2020

Effects of Energy Resolution and Bin Size on the Measurements of Nuclear Neutron Distributions Using Coherent Elastic Neutrino-nucleus Scattering

Hongyong Zhang
Colby College

Effects of Energy Resolution and Bin Size on the Measurements of Nuclear Neutron Distributions Using Coherent Elastic Neutrino-nucleus Scattering

Hongyong Zhang

Department of Physics and Astronomy
Colby College
May, 2020

Abstract

Coherent elastic neutrino-nucleus scattering(CEνNS) is a process that involves the neutral-current scattering of a neutrino with an entire nucleus, which probes more bulk properties of the nucleus. Using the Taylor expansion of the form factors and calculation of effective moments that sums over isotopes, we successfully simulate the events for coherent elastic neutrino-nucleus scattering for detectors using Ar and Ge and demonstrate that CEνNS can determine both the second moment and the fourth moment. Running the Monte Carlo simulation using a detector filled with 10 tonnes of natural germanium, a neutrino flux of 3×10^7 neutrinos per second per cm^2, and a bin size of 5 keV, we demonstrate that the neutron radii can be determined with less than 1% uncertainty if the detector efficiency remains constant and that a lower energy threshold can lead to a more accurate result.

Acknowledgements

I would like to thank Professor Kelly Patton for her ongoing guidance and assistance on this research project throughout the entire academic year, especially during this difficult time period of the COVID-19 pandemic. Furthermore, I would like to thank the Department of Physics and Astronomy at Colby College for this precious opportunity to conduct research and draft the honors thesis for my senior year. Last, I would like to thank my friends and my parents for their encouragement and support that help me stay strong and positive under the COVID-19 situation and make this thesis a reality.

Contents

1 Introduction **2**

1.1 Energy Density Functional Theory (DFT) 2

1.2 Neutrino-Nucleus Coherent Scattering 3

1.3 Applications in Nuclear Physics and Astrophysics 5

1.4 Detectors 5

2 Methodology **10**

2.1 Calculation of the Form Factor and the Number of Scattering Events 10

2.2 Form Factor 13

2.3 Form Factor Expansion 14

2.4 Effective Moments 16

3 Results and Discussion **24**

3.1 Binned Scattering Events 24

3.2 Monte-Carlo Simulations 24

3.3 Discussion 27

3.4 Future Work 30

4 Conclusion **31**

A Python Code for the Simulations **34**

1 Introduction

The neutron radius and the proton radius are two of the most fundamental and important properties of a nucleus. Since protons carry positive charges, the proton charge radii can be obtained with accuracy through electron scattering. However, the uncertainties in calculations and the discrepancies between different theoretical fit models make it hard for us to determine the neutron radii for nuclei with medium to heavy weight.[1] Since the neutron and proton radii are related to the neutron and proton density distributions in nuclei, we can obtain the neutron radii with confidence by measuring the neutron density distributions accurately. Determining neutron radii with precision can generate significant impacts on many related fields, such as astrophysics and experimental particle physics, and provide inspirations for the development of new physics.

1.1 Energy Density Functional Theory (DFT)

The most common method for determining neutron densities is the energy density functional theory (DFT).[2] In the Skyrme Hartree-Fock model, the total binding energy is given by

$$E = E_{kin} + \int d^3r \epsilon_{Sk} + E_{Coul} + E_{pair} - E_{corr}, \tag{1}$$

which is the sum of the kinetic energy, the Skyrme energy functional that includes the interactions between nucleons, the Coulomb energy, the pairing energy, and the energy corrections. [3] The Skyrme energy functional is given by

$$\epsilon_{Sk} = \sum_{T=0,1} (\epsilon_T^{even} + \epsilon_T^{odd}), \tag{2}$$

where the odd part contains the time-odd densities and the even part contains the time-even densities that sum over the isospin T. Ref. [3] contains a detailed description of the mean-field models for nuclear structures. By fitting the nuclear observables obtained from experiments to the various parameters in the model, we can predict the neutron density distribution inside a nucleus.

One interesting observable is the neutron halos for light nuclei with abundant neutrons, which is caused by the weak binding of the outermost neutrons.[3] Although

there are vast disagreements in the parametrization of the halos, we can perform a robust measurement by implementing the Skyrme interactions and measuring parameters such as the root-mean-squared radius, the diffraction radius, and the surface thickness.[3] Another interesting observable is the neutron skin, which can help us determine the difference between neutron and proton radii by determining the relation between the neutron skin and the isovector forces.[3] The asymmetry energy from the interactions predicted by the Skyrme Hartee-Fock model forms a special relation with the neutron skin with higher asymmetry energy producing a larger skin. Therefore, the Skyrme Hartee-Fock model can help predict neutron radii and accurate measurements of neutron radii can help revise the model and improve its predictions.

1.2 Neutrino-Nucleus Coherent Scattering

Coherent elastic neutrino-nucleus scattering(CEνNS) is a process that involves the neutral-current scattering of a neutrino with an entire nucleus, which probes more bulk properties of the nucleus. The idea of probing the neutron form factor by measuring the neutrino-nucleus coherent scattering events to determine neutron density distributions is first proposed in Ref. [1] using a detector filled with one tonne of ^{40}Ar. Traditionally, the neutrino-nucleus coherent elastic scattering events have been difficult to detect since the recoil energies are very small. The recoil energies are mostly in the sub-MeV range, which is below the threshold for most neutrino detectors. However, the recent developments of the ultra low threshold detectors enable the detection of weakly interacting massive particles and low energy solar neutrinos with thresholds of 10 keV or even lower. [4] We will introduce some of the different types of these detectors afterwards. These detectors allow us to measure the counts for the neutrino-nucleus coherent scattering and determine the neutron form factors. Apart from the neutron form factors which are our main focus, this section will continue to introduce briefly the other aspects of neutrinos and nuclear physics behind the coherent scattering that can be explored using CEνNS.

The rate of the neutrino-nucleus coherent elastic scattering is predicted by the standard model, which is proportional to the weak charge Q_W. The weak charge is defined as

$$Q_W = N - (1 - 4\sin^2\theta_W)Z \tag{3}$$

where N is the number of neutrons, Z is the number of protons, and θ_W is the weak

mixing angle. If we can measure the CEνNS events under a 10% uncertainty, then we can extract the mixing angle within a 5% uncertainty. This provides a new method to determine the weak mixing angle if we can measure the scattering events accurately.

The non-standard interactions of neutrinos also impact the cross section. We can assume an effective Lagrangian for interactions between neutrinos and hadrons as[4]

$$\begin{aligned} L_{\nu H}^{NSI} = &- \frac{G_F}{\sqrt{2}} \sum_{\substack{q=u,d,\alpha \\ \beta=e,\mu,\tau}} [\bar{\nu}_\alpha \gamma^\mu (1-\gamma^5)\nu_\beta] \\ &\times (\epsilon_{\alpha\beta}^{qL}[\bar{q}\gamma_\mu(1-\gamma^5)q] + \epsilon_\alpha^{qR}\beta[\bar{q}\gamma_\mu(1+\gamma^5 q)]). \end{aligned} \tag{4}$$

The cross section for coherent elastic scattering of neutrinos of flavor α is [4]

$$\begin{aligned} \frac{d\sigma}{dE}_{\nu\alpha A} = &\frac{G_F^2}{\pi} F^2(2ME)\left[1 - \frac{ME}{2k^2}\right] \times \\ &\{[Z(g_V^p + 2\epsilon_{\alpha\alpha}^{\mu V} + \epsilon_{\alpha\alpha}^{dV}) + N(g_V^n + \epsilon_{\alpha\alpha}^{\mu V} + 2\epsilon_{\alpha\alpha}^{dV})]^2 \\ &+ \sum_{\alpha\neq\beta}[Z(2\epsilon_{\alpha\beta}^{\mu V} + \epsilon_{\alpha\beta}^{dV}) + N(\epsilon_{\alpha\beta}^{\mu V} + 2\epsilon_{\alpha\beta}^{dV})]^2\}, \end{aligned} \tag{5}$$

where $g_V^p = (\frac{1}{2} - 2\sin^2\theta_W)$ and $g_V^n = -\frac{1}{2}$ are the standard model weak constants. For a neutrino source that uses pion decay at rest, it contains ν_μ, $\bar{\nu}_\mu$, and ν_e. A coherent elastic scattering experiment can generate very small sensitivity to the ϵ_{ee}^{qV} and $\epsilon_{e\tau}^{qV}$ parameters that are currently allowed. [4]

The standard model also predicts that there is a neutrino magnetic moment, $\mu_\nu \leq 10^{-19}\mu_B(m_\nu/1\text{eV})$, which is very small compared to other limits that the standard model predicts. For example, the limit set from the lack of observed energy loss in red giant revolution's electromagnetic couplings is $\mu_\nu \leq 10^{-12}\mu_B$.[4] This small but non-zero magnetic moment can change the results of the coherent elastic scattering at low energies. The magnetic scattering cross section is given by[4]

$$\left(\frac{d\sigma}{dE}\right) = \frac{\pi\alpha^2\mu_\nu^2 Z^2}{m_e^2}\left(\frac{1-E/k}{E} + \frac{E}{4k^2}\right). \tag{6}$$

The change in the scattering events happen when neutrino magnetic moment is near the limit for ν_e for recoil energies less than 10 keV, which may not be noticeable due to slightly higher thresholds for most detectors. For ν_μ, there could be a measurable difference if the detector has a 10 keV threshold. For nuclei with spin, they have

extra terms that could potentially change the cross sections and vary the scattering results.

The neutrino-nucleus coherent elastic scattering opens up possibilities for discovering new physics and improving some of the existing theories and models. Made possible by the invention of the low threshold detectors, the measurement of the scattering events has many applications, such as the detection of supernova neutrinos, reactor monitoring, measuring the Weinberg angle, and measuring form factors with precision which motivates this thesis.

1.3 Applications in Nuclear Physics and Astrophysics

A better understanding of neutron densities and neutron radii has many applications. One example is that obtaining a more accurate neutron density distribution can reduce uncertainties in atomic parity violation experiments.[1] Further, as mentioned previously, accurate measurements of neutron radii can improve the prediction of the energy functionals in different mean field theories significantly. Additionally, measuring neutron density distributions can help determine the neutron skin thickness, which has important implications in astrophysics.[2] Although the orbits and masses for neutron stars can be determined accurately, the radius of neutron stars and their moments of inertia can only be obtained through theories, which uses the equation of state for neutron-rich nuclear matter that relates to the density dependence of the nuclear symmetry energy. As mentioned in the first section, the symmetry energy is directly related to the neutron skin since neutron skins are sensitive to the symmetry energy at low densities and the neutron star radius is directly related to the symmetry energy at higher densities.[5] Therefore, measuring neutron density distributions provides information on the neutron skin and the equation of state of neutron matter, which helps determine the size of the neutron stars.

1.4 Detectors

There are many ongoing efforts around the world to conduct CEνNS experiments. This section presents an overview of the detectors, their corresponding technologies, specifications, and anticipated energy thresholds that different laboratories worldwide implement for the CEνNS experiments.

