Project Title*

Shaun Alsum
University of Wisconsin-Madison, Department of Physics,
1150 University Ave., Madison, WI 53706, USA
(LUX Collaboration)
(Dated: December 14, 2016)

The LUX experiment is a dual phase liquid xenon time projection chamber whose primary purpose is the detection of WIMP dark matter. To distinguish between background interactions and WIMP interactions both electron and nuclear recoil responses need to be calibrated. LUX uses 2.45 MeV neutrons from deuterium-deuterium fusion to carry out the nuclear calibration. A simple Monte Carlo simulation of this calibration was carried out. Many different interaction scenarios were considered, and discussed. These were classified into pristine, false-pristine, or vetoable events, depending on whether they perfectly fulfilled the requirements of the calibration method, did not but were indistinguishable from those that did, or were distinguishable respectively. The probability of an event being pristine was found to be $\approx 1.4\%$ and the probability that a pristine-looking event was actually false-pristine was found to be $\approx 30\%$.

I. INTRODUCTION

The Large Underground Xenon (LUX) experiment is a two-phase Time Projection Chamber (TPC) that uses xenon as its active volume. Its primary purpose is to attempt to detect the recoil of Weakly Interacting Massive Particles (WIMPs) off of nuclei in its active volume, and thus discover the existence (or non-existence) of the WIMP dark-matter candidate. (citation)

A. The LUX Detector

The LUX TPC is a nearly-cylindrical dodecagonal prism 50 cm in diameter with an active vertical length of 49 cm. The TPC encloses nearly 300 kg of xenon.

The TPC works by comparing the yield of two signals. The first is a prompt scintillation signal emitted when energy is first deposited a particle into the Xenon (S1). Electrons freed by the initial interaction are drifted by an electric field to a surface between gaseous and liquid Xenon maintained between two wire grids. Upon encountering the surface these electrons are extracted from the liquid by means of a stronger magnetic field created by these two grids. Extracted electrons, being no longer encumbered by a dense liquid, accelerate in the field producing a secondary scintillation signal in the Xenon gas (S2). Both the S1 and S2 signals are detected by an array of photomultiplier tubes (PMTs) located at both the top and the bottom of the detector.

The location of the S2 signal (reconstructed by template matching the hit-pattern of the signal on the upper PMT array) is used to determine the xy (parallel to liquid surface) position of a given recoil, while the time-delay between the S1 (prompt) and S2 (electroluminescent) scintillation signals gives a precise measurement of a the z position. The total amount of light measured by either signal, or more commonly a combination of both, can be used to determine the amount of energy deposited during the course of a recoil. Figure 1 illustrates this process.

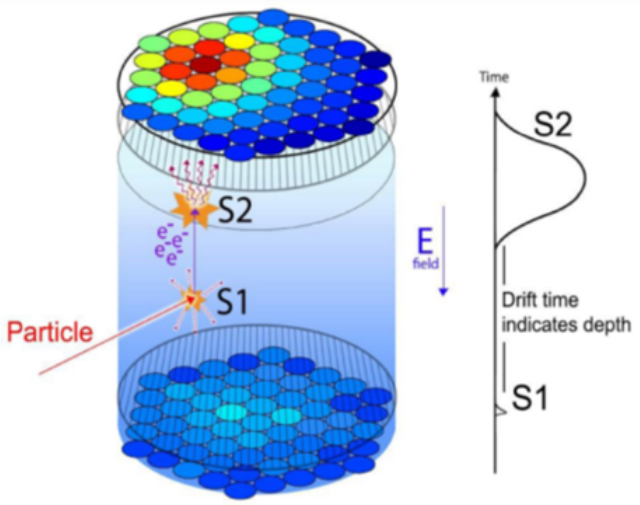

FIG. 1. A particle interaction creates prompt S1 light. Ionization electrons created in this interaction drift to the surface and create S2 light. Light is detected by an array of PMTs on the top and bottom. More red corresponds to a higher number of photons seen by they PMT where deeper blue indicates fewer. The curves on the right indicate the total amount of light observed in all PMTs as a function of time. In principle a particle may continue on after scattering but this is not shown.

Because the WIMP-nucleon cross-section is very small (if it exists at all), interactions are expected to be rare, and backgrounds must therefore be minimized. Background minimization is accomplished in three ways. First, the experiment is located 1,478 meters (4850 feet) underground in the Sanford Underground Research Facility (SURF) in Lead, South Dakota. This protects the

* Prepared for Medical Physics 699 with Professor Michael Kissick

detector from all but a small number of cosmic rays (citation). Second, a tank of water (need number) is placed around the detector to shield radiation emanated from heavy metals in the cavern rock wall. Third, only data from the innermost portion of Xe is used for WIMP search, the outer shell being used as a shield against radiation produced by the detector walls themselves. A more complete description detailing the physical design of LUX can be found in (citation).

Despite all of this shielding, background events are still present in small amounts. Therefore, any differences between the signals arising from backgrounds, and signals arising from possible WIMP interactions must be well understood. There are two primary types of signals in the LUX detector. Electron recoils, in which an incident particle interacts with the coulomb field created by a xenon atom's surrounding electrons, and nuclear recoils, in which a particle interacts directly with the nucleus, or one of its nucleons, typically, via either the strong or the weak nuclear force.

Particles which have the capability of interacting via the electromagnetic force nearly always interact via electronic recoils. This includes electrons, positrons, protons, alphas, and photons. Particles which cannot interact via the electromagnetic force, interact via nuclear recoils. The list of particles which interact in this manner includes neutrons, neutrinos, and, if they exist, WIMPs. The interaction rate of neutrons and neutrinos outside of calibration runs are both estimated at less than one event for the duration of the experiment, so a distinction between electron recoil events and nuclear recoil events is effectively a distinction between signal and background. (citation)

Given the importance of distinguishing between electronic, and nuclear recoil events, both must be calibrated carefully in order to construct a model for classifying each event (or, depending on statistical methodology constructing an overall model of expected outcome based on various WIMP signal strengths). Models usually differentiate between electron and nuclear recoils based on the ratio of light detected via the S2 signal vs the S1 signal. Electron recoil calibration is done using gamma rays from the de-excitation of the krypton metastable state ^{83m}Kr, and using betas from the beta decay of tritium bound into methane CH_3T. Nuclear recoil calibration utilizes neutrons created via deuterium-deuterium fusion. (Citation)

B. Using Neutrons to Calibrate for WIMP Deposited Energy

Neutrons, being neutral, interact with atomic nuclei and thus neutron recoils can be used to precisely calibrate nuclear recoils. The long mean free path of fast neutrons also lends itself well to calibration of a single scatter event, simulating the single hit that would occur if a WIMP (very small cross-section) were to interact with the detector.

The frequency of spin-independent WIMP interactions is expected to decrease exponentially as a function of the interaction's deposited energy in the detector. (citation) The LUX detector has an energy detection threshold that ranges from 0% at 1.1 keV of energy deposited increasing to nearly 100% at a deposition energy of 8.3. (citation) Thus, the desired range of energy depositions from a neutron calibration source is the tens-of-keV range.

Deuterium-deuterium fusion has two possible outcomes, the fusion can produce either a tritium and proton, or a helium-3 (^{3}He) and a neutron, both with a nearly 50% branching ratio at all energies. (citation) We are only concerned with the neutron-producing reaction which has a Q-value of $2m_{D_2} - (m_{^3He} + m_n) = 3.268\,\text{MeV}$. (citation) Deuterium only needs to be accelerated to 80 keV in order to stimulate the reaction, this is insignificant compared to the 3.268 MeV and leaves us squarely in the non-relativistic regime, where kinetic energy $T = \frac{p^2}{2m}$ where p is momentum. This implies that since $p_{^3He} \approx p_n$ the neutron always recieves $\frac{3}{4}$ of the 3.268 MeV, or 2.45 MeV since $m_{^3He} \approx 3m_n$.

A straightforward application of kinematics shows that the maximum possible xenon recoil energy from a single scatter is

$$T'_{Xe} = \frac{4m_n m_{Xe}}{(m_n + m_{Xe})^2} T_n$$

which, for a 2.45 MeV neutron is 74 keV. This fits well into the desired range for recoil energies.

Though the recoil track of a Xe nucleus cannot be observed directly in the LUX detector, the amount of energy deposited in an elastic collision can be precisely determined in the non-relativistic limit if the initial energy and scattering angle of the neutron are well-known. In the non-relativistic limit, $T = \frac{p^2}{2m}$, or $p = \sqrt{2mT}$, the elastic collision of a neutron with a stationary (or nearly stationary) nucleus can be written as follows:

$$\begin{aligned} \frac{p_n^2}{2m_n} &= \frac{p_n'^2}{2m_n} + \frac{p_{Xe}'^2}{2m_{Xe}} \\ p_n &= p'_n \cos\theta + p'_{Xe} \cos\phi \\ 0 &= p'_n \sin\theta + p'_{Xe} \sin\phi \end{aligned}$$

solving this system for T'_{Xe} in terms of the incident energy $T_n = \frac{p_n^2}{2m_n}$ and the neutron scattering angle θ gives

$$T'_{Xe} = T_n \frac{4m_n m_{Xe}}{(m_n + m_{Xe})^2} \frac{1 - \cos\theta}{2}.$$

Therefore, as long as the incident energy T_n and scattering angle of the neutron θ are known, the energy deposited by a given scatter can be determined. (citation) Of course, this result does not hold if the collision is inelastic.

