCuts:"
    }
    // "G4VUserPhysicsList::SetCutsWithDefault" method sets
    // the default cut value for all particle types
    SetCutsWithDefault();
}

//....oooOO0OOooo ........ oooO0000oo0 ........ oooO000000000
//.... oooO0000000 ....... oooO0000000o00 ....... oooO000000000 ....
E The Physics List

```cpp
#include "PhysicsList.hh"
#include "G4SystemOfUnits.hh"

PhysicsList::PhysicsList() : G4VModularPhysicsList() {
    // SetDefaultCutValue(1*nanometer);
    // SetVerboseLevel(1);
}

PhysicsList::~PhysicsList() {}
```

F The Detector Construction

```cpp
G4double ice_pressure = 101325*pascal; // 1 atm
G4double ice_temperature = 273.15*kelvin; // 0 deg C
//G4double ice_molarmass = 18.0153*g/mole; // from WolframAlpha
G4Material* Ice = new G4Material("Ice", ice_density, 2,
    kStateUndefined, ice_temperature, ice_pressure);
G4Element* elH = new G4Element("Hydrogen", "H", 1, 0.9169*g/mole);
G4Element* elO = new G4Element("Oxygen", "O", 8, 16.00*g/mole);
Ice->AddElement(elH, 2);
Ice->AddElement(elO, 1);

const G4int NUMENTRIES = 11;
G4double ppckov[NUMENTRIES] = {1.771*eV, 1.851*eV, 1.937*eV, 2.033*eV, 2.138*eV, 2.254*eV, 2.384*eV, 2.530*eV, 2.695*eV, 2.883*eV, 3.099*eV};
G4double rindex[NUMENTRIES] = {1.306, 1.307, 1.308, 1.309, 1.310, 1.311, 1.312, 1.313, 1.315, 1.317, 1.319};
G4double absorption[NUMENTRIES] = {192.0*cm, 282.1*cm, 417.4*cm, 704.5*cm, 1142.4*cm, 1407.3*cm, 1830.9*cm, ...}
```
2190.6*cm, 2392.5*cm, 1645.1*cm, 1174.5*cm }
G4MaterialPropertiesTable* MPT = new
G4MaterialPropertiesTable();
MPT->AddProperty("RINDEX", ppckov, rindex, NUMENTRIES);
MPT->AddProperty("ABSLENGTH", ppckov, absorption, NUMENTRIES);
Ice->SetMaterialPropertiesTable(MPT);

// Materials defined using NIST Manager
G4NistManager* nistManager = G4NistManager::Instance();
nistManager->FindOrBuildMaterial("G4_WATER");

// Geant4 conventional definition of a vacuum
G4double vacuum_dens = universe_mean_dens; // from PhysicalConstants.h
G4double vacuum_pres = 1.0e-19*pascal;
G4double vacuum_temp = 0.1*kelvin;
G4double vacuum_molarmass = 1.01*g/mole;
new G4Material("Vacuum", 1., vacuum_molarmass,
               vacuum_dens,
               kStateGas, vacuum_temp,
               vacuum_pres);

// Print materials
G4cout << *(G4Material::GetMaterialTable()) << G4endl;
}

////// Geometry parameters
G4VPhysicalVolume* DetectorConstruction::DefineVolumes() {

    // Declarations
    G4double IceCubeLength = 7*m; // side length of bulk ice
    G4double IceCubeHalfLength = 3.5*IceCubeLength; // half side length of bulk ice
G4double worldBoxHalfLength = 2*IceCubeHalfLength; // half side length of world
G4double DomRadius = 10*2.54*cm/2;
// Get materials
G4Material* defaultMaterial = G4Material::GetMaterial("Vacuum");
G4Material* iceMaterial = G4Material::GetMaterial("Ice");
G4Material* DOMMaterial = G4Material::GetMaterial("Ice");

// Throw exception to ensure material usability
if (!defaultMaterial) {
    G4ExceptionDescription msg;
    msg << "Cannot retrieve materials already defined.");
    G4Exception("DetectorConstruction::DefineVolumes()", "MyCode0001", FatalException, msg);
}

// World box
// Box solid definition
G4VSolid* worldS = new G4Box(  
    "worldBox", // its name  
    worldBoxHalfLength, // parameters  
    worldBoxHalfLength,  
    worldBoxHalfLength  
);