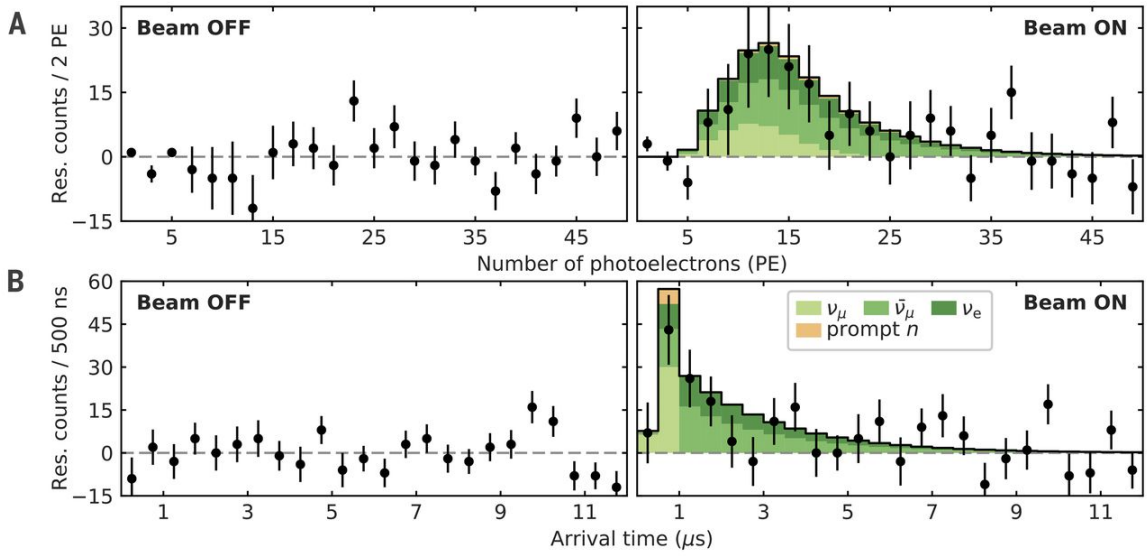

Figure 1: Observation of CEνNS events. Part A and B shows the residual differences between CsI signals 12 microseconds before and after the POT triggers. Approximately 1.17 photoelectrons are expected per keV of cesium or iodine nuclear recoil energy. Figure credit to Ref. [6]

The COHERENT experiment at the Spallation Neutron Source is the first and only experiment to successfully measure the CEνNS events. The results of the measurements are shown in Fig. 1. Approximately 1.17 photoelectrons are detected per keV of nuclear recoil energy.[6] The COHERENT experiment implements four different types of low-threshold/low-background technologies: CsI[Na] scintillating crystal, p-type point-contact germanium detectors, single-phase liquid argon, and NaI[Tl] scintillating crystals. The detailed parameters of the four subsystems is shown in Table 1. Fig. 2 shows the apparatus of the detector.[7]

Target	Technology	Mass (kg)	Distance from source (m)	Recoil threshold (keV)
CsI[Na]	Scintillating crystal	14.6	19.3	6.5
Ge	HPGe PPC	10	22	5
LAr	Single-phase	22	29	20
NaI[Tl]	Scintillating crystal	0.0925	28	13

Table 1: Parameters for the four subsystems of the COHERENT detector. Table credit to Ref. [7]

The NUCLEUS experiment at the Chooz nuclear power plant in France uses cryogenic detectors which are based on CRESST cryogenic detectors that have the world-leading sensitivity for searching low mass dark matter. The experiments have two phases: use a detector with 10g target for the first phase and use a detector with 1kg target for the second phase. The 10g target detector uses 6g of 3×3 array of $CaWO_4$ and 4g of 3×3 array of Al_2O_3. Fig. 3 shows the detailed apparatus of the 10g target detector. The NUCLEUS detector has an unprecedentedly low energy threshold $\leq$ 20eV and has the potential to have an eventual precision around 10% on the cross section of the coherent elastic scattering. The 1kg detector, planned to be ready for use in 2023, targets the measurement of the cross section with a 1% uncertainty. [8]

The CryoCube detector for RICOCHET aims to implement a compact array of 27 cubic 32g detectors using either germanium to build a semiconductor detector or use zinc to build a superconductor detector. The germanium detector allows a low energy threshold at around 10 eV while the zinc detector may achieve an extremely low threshold in the meV range. The detector also purposes to have a $\geq 10^3$ EM background rejection and 1 kg target mass.[9]

This thesis is organized in the following way. Section II will talk about the methodology that we uses to simulate and calculate the coherent scattering events, including cross section, expansion of the form factor, and calculation of the effective moment. Section III will discuss the results generated from the methods in section II and perform a Monte-Carlo simulation to analyze the data in detail.

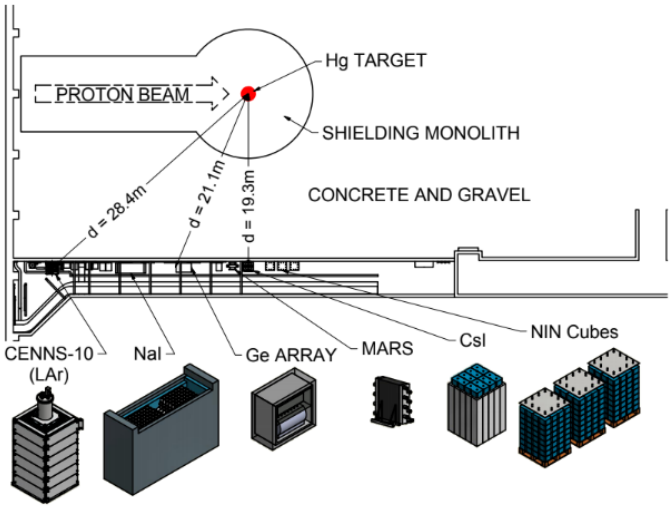

Figure 2: The setup of the four subsystems for the COHERENT detector. Figure credit to Ref. [7]

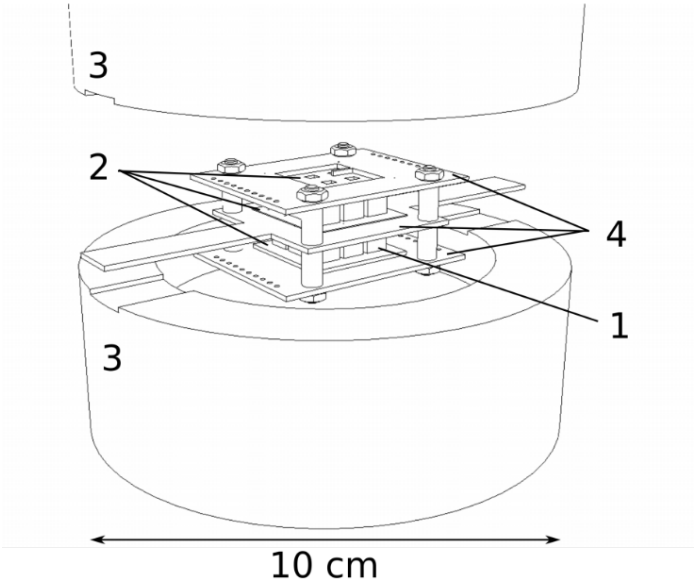

Figure 3: 3D sketch of the NUCLEUS-10g detector. it consists of three different kinds of cryogenic calorimeters - two 3×3 arrays of gram-scale cryogenic calorimeters as the target (1), an inner veto (2) and an outer veto (3) with a 10cm diameter. A non-instrumented support structure (4) is implemented to hold the assembly. Figure credit to Ref. [8]

2 Methodology

In this section, we present the detailed steps to simulate the neutrino-nucleus coherent scattering, including the calculation of the cross section, the use and expansion of the form factor, and the calculation of effective moments for atoms like germanium that exist in isotopes with different abundances.

2.1 Calculation of the Form Factor and the Number of Scattering Events

For the neutrino-nucleus coherent elastic scattering, the cross section for a spherical and spin zero nucleus is given by [1]

$$\frac{d\sigma}{dT}(E,T) = \frac{G_F^2}{2\pi}\left[2 - \frac{2T}{E} + \left(\frac{T}{E}\right)^2 - \frac{MT}{E^2}\right]\frac{Q_W^2}{4}F^2(Q^2), \tag{7}$$

where G_F is the Fermi constant, E is the energy of the incoming neutrino, T is the recoil energy of the nucleus, M is the mass of the nucleus, $Q_W = N - Z(1-4\sin^2\theta_W)$ is the weak charge of the nucleus ($\sin^2\theta_W \approx 0.231$). The equation for the cross section contains the squared momentum transfer, $Q^2 = 2E^2TM/(E^2 - ET)$, and the form factor $F(Q^2)$ as a function of Q^2, containing information about the nuclear densities by a Fourier transform.

The form factor for a spherical nucleus is given by [1]

$$F(Q^2) = \frac{1}{Q_W}\int[\rho_n(r) - (1-4\sin^2\theta_W)\rho_p(r)]\frac{\sin Q_r}{Q_r}r^2dr, \tag{8}$$

where ρ_n and ρ_p are the neutron and proton densities respectively. The form factor modifies the non-coherent scattering at higher energies. Some effects such as the ones because of the finite size of the nucleon have been neglected, which changes the form factor at high moment transfer and modifies the neutron density. However, based on the sensitivity of our simulation, the differences that these effects cause are insignificant.

We can separate the neutron and proton parts in the form factor and obtain the following

$$F(Q^2) = \frac{1}{Q_W}\int[F_n(Q^2) - (1-4\sin^2\theta_W)F_p(Q^2)]. \tag{9}$$

Since $1-4\sin^2\theta_W \approx 0.076$, the contribution to the overall form factor from the proton term is quite small. Therefore, the scattering mainly depends on the neutrons and our simulations focus primarily on the neutron parts.

There are primarily two types of neutrino in our simulations, the ones from fission inside the nuclear reactors and the ones from decaying stopped pions. Since the reactor neutrinos have lower energies and the recoil energies generated by these neutrinos are hardly distinguishable from the background, we only look at the neutrinos from the pion decay. The pion decays through $\pi^+ \rightarrow \nu_\mu + \mu^+$. The muons can further decay and generate antineutrinos by $\mu^+ \rightarrow e^+ + \nu_e + \bar{\nu}_\mu$. We neglect the initial pion decay and only focus on the $\bar{\nu}_\mu$ from μ^+. The normalized spectra for the neutrinos ν_e and antineutrinos $\bar{\nu}_\mu$ are [2]

$$\begin{aligned} f_{\nu_e} &= \frac{96}{m_\mu^4}(m_\mu - E_{\nu_e}^2 - 2E_{\nu_e}^3)dE_{\nu_e}, \\ f_{\bar{\nu}_\mu} &= \frac{16}{m_\mu^4}(3m_\mu E_{\bar{\nu}_\mu}^2 - 4E_{\bar{\nu}_\mu}^3)dE_{\bar{\nu}_\mu}, \end{aligned} \tag{10}$$

where m_μ is the muon mass. These two equations give the probability density of finding an emitted neutrino between energy E and $E + dE$. The energy spectrum of the two fluxes is shown below in Fig. 4.

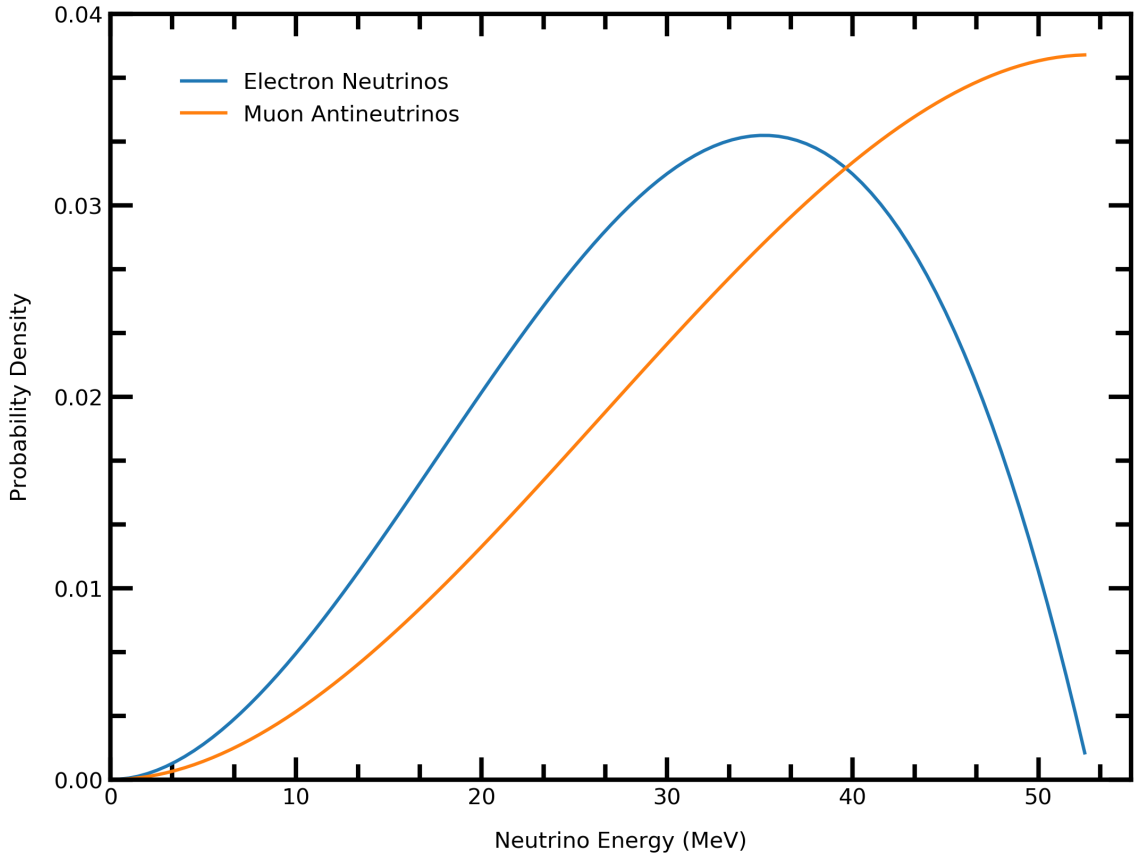

Figure 4: The probability density of finding electron neutrinos and muon antineutrinos at different neutrino energies.

Combining the cross section with the neutrino flux, we can calculate the number of scattering events based on the recoil energy: [2]

$$\frac{dN}{dT}(T) = N_t C \int_{E_{min}(T)}^{m_\mu/2} f(E) \frac{d\sigma}{dT}(E,T) dE, \tag{11}$$

where N_t is the number of targets in the detector, C is the total number of neutrinos of a specific flavor approaching the target per second per cm squared, $E_{min}(T) = \frac{1}{2}(T + \sqrt{T^2 + 2TM})$ is the minimum energy required for a neutrino to generate the recoil energy T at the nucleus, $m_\mu/2$ is the maximum energy from a neutrino, and $f(E)$ is the sum of the fluxes for all types of neutrinos produced.