The monoenergetic nature of neutrons resulting from deuterium-deuterium fusion provides further motivation

for its use, because the incident energy of the first scatter is always known. The primary remaining concern, then, is the determination of the neutron scattering angle θ.

C. The LUX Neutron Calibration Scheme

LUX uses an Adelphi Technology, Inc. DD108 neutron generator to create the neutrons for use in calibration. The generator is placed outside the water tank, and emits nearly isotropically into 4π. Shielding is placed around the majority of the generator and an air-filled tube 4.9 cm in diameter (inner-diameter) was placed in the water tank to allow passage of neutrons into the detector. The tube is centered along the side of the detector and placed at 16.1 cm below the liquid surface. The tube is leveled so as to be nearly (but unfortunately not quite) horizontal. This tube serves to allow passage for neutrons into the detector, but also serves to collimate the neutron flux. The generator is typically placed ≈ 45 cm from the water tank, the tube is 377 cm long, and there is ≈ 3 cm of water before the tube begins inside the tank. (Citation) So the neutrons must emerge from a disk 2.45 cm in radius at a distance of 425 cm. This yields an acceptance of $\frac{\pi * 2.45^2}{4 * \pi * 425^2} = 8.6 \times 10^{-6}$ and a maximum incident angle of 0.0058 rad (0.33 deg) from parallel to the tube. In this way, we know the kinetic energy and incident angle of incident neutrons to a high degree of precision.

The LUX detector only detects the position, time, and magnitude of energy depositions. Because we with to know the angle at which the neutron scattered, we must select only events that scatter multiple times (preferably exactly two) in the active region of the detector. This allows us to measure the position of the two events, and thus, the angle at which the neutron scattered off of the Xe. Knowing this, the energy of the first scatter can be determined and compared against its S2 signal, assuming the S2 signals from the two events can be resolved separately. An illustration of this process is displayed in figure 2.

The scheme described above works very well. However, there are many scattering scenarios that can either fool us, or render it very difficult, if not impossible, to use a given event. Determining what fraction of events are tricking us, how many we can reject as not useful, and how many we can actually use is the focus of the rest of this paper.

II. METHODS

The existence of many of these subtleties in the scatters of individual events, as well as the high density of resonances in the neutron-Xe scattering cross section near (but thankfully not at) 2.45 MeV makes analytical analysis of this multiple scattering process intractable. Instead, monte carlo simulation is used to get a handle on

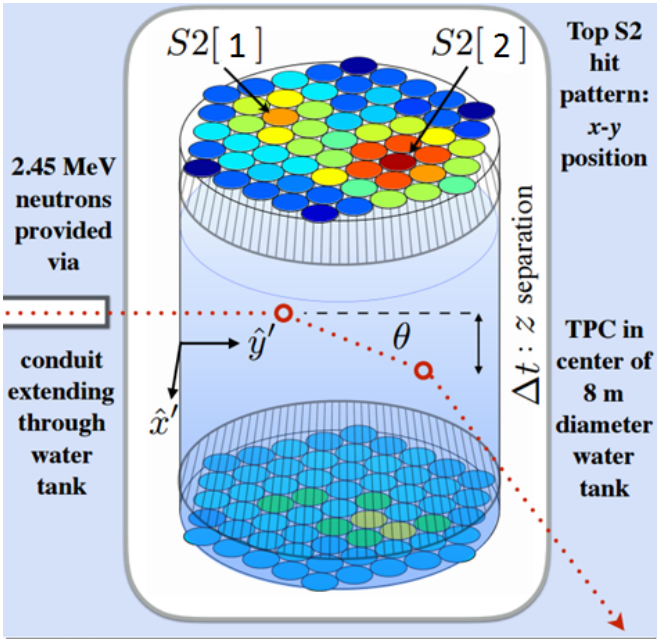

FIG. 2. Diagram of an ideal event in the LUX nuclear calibration setup. 2.45 MeV neutrons produced by the generator emerge from the collimator and scatter in the liquid Xe portion of the TPC. Two scatters take place in the active region and the surviving neutron exits the detector. Two distinct S2 signals originating from the two scatters can be resolved. Note that they y' axis in this figure corresponds to the z axis in the simulation described in section II B

.

these events, with a brief analytical analysis used only to cross-check simulation results.

A. Expected Simulation Results for the First Scatter

According to the National Nuclear Data Center curated by Brookhaven National Laboratory, the neutron-Xe total cross section at 2.45 MeV is ≈ 5.95 b. (citation) The narrow-beam attenuation coefficient is therefore

$$\mu = \frac{\rho N_A}{A}\sigma \approx \frac{3.1\,\mathrm{g\,cm^{-3}} \cdot 6.022 \times 10^{23}}{131.1\,\mathrm{g\,mol^{-1}}} \cdot 5.95 \times 10^{-24}\ \mathrm{cm^2}$$
$$= 0.085\,\mathrm{cm^{-1}}$$

where μ is the narrow-beam attenuation coefficient, ρ is the density of the medium, N_A is Avogadro's Number, A is the atomic weight of the element, and σ is the total cross section. This means that the distribution of penetration depth for a neutron first-scatter should be a decaying exponential $d(x) = ke^{-\mu x}$ where k is a constant and $\mu = 0.085\,\mathrm{cm^{-1}}$ as determined above. (citation) $\mu = 0.085\,\mathrm{cm^{-1}}$ corresponds to a mean free path of 11.8 cm. We expect to see this reproduced in our simulation.

Due to excess radiation from the wall of the detector, and possible contamination from slower neutrons that escaped from the tube have scattered in from the water tank, only first scatters that occur more than 15 cm from the point the beam enters the detector. (citation) The probability of interaction within the first chunk of length x in a medium is given by

$$P(x) = 1 - e^{-\mu x}.$$

Therefore the probability of first interaction occurring after between 15 cm and 50 cm (the back of the detector) should be

$$\begin{aligned} P(15\,\text{cm}, 50\,\text{cm}) = \; & P(50\,\text{cm}) - P(15\,\text{cm}) \\ = \; & 1 - e^{-0.085\,\text{cm}^{-1}\cdot 50\,\text{cm}} - (1 - e^{-0.085\,\text{cm}^{-1}\cdot 15\,\text{cm}}) \\ = \; & 0.279 - .014 = 0.265. \end{aligned}$$

Furthermore, recall that our energy reconstruction method only works for elastic scattering. We can determine the number of elastic scatters in our region of interest by replacing our previous $P(x)$ by

$$P_{es}(x) = \frac{\mu_{es}}{\mu} e^{-\mu x}$$

where μ is still the total thin-beam attenuation coefficient, but μ_{es} is the elastic scatter only attenuation coefficient. (citation) The National Nuclear Data Center gives σ_{es} a value of ≈ 5.72 b which corresponds to a μ_{es} of $0.81\,\text{cm}^{-1}$. Therefore, the fraction of first scatters that are actually elastic is $\approx \frac{.81}{.85} \cdot .265 = .253$ or $\approx 95\%$ of the total interactions. We also expect to see this approximate result reproduced in our simulation.

B. The Monte Carlo Simulation

The simulations were done using an application built from the Geant4 simulation toolkit. Geant4 version 10.02.p02, the most recent version as of the date of this publication, was used. Geant4 requires user-definition of all physics processes that are allowed to take place during a simulation. A number collections of processes are included with Geant4 and are called "reference physics lists." (citation) One of these called "QGSP_BIC_HP" that, among other things, specializes in the correct handling of neutron interactions below 20 MeV (citation) was chosen.

The geometry used in this study was simply a cube of liquid Xe 12 m on a side embedded at the center of the world defined as a vacuum cube 14 m on a side. The liquid xenon was created in the simulation by telling Geant4 its atomic number, atomic weight, and mass density. Neutrons are fired in the +Z direction from a disk 2.45 cm in radius centered at the origin with extent in the xy plane. Initial particle positions are chosen randomly from a uniform probability distribution on this disk. All neutrons are fired with a kinetic energy of exactly 2.45 MeV. These particles and any secondaries created via particle interaction are simulated until all particle kinetic energies reach 0, or the particle in question leaves the 14 m world. In practice, the particles in this particular simulation do not reach the edge, but come to a stop within the liquid Xe.

Each event (a single firing of a neutron) is made up of a number of "tracks," each representing a particle as it travels through the medium (In fact, some particles are split into multiple tracks, a fact that caused no end of headaches in performing the analysis). Each track is made up of "steps," each representing an interaction, or transition from one material to another. After an event is simulated, this information is contained in class objects called "Trajectory" and "Point" respectively, which can be used for analysis until the next event is simulated. (citation)

1.5 million events were simulated in total.