// World volume
G4LogicalVolume* worldLV = new G4LogicalVolume(  
    worldS, // its solid  
    defaultMaterial, // its material  
    "world" // its name  
);

// World physical volume of solid placed in LV
G4VPhysicalVolume* worldPV = new G4PVPlacement(  

G4ThreeVector(), // no translation
worldLV, // its logical volume
"worldLV", // its name
0, // its mother volume
false, // no boolean operation
0, // copy number
fCheckOverlaps // checking overlaps
);

// World visualization attributes
G4VisAttributes* defaultVisAtt = new G4VisAttributes(
    G4Colour(1.0, 1.0, 1.0));
defaultVisAtt->SetVisibility(true);
worldLV->SetVisAttributes(defaultVisAtt);

// IceCube box
// Box solid definition
G4VSolid* IceCubeS = new G4Box(
    "IceCubeBox", // its name
    IceCubeHalfLength, // parameters
    IceCubeHalfLength,
    IceCubeHalfLength
);

// IceCube volume
G4LogicalVolume* IceCubeLV = new G4LogicalVolume(
    IceCubeS, // its solid
    iceMaterial, // its material
    "IceCubeBox" // its name
);

// IceCube Physical volume of solid placed in LV
G4VPhysicalVolume* IceCubePV = new G4PVPlacement(0, // no rotation
    G4ThreeVector(), // no translation (from center
of mother volume)
IceCubeLV, // its logical volume
"IceCubeLV", // its name
worldLV, // its mother volume
false, // no boolean operation
0, // copy number
fCheckOverlaps // checking overlaps
);

// DOM Construction
G4VSolid* IceCubeDOMS
= new G4Orb(
    "DOM1",
    DomRadius);

// DOM Volume
G4LogicalVolume* IceCubeDOMLV
=new G4LogicalVolume( 
    IceCubeDOMS,
    DOMMaterial,
    "Dom1"
);

// DOM Physical volume of solid placed in LV
G4VPhysicalVolume* IceCubeDOMPV
= new G4PVPlacement(
    0,
    G4ThreeVector(),
    IceCubeDOMLV,
    "IceCubeDOMLV",
    IceCubeLV,
    false,
    0,
    fCheckOverlaps
);
// IceCube Visualization attributes
G4VisAttributes* iceVisAtt = new G4VisAttributes(G4Colour(0, 0, 1.0));
iceVisAtt->SetVisibility(true);
IceCubeLV->SetVisAttributes(iceVisAtt);
G4VisAttributes* DOMVisAtt = new G4VisAttributes(G4Colour(0, 1.0, 0));
DOMVisAtt->SetVisibility(true);
IceCubeDOMLV->SetVisAttributes(DOMVisAtt);