2.2 Form Factor

Using the analytic method from Ref.[10], we can represent form factors using the analytic approximations of the density distributions instead of nuclear structure calculations, which simulates the nucleons with a thickness s on the surface and a constant interior density. The skin thickness s represents how fast the density falls off from the interior to zero. We obtain the form factors by the Fourier transform of the density distributions, also called the Helm form factor: [10]

$$F(Q^2) = \frac{3j_1(QR_0)}{QR_0}\exp[-\frac{1}{2}(Qs)^2], \tag{12}$$

where j_1 represents the first spherical Bessel function, $R_0^2 = R^2 - 5s^2$ is around 3 fm to 5 fm, R is the radius of the nucleus, and s is the thickness of the surface. Fig. 5 shows the relationship between the Helm form factor and the momentum transfer Q with different possible R_0. Using Eq. (12) to calculate the form factors for protons and neutrons, we can get the full form factor by incorporating the two parts into Eq. (9).

For detectors filled with ^{40}Ar where $N = 22$ and $Z = 18$, we use the data from Ref.[11], which gives the value for the mean square radius of the proton, $\langle R_p^2 \approx 11.75 \text{ fm}\rangle$. We then take the square root of the value and get the proton radius, $R_p = 3.43$ fm. For the surface thickness, we set the same value, $s = 0.5$ fm.[1] Since the purpose of our simulation is to demonstrate the possibility of conducting such an experiment in reality, it is adequate to assign the values to the parameters to generate an analytic model for demonstration while specific calculations and values need to be made and determined for performing a real experiment. For a nucleus that has the same number of protons and neutrons, the proton and neutron density distribution should be roughly the same, which means that $R_p \approx R_n$. Therefore, for approximations, we can set $R_p = R_n$, which means that we can compute both $F_n(Q^2)$ and $F_p(Q^2)$ using Eq. (12). The calculated Helm form factor for argon is shown in Fig. 6. We can then use Eq. (9) to calculate the full form factor, which we can plug into Eq. (7) to compute the cross section. With the cross section calculated, we can combine it with the neutrinos flux from Eq. (10) to simulate the scattering events by integrating over Eq. (11) and plot it out as a base curve for reference, which is demonstrated in Fig. 7.

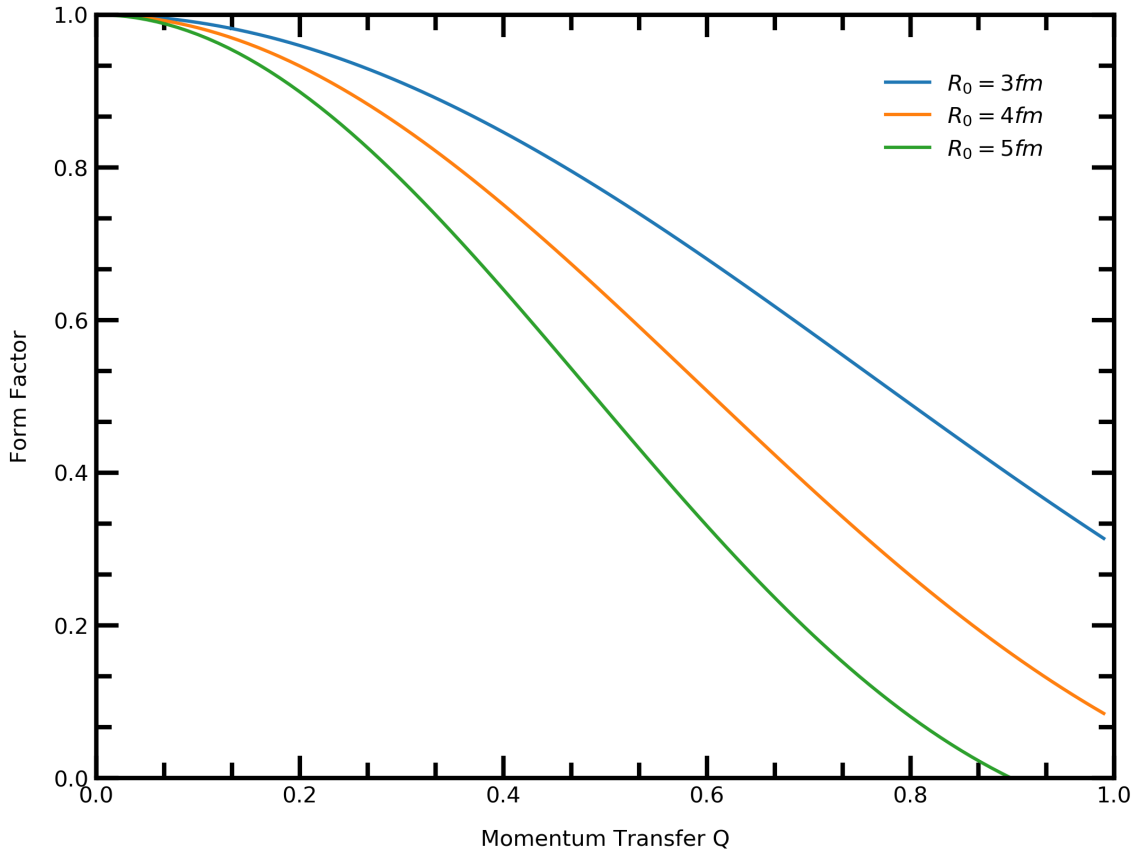

Figure 5: Helm form factor against momentum transfer Q at $R_0 = 3$ fm, $R_0 = 4$ fm, and $R_0 = 5$ fm. The variation in the R_0 value changes the shape of the curve for the form factor.

2.3 Form Factor Expansion

Since Q is very small, we can apply Taylor expansion to the form factor and truncate the series so that we can determine the form factor using only a few parameters that we can obtain from the nuclear structure calculations. The Taylor expansion of the neutron form factor is [2]

$$\begin{aligned} F_n(Q^2) &\approx \int \rho_n(r) \left(1 - \frac{Q^2}{3!}r^2 + \frac{Q^4}{5!}r^4 - \frac{Q^6}{7!}r^6 + \cdots\right) r^2 dr \\ &\approx N \left(1 - \frac{Q^2}{3!}\langle R_n^2\rangle + \frac{Q^4}{5!}\langle R_n^4\rangle + -\frac{Q^6}{7!}\langle R_n^6\rangle + \cdots\right) \end{aligned} \tag{13}$$

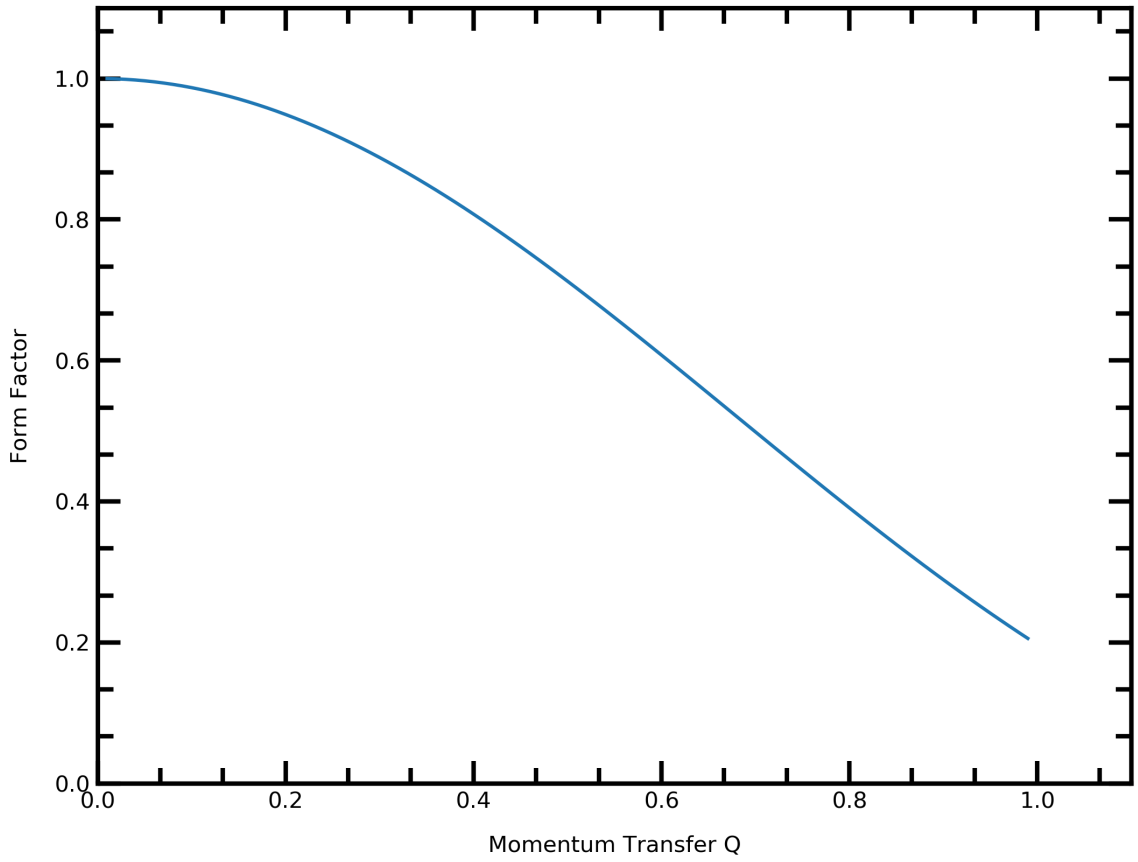

Figure 6: Helm form factor against momentum transfer Q for ^{40}Ar. We set $R_0 = 3.43$ for both proton and neutron full factors, $N = 22$, $Z = 18$, and $s = 0.5$ fm.

where

$$\langle R_n^k \rangle = \frac{\int \rho_n r^k d^3 r}{\int \rho_n d^3 r}. \tag{14}$$

Here we only focus on the neutron part of the form factor since the proton terms will barely change the result. After the expansion, the form factor consists only of the even moments of the neutron density distributions, which are easy to calculate and represent physical and measurable parameters.[2] Due to the low energy of the neutrinos, we can omit the terms after the first two terms for nuclei with lighter weights such as Ar and Ge and after the first three terms for heavier ones like Xe. Fig. 8 shows the comparison between the results obtained from the Helm form factor and the truncated form factor from Taylor expansion. The difference in the counts is noticeable but gradually becomes smaller at lower energies from 0 to 60 keV, and approaches 0 as the energy goes higher than 60 keV, suggesting that the truncated

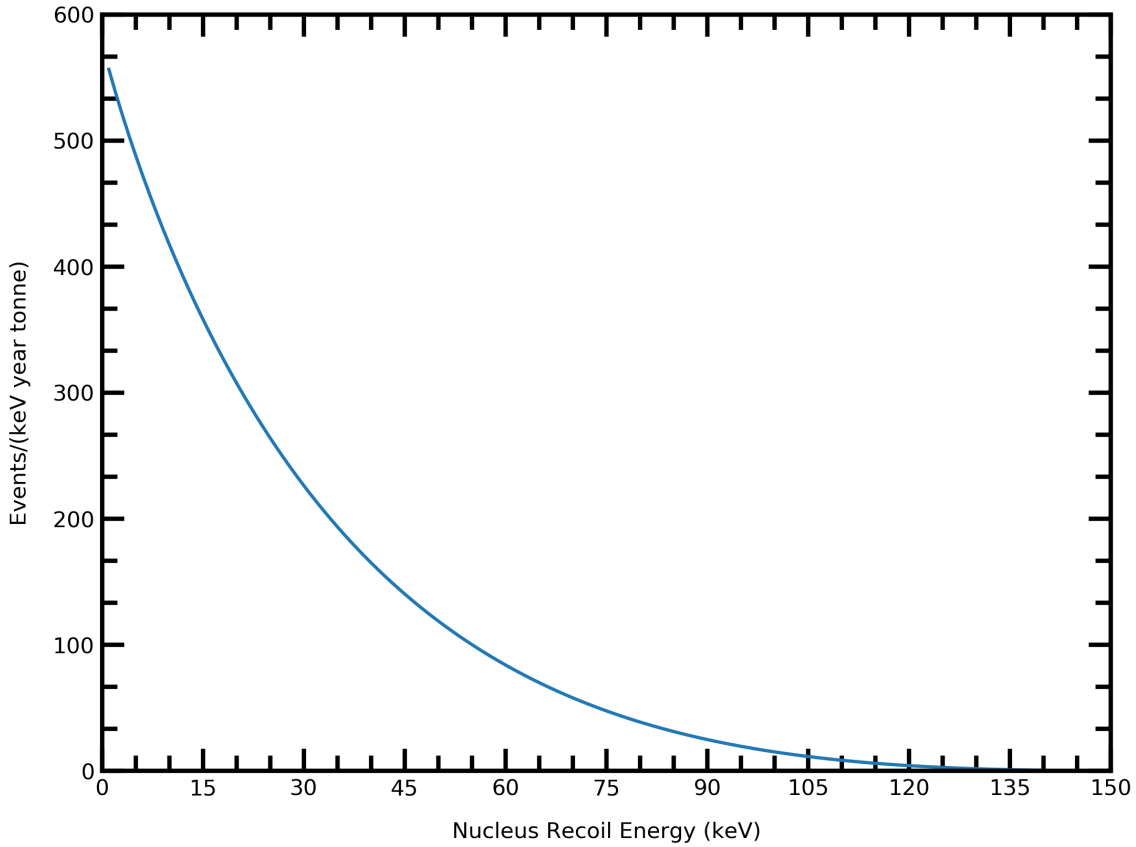

Figure 7: Number of ν_e and $\bar{\nu}_\mu$ events in detector filled with ^{40}Ar in units of number per (keV year tonne) assuming 10^7 neutrinos of the two flavors are emitted from the sources every second. Assume $R_n = R_p$.

series is an applicable implementation that preserves the accuracy of the result. Fig. 9 shows the results at different R values. The changes in R have enough impacts on the results to the curves distinguishable.