C. Analysis of the Monte Carlo

After each event was simulated, it was analyzed to determine whether it is a pristine event, or a non-pristine event. A so-called pristine event is one that fully satisfies the assumptions made in our calculation of the energy deposited by the first scatter, and in which the first and second events can be uniquely identified and resolved. The non-pristine events come in two flavors, vetoable, which leave some hint that we don't want to, or can't, use them, and non-vetoable, which would pass as pristine events in the detector. The process of working through the various scenarios that result in classification into these categories is what follows.

Unfortunately, despite being non-relativistic, the neutrons are still traveling fast enough that time-of-interaction alone is not a very reliable indicator of what order scatters occurred in. Therefore, we must determine this via some other method. The method we use is event selection from inspection of geometry. As long as one, and only one, scatter occurs in the cylindrical region $x^2+y^2 < 2.45^2$ cm^2 (the projection of the collimator tube into the detector), that scatter must be (barring possible unusual circumstance we will get to later) the first scatter. Further, if there are more than two scatters detected, it is difficult to determine which is the second, even if the first is known, since scattering can occur at any angle. Because of these reasons we make the requirements that there be only one detected interaction in the aforementioned cylindrical region, and that there must be exactly two detected interactions total.

1. *The IsDetected Method*

The above requirements themselves require that we know whether a given scatter will be detected or not.

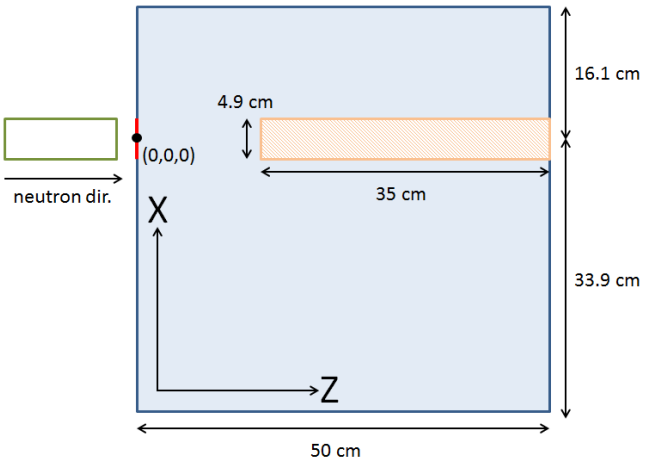

FIG. 3. A diagram of the simulation layout. Although no actual geometry apart from liquid xenon has been constructed, several regions are defined in analysis. The green box illustrates where neutrons from the generator would come from, if the generator was simulated. Instead, The red line centered at the origin indicates the disk on which neutrons were actually created. The blue region marks out the boundaries of the detector. Any interaction occurring outside the blue region (which contains the pink hashed region) was declared to have not been detected. The pink hashed region indicates the so-called "first scatter" region. Any event whose first scatter does no occur in this region, was rejected. Note that the neutrons are fired along the z-axis, and the x-axis indicates the vertical direction in the TPC. Also note that no actual structure was simulated, these are simply regions defined in the analysis.

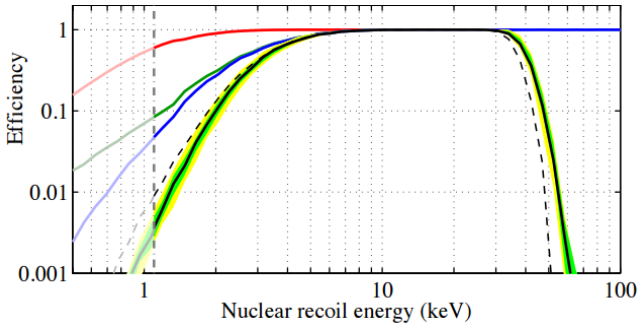

FIG. 4. The probability of accepting a nuclear recoil with given energy. There is a hard cutoff at 1.1 keV. Red is the probability of S2 detection, green is S1, blue is the probability of some detection in both, and black is the probability of both being above threshold for analysis. The dashed line is the predicted black curve using an alternate model. I use the Black line as the probability of detection in this work.

This brings us to the algorithm that is the workhorse of this analysis, a method named "IsDetected." IsDetected does just what its name implies, it predicts whether a given simulated scattering interaction would be, or would not be detected by the LUX detector. IsDetected has two primary conditions that must be met to return an affirmative; the given event must be within the geometrical confines of the LUX detector, were it to be simulated, and it must be above the energy threshold of the detector.

The first check is fairly straightforward, the method pulls the position of the interaction from the point in question, and checks that $-32.9\,\text{cm} < x < 16.1\,\text{cm}$ (the vertical direction), and that $y^2 + (z - 25\,\text{cm})^2 < 25\ \text{cm}^2$ (the horizontal plane). This approximates the detector as a right cylinder, instead of using its actual dodecagonal shape. This region is shaded in blue in figure 3. Note that this is only an analysis cut, no actual physical walls were simulated.

The second criterion is slightly more complicated. As mentioned in section I B, the energy detection threshold of LUX is not a single value, but ranges from 0% acceptance at 1.1 keV, to 100% at ≈ 8.3 keV. The Acceptance probability can be viewed in figure 4. This covers a significant range of expected energy depositions, and so must be simulated instead of simply accepted above a certain energy. To determine whether an event was accepted or not, the energy of the interaction had to be extracted. In most cases, this extraction was done by simply subtracting the kinetic energy of a point post-interaction from the kinetic energy of the point pre-interaction. However, on occasion a track was split into two, which results in the post-interaction energy of the last point on the first track having a value of 0. In cases where this is the point in question, the daughter track was found and the energy that it was initialized with was used instead of the post-interaction energy. Once the energy deposition was determined, the acceptance probability for this energy was found via linear interpolation of a data table loaded in from a file if the energy deposition was found to be between 1.1 keV and 8.3 keV. Otherwise the acceptance probability was taken to be 0, if the energy was below 1.1 keV, and 1 if above 8.3 keV. Finally, a random number between 0 and 1 was generated and compared to the obtained acceptance probability; if the number was less than the acceptance probability, the energy threshold criterion was determined to be met, if the number was greater, the criterion was not met.

Note that if an interaction is an inelastic scatter, it is possible for the energy from an excited nucleus to be released in the form of a gamma ray that would be detected elsewhere, thus causing the initial interaction site to have low enough local energy deposited that it was not detected. This case is not considered.

2. *Multiple Scatters*

Once we have a method for determining if a given interaction is detected or not, one of the easiest things to do, is to make sure that an event consists only of exactly two detectable scatters. Recall that this is one of the criteria for being a pristine event. We simply look at every single track, determine if any given track is a

neutron, and if it is, run the IsDetected Method on every single point in that track. If IsDetected determines that exactly two interactions were detected, the event may be pristine. I call this the "DS criterion." If any number of interactions other than two were detected, the event is classified as Non-pristine, but vetoable.

The rest of the analyses are still run on events not passing this cut because they are interesting, but they will not be classified as pristine events.

3. First Scatters

We now look at what is required of the first scatter so that the overall event can be considered pristine. Recall from section I B that our energy measurement only works if the incident neutron is at 2.45 MeV and is coming in straight from the direction of the collimator. This requires that the first detected scatter is the first scatter to occur. So for now, we shall ignore everything except the second point of the first track, which must be the first interaction of the incident neutron. The first criterion that we must meet is that the event must take place at least 15 cm into the detector. For future reference, we will call this the "A15 criterion." The reasons for this were discussed in section II A. The status of the A15 criterion is easily determined through use of the "IsInBeam" method.

The, perhaps poorly named, IsInBeam method simply checks whether a given interaction takes place in the cylindrical region defined at the top of this subsection (Analysis of the Monte Carlo) at a distance of between 15 cm and 50 cm (the opposite end of the detector) from the particle generation point. If the interaction is indeed in this region, it returns true, otherwise it returns false.

The second criterion we consider is that the second point must be detected. If it is low enough in energy deposition, as previously discussed, it may not be. This is easily determined through the employment of the IsDetected method discussed earlier. We call this the "secDet criterion."

Thirdly, we recall that our energy measurement is only valid in the case of elastic scatters. The only processes that occur with a non-negligible cross section for neutrons in Xe around 2.45 MeV is elastic and inelastic scattering (citation), so to deal with this "elastic criterion" we simply ask the interaction point for what process it was using and reject the event if it is an inelastic process.

4. Foolers

As alluded to in the previous section, it is possible that the first interaction detected, was not the first interaction that occurred. In this case, if the first interaction detected meets all the criteria that make the first scatter pristine, it will fool us into thinking we can use it for calibration, despite the fact that it probably satisfies neither the initial energy, nor the incident angle conditions necessary for us to use our energy deposition equation. For this reason, I have dubbed such events "foolers." An example of a fooler is illustrated in figure 6 on page 7.

If an event doesn't meet the secDet criterion, each point in the initial neutron track is investigated in turn, starting with the next. If the point in question was detected, it is determined to be a fooler if IsInBeam confirms it is in the acceptable first scatter region. If it was detected but was not in this region, it is determined that the point is not a fooler. If no points in the initial neutron track were detected, any daughter neutron tracks are treated in the same manner. If the point was not detected at all, the investigation moves on to the next point. If no points at all were detected, the event is, of course, not a fooler as it would be vetoed for not having 2 detected interactions.