// Always return the physical World
return worldPV;
Primary Generator Action

```cpp
#include "PrimaryGeneratorAction.hh"

#include "G4RunManager.hh"
#include "G4LogicalVolumeStore.hh"
#include "G4LogicalVolume.hh"
#include "G4Box.hh"
#include "G4Tubs.hh"
#include "G4Event.hh"
#include "G4ParticleGun.hh"
#include "G4ParticleTable.hh"
#include "G4ParticleDefinition.hh"
#include "G4SystemOfUnits.hh"

#include "Randomize.hh"
#include <math.h>

PrimaryGeneratorAction::PrimaryGeneratorAction()
    : G4VUserPrimaryGeneratorAction(), 
fParticleGun(0), 
fIceBox(0) {

    G4int nofParticles = 1;
    fParticleGun = new G4ParticleGun(nofParticles);
}

PrimaryGeneratorAction::~PrimaryGeneratorAction() { delete 
    fParticleGun; }

void PrimaryGeneratorAction::GeneratePrimaries(G4Event* 
anEvent) {

    // Find geometry–defined length to orient gun, spit error 
    and center if not found
    if (!fIceBox) {
        G4LogicalVolume* IceBoxLV
```
= G4LogicalVolumeStore::GetInstance()->GetVolume("IceCubeBox");
if (IceBoxLV) fIceBox = dynamic_cast<G4Box*>(IceBoxLV->GetSolid());
}
// Place gun side of universe
fParticleGun->SetParticlePosition(G4ThreeVector());
if (!fIceBox) {
// G4double IceBoxHalfLength = fIceBox->GetXHalfLength();
// // Replace gun at 2 * side length from origin
// //
// }
// else {
G4ExceptionDescription msg;
msg << "IceBox volume of box shape not found.\n";
msg << "Perhaps you have changed geometry.\n";
msg << "The gun will be place at the center."
G4Exception("B1PrimaryGeneratorAction::GeneratePrimaries()
", 
"MyCode0002", JustWarning, msg);
}
// Define neutrino interaction product flux statistics (from GENIE)
G4double averagePxProton = -0.003811, sigmaPxProton = 0.349640,
averagePyProton = 0.055780, sigmaPyProton = 0.308303,
averagePzProton = 2.661914, sigmaPzProton = 4.118808,
averageEProton = 3.207225, sigmaEProton = 3.959008,
averagePxAntiProton = -0.183272,
sigmaPxAntiProton = 0.812069,
averagePyAntiProton = 0.219636,
sigmaPyAntiProton = 0.560076,
averagePzAntiProton = 13.869272,
sigmaPzAntiProton = 13.076842,
averageEAntiProton = 13.996364,
sigmaEAntiProton = 13.015191,
averagePxPiplus = -0.000874, sigmaPxPiplus = 
0.426639,
averagePyPiplus = 0.023405, sigmaPyPiplus = 
0.485114,
averagePzPiplus = 6.872254, sigmaPzPiplus = 
7.702132,
averageEPiplus = 6.960894, sigmaEPiplus = 
7.680546,
averagePxPiminus = -0.009821, sigmaPxPiminus = 
0.335449,
averagePyPiminus = 0.015561, sigmaPyPiminus = 
0.432684,
averagePzPiminus = 5.019251, sigmaPzPiminus = 
7.050697,
averageEPiminus = 5.102711, sigmaEPiminus = 
7.050697,
averagePxTau = 0.061113, sigmaPxTau = 1.778832,
averagePyTau = -0.387325, sigmaPyTau = 1.871563,
averagePzTau = 55.285562, sigmaPzTau = 
27.685677,
averageETau = 55.417026, sigmaETau = 27.602968,
averagePxLambdaCharmedPlus = 0.456625,
sigmaPxLambdaCharmedPlus = 1.406648,
averagePyLambdaCharmedPlus = -0.107375,
sigmaPyLambdaCharmedPlus = 0.870668,
averagePzLambdaCharmedPlus = 26.19275,
sigmaPzLambdaCharmedPlus = 14.665224,
averageELambdaCharmedPlus = 26.401125,
sigmaELambdaCharmedPlus = 14.569784,
averagePxKaonPlus = 0.0638, sigmaPxKaonPlus = 
0.910747,
averagePyKaonPlus = 0.1505, sigmaPyKaonPlus = 
0.774223,
averagePzKaonPlus = 13.2971, sigmaPzKaonPlus = 
15.375774,
averageEKaonPlus = 13.45424, sigmaEKaonPlus = 
15.294512,
averagePxKaonMinus = 0.151294, sigmaPxKaonMinus = 0.493436,
averagePyKaonMinus = 0.049706, sigmaPyKaonMinus = 0.470412,
averagePzKaonMinus = 6.775588, sigmaPzKaonMinus = 6.985317,
averageEKaonMinus = 6.857058, sigmaEKaonMinus = 6.9584279;

// Construct firing position and direction
fParticleGun->SetParticlePosition(G4ThreeVector(0, 0, -0.5*m));
// G4double beamPx = G4RandGauss::shoot(averagePxTau, sigmaPxTau/pow(2, 0.5)),
// beamPy = G4RandGauss::shoot(averagePyTau, sigmaPyTau/pow(2, 0.5)),
// beamPz = G4RandGauss::shoot(averagePzTau, sigmaPzTau/pow(2, 0.5));
// fParticleGun->SetParticleMomentumDirection(G4ThreeVector(beamPx, beamPy, beamPz));
// fParticleGun->SetParticleMomentumDirection(G4ThreeVector(0, 0, 1));
// fParticleGun->SetParticleMomentumDirection(G4ThreeVector(averagePxTau, averagePyTau, averagePzTau));
fParticleGun->SetParticleMomentumDirection(G4ThreeVector(averagePxProton, averagePyProton, averagePzProton));