2.4 Effective Moments

When previously generating the base curve for Ar, we used ^{40}Ar to calculate the number of scattering event because 99.6% of Ar is constituted of ^{40}Ar. However, for elements such as germanium and xenon, they exist in multiple isotope forms with sufficient abundances that we need to define the effective moments in order to include the different isotopes. The isotopes and abundances for germanium and xenon are shown in Table 2.

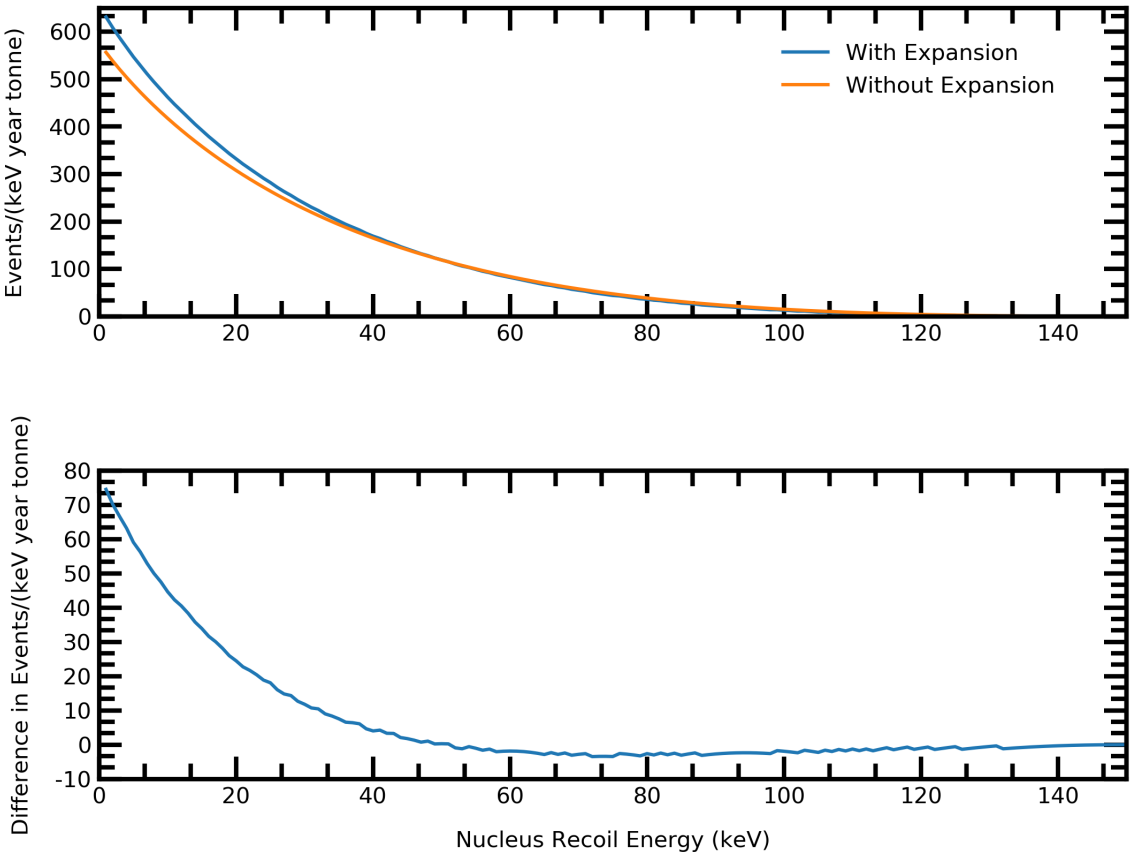

Figure 8: Number of ν_e and $\bar{\nu}_\mu$ events in detector in units of number per (keV year tonne) using Helm form factor and the first two terms of Taylor expansion of the form factor neglecting the proton part. The subplot below shows the difference in counts between the two methods.

To account for the isotopes, we need to sum over all the isotopes. The equation for calculating the scattering events then becomes: [2]

$$\frac{dN}{dT}(T) = N_A M_{detector} C \int f(E) \sum_i \left[\frac{X_i}{M_i} \left(\frac{d\sigma}{dT}(T, E) \right)_i \right] dE, \tag{15}$$

where $N_A M_{detector}$ substitutes the N_t in Eq. 11 and the summation adds up all the isotopes. X_i is the natural abundance of isotope i, M_i is the mass of the isotope, N_A is the Avogadro's number, and $M_{detector}$ is the total mass of the element in the detector.[2]

For simulating the scattering event for germanium, neglecting the proton terms and focusing only on the neutron terms, we can use the first two terms of the expansion

Eq. 13 and plug them into the cross section given by Eq. 7 for calculating the sum of the cross sections for all isotopes in Eq. 15. We then get:

$$\begin{aligned}\sum_i \left[\frac{X_i}{M_i}\left(\frac{d\sigma}{dT}(T,E)\right)_i\right] =& \frac{G_F^2}{8\pi}\Bigg[\sum_i \left(X_i N_i^2\right)\left(2-\frac{2T}{E}+\left(\frac{T}{E}\right)^2\right)-\sum_i \left(X_i N_i^2 M_i\right)\left(\frac{T}{E^2}\right)\\ &-\sum_i \left(X_i N_i^2 M_i \langle R_n^2\rangle_i\right)\left(2-\frac{2T}{E}+\left(\frac{T}{E}\right)^2\right)\frac{Q^2}{3\langle M\rangle}\\ &+\sum_i \left(X_i N_i^2 M_i^2 \langle R_n^2\rangle_i\right)\left(\frac{T}{E^2}\right)\frac{Q^2}{3\langle M\rangle}\\ &+\sum_i \left(X_i N_i^2 M_i^2 \langle R_n^2\rangle_i^2\right)\left(2-\frac{2T}{E}+\left(\frac{T}{E}\right)^2\right)\frac{Q^4}{36\langle M\rangle^2}\\ &-\sum_i \left(X_i N_i^2 M_i^3 \langle R_n^2\rangle_i^2\right)\left(\frac{T}{E^2}\right)\frac{Q^4}{36\langle M\rangle^2}\\ &+\sum_i \left(X_i N_i^2 M_i^2 \langle R_n^4\rangle_i\right)\left(2-\frac{2T}{E}+\left(\frac{T}{E}\right)^2\right)\frac{Q^4}{60\langle M\rangle^2}\\ &-\sum_i \left(X_i N_i^2 M_i^3 \langle R_n^4\rangle_i\right)\left(\frac{T}{E^2}\right)\frac{Q^4}{60\langle M\rangle^2}\\ &-\sum_i \left(X_i N_i^2 M_i^3 \langle R_n^2\rangle_i \langle R_n^4\rangle_i\right)\left(2-\frac{2T}{E}+\left(\frac{T}{E}\right)^2\right)\frac{Q^6}{360\langle M\rangle^3}\\ &+\sum_i \left(X_i N_i^2 M_i^4 \langle R_n^2\rangle_i \langle R_n^4\rangle_i\right)\left(\frac{T}{E^2}\right)\frac{Q^6}{360\langle M\rangle^3}+\cdots\Bigg],\end{aligned} \tag{16}$$

where $\langle M\rangle = \sum_i X_i M_i$ and $Q^2 = 2E^2T\langle M\rangle/(E^2 - ET)$. Here we are keeping the

Isotope	Abundance	Isotope	Abundance
^{70}Ge	0.205	^{128}Xe	0.0191
^{72}Ge	0.274	^{129}Xe	0.264
^{73}Ge	0.078	^{130}Xe	0.041
^{74}Ge	0.365	^{131}Xe	0.212
^{76}Ge	0.205	^{132}Xe	0.269
		^{134}Xe	0.104
		^{136}Xe	0.089

Table 2: Isotopes and abundances for germanium and xenon[12]

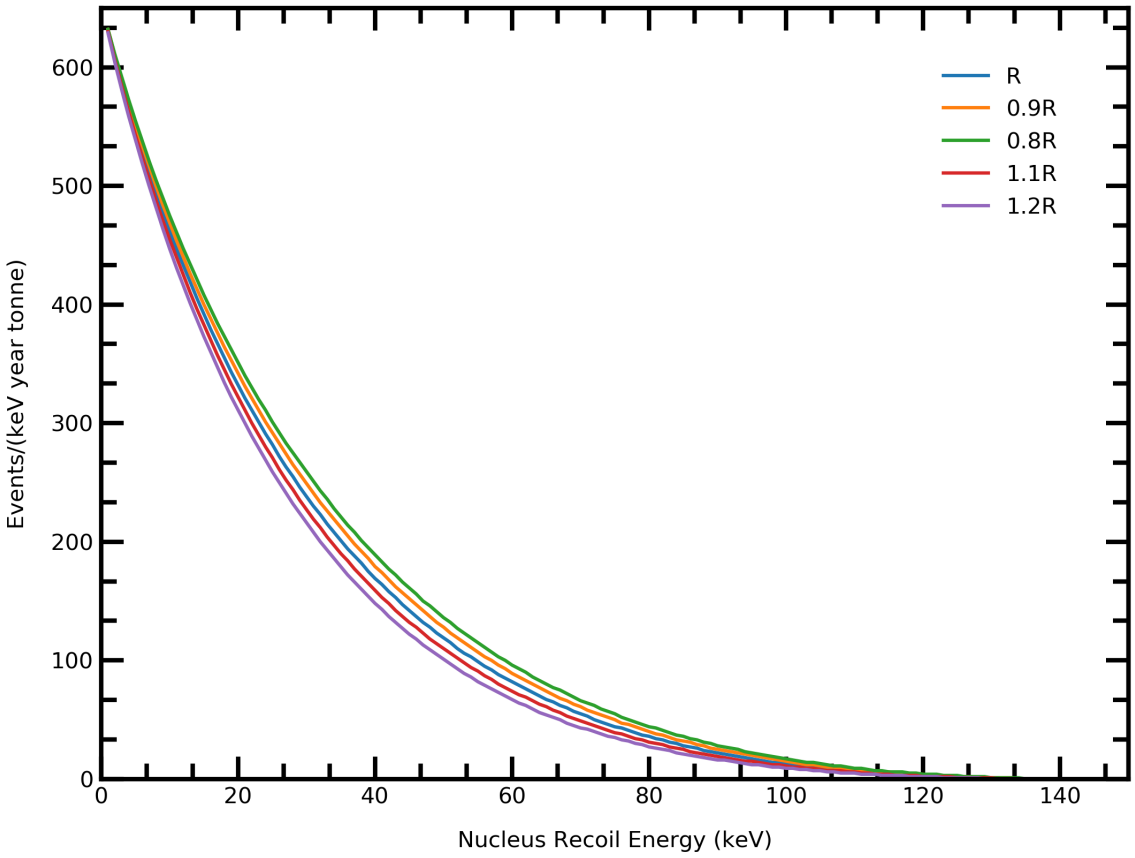

Figure 9: Number of ν_e and $\bar{\nu}_\mu$ events in detector in units of number per (keV year tonne) for ^{40}Ar at slightly different R values. The base values are $\langle R_n^2\rangle^{1/2} = 3.417$ fm and $\langle R_n^4\rangle^{1/4} = 3.723$ fm, adopted from the SkM* standard values.[14] The 10% increments and decrements only slightly vary the resulting curves.