5. Second Scatters

We now look at what is required of the second scatter to make the event pristine. I considered only events that had passed the criteria discussed for the first scatter (A15, secDet, and elastic). Because the neutron must have come directly from the point of first scatter, I currently consider only the third point in the initial neutron track, or if the initial neutron track ended after the second point, I consider the *second* point of the daughter neutron track (the first is the initialization) if it exists.

The first criterion for the second scatter is that it must have occurred outside the first scatter region. If it did not, the first and second scatters may be confused. This was easily determined by again utilizing the IsInBeam method. The difference from previous uses being that it must this time return false. I name this criterion "GDS." The second criterion for acceptance concerning the second scatter is that the scatter be detected (DGDS). This is readily determined through another call to IsDetected.

The third criterion (RGDS) is that the second interaction must be far enough from the first that the spatial resolution of the two interactions does not significantly impede the reconstruction of the scattering angle. This is the case if the two events are closer than ≈ 5 cm. I simply locate the two events and exclude the event if this is the case. This has the added benefit of excluding the case where the two events are close enough together that, while each would be detected individually, and so IsDetected would return true, the actual LUX detector would detect only a single, larger, event.

Note that it does not matter whether the second event is elastic or inelastic, it only needs to be detected and be in the correct location.

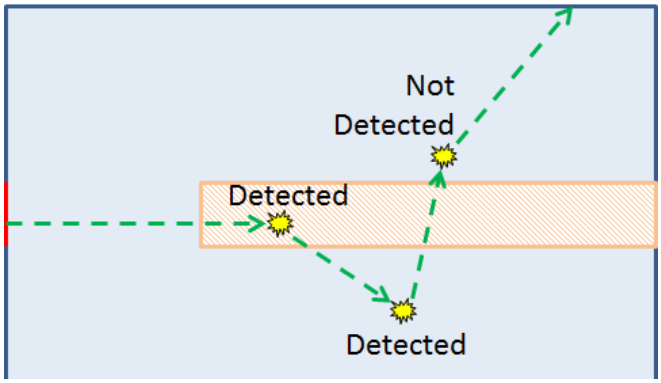

FIG. 5. Example of a pristine event. One scatter detected inside the first scatter region, one detected outside. The green arrows represent the neutrons, and the yellow stars depict scatters. The following two figures are scenarios that may masquerade as pristine events.

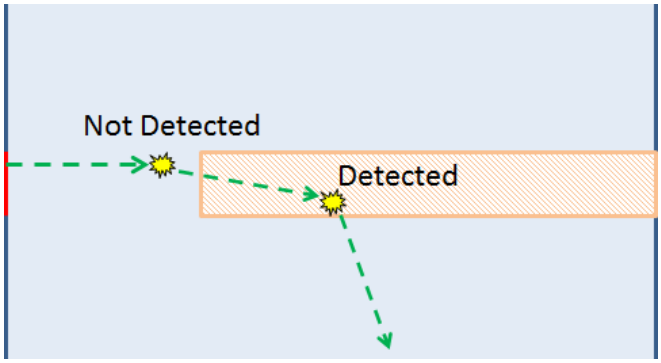

FIG. 6. Example of a fooler. The first detected scatter happens in the first scatter region, but was preceded by another scatter that was not detected.

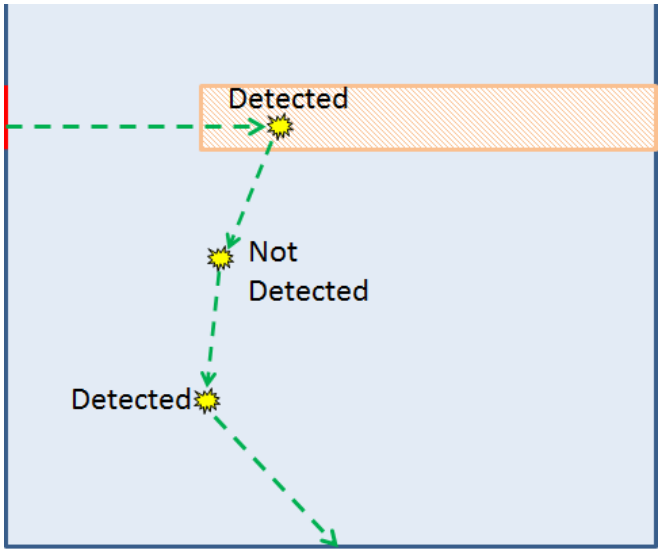

FIG. 7. Example of a Trickster. The scatter occurring immediately after the first detected scatter was not detected, but a future one was.

6. *Tricksters*

I have named the analog to foolers from the first scatter case "tricksters." An event is said to be a trickster if an event meets all the criteria concerning the first scatter, the second interaction was not detected, and the first interaction *detected after* the second interaction meets the GDS, DGDS, and RGDS criteria. An event is determined to be a trickster using the same method as a fooler, except that the hunt begins after the second event, and, of course, that the detected event follow the rules for a second scatter rather than a first. An example of a trickster is shown in figure 7.

7. *Pristine and Non-Pristine Events*

If an event has only two detected scatters and meets all 6 discussed criteria (A15, secDet, elastic, GDS, DGDS, and RGDS), it is classified as a pristine event. See figure 5 for an example. Nearly all other events can be vetoed unless they are either a fooler or a trickster. The number of tricksters can be determined from the results of the simulation, however I, unfortunately, only analyzed second scatters if all three of the first scatter criteria were met. Because of this, I don't have a direct measure of how many foolers there were that would be indistinguishable from pristine events. I expect the second scatter to depend minimally on whether the first detected scatter was a fooler or not, so multiplying the ratio of foolers to total events by the ratio of pristine events to events meeting the first scatter criteria should give a good estimate of the fraction of total events that would be pristine if there hadn't been a fooler.

III. RESULTS

A. Validation

In section II A we determined that the mean free path of 2.45 MeV neutrons in liquid Xe should be ≈ 11.8 cm. We find that the simulation reproduces this result with good agreement. After determining the distance from initialization to first scatter, we find that the mean free path coming from the simulation is 12.05 cm. This can be seen in figure 8. Further, we predicted the fraction of events interacting for the first time between 15 cm and 50 cm would be .265, with the number that are first interacting in that region and are also elastic collisions to be about 95% of that number. Our results for overall attenuation match quite well. we measure 408,985 events matching the first criterion which yields a fraction of $0.27265 \pm 3.6 \times 10^{-4}$ (error quoted is the standard error of the mean obtained using binomial statistics) (citation). These agreements with the predicted results give me confidence that the simulation is behaving in a manner consistent with reality. However, our result for the

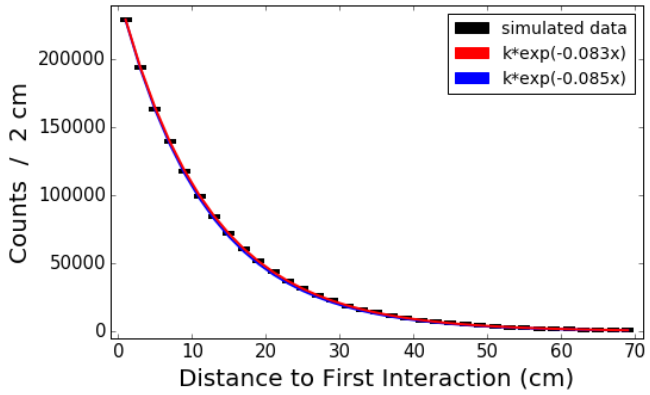

FIG. 8. This plot shows the agreement in fit between the simulated distance to first scatter in black and an exponential fit with $\mu = 0.083$ in red. The curve corresponding to the looked-up value of $\mu = 0.085$ is displayed in blue but is difficult to see because much of it lies under the red curve.

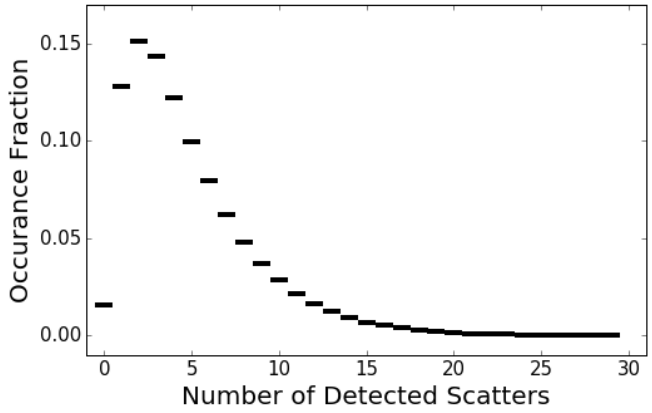

FIG. 9. This displays the fraction of events that occured with a given number of detected scatters. The plot is truncated after 29, but there are still a non-zero number of events that fall into this category. For reference 133 events had 29 detected scatters.

number of inelastic collisions is quite far off, we obtain a fraction of $0.31610 \pm 3.8 \times 10^{-4}$; nevertheless, we press on.