// Fire
fParticleGun->GeneratePrimaryVertex(anEvent);
H Run Action

#include "RunAction.hh"
#include "PrimaryGeneratorAction.hh"
#include "Analysis.hh"
#include "DetectorConstruction.hh"
#include "G4Run.hh"
#include "G4RunManager.hh"
#include "G4UnitsTable.hh"
#include "G4SystemOfUnits.hh"
#include "G4LogicalVolume.hh"
#include "G4LogicalVolumeStore.hh"
#include "G4PhysicalVolumeStore.hh"
#include "G4SolidStore.hh"
#include "G4GeometryManager.hh"
#include "G4EmCalculator.hh"

#include <iostream>
#include <fstream>
#include <string>
#include <stdio.h>
#include <sys/types.h>

RunAction::RunAction() : G4UserRunAction() {}

RunAction::~RunAction() { /* delete G4AnalysisManager::
 Instance(); */ }

void RunAction::BeginOfRunAction(const G4Run* run) {

    // Acquire analysis instance
    Analysis* simulationAnalysis = Analysis::GetAnalysis();

    // Set number of events for processing
    G4int numEvents = run->GetNumberOfEventToBeProcessed();
    simulationAnalysis->SetNumEvents(numEvents);
}
void RunAction::EndOfRunAction(const G4Run* /*run*/) {

    // Acquire analysis instance
    Analysis* simulationAnalysis = Analysis::GetAnalysis();
    G4int i, numEvents = simulationAnalysis->GetNumEvents();

    // Declare data filename
    std::ostringstream fileNameStream;
    G4String fileName;
    fileNameStream << "datafile.txt";
    fileName = fileNameStream.str();

    // Export histogram values into data file
    ofstream theFile;
    theFile.open(fileName);
    theFile << "// Time [0.25 x ns], Average, StDev (" << numEvents << " events processed)" << G4endl;
    for (i=0; i<100; i++) {
        // G4double binTally = (float)(simulationAnalysis->callOpticalHistogram(i));
        // G4double binSquaredTally = (float)(simulationAnalysis->callSquaredOpticalHistogram(i));
        // G4double tallyAverage = binTally/((float)(numEvents));
        // G4double tallyStDev = pow(binSquaredTally - (pow(binTally, 2)/((float)(numEvents))), 0.5);
        G4double tallyAverage = simulationAnalysis->callAverageOpticalHistogram(i);
        G4double tallyStDev = pow(tallyVar, 0.5);
        theFile << tallyAverage << " " << tallyStDev
    }
}
<< " " << G4endl;
    
    theFile.close();
  