$\langle R_n^2\rangle$ and $\langle R_n^4\rangle$ terms for germanium since we truncate the series for the neutron form factor after the first two terms that are consistent with the $\langle R_n^2\rangle$ and $\langle R_n^4\rangle$ terms. For heavier elements like xenon, terms up to $\langle R_n^6\rangle$ need to be preserved. The two effective second moments are [2]

$$\begin{aligned}\langle R_n^2\rangle_{eff,1} &= \frac{\sum_i X_i N_i^2 M_i \langle R_n^2\rangle_i}{\sum_i X_i N_i^2 M_i},\\ \langle R_n^2\rangle_{eff,2} &= \frac{\sum_i X_i N_i^2 M_i^2 \langle R_n^2\rangle_i}{\sum_i X_i N_i^2 M_i^2},\end{aligned} \tag{17}$$

which are derived from the third and fourth terms in Eq. 16, and the two effective

fourth moments are [2]

$$\begin{aligned}\langle R_n^4\rangle_{eff,1} &= \frac{\sum_i X_i N_i^2 M_i^2 \langle R_n^4\rangle_i}{\sum_i X_i N_i^2 M_i^2},\\ \langle R_n^4\rangle_{eff,2} &= \frac{\sum_i X_i N_i^2 M_i^3 \langle R_n^4\rangle_i}{\sum_i X_i N_i^2 M_i^3},\end{aligned} \tag{18}$$

which are derived from the seventh and eighth terms in Eq. 16.

Since the differences between the two effective second moments and the two effective fourth moments are negligibly small, we can assume that the two equations give the same result. Plugging the effective moments equations into Eq. 16, we get the new form for the the cross section, which only differs the original form by 0.01%, [2]

$$\begin{aligned}\sum_i \left[\frac{X_i}{M_i}\left(\frac{d\sigma}{dT}(T,E)\right)_i\right] =& \frac{G_F^2}{8\pi}\Bigg[\sum_i \left(X_i N_i^2\right)\left(2-\frac{2T}{E}+\left(\frac{T}{E}\right)^2\right) - \sum_i \left(X_i N_i^2 M_i\right)\left(\frac{T}{E^2}\right)\\ &- \langle R_n^2\rangle_{eff}\sum_i \left(X_i N_i^2 M_i\right)\left(2-\frac{2T}{E}+\left(\frac{T}{E}\right)^2\right)\frac{Q^2}{3\langle M\rangle}\\ &+ \langle R_n^2\rangle_{eff}\sum_i \left(X_i N_i^2 M_i^2\right)\left(\frac{T}{E^2}\right)\frac{Q^2}{3\langle M\rangle}\\ &+ \langle R_n^2\rangle_{eff}^2\sum_i \left(X_i N_i^2 M_i^2\right)\left(2-\frac{2T}{E}+\left(\frac{T}{E}\right)^2\right)\frac{Q^4}{36\langle M\rangle^2}\\ &- \langle R_n^2\rangle_{eff}^2\sum_i \left(X_i N_i^2 M_i^3\right)\left(\frac{T}{E^2}\right)\frac{Q^4}{36\langle M\rangle^2}\\ &+ \langle R_n^4\rangle_{eff}\sum_i \left(X_i N_i^2 M_i^2\right)\left(2-\frac{2T}{E}+\left(\frac{T}{E}\right)^2\right)\frac{Q^4}{60\langle M\rangle^2}\\ &- \langle R_n^4\rangle_{eff}\sum_i \left(X_i N_i^2 M_i^3\right)\left(\frac{T}{E^2}\right)\frac{Q^4}{60\langle M\rangle^2}\\ &- \langle R_n^2\rangle_{eff}\langle R_n^4\rangle_{eff}\sum_i \left(X_i N_i^2 M_i^3\right)\left(2-\frac{2T}{E}+\left(\frac{T}{E}\right)^2\right)\frac{Q^6}{360\langle M\rangle^3}\\ &+ \langle R_n^2\rangle_{eff}\langle R_n^4\rangle_{eff}\sum_i \left(X_i N_i^2 M_i^4\right)\left(\frac{T}{E^2}\right)\frac{Q^6}{360\langle M\rangle^3}+\cdots\Bigg].\end{aligned} \tag{19}$$

Adopting the new expression of the cross section that sums over all isotopes from

Eq. 19, we can simulate and make predictions on the number of scattering events for detectors filled with germanium using the data shown in Table 3, as shown in Fig. 10.

Remember that when deriving the cross section for light elements with multiple isotopes, we neglect the proton terms since it has very small impact on our results. In order to make the result more accurate, we can add back the proton terms and the terms including both protons and neutrons. Remember that we previously separate the two parts of the full form factor using Eq. 9, which gives us

$$F^2(Q^2) = F_n^2(Q^2) - 2\alpha F_n(Q^2)F_p(Q^2) + \alpha^2 F_p^2(Q^2) \tag{20}$$

where $\alpha = 1 - 4\sin^2\theta_W$. Since the $F_p^2(Q^2)$ terms are the same as the $F_n^2(Q^2)$ terms shown in Eq. 19 with all the parameters relevant to neutrons changing to the ones with protons in them, we only need simple substitution of some constants to obtain the $F_p^2(Q^2)$ part. For the part with both protons and neutrons, we only need to add the cross term,

$$\begin{aligned} F_n(Q^2)F_p(Q^2) =& 1 - \frac{Q^2}{3!}\left(\langle R_n^2\rangle + \langle R_p^2\rangle\right) + \frac{Q^4}{5!}\left(\langle R_n^4\rangle + \langle R_p^4\rangle\right) + \frac{Q^4}{(3!)^2}\left(\langle R_n^2\rangle\langle R_p^2\rangle\right) \\ & - \frac{Q^6}{3!\cdot 5!}\left(\langle R_n^2\rangle\langle R_p^4\rangle + \langle R_n^4\rangle\langle R_p^2\rangle\right) + \frac{Q^8}{(5!)^2}\left(\langle R_n^4\rangle\langle R_p^4\rangle\right), \end{aligned} \tag{21}$$

which is obtained by multiplying the terms up to Q^4 in the neutron form factor with the ones from the proton form factor. The coefficient

$$\frac{N_i^2 X_i M_i^k}{\langle M\rangle^k} \tag{22}$$

then becomes

$$\frac{N_i Z_i X_i M_i^k}{\langle M\rangle^k} \tag{23}$$

for the cross term. Adding the proton term and the cross term, we can get the cross section with the full form factor. We can then simulate the scattering event using the new cross section and get a more accurate curve that contains slightly fewer events at each energy bin compared to the previous one. Fig. 10 shows the result and compares it with the one that we previously obtained using only the neutron form factor. The

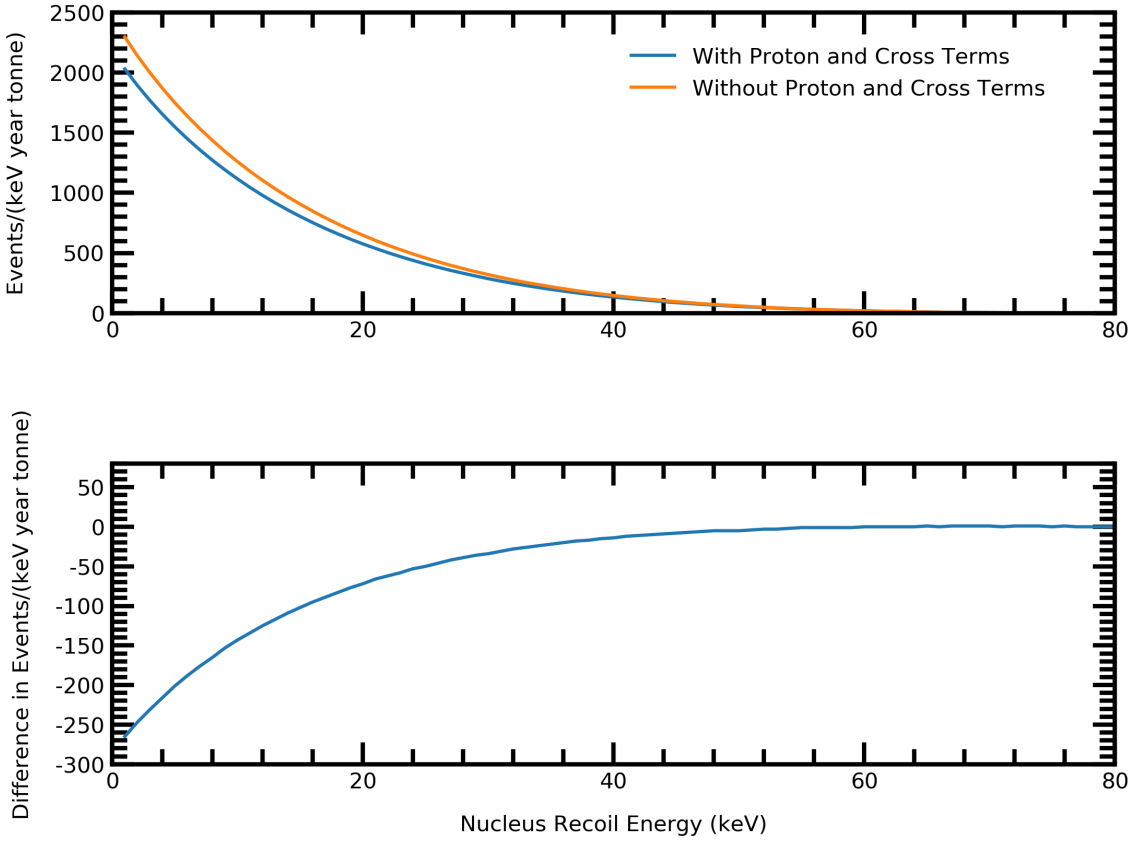

Figure 10: Number of ν_e and $\bar{\nu}_\mu$ events in detector in units of number per (keV year tonne) for Ge using only the neutron form factor and for adding the proton and cross terms. The subplot below shows the difference in results between the two methods.

difference is quite large at lower energies, but goes to zero as the energy increases to around 50 keV.

In order to predict different neutron distributions for different nuclear structure calculations and predict the number of scattering events expected in the detector, we need to vary the neutron radius. We vary the radius from -20% to $+20\%$ with an incremental step of 10% and create a database containing the scattering events at each radius. We then run a Monte-Carlo simulation for our data as an analysis, which we will discuss in the next section.

Atomic Mass (amu)	N	Abundance	$\langle R_p^2 \rangle$ (fm^2)	$\langle R_p^4 \rangle$ (fm^4)	$\langle R_n^2 \rangle$ (fm^2)	$\langle R_n^4 \rangle$ (fm^4)
69.924	38	0.206	15.428	323.689	15.832	343.030
71.922	40	0.275	15.895	344.108	16.568	377.481
72.923	41	0.078	15.665	330.939	16.457	368.242
73.921	42	0.365	15.974	345.718	16.893	390.827
75.921	44	0.077	16.026	345.900	17.170	401.341

Table 3: Atomic mass, number of neutrons, abundance, $\langle R_p^2 \rangle$, $\langle R_p^4 \rangle$, $\langle R_n^2 \rangle$, and $\langle R_n^4 \rangle$ values for germanium isotopes. Table obatined from Ref. [13].

3 Results and Discussion

3.1 Binned Scattering Events

Instead of showing the counts at every keV, we present and store our results in a more realistic way. We add up the counts in every 5 keV range together and represent our scattering curves as binned bar plots. This is because most detectors do not have resolutions small enough to distinguish the difference in 1 keV. Most simulations similar to ours, such as the ones in Ref. [1] and Ref. [2], use 10 keV as the energy bin and resolution. Therefore, our implementation of the 5 keV energy bin helps determine whether a finer resolution can lead to a more satisfactory result. The detailed method using Python is shown in Appendix A. Examples of our binned events are shown in Fig. 11, Fig. 12, and Fig. 13. Comparing Fig. 11 and Fig. 12, we notice that Ar goes to a higher recoil energy than Ge since Ar has a lighter atomic mass. Ge has more counts overall because the cross section is proportional to N^2 and Ge contains more neutrons in the atom. In Fig. 12 and Fig. 13, we can again see the effect of the added proton and cross terms in the form factor since the scattering events at each bin decrease.