(maybe something about usual accuracy of neutron simulations with citation)

B. Multiple Scatters

The distribution of number of scatters detected fortunately peaks at 2 interactions detected with a fraction of 0.1513. The distribution is displayed in figure 9 We can veto all but this fraction.

C. First Scatters

After considering only the subset of events with two detectable scatters, the fraction of first scatters occurring in the first scatter region remains nearly unchanged with 60,854 occurring out of 226,884 double scatter events. I also report that 56,495 events were both detected, and in this region, while 41,478 of the events in this region were elastic. The number of events meeting all three criteria (GFS) while being the first of only two detected scatters, was 37,121, giving a fraction of $0.02475 \pm 1.3 \times 10^{-4}$.

The number of foolers simulated was 31,717 out of the full simulated 1.5 million. Subsequent detected scatters should not influence, nor be influenced by, the fooler fraction, so all events, not only double scatters, were considered. This implies that a fraction of 0.02114 ± 0.00012 of all events detected to be meeting the criteria for a good first scatter are misleading.

D. Second Scatters

The number of events that met the pristine event criteria for the first scatter and have only two detected scatters is 37,121. Out of these, 29,600 of the second scatters meet the GDS criterion (located outside of the first scatter region), 24,837 meet DGDS (are detected) and 21,253 meet RGDS (enough distance between first and second scatters) in addition to the others. This makes the fraction of events that meet all criteria (and are therefore pristine events) 0.014168 ± 0.000096 of all 2.45 MeV neutrons making it into the detector from the calibration source produce pristine events.

The number of tricksters was simulated to be 2,086, giving a fraction of $1.390 \times 10^{-3} \pm 3.0 \times 10^{-5}$.

All of these results are summarized in table I.

IV. DISCUSSION

A. Pristine Event Rate

The simulation implies that $\approx 1.4\%$ of neutrons making it into the detector at the correct incident angle and energy will be pristine. This is an appreciable fraction of the total. For a given LUX calibration run, at least 1,000 pristine events are desired. The LUX neutron generator is reported to emit $\approx 2.5 \times 10^6$ neutrons/s into 4π (citation). An acceptance fraction of 8.6×10^{-6} was calculated in section I C, this yields a rate of 21.5 neutrons/s. However, recall that ≈ 6 cm of water is located along the neutron path before it reaches the detector. According to the NNDC, the neutron-hydrogen total cross section at 2.45 MeV is 19.2 b while the neutron-oxygen cross section is 3.5 b. (citation) This yields an attenuation coefficient

Criteria Met	# Events	Probability
A15	408,985	0.273
DS	226,884	0.151
DS, A15	60,854	0.041
DS, A15, secDet	56,495	0.038
DS, A15, elastic	41,478	0.028
DS, GFS	37,121	0.025
DS, GFS, GDS	29,600	0.020
DS, GFS, DGDS	24,837	0.017
pristine	21,253	0.014
inelastic 1st scatter	474,155	0.316
fooler	31,717	0.021
DS, trickster	2,086	0.0014
pristine	21,253	0.014
false-pristine		0.0061

TABLE I. This table displays the number of events meeting the criteria defined in section II C, as well as the associated probability derived from these numbers. All probabilities are displayed to their third decimal points with the exception of those with relevance to false-pristine events since their main value is in being compared to pristine events, which themselves have a small probability. Also note that inelastic and fooler events are defined for all events, whereas tricksters are only defined for those meeting the GFS criterion. The false-pristine event probability is calculated (see section IV B) and so there is no associated # Events entry.

of

$$\begin{aligned}\mu &= \mu_H + \mu_O \\ &= \frac{2\rho_{water}N_A}{A_{water}}\sigma_H + \frac{\rho_{water}N_A}{A_{water}}\sigma_O \\ &= \frac{2\cdot 1\,\mathrm{g\,cm^{-3}}\cdot 6.022\times10^{23}}{18.0\,\mathrm{g\,mol^{-1}}}\cdot 19.2\times10^{-24}\ \mathrm{cm^2} \\ &+ \frac{1\,\mathrm{g\,cm^{-3}}\cdot 6.022\times10^{23}}{18.0\,\mathrm{g\,mol^{-1}}}\cdot 3.5\times10^{-24}\ \mathrm{cm^2} \\ &= 1.39\,\mathrm{cm^{-1}}.\end{aligned}$$

Applying the attenuation equation with this number gives a survival fraction of

$$\frac{I}{I_0} = e^{-\mu x} = e^{-1.39\,\mathrm{cm^{-1}}\cdot 6\,\mathrm{cm}} = 2.4\times10^{-4}.$$

Given the fraction of events that are pristine, this gives an estimate for the pristine event rate of $21.5\,\mathrm{s^{-1}}\cdot 2.4\times10^{-4}\cdot 0.014 = 7.2\times10^{-5}\ \mathrm{s^{-1}}$. This is about 6 events per day and was not what was observed. This leads the author to believe there was an error in one of the numbers reported in (citation). Factoring in events that forward scatter and so are still detected and also events that are not pristine but are only minimally scattered don't seem enough to bridge the gap between this rate and the observed rate (1031 used events in 107.2 live-hours during the 2013 calibration (citation)). At this rate one calibration run would take ≈ 160 days which would be devastating. On the other hand, if the detection efficiency of the Bonner sphere (≈ 5% (citation)) were neglected in the determination of the neutron flux, the resulting error would lead to a result similar to the one obtained here. Regardless, the results of this study reveal that the length of a calibration run must consist of at least ≈ 71,400 neutrons entering the detector, at whatever rate they arrive.

A number of improvements could be made to this study in order to obtain a better estimate for the length of time needed for calibration. Most obviously, the study could benefit greatly from full simulation of all detector components, instead of just the Xe. A more complete inclusion of the air-filled neutron tube, water tank, titanium cryostats, and teflon TPC could reveal with more clarity how many neutrons successfully make it into the active region of the detector. Additionally, this would include forward-scattering effects, not simply use assumptions based on the narrow-beam attenuation coefficient. A further improvement would be to include interactions of long-ranged secondary particles. Some additional events may be vetoable due to this effect, or some events classified by this analysis as single scatter may masquerade as pristine events due to these secondaries. A third improvement would be to check the energy distribution of the pristine events to ensure the whole range is well represented. If there were deficiencies in any region, the exposure time may have to be increased.

B. False Events

The class of events which are not pristine, and yet will appear so in the detector, consists of events with inelastic first scatters, foolers, and tricksters. The probability of an event having its first scatter be inelastic, but being otherwise pristine, was not simulated. However if we assume that the probability of an event's second scatter is unaffected by the first being inelastic (a dubious assumption, given the proximity to resonances), the probability of being fooled by this is simply $p_{inelastic} = f_{inelastic}\cdot f_{pristine} = 0.316\cdot 0.014 = 4.4\times10^{-3}$. The calculation for number of foolers is similar: $p_{foolers} = f_{fooler}\cdot f_{pristine} = 0.021\cdot 0.014 = 2.9\times10^{-4}$. The number of tricksters was recorded for the entire simulation, and was 1.4×10^{-3}. The probability of having a false-pristine event, is then the sum of these three probabilities, minus the chance that an event could fall into multiple of these categories. The probability of this happening, however, is negligible. This gives $p_{false-pristine} = p_{inelastic} + p_{fooler} + p_{trickster} = 6.1\times10^{-3}$. This means that since the probability of being a pristine event was 0.0142, about 30% of all events accepted as pristine are actually flawed in some way.

Although I did not address it in this analysis, it may be possible to veto a large number of inelastic collisions, either because they tend to deposit more energy, and so a population may therefore be identified and cut despite the lack of calibration, or because they may emit a γ-ray that could be detected and vetoed upon. If the majority of these events could be vetoed, or if the cross section in Geant4 is wrong and the inelastic fraction were much

smaller, it could bring the probability of a false-pristine event to just those contributed by foolers or tricksters, 1.7×10^{-3}. Which would make only 11% of accepted events false-pristines.

There are also many ways to improve this part of the study. Again a more complete simulation geometry would be useful. This would allow the study of events in-scattering from components external to the liquid Xe, which may also fool us. It would also be prudent to take secondaries, like the γs mentioned above, into account for the purposes both of veto, and false signal. It may also be possible to study both the energy and position distributions of these false-pristine events to see if there are any differences between them and real pristine events. If there are, regions corresponding to false-pristines can be de-weighted relative to the rest of the calibration.

Even once we know how many false-pristine events there are in our calibration data, we still don't know their affects on its results. To determine the affects on our results, the simulation could be extended to include the actual S1 and S2 signal yields for every given event, and the whole calibration analysis done with this data. We could then compare this model directly to the data, varying the signal yields to match the whole simulation, false-pristines and all, in order to obtain calibrated values for nuclear recoils.

V. CONCLUSION

Monte Carlo results show that $\approx 1.4\%$ of calibration neutrons entering the detector are perfect candidates for use in the calibration for the nuclear recoil response in LUX. Although 1.4% seems like a reasonable probability for acquiring useful events, a brief analysis of neutron production rate and propagation reveal a prohibitively long calibration period, or an error in either the information reported in (citation), or in the analysis method. Further results show that as many as 30% of perfect-looking results may not satisfy assumptions made in the calibration analysis, though the impact of these results need further study. Ways of exploring this impact, or minimizing its effects have been proposed.