}
I Stepping Action

```c++
#include "SteppingAction.hh"
#include "EventAction.hh"
#include "DetectorConstruction.hh"
#include "Analysis.hh"
#include "G4SteppingManager.hh"

#include "G4Step.hh"
#include "G4Run.hh"
#include "G4Event.hh"
#include "G4RunManager.hh"
#include "G4SystemOfUnits.hh"
#include "G4UnitsTable.hh"

#include "G4LogicalVolume.hh"
#include "G4LogicalVolumeStore.hh"
#include "G4Tubs.hh"

#include <iostream>
#include <fstream>
#include <string>
#include <stdio.h>
#include <sys/types.h>
#include <math.h>
#include <stdio.h>

// Initialize Step Procedure
SteppingAction::SteppingAction( const DetectorConstruction* detectorConstruction,
                                EventAction* eventAction )
    : G4UserSteppingAction( ),
      fDetConstruction( detectorConstruction ),
      fEventAction( eventAction )
{ }

SteppingAction::~SteppingAction() {}
// Step Procedure (for every step...)
void SteppingAction::UserSteppingAction(const G4Step* step)
{

// Get step info of particle and event
G4String particlename = step->GetTrack()->GetDefinition()->GetParticleName();
G4int stepnum = step->GetTrack()->GetCurrentStepNumber();
G4int parentnum = step->GetTrack()->GetParentID();
G4int tracknum = step->GetTrack()->GetTrackID();
G4int eventnum = G4RunManager::GetRunManager()->GetCurrentEvent()->GetEventID();

//// Declare data filename – CAREFUL, THIS CREATES IMPOSSIBLY LARGE FILES
// std::ostringstream fileNameStream;
// G4String fileName;
// fileNameStream << "processfile.txt";
// fileName = fileNameStream.str();
//
// ///// Append process into process file
// std::ofstream theFile;
// theFile.open(fileName, std::ios_base::app);
// theFile << "Event " << eventnum << ", step " << stepnum << " of track: " << tracknum << " from parent " << parentnum << " - " << particlename << " undergoing " << processname << " process." << G4endl;
// theFile.close();

// Get name of volume at step location
G4String volumeName = step->GetTrack()->GetVolume()->GetLogicalVolume()->GetName();

// if ( particlename == "opticalphoton" && volumeName == "Doml" ) {
//
93
G4double waveformTime = 6*ns;
G4double waveformTimeOffset = 1*ns;
G4cout << "Volume is: " << volumeName << G4endl;
// changed from 10 to 15 ns on october 11 and from 5 to 0
// for offset to test why histogram is giving zeros for all
data
// If optical photon at end of its track (note: I commented
the latter condition; why?)
// Not checking "if dead" below because we stopped any
entering the dom on line 49.
if ( particlename == "opticalphoton" /* && step->GetTrack
()->GetTrackStatus() != fAlive */ ) {

    // Get name of volume at track origin (vertex)
    G4String volumeNameVertex = step->GetTrack()->
    GetLogicalVolumeAtVertex()->GetName();

    // If began in ice and ended in DOM
    if ( volumeNameVertex == "IceCubeBox" && volumeName == "Dom1"
    ) {
        step->GetTrack()->SetTrackStatus(fStopButAlive);
        G4cout << "Volume is: " << volumeName << ", vertex
        volume is " << volumeNameVertex << G4endl;

        // Get time of incidence (since beginning of event )
        G4double globalTime = step->GetTrack()->
        GetGlobalTime();

        // Round down to nearest histogram time bin ( 
        ignore delayed stats)
        if ( globalTime > waveformTimeOffset &&
        globalTime < (waveformTimeOffset + waveformTime) ) {
            G4int binTime = ((globalTime-waveformTimeOffset)

94
// Acquire analysis instance
Analysis* simulationAnalysis = Analysis::GetAnalysis();

// Add to optical histogram
simulationAnalysis->appendOpticalHistogram(binTime);
// G4cout << "Time is: " << globalTime << ", bin is: " << binTime << G4endl;
J  Event Action

#include "EventAction.hh"
#include "RunAction.hh"
#include "Randomize.hh"
#include "Analysis.hh"

#include "G4RunManager.hh"
#include "G4Run.hh"
#include "G4Event.hh"
#include "G4UnitsTable.hh"

#include <iomanip>
#include <iostream>
#include <fstream>
#include <string>
#include <stdio.h>
#include <sys/types.h>

// Useful module inserts
EventAction::EventAction() : G4UserEventAction() {}
EventAction::~EventAction() {}

void EventAction::BeginOfEventAction(const G4Event* /*event*/) {
        // Acquire analysis instance
        Analysis* simulationAnalysis = Analysis::GetAnalysis();

        // Re-zero event's optical histograms
        simulationAnalysis->resetEventOpticalHistogram();
}

void EventAction::EndOfEventAction(const G4Event* /*event*/) {

// Acquire analysis instance
Analysis* simulationAnalysis = Analysis::GetAnalysis();
G4int numEvents = simulationAnalysis->GetNumEvents();

// Move event tallies over to sum and squared sum histograms
simulationAnalysis->tallyOpticalHistograms(numEvents);
}
K Action Initialization

```cpp
#include "ActionInitialization.hh"
#include "PrimaryGeneratorAction.hh"
#include "RunAction.hh"
#include "EventAction.hh"
#include "SteppingAction.hh"
#include "DetectorConstruction.hh"

// Construct 'action' (whole routine) object
ActionInitialization::ActionInitialization(
    DetectorConstruction* detConstruction)
    : G4VUserActionInitialization(), fDetConstruction(detConstruction) {}

ActionInitialization::~ActionInitialization() {}

void ActionInitialization::BuildForMaster() const {
    SetUserAction(new RunAction);
}

void ActionInitialization::Build() const {
    SetUserAction(new PrimaryGeneratorAction);
    SetUserAction(new RunAction);
    EventAction* eventAction = new EventAction;
    SetUserAction(eventAction);
    SetUserAction(new SteppingAction(fDetConstruction, eventAction));
```