3.2 Monte-Carlo Simulations

With the simulations running correctly, we then perform a simple Monte Carlo simulation to test the accuracy of determining the nuclear moments via the scattering events. We assume that the detector is filled with 10 tonnes of natural germanium and experiences a flux from pion decay of 3×10^7 neutrinos per second per cm squared in each flavor for one year. According to Ref. [2], the neutrino production rate at multiple laboratories such as the Spallation Neutron Source at Oak Ridge National Laboratory, DAEδALUS, and the European Spallation Source varies from 1×10^{15} neutrinos per second per cm^2 to 3.5×10^{15} neutrinos per second per cm^2. Therefore, our simulation represents placing the detectors from 16 to 30 meters from the neutrino sources, depending on the setup at different laboratories.[2]

To set up our Monte Carlo simulation, we first create a database of the scattering events from 5 to 80 keV and binned with a 5 keV bin size. The lower cutoff at 5 keV is due to the detector's possible threshold. To generate a database with sufficient data, we vary the $\langle R_n^2 \rangle$ value from -15% to $+15\%$ and vary the $\langle R_n^4 \rangle$ value from -30% to

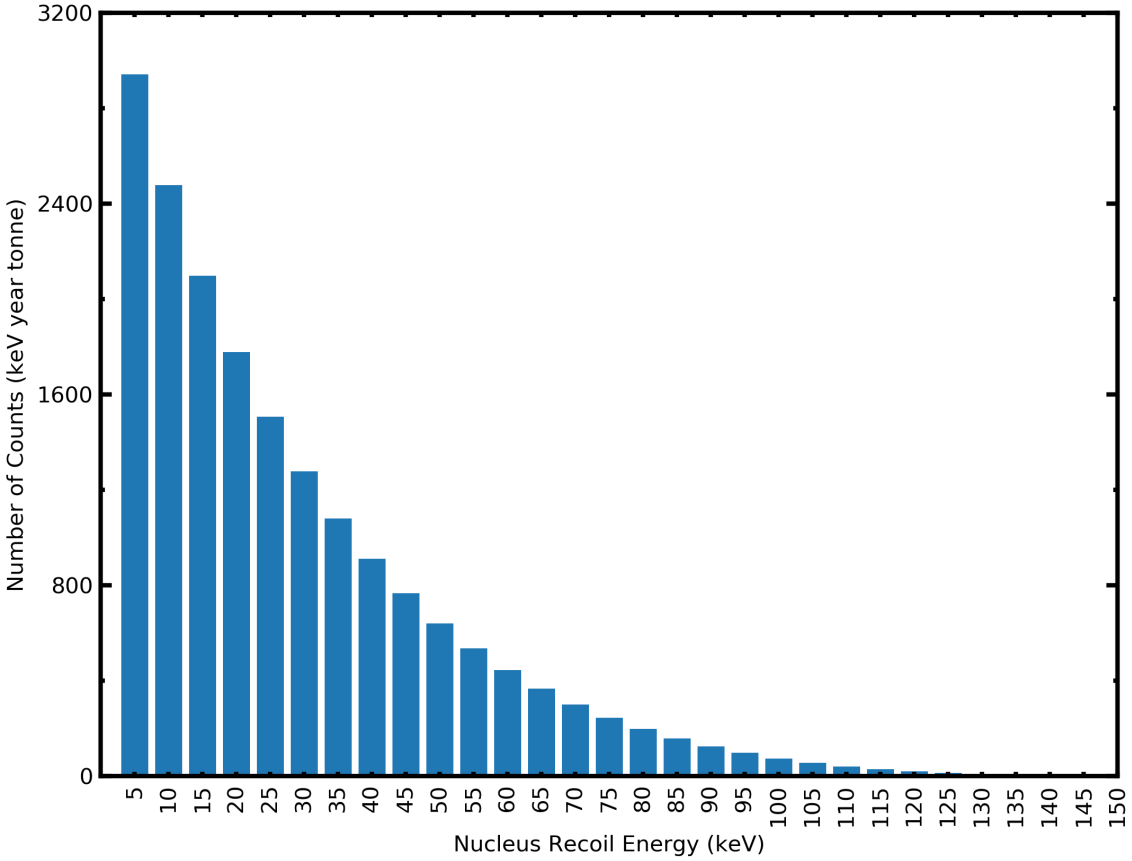

Figure 11: Number of ν_e and $\bar{\nu}_\mu$ events per (keV year tonne) summed in 5 keV bins for ^{40}Ar.

+30%. Each value changes in a small step to give 100 different values. Combining the $\langle R_n^2 \rangle$ and the $\langle R_n^4 \rangle$ values, our database contains the scattering events for 10000 scenarios of different $\langle R_n^2 \rangle$ and $\langle R_n^4 \rangle$ values. Our Monte Carlo simulation assumes that the detection efficiency is always 100% and the neutrino flux is held constant so that we do not need to change the flux parameter during our simulation. For each run, we take the standard scattering events computed using the SkM* $\langle R_n^2 \rangle$ and the $\langle R_n^4 \rangle$ values and add random noise to each bin of the scattering events, which we present mathematically as [14]

$$N'_{events} = N_{events} + \sqrt{N_{events}}s, \tag{24}$$

where N_{events} is the standard scattering events obtained using the SkM* values[14] and s is a random number between -1 and 1 generated from a Gaussian distribution.

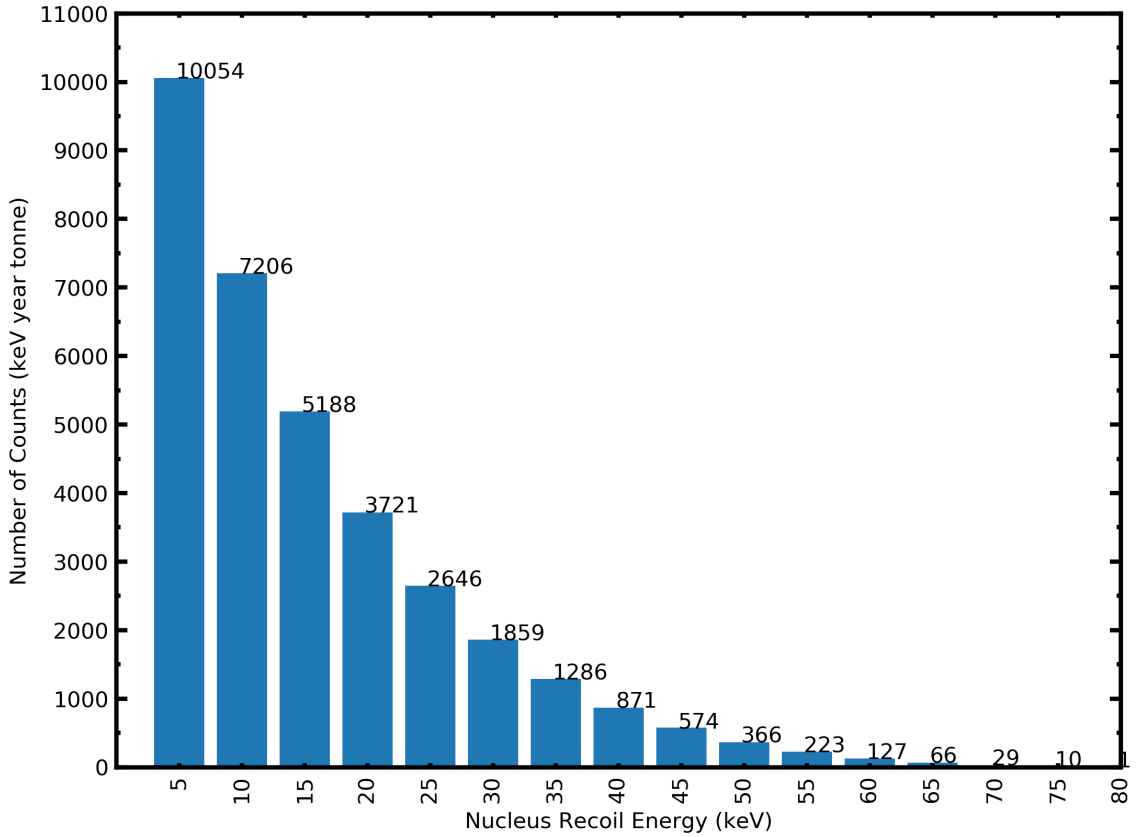

Figure 12: Number of ν_e and $\bar{\nu}_\mu$ events per (keV year tonne) summed in 5 keV bins for Ge, calculated using only the neutron terms in the form factor.

We then take the events with random noise and calculate the χ^2 value for each of 10000 events in our database using

$$\chi^2 = \sum \frac{(N'_{events} - N_{database})^2}{N_{database}}. \tag{25}$$

We then find the corresponding $\langle R_n^2 \rangle$ and the $\langle R_n^4 \rangle$ pair in our database that give the smallest χ^2 value. We repeat this process 5000 times and analyze the trends in the results. Fig. 14 shows the scatter plot of the 5000 $\langle R_n^2 \rangle$ and the $\langle R_n^4 \rangle$ pairs.

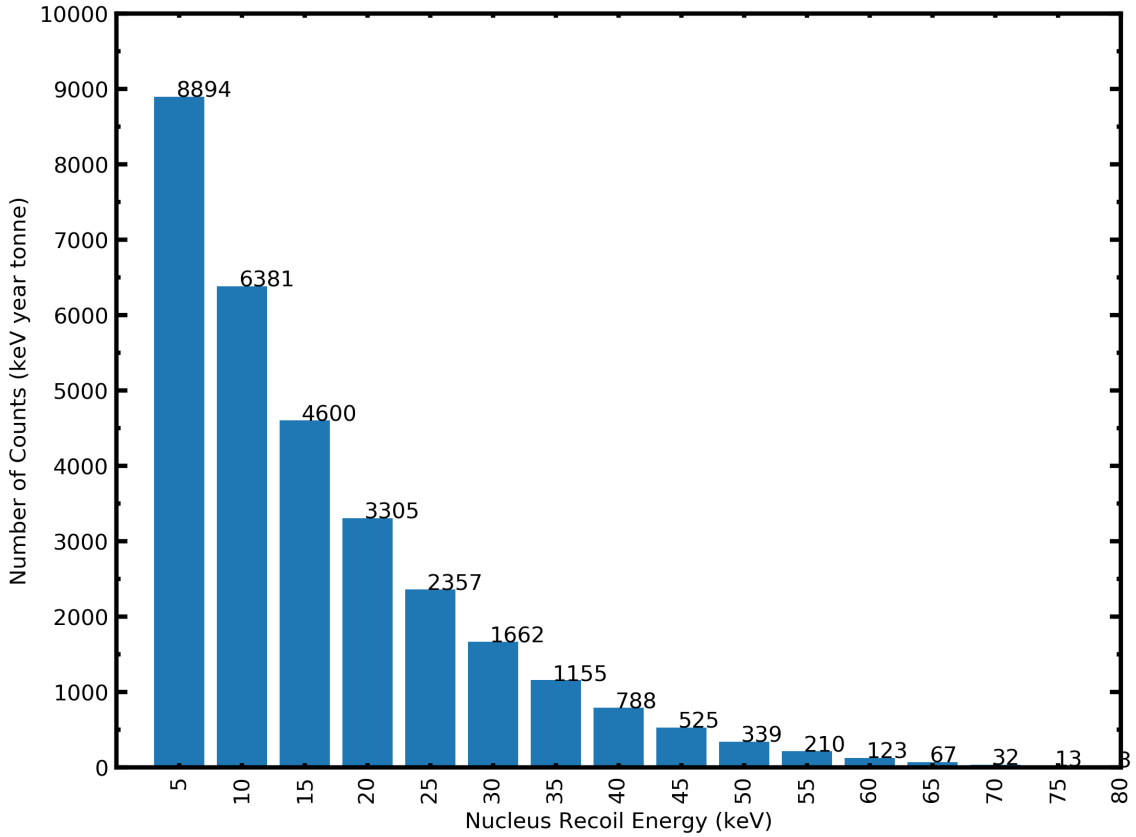

Figure 13: Number of ν_e and $\bar{\nu}_\mu$ events per (keV year tonne) summed in 5 keV bins for Ge, calculated using the full form factor with the proton and cross terms.

3.3 Discussion

From Fig. 14, we can see that the range of possible $\langle R_n^4 \rangle^{1/4}$ values is a lot larger than that of $\langle R_n^2 \rangle^{1/2}$ values. This issue about the dependence on $\langle R_n^4 \rangle^{1/4}$ can be explained by looking at the contribution to the recoil distributions of the effective fourth moment in the form factors. For germanium, a 10% change in the fourth moment will lead to 1.3% more scattering events while a 10% change in $\langle R_n^2 \rangle^{1/2}$ will result in a 6% change in scattering events.[2] Since the effective second moment relates to the nuclear recoil energy more closely, its value varies in a smaller range and has a lower uncertainty compared to the $\langle R_n^2 \rangle^{1/2}$ values.