VI. AKNOWLEDGEMENTS

VII. REFERENCES

Appendix A: Analysis Code

The following is a code dump of the file in which the bulk of the analysis was done. It is the "EventAction.cc" file which is executed once per Geant4 event. Any references to "HistoManager" are references to another file not included, but this is simply used for filling histograms for analysis.

```
//
// ********************************************************************
// * License and Disclaimer                                           *
// *                                                                  *
// * 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.          *
// ********************************************************************
//
/// \file runAndEvent/RE01/src/RE01EventAction.cc
/// \brief Implementation of the RE01EventAction class
//
// $Id: RE01EventAction.cc 66379 2012-12-18 09:46:33Z gcosmo $
//

#include "EventAction.hh"
#include "Trajectory.hh"
#include "HistoManager.hh"
#include "TrackInformation.hh"
#include "TrajectoryPoint.hh"
#include <vector>
#include <iostream>
#include <fstream>
#include <sstream>
#include <cstring>
#include <typeinfo>
#include <regex>
#include <libgen.h>
#include <string>
#include <cmath>

#include "G4Event.hh"
#include "G4EventManager.hh"
#include "G4HCofThisEvent.hh"
#include "G4VHitsCollection.hh"
#include "G4TrajectoryContainer.hh"
```

```
#include "G4VVisManager.hh"
#include "G4SDManager.hh"
#include "G4UImanager.hh"
#include "G4ios.hh"
#include "G4PrimaryVertex.hh"
#include "G4PrimaryParticle.hh"
#include "G4SystemOfUnits.hh"
#include "G4NistManager.hh"
#include "Randomize.hh"


//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
EventAction::EventAction()
  :G4UserEventAction()
{
  fHisto = new HistoManager();
  fDeadLengthX = 3.0*cm;//3.0*cm;
  fDeadLengthY = 4.0*cm;//4.0*cm;
  fDeadLengthZ = 6.0*cm;//6.0*cm;
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
EventAction::~EventAction()
{
  delete fHisto;
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
void EventAction::BeginOfEventAction(const G4Event* evt)
{
  G4int eventID = evt->GetEventID();
  if( eventID % 1000 == 0 )
    { G4cout << "Execution " << double(eventID)/10000 << "% Complete." << G4endl; }
  //  G4cout << "Event Number " << eventID << G4endl;

}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
void EventAction::EndOfEventAction(const G4Event* evt)
{

  ////////////////////// MuRS ANALYSIS ////////////////////
  // get number of stored trajectories
  G4TrajectoryContainer* trajectoryContainer = evt->GetTrajectoryContainer();

  // extract the trajectories and print them out
  doTheMuRSAnalysis(trajectoryContainer);

}


void EventAction::doTheMuRSAnalysis(G4TrajectoryContainer* trajectoryContainer)
{
  // Load in acceptance data
  std::vector<double> aEnergy;
  std::vector<double> aProb;
  std::ifstream fin("/home/shaun/G4/userMade/userMade/LUX_acceptance.csv");
  std::string line = "";
```

```
//int count = 0;
while(std::getline(fin,line))
  {
    std::stringstream strstr(line);
    std::string word = "";
    bool xyState = 0;
    while(std::getline(strstr,word, ' '))
    {
      if(xyState == 0)
        {
          aEnergy.push_back((G4double)atof(word.c_str()));
          //G4cout << aEnergy[count] << " KeV" << G4endl;
          xyState = 1;
        }
      else if(xyState == 1)
        {
          aProb.push_back((G4double)atof(word.c_str()));
          //G4cout << aProb[count] << " probability" << G4endl;
          xyState = 0;
        }
    }
    //count++;
  }

//Incriment counter hist for total events
fHisto->FillCounter(1);  // 1 is the code assigned to total event #

G4int n_trajectories = 0;
if (trajectoryContainer) n_trajectories = trajectoryContainer->entries();


for ( int aaTraj = 0; aaTraj < n_trajectories; aaTraj++ )
  {
    /*      G4VTrajectory* Traj = trajectoryContainer[aaTraj]);
    G4int parentID = Traj->GetParentID();
    if( parentID == 0){
      int numPoints = Traj->GetPointEntries();
      for(int bbPoint = 0; bbPoint < numPoints; bbPoint++){
        if(bbPoint == 0){
          G4VTrajectoryPoint* Point = Traj->GetPoint(bbPoint);
          G4ThreeVector pointPosition = Point->GetPosition();
          fHisto->FillEndofEventTotalNum(pointPosition.getX());
        }
      }
    }
    */
    // Make the Trajectory
    Trajectory* Traj = (Trajectory*)((*(trajectoryContainer))[aaTraj]);
    G4int parentID = Traj->GetParentID();
    G4int trackID = Traj->GetTrackID();
    // Look at primaries
    if(parentID == 0)
    {
      //Create flag that will determine if the event is kept at various stages
      G4bool keep = true;
      // Look at initialization data
      G4double Pinitial = Traj->GetInitialMomentum().mag();
      fHisto->FillInitialMomentum(Pinitial);
```

```
TrajectoryPoint* iniTrajPt = Traj->GetPoint(0);
G4ThreeVector iniPos = iniTrajPt->GetPosition();
G4double iniPosX = iniPos.x();
G4double iniPosY = iniPos.y();
G4double iniPosZ = iniPos.z();
fHisto->FillInitialPosX(iniPosX);
fHisto->FillInitialPosY(iniPosY);
fHisto->FillInitialPosZ(iniPosZ);

G4double secPosX = 0;
G4double secPosY = 0;
G4double secPosZ = 0;

Traj->SetNumPoints();
G4int numPoints = Traj->GetNumPoints();
if(numPoints > 1)  // should always be true
{
  // compare second point to first
  TrajectoryPoint* secTrajPt = Traj->GetPoint(1);
  G4ThreeVector secPos = secTrajPt->GetPosition();
  secPosX = secPos.x();
  secPosY = secPos.y();
  secPosZ = secPos.z();
  fHisto->FillSecPosX(secPosX);
  fHisto->FillSecPosY(secPosY);
  fHisto->FillSecPosZ(secPosZ);


  G4double dist12 = EventAction::Distance(iniPos, secPos);
  fHisto->FillDistFirstScatter(dist12);  // plot of first interaction distance

  // count events that have first scatter after 15 cm
  G4bool A15 = false;
  if(EventAction::IsInBeam(secTrajPt)){A15 = true;}
  if(A15 == true){fHisto->FillCounter(2);}

  // count DETECTED events that have first scatter after 15 cm
  // energy deposited not so simple because new neutron is created if it
  //   is inelastic
  G4bool secDet = EventAction::IsDetected(secTrajPt, trackID, trajectoryContainer, aE

  // check to see if the first scatter was inelastic
  G4bool inelasticFirstScatter = false;
  if(secTrajPt->GetProcessName() == "neutronInelastic")
    {inelasticFirstScatter=true;}

  G4bool goodFirstScatter = false;
  if(A15 and secDet and !inelasticFirstScatter){goodFirstScatter=true;}
  if(goodFirstScatter){fHisto->FillCounter(3);}


  // count how many tracks are DETECTED for the first time in
  //   the beam (foolers)
  G4bool fooler = false;
  G4int tmpTrackID = trackID;
  Trajectory* tmpTraj = Traj;
  G4int tmpNumPoints = numPoints;
  //if(!A15 and !secDet)
```

```
if(!secDet)
{
  G4int bbPt;
  //if(secTrajPt->GetProcessName() == "neutronInelastic"){bbPt = 1;}
  if(numPoints == 2){bbPt = 1;}
  else {bbPt = 2;}
  while(true)
    {
      TrajectoryPoint* bbTrajPt = tmpTraj->GetPoint(bbPt);;
      // If detected, check if it was in the beam
      if(EventAction::IsDetected(bbTrajPt, tmpTrackID,
                                 trajectoryContainer, aEnergy, aProb))
        {
          if(EventAction::IsInBeam(bbTrajPt)){fooler = true;}
          else{fooler = false;}
          break;
        }
      // If not detected, check if last point in track
      //   if so, check if a daughter neutron exists and start down
      //   that track
      else if(bbPt == tmpNumPoints-1)
        {
          if(bbTrajPt->GetProcessName() == "neutronInelastic")
            {
              for(int ccTraj = 0; ccTraj < n_trajectories; ccTraj++)
                {
                  Trajectory* ttTraj = (Trajectory*)
                    ((*(trajectoryContainer))[ccTraj]);
                  G4int ttParentID = ttTraj->GetParentID();
                  if(ttParentID == tmpTrackID){
                    tmpTraj = ttTraj;;
                    break;
                  }
                }
              tmpTrackID = tmpTraj->GetTrackID();
              tmpTraj->SetNumPoints();  //seemed to be working
              //without this, hopefully doesn't stop working with
              tmpNumPoints = tmpTraj->GetNumPoints();
              bbPt = 1;
            }
          else{fooler=false; break;}
        }
      // If not detected and not last point in track, continue
      else{bbPt++;}
    }
}  // end of fooler checks
if(fooler){fHisto->FillCounter(4);}



G4bool goodDoubleScatter = false;  //1st scat det in beam 2nd out
G4bool detGoodDoubleScatter = false;  //gDS that is detected
G4bool trickster = false;  // gDS not detected then detected later
G4bool outActive = false;  // second scatter farther than x away in
                           //one direction (simulate going above grid)
G4bool resolvedGoodDoubleScatter = false;
G4bool doubleScatter = false;
G4bool useableEvent = false;
```

```
G4double thiPosX = 0;
G4double thiPosY = 0;
G4double thiPosZ = 0;

// only consider good in-beam events from here on
if(A15 and secDet and !inelasticFirstScatter)
  {
    if(numPoints > 2)
      {
        // Find good events (1st scatter detectable in-beam,
        //   2nd detectable out-of-beam)
        TrajectoryPoint* thiTrajPt = Traj->GetPoint(2);
        G4ThreeVector thiPos = thiTrajPt->GetPosition();
        thiPosX = thiPos.x();
        thiPosY = thiPos.y();
        thiPosZ = thiPos.z();
        fHisto->FillThiPosX(thiPosX);
        fHisto->FillThiPosY(thiPosY);
        fHisto->FillThiPosZ(thiPosZ);

        if(!EventAction::IsInBeam(thiTrajPt)){goodDoubleScatter=true;}
        if(goodDoubleScatter and
           EventAction::IsDetected(thiTrajPt, trackID,
                                   trajectoryContainer, aEnergy,
                                   aProb)){detGoodDoubleScatter=true;}
        // find tricksters, assume will detect any Inelastics
        if(goodDoubleScatter and !detGoodDoubleScatter)
          {
            if(numPoints>3)
              {
                for(G4int eePt = 3; eePt < numPoints; eePt++)
                  {
                    TrajectoryPoint* eeTrajPt = Traj->GetPoint(eePt);
                    if(EventAction::IsDetected(eeTrajPt, trackID,
                                               trajectoryContainer,