We also obtain the numerical results from the simulation by calculating the average, the minimum, the maximum, their percentage difference compared with the standard value, and the standard deviation. The results are shown in Table 4. The

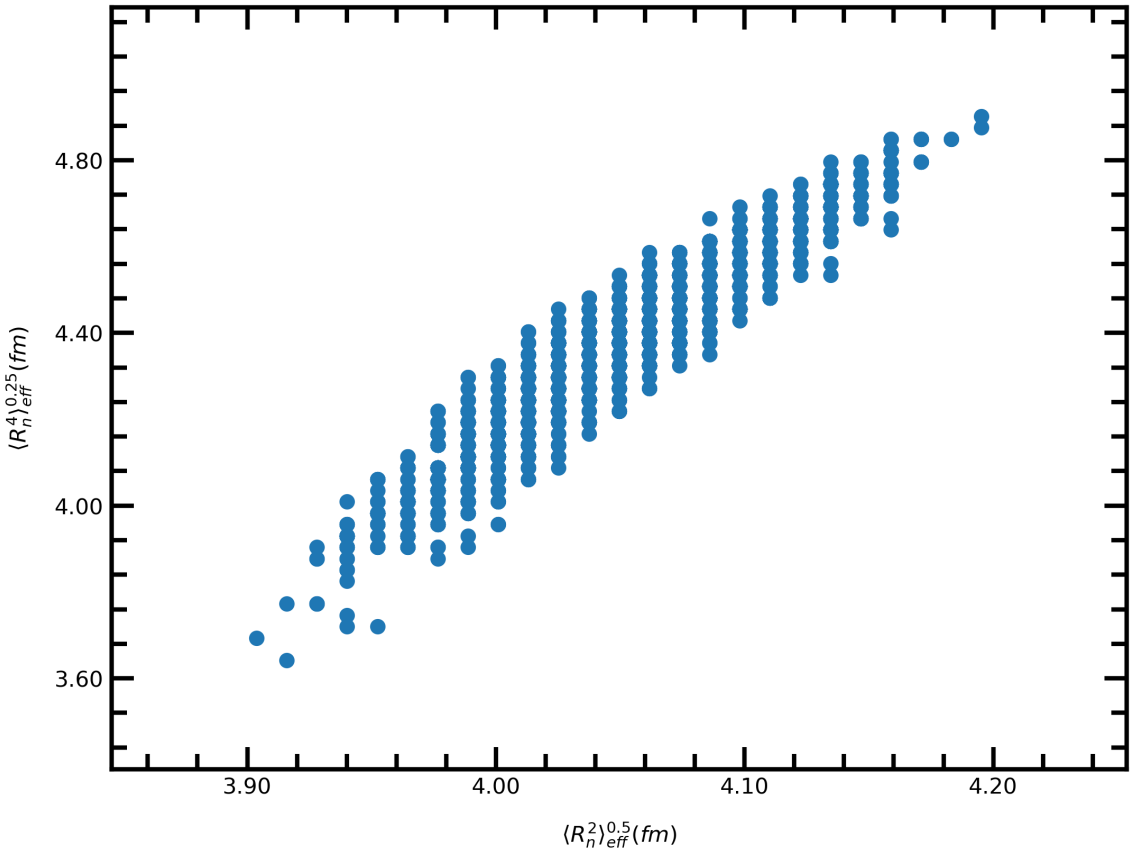

Figure 14: 5000 $\langle R_n^2 \rangle$ and the $\langle R_n^4 \rangle$ pairs from the Monte-Carlo Simulation. Assume that the neutrino flux is 3×10^7 neutrinos per second per cm squared and the detector is filled with 10 tonnes of germanium with 100% detecting efficiency. The scattering events are calculated using the truncated series from Taylor expansion and ignoring the proton and cross terms.

data shows that if the detecting efficiency is constantly 100%, then the measurement of the effective second moment can give us the result that is within $\pm 0.05\%$ of the standard value while the measurement of the effective fourth moment gives us the result within $\pm 0.2\%$ of the standard value. Comparing our result with Table 5, the result from Ref. [2] that uses 10 keV bins instead of 5 keV bins, we can see that our $\langle R_n^4 \rangle^{1/4}$ average value is a lot closer to the standard value, which means that the measurement would give a more accurate result. The comparison shows that the smaller bins can generate higher resolutions, leading to more accurate measurements and results.

Ge	SkM* values	Mean	Max	Min	Standard Deviation
$\langle R_n^2\rangle^{1/2}$ (fm)	4.0495	4.0505	4.1953	3.9037	0.0433
% Difference		0.025	3.57	-3.62	
$\langle R_n^4\rangle^{1/4}$ (fm)	4.3765	4.3726	4.9017	3.6412	0.1806
% Difference		-0.089	12.1	-16.7	

Table 4: Numerical results for detectors filled with 10 tonnes of natural germanium. Assume that the detecting efficiency is always 100%. The first row is the standard value, minimum, maximum, the percentage differences from the standard value, and the standard deviation for $\langle R_n^2\rangle^{1/2}$. The second row shows the same results for $\langle R_n^4\rangle^{1/4}$.

Ge	SkM* values	Mean	Max	Min
$\langle R_n^2\rangle^{1/2}$ (fm)	4.0495	4.0516	4.2697	3.8792
% Difference		0.05	5	-4
$\langle R_n^4\rangle^{1/4}$ (fm)	4.3765	4.3603	5.0096	3.7276
% Difference		-0.4	15	-15
Fixing detector efficiency	SkM* values	Mean	Max	Min
$\langle R_n^2\rangle^{1/2}$ (fm)	4.0495	4.0491	4.1175	3.9857
% Difference		-0.009	1.7	-1.6
$\langle R_n^4\rangle^{1/4}$ (fm)	4.3765	4.3679	4.6546	4.0826
% Difference		-0.2	7	-7

Table 5: Same as Table. 4. Results from Ref. [2] using the 1.5 tonne germanium detector. The first two rows show the results for allowing the detector efficiency to vary by 10%. The last two rows fixed the detector efficiency.

3.4 Future Work

Continuing with our research, there are a few directions that still require more time and effort to investigate. First, in the flux, we only take into account two types of neutrinos, the electron neutrinos and the muon antineutrinos. There is another mono-energetic source that we have not added into our calculation, which gives a constant flux of 29.9 MeV and can make our results more accurate. Second, we have only worked with argon and germanium and only run Monte Carlo simulations with germanium. A good idea would be to add another element like xenon, run Monte Carlo simulations with each, and compare and analyze the trends and differences in their results. Third, during the Monte Carlo simulations, we have been keeping the detecting efficiency at 100%, which is an ideal assumption that is rarely achieved in reality. Varying the detecting efficiency and analyzing its impact on the results for our Monte-Carlo simulations can give us a direct indication of how the consistency of the detectors' performances influences our measurements. Fourth, we used the 5 keV bins which appears to generate better results than the previous implementations using 10 keV bins. It's worth further investigating to run simulations with even lower energy bins like 2 keV bins and compare the results to see if a higher energy resolution can lead to even better measurements. Last, we have calculated the scattering events using the full form factor that adds the proton and cross terms but have not continue our research with this. As shown in Fig. 10, adding the extra terms will slightly reduce the events occurred at lower energies. Therefore, using the full factor and add proton's effective moments as new parameters in the Monte Carlo simulation will give us more accurate results in measuring neutron radius and neutron density distributions.

4 Conclusion

We have proven that the neutrino-nucleus coherent elastic scattering can help determine both the second moment and the fourth moment, which also provides information on the neutron radius. By comparing the results of the scattering simulations using the Helm form factor and the first two terms of the form factor expansion for Ar, we proved the applicability of the implementation of the truncated form factor series instead of the full version. Then, by simulating the scattering events using the truncated form factor for a detector filled with 10 tonnes of natural germanium and a neutrino flux of 3×10^7 neutrinos per second per cm^2, running the Monte Carlo simulation using a bin size of 5 keV, and comparing the results with the previous simulations that used 10 keV bin size, we conclude that if the detector efficiency is fixed, then the CEνNS results can determine the neutron radius and the second and fourth moment with less than 1% uncertainty. We also conclude that the smaller energy bins and higher energy resolutions for detectors reduce the uncertainty in the neutron form factor and yield a more accurate measurement of the second and fourth moments. As mentioned previously, future work is needed to make the simulations more realistic and put limits on what is needed and what is possible for measurements.

References

[1] Amanik, Philip S, and Gail C Mclaughlin, *"Nuclear Neutron Form Factor from Neutrino–Nucleus Coherent Elastic Scattering."* (Journal of Physics G: Nuclear and Particle Physics, vol. 36, no. 1, 2008, p. 015105., doi:10.1088/0954-3899/36/1/015105.).

[2] Patton, Kelly, et al., *"Neutrino-Nucleus Coherent Scattering as a Probe of Neutron Density Distributions."* (Physical Review C, vol. 86, no. 2, 2012, doi:10.1103/physrevc.86.024612.).

[3] Bender, Michael, et al., *"Self-Consistent Mean-Field Models for Nuclear Structure."* (Reviews of Modern Physics, vol. 75, no. 1, 2003, pp. 121–180., doi:10.1103/revmodphys.75.121.).

[4] Scholberg, K., *"Prospects for Measuring Neutrino–Nucleus Coherent Scattering with a Stopped-Pion Neutrino Source."* (Nuclear Physics B - Proceedings Supplements, vol. 229-232, 2012, p. 505., doi:10.1016/j.nuclphysbps.2012.09.142.).

[5] Horowitz, C. J., *"Parity Violating Measurements Of Neutron Densities."* (Nuclear Structure, 2001, doi:10.1142/9789812799753_0067.).

[6] Akimov D. et al., *"Observation of Coherent Elastic Neutrino-Nucleus Scattering."* (Science, vol. 357, no. 6356, 2017, pp. 1123–1126., doi:10.1126/science.aao0990.).

[7] Akimov D. et al., *"COHERENT 2018 at the Spallation Neutron Source."* (Springer Theses First Observation of Coherent Elastic Neutrino-Nucleus Scattering, 2018, pp. 15–20., doi:10.1007/978-3-319-99747-6_3.).

[8] Angloher G. et al., *"Exploring CEvNS with NUCLEUS at the Chooz Nuclear Power Plant"* (2019,arxiv.org/abs/1905.10258.).

[9] Misiak D., *"The CryoCube Detector Array for RICOCHET"* (2018,The Magnificent CEνNS Workshop).

[10] Engel, J., *"Nuclear Form Factors for the Scattering of Weakly Interacting Massive Particles."* (Physics Letters B, vol. 264, no. 1-2, 1991, pp. 114–119., doi:10.1016/0370-2693(91)90712-y.).

[11] Blaum, K., et al., *"Nuclear Moments and Charge Radii of Argon Isotopes between the Neutron-Shell Closures N=20 and N=28."* (Nuclear Physics A, vol. 799, no. 1-4, 2008, pp. 30–45., doi:10.1016/j.nuclphysa.2007.11.004.).

[12] Tuli,J.K., *"Nuclear Wallet Cards."* (2008).

[13] Patton K., *Private Communication.* (2020).

[14] Bartel, J., et al., *"Towards a Better Parametrisation of Skyrme-like Effective Forces: A Critical Study of the SkM Force."* (Nuclear Physics A, vol. 386, no. 1, 1982, pp. 79–100., doi:10.1016/0375-9474(82)90403-1.).