                                               aEnergy, aProb))
                      {
                        if(!EventAction::IsInBeam(eeTrajPt))
                          {trickster = true; break;}
                        else{trickster = false; break;}
                      }
                    // if not detected move on, if not detected on
                    //   last round, either inelastic and detected
                    //   next, or dies off, either way not trickster
                  }
              }
            // if numPoints <= 3, inelastic and so detected
          }
        // determine if the second scatter is maybe ranged out
        if(detGoodDoubleScatter)
          {if(thiPosX > 20*cm){outActive = true;}}  //placeholder
        // determine whether event would have been resolvable
        if(detGoodDoubleScatter)
          {
            if(EventAction::Distance(secPos, thiPos) > 5*cm)
              {resolvedGoodDoubleScatter=true;}
          }
```

```
      }
    else if(secTrajPt->GetProcessName() == "neutronInelastic")
      {
        G4int n_traj = trajectoryContainer->entries();
        for(int ddTraj = 0; ddTraj< n_traj; ddTraj++)
          {
// Make a Trajectory to zero in on the daughter
Trajectory* dTraj = (Trajectory*)((*(trajectoryContainer))[ddTraj]);
G4int dParentID = dTraj->GetParentID();
G4String dParticleName = dTraj->GetParticleName();
if(dParentID == trackID and dParticleName == "neutron")
  {
    TrajectoryPoint* dIniPt = dTraj->GetPoint(1);
    G4ThreeVector thiPos = dIniPt->GetPosition();
    thiPosX = thiPos.x();
    thiPosY = thiPos.y();
    thiPosZ = thiPos.z();
    fHisto->FillThiPosX(thiPosX);
    fHisto->FillThiPosY(thiPosY);
    fHisto->FillThiPosZ(thiPosZ);

    if(!EventAction::IsInBeam(dIniPt)){goodDoubleScatter=true;}
    G4int dTrackID = dTraj->GetTrackID();
    if(goodDoubleScatter and
          EventAction::IsDetected(dIniPt, dTrackID,
                                  trajectoryContainer, aEnergy,
                                  aProb)){detGoodDoubleScatter=true;}

    dTraj->SetNumPoints();
    G4int dNumPoints = dTraj->GetNumPoints();
    // find tricksters, assume will detect any Inelastics
    if(goodFirstScatter and !detGoodDoubleScatter)
      {
        for(G4int eePt = 2; eePt < dNumPoints; eePt++)
          {
            TrajectoryPoint* eeTrajPt = dTraj->GetPoint(eePt);
            if(EventAction::IsDetected(eeTrajPt, trackID,
                                  trajectoryContainer,
                                  aEnergy, aProb))
              {
                if(!EventAction::IsInBeam(eeTrajPt))
                  {trickster = true; break;}
                else{trickster = false; break;}
              }
            // if not detected move on, if not detected on
            //   last round, either inelastic and detected
            //   next, or dies off, either way not trickster
          }

        // determine if the second scatter is maybe ranged out
        if(detGoodDoubleScatter)
          {if(thiPosX > 20*cm){outActive = true;}}  //placeholder
        // determine whether event would have been resolvable
        if(detGoodDoubleScatter)
          {
            if(EventAction::Distance(secPos, thiPos) > 5*cm)
              {resolvedGoodDoubleScatter=true;}
          }
```

```
        }
    }
break;
          }
        }
      else
        {
          G4cout << "Only One Scatter?!" << G4endl;
        }
    }

//check to see if only two scatters were detected
G4int numDetected = 0;
for(G4int ffTraj = 0; ffTraj < n_trajectories; ffTraj++)
  {
    Trajectory* fTraj=(Trajectory*)((*(trajectoryContainer))[ffTraj]);
    if(fTraj->GetParticleName() == "neutron")
      {
        G4int fTrackID = fTraj->GetTrackID();
        fTraj->SetNumPoints();
        G4int fNumPoints = fTraj->GetNumPoints();
        if(fNumPoints > 1)
          {
            for(G4int hhPt = 1; hhPt < fNumPoints; hhPt++)
              {
                TrajectoryPoint* hTrajPt = fTraj->GetPoint(hhPt);
                if(EventAction::IsDetected(hTrajPt, fTrackID,
                                           trajectoryContainer,
                                           aEnergy, aProb))
                  {
                    numDetected++;
                  }
              }
          }
      }
  }
fHisto->FillNumScatters(numDetected);

if(resolvedGoodDoubleScatter and !outActive
   and !inelasticFirstScatter and numDetected==2){useableEvent=true;}

if(goodDoubleScatter){fHisto->FillCounter(5);}
if(detGoodDoubleScatter){fHisto->FillCounter(6);}
if(resolvedGoodDoubleScatter){fHisto->FillCounter(7);}
if(useableEvent){fHisto->FillCounter(8);}
if(trickster){fHisto->FillCounter(9);}
if(outActive){fHisto->FillCounter(10);}
if(inelasticFirstScatter){fHisto->FillCounter(11);}
if(inelasticFirstScatter and A15){fHisto->FillCounter(20);}

if(numDetected==2)
  {
    if(goodDoubleScatter){fHisto->FillCounter(12);}
    if(detGoodDoubleScatter){fHisto->FillCounter(13);}
    if(resolvedGoodDoubleScatter){fHisto->FillCounter(14);}
    if(useableEvent){fHisto->FillCounter(15);}
    if(trickster){fHisto->FillCounter(16);}
    if(outActive){fHisto->FillCounter(17);}
```

```
            if(inelasticFirstScatter){fHisto->FillCounter(18);}
            if(goodFirstScatter){fHisto->FillCounter(19);}
            if(fooler){fHisto->FillCounter(21);}
            if(A15){fHisto->FillCounter(22);}
            if(A15 and inelasticFirstScatter){fHisto->FillCounter(23);}
            if(A15 and secDet){fHisto->FillCounter(24);}
          }

          // plot the R and Z positions for the pristine events
        if(useableEvent)
          {
            G4double secPosR = std::sqrt(secPosY*secPosY + secPosX*secPosX);
            G4double thiPosR = std::sqrt(thiPosX*thiPosX + thiPosY*thiPosY);
            fHisto->FillPristineSecPosR(secPosR);
            fHisto->FillPristineSecPosZ(secPosZ);
            fHisto->FillPristineThiPosR(thiPosR);
            fHisto->FillPristineThiPosZ(thiPosZ);
          }



          // Where does the first particle die?
          TrajectoryPoint* fnlTrajPt = Traj->GetPoint(numPoints-1);
          G4double fnlPosX = fnlTrajPt->GetPosition().x();
          G4double fnlPosY = fnlTrajPt->GetPosition().y();
          G4double fnlPosZ = fnlTrajPt->GetPosition().z();
          fHisto->FillFinalPosX(fnlPosX);
          fHisto->FillFinalPosY(fnlPosY);
          fHisto->FillFinalPosZ(fnlPosZ);

        }
      }

    }



}

// Determines the distance between two 3vectors
G4double EventAction::Distance(G4ThreeVector V1, G4ThreeVector V2)
{
  return std::sqrt((V1.x() - V2.x())*(V1.x() - V2.x()) +
                   (V1.y() - V2.y())*(V1.y() - V2.y()) +
                   (V1.z() - V2.z())*(V1.z() - V2.z()));
}

// Rolls the dice to see whether an interaction with a given energy could be detected
G4bool EventAction::Detected(std::vector<double> aEnergy, std::vector<double> aProb, G4double
{
  G4double energy = rawenergy/keV;
  if(energy <= aEnergy[0]){return false;}
  else if(energy >= aEnergy[-1]){return true;}
  else
    {
      G4int nEntries = aEnergy.size();
      for(G4int aaEntry = 1; aaEntry < nEntries; aaEntry++)
        {
```

```
            //interpolate to get the probability of acceptance for this energy
            if(energy < aEnergy[aaEntry])
              {
                G4double prob = ((aProb[aaEntry] - aProb[aaEntry-1])/
                                 (aEnergy[aaEntry] - aEnergy[aaEntry-1])) *
                                (energy - aEnergy[aaEntry-1]) +
                                aProb[aaEntry-1];
                G4double randomNum = G4UniformRand();
                if(randomNum < prob){return true;}
                else{return false;}
              }
          }
        G4cout << "-------------------------------------------------------" << G4endl;
        G4cout << "Turn Back, Traveller.  You should not venture here!" << G4endl;
        G4cout << "-------------------------------------------------------" << G4endl;
        return true;
      }
}

// Determines whether an interaction is geometrically located in acceptance region
G4bool EventAction::IsInBeam(TrajectoryPoint* TrajPt)
{
  G4ThreeVector pos = TrajPt->GetPosition();
  G4double posX = pos.x();
  G4double posY = pos.y();
  G4double posZ = pos.z();

  if(posX*posX + posY*posY < 6.0025*cm*cm and posZ > 15*cm and posZ < 50*cm)
    {return true;}
  else{return false;}
}

// Determines interaction's energy then uses "Detected" to determine if an event
//   would be detected in LUX
G4bool EventAction::IsDetected(TrajectoryPoint* TrajPt, G4int TrackID,
                               G4TrajectoryContainer* trajCont, std::vector<double> aEnergy,
                               std::vector<double> aProb)
{
  // find the energy deposited.  Trivial in the case of non-inelastic scattering.
  G4double preE = TrajPt->GetPreKineticEnergy();
  G4double postE = 0;  //initializing
  // look for the daughter neutron if inelastic

  try{ TrajPt->GetProcessName();
  } catch (const std::length_error& e) {
    G4cout << "caught exception" << G4endl;
    return false;
  }

  if(TrajPt->GetProcessName() == "neutronInelastic")
    {
      G4int n_trajectories = trajCont->entries();
      for(int aaTraj = 0; aaTraj< n_trajectories; aaTraj++)
        {
          // Make a Trajectory to zero in on the daughter
          Trajectory* dTraj = (Trajectory*)((*(trajCont))[aaTraj]);
          G4int dParentID = dTraj->GetParentID();
          G4String dParticleName = dTraj->GetParticleName();
```

```
            if(dParentID == TrackID and dParticleName == "neutron")
              {postE = dTraj->GetKinEnergy();}   // gets kin energy when initialized, same as pt
          }
      }
    else{postE = TrajPt->GetPostKineticEnergy();}
    G4double depE = preE - postE;

    G4ThreeVector pos = TrajPt->GetPosition();
    G4double posX = pos.x();
    G4double posY = pos.y();
    G4double posZ = pos.z();
      if(-32.9*cm < posX and posX < 16.1*cm and
       posY*posY + (posZ-25*cm)*(posZ-25*cm) < 25*cm*25*cm)
      {return EventAction::Detected(aEnergy, aProb, depE);}
    else{return false;}
}
```

Appendix B: Second Scatter Positions

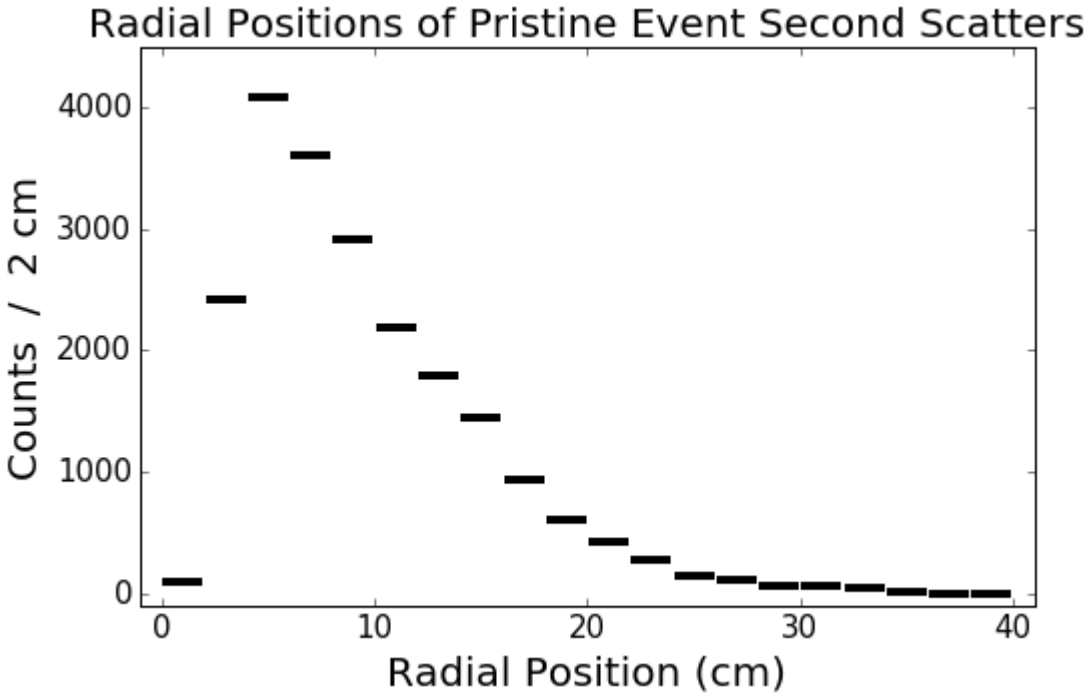

FIG. 10. This figure displays the radial positions (xy plane) of second scatters for all of the pristine events simulated. The peak value is 4085 out of a total 21253 pristine events.

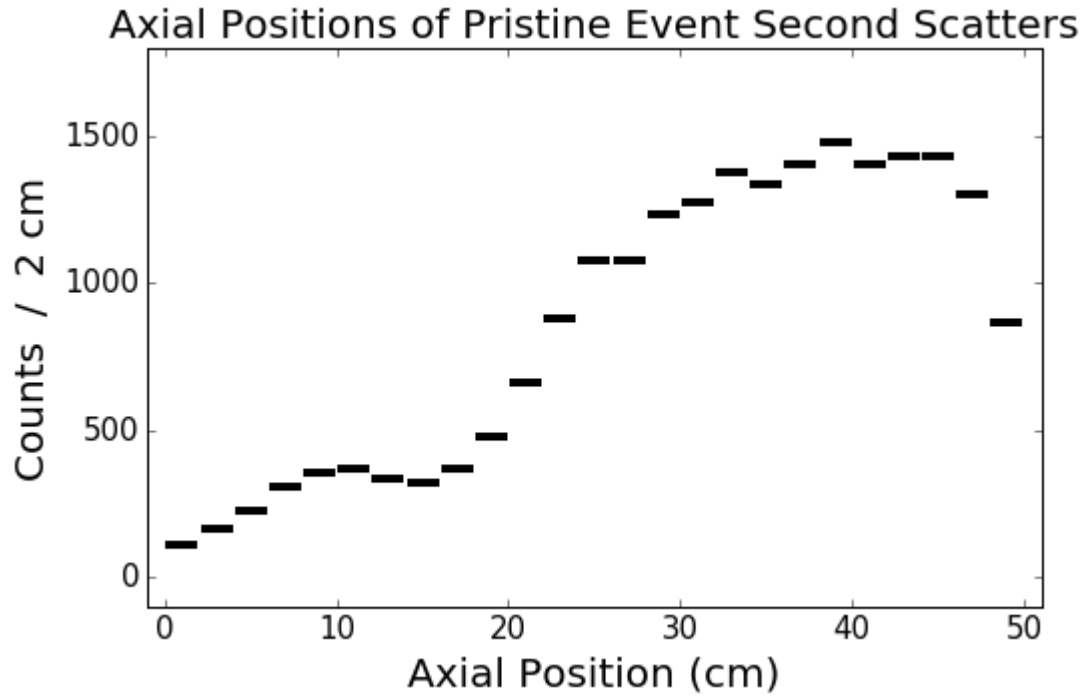

FIG. 11. This figure displays the axial positions (z direction) of second scatters for all of the pristine events simulated. The peak value is 1480 counts. Interestingly, there seems to be two peaks; perhaps corresponding to events that leave the active region more vertically out of the liquid surface (16.1 cm from the beam) and those that leave horizontally out of the sides (25 cm radius, distance from the beam varies.