Appendix A Python Code for the Simulations

Starting on the next page, this section will show all the python codes developed in this research that are used to simulate the coherent elastic neutrino-nucleus scattering events and run Monte Carlo simulations. There are a total of 9 files and each file contains detailed description of its purpose.

```python
## Hongyong Zhang
## Compute the two types of flux.
## 05/19/2020
import math
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
from matplotlib import rc
from matplotlib.ticker import (MultipleLocator, FormatStrFormatter,AutoMinorLocator)

## Part1: Constants and Calculations
#############################################################################################
m = 105.6583745
Ev = np.arange(0,m/2,0.5)
fe = 96/m**4*(m*Ev**2-2*Ev**3)
fu = 16/m**4*(3*m*Ev**2-4*Ev**3)

## Part2: Plot
#############################################################################################
plt.rcParams['font.size'] = 10
plt.rcParams['axes.linewidth'] = 2
fig = plt.figure()
ax = fig.add_axes([0, 0, 1, 1])
# Edit the major and minor ticks of the x and y axes
ax.xaxis.set_tick_params(which='major', size=10, width=2, direction='in', top='on')
ax.xaxis.set_tick_params(which='minor', size=7, width=2, direction='in', top='on')
ax.yaxis.set_tick_params(which='major', size=10, width=2, direction='in', right='on')
ax.yaxis.set_tick_params(which='minor', size=7, width=2, direction='in', right='on')

ax.plot(Ev,fe,label='Electron Neutrinos')
ax.plot(Ev,fu,label='Muon Antineutrinos')

# Set the axis limits
ax.set_xlim(0, 55)
ax.set_ylim(0, 0.04)
# Edit the major and minor tick locations
ax.xaxis.set_major_locator(MultipleLocator(10))
ax.xaxis.set_major_formatter(FormatStrFormatter('%d'))
ax.xaxis.set_minor_locator(AutoMinorLocator(3))
ax.yaxis.set_major_locator(MultipleLocator(0.01))
ax.yaxis.set_major_formatter(FormatStrFormatter('%.2f'))
ax.yaxis.set_minor_locator(AutoMinorLocator(3))

# Add legend to plot
ax.legend(bbox_to_anchor=(0.05, 0.95),loc=2, frameon=False, fontsize=10)
ax.set_xlabel('Neutrino Energy (MeV)', labelpad=10)
ax.set_ylabel('Probability Density', labelpad=10)
plt.savefig('Flux.png', dpi=300, transparent=False, bbox_inches='tight')
plt.show()
```

```python
## Hongyong Zhang
## Calculate the Helm form factor for different R values.
## 05/19/2020
import math
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
from matplotlib import rc
from matplotlib.ticker import (MultipleLocator, FormatStrFormatter,AutoMinorLocator)

## Part1: Calculations
################################################################################################
s = 0.5
def fourier_transform(R0):
    Q = np.arange(0, 1, 0.01 )
    a = np.sin(Q*R0)/((Q*R0)**2)
    b = np.cos(Q*R0)/(Q*R0)
    j1 = a - b
    e = np.exp(-0.5*((Q*s)**2))
    F = 3*j1/(Q*R0)*e
    return (Q,F)


## Part2: Plot
################################################################################################
plt.rcParams['font.size'] = 10
plt.rcParams['axes.linewidth'] = 2
fig = plt.figure()
ax = fig.add_axes([0, 0, 1, 1])
# Edit the major and minor ticks of the x and y axes
ax.xaxis.set_tick_params(which='major', size=10, width=2, direction='in', top='on')
ax.xaxis.set_tick_params(which='minor', size=7, width=2, direction='in', top='on')
ax.yaxis.set_tick_params(which='major', size=10, width=2, direction='in', right='on')
ax.yaxis.set_tick_params(which='minor', size=7, width=2, direction='in', right='on')
# Set the axis limits
ax.set_xlim(0, 1)
ax.set_ylim(0, 1)
# Edit the major and minor tick locations
ax.xaxis.set_major_locator(MultipleLocator(0.2))
ax.xaxis.set_major_formatter(FormatStrFormatter('%.1f'))
ax.xaxis.set_minor_locator(AutoMinorLocator(3))
ax.yaxis.set_major_locator(MultipleLocator(0.2))
ax.yaxis.set_major_formatter(FormatStrFormatter('%.1f'))
ax.yaxis.set_minor_locator(AutoMinorLocator(3))


(Q,F) = fourier_transform(3)
ax.plot(Q,F,label='$R_0 = 3fm$')
(Q,F) = fourier_transform(4)
ax.plot(Q,F,label='$R_0 = 4fm$')
(Q,F) = fourier_transform(5)
ax.plot(Q,F,label='$R_0 = 5fm$')

# Add legend to plot
ax.legend(bbox_to_anchor=(0.95, 0.95),loc=1, frameon=False, fontsize=10)
ax.set_xlabel('Momentum Transfer Q', labelpad=10)
ax.set_ylabel('Form Factor', labelpad=10)
plt.savefig('Form_factor.png', dpi=300, transparent=False, bbox_inches='tight')
plt.show()
```

```python
## Hongyong Zhang
## Calculate the Helm form factor for Ar.
## 05/19/2020
import math
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
from matplotlib import rc
from matplotlib.ticker import (MultipleLocator, FormatStrFormatter,AutoMinorLocator)
import scipy.integrate

## Part1: Calculations
################################################################################################
s = 0.5
Q = np.arange(0.01, 1, 0.01 )
def fourier_transform(R0):
    a = np.sin(Q*R0)/((Q*R0)**2)
    b = np.cos(Q*R0)/(Q*R0)
    j1 = a - b
    e = np.exp(-0.5*((Q*s)**2))
    F = 3*j1/(Q*R0)*e

    return F

# Calculate the full form factor for argon
Qw = 22 - 18*(1-4*0.231)
ff = fourier_transform(3.43)
full_ff = 1/Qw*(22*ff-18*(1-4*0.231)*ff)


## Part2: Plot
################################################################################################
plt.rcParams['font.size'] = 10
plt.rcParams['axes.linewidth'] = 2
fig = plt.figure()
ax = fig.add_axes([0, 0, 1, 1])
# Edit the major and minor ticks of the x and y axes
ax.xaxis.set_tick_params(which='major', size=10, width=2, direction='in', top='on')
ax.xaxis.set_tick_params(which='minor', size=7, width=2, direction='in', top='on')
ax.yaxis.set_tick_params(which='major', size=10, width=2, direction='in', right='on')
ax.yaxis.set_tick_params(which='minor', size=7, width=2, direction='in', right='on')


ax.plot(Q,full_ff)

# Set the axis limits
ax.set_xlim(0, 1.1)
ax.set_ylim(0, 1.1)
# Edit the major and minor tick locations
ax.xaxis.set_major_locator(MultipleLocator(0.2))
ax.xaxis.set_major_formatter(FormatStrFormatter('%.1f'))
ax.xaxis.set_minor_locator(AutoMinorLocator(3))
ax.yaxis.set_major_locator(MultipleLocator(0.2))
ax.yaxis.set_major_formatter(FormatStrFormatter('%.1f'))
ax.yaxis.set_minor_locator(AutoMinorLocator(3))

ax.set_xlabel('Momentum Transfer Q', labelpad=10)
ax.set_ylabel('Form Factor', labelpad=10)
plt.savefig('Form_factor_Argon.png', dpi=300, transparent=False, bbox_inches='tight')
plt.show()
```

```
## Hongyong Zhang
## Scattering Events for Ar using the Helm form factor.
## 05/19/2020
import matplotlib as mpl
import matplotlib.pyplot as plt
from matplotlib import rc
from matplotlib.ticker import (MultipleLocator, FormatStrFormatter,AutoMinorLocator)
import numpy as np
import pandas as pd
import scipy.integrate
from numpy import cos, sin, exp, sqrt

## Part1: Constants
#############################################################################################
## Mass are in Mev/c^2
M = 37215.8
m = 105.6583745
hbar = 6.582119569e-22
c = 3e8
event = []
energy = []

## Part2: Integrations
#############################################################################################
for i in range(1,151):
    t = 0.001*i
    def integrand1(z):
        # Calculate dN/dt numerically, variable z is the energy E.
        s = 0.5e-15
        R0 = 3.43e-15


        Q = (2*(z**2)*t*M/(z**2-z*t))**0.5
        a = sin(Q*R0/(hbar*c))/((Q*R0/(hbar*c))**2)
        b = cos(Q*R0/(hbar*c))/(Q*R0/(hbar*c))
        j1 = a - b
        e = exp(-0.5*((Q*s/(hbar*c))**2))

        ## form factor
        F = 3*j1/(Q*R0/(hbar*c))*e
        Qw = 22 - 18*(1-4*0.231)
        ## Fermi Constant is in MeV^(-2)
        Gf = 1.1663787e-5/(1000**2)

        ## full form factor, cross section, and flux
        full_ff = 1/Qw*(22*F-18*(1-4*0.231)*F)
        Cross_Section = (Gf**2)/(2*3.1415926)*M*(2-2*t/z+(t/z)**2-
M*t/(z**2))*Qw**2/4*(full_ff**2)
        fu = 16/m**4*(3*m*z**2-4*z**3)
        fe = 96/m**4*(m*z**2-2*z**3)

        ## The full integral,multiply by (hbar*c)^2 to normalize.
        Nt = 54.63e28/39.962383
        C = 10**11
        f1 = (hbar*c)**2*C*fu*Cross_Section*(3.154e7)*Nt/1000
        return f1

    def integrand2(z):
        # Calculate dN/dt numerically, variable z is the energy E.
        s = 0.5e-15
```

```
        R0 = 3.43e-15


        Q = (2*(z**2)*t*M/(z**2-z*t))**0.5
        a = sin(Q*R0/(hbar*c))/((Q*R0/(hbar*c))**2)
        b = cos(Q*R0/(hbar*c))/(Q*R0/(hbar*c))
        j1 = a - b
        e = exp(-0.5*((Q*s/(hbar*c))**2))

        ## form factor
        F = 3*j1/(Q*R0/(hbar*c))*e
        Qw = 22 - 18*(1-4*0.231)
        ## Fermi Constant is in MeV^(-2)
        Gf = 1.1663787e-5/(1000**2)

        ## full form factor, cross section, and flux
        full_ff = 1/Qw*(22*F-18*(1-4*0.231)*F)
        Cross_Section = (Gf**2)/(2*3.1415926)*M*(2-2*t/z+(t/z)**2-
M*t/(z**2))*Qw**2/4*(full_ff**2)
        fu = 16/m**4*(3*m*z**2-4*z**3)
        fe = 96/m**4*(m*z**2-2*z**3)

        ## The full integral,multiply by (hbar*c)^2 to normalize.
        Nt = 54.63e28/39.962383
        C = 10**11
        f2 = (hbar*c)**2*C*fe*Cross_Section*(3.154e7)*Nt/1000
        return f2


    I1 = scipy.integrate.quad(integrand1,(t+sqrt(t**2+2*t*M))/2,m/2)
    I2 = scipy.integrate.quad(integrand2,(t+sqrt(t**2+2*t*M))/2,m/2)
    event.append(I1[0]+I2[0])
    energy.append(t*1000)


## Part3: Plots and Tables
################################################################################################
## Plot:
plt.rcParams['font.size'] = 10
plt.rcParams['axes.linewidth'] = 2
fig = plt.figure()
ax = fig.add_axes([0, 0, 1, 1])
# Edit the major and minor ticks of the x and y axes
ax.xaxis.set_tick_params(which='major', size=10, width=2, direction='in', top='on')
ax.xaxis.set_tick_params(which='minor', size=7, width=2, direction='in', top='on')
ax.yaxis.set_tick_params(which='major', size=10, width=2, direction='in', right='on')
ax.yaxis.set_tick_params(which='minor', size=7, width=2, direction='in', right='on')

ax.plot(energy,event)

# Set the axis limits
ax.set_xlim(0, 150)
ax.set_ylim(0, 600)
# Edit the major and minor tick locations
ax.xaxis.set_major_locator(MultipleLocator(15))
ax.xaxis.set_major_formatter(FormatStrFormatter('%d'))
ax.xaxis.set_minor_locator(AutoMinorLocator(3))
ax.yaxis.set_major_locator(MultipleLocator(100))
ax.yaxis.set_major_formatter(FormatStrFormatter('%d'))
ax.yaxis.set_minor_locator(AutoMinorLocator(3))
```

```
119
120 ax.set_xlabel('Nucleus Recoil Energy (keV)', labelpad=10)
121 ax.set_ylabel('Events/(keV year tonne)', labelpad=10)
122 plt.savefig('Scattering_Events_Ar_Amanik.png', dpi=300, transparent=False,
    bbox_inches='tight')
123 plt.show()
124
125 table = pd.DataFrame({ 'Bin Range(keV)': energy, 'Events' : event})
126 #print(event1)
127 table.to_csv('table_Ar.csv', index=False, encoding='utf-8')
128
```

```
## Hongyong Zhang
## Scattering Events for Ar using the first two terms of the form factor expansion.
## 05/19/2020
import matplotlib as mpl
import matplotlib.pyplot as plt
from matplotlib import rc
from matplotlib.ticker import (MultipleLocator, FormatStrFormatter,AutoMinorLocator)
import numpy as np
import scipy.integrate
import pandas as pd
from numpy import cos, sin, exp, sqrt

## Part1: Constants
#############################################################################################
## Mass are in Mev/c^2
M = 37215.8
m = 105.6583745
hbar = 6.582119569e-22
c = 3e8
R2 = (3.4168e-15)**2
R4 = (3.7233e-15)**4


## Part2: Integrations
#############################################################################################
def dNdt(R2,R4):
    event = []
    for p in range(1,31):
        ssum = 0
        for i in range(1,6):
            sum = 0
            t = ((p-1)*5+i)*0.001
            def integrand1(z,R2,R4):
                # Calculate dN/dt numerically, variable z is the energy E.
                s = 0.5e-15
                R0 = 3.43e-15
                Qw = 22 - 18*(1-4*0.231)
                Q = (2*(z**2)*t*M/(z**2-z*t))**0.5


                ## full form factor using expansions
                full_ff = 22*(1-(Q/(hbar*c))**2/6*R2+(Q/(hbar*c))**4/120*R4)/Qw

                ## Fermi Constant is in MeV^(-2)
                Gf = 1.1663787e-5/(1000**2)

                ## cross section and flux
                Cross_Section = (Gf**2)/(2*3.1415926)*M*(2-2*t/z+(t/z)**2-
M*t/(z**2))*Qw**2/4*(full_ff**2)
                fu = 16/m**4*(3*m*z**2-4*z**3)
                fe = 96/m**4*(m*z**2-2*z**3)

                ## The full integral,multiply by (hbar*c)^2 to normalize.
                Nt = 54.63e28/39.962383
                C = 10**11
                f1 = (hbar*c)**2*C*fu*Cross_Section*(3.154e7)*Nt/1000
                return f1

            def integrand2(z,R2,R4):
            # Calculate dN/dt numerically, variable z is the energy E.
```

```
60                 s = 0.5e-15
61                 R0 = 3.43e-15
62                 Qw = 22 - 18*(1-4*0.231)
63                 Q = (2*(z**2)*t*M/(z**2-z*t))**0.5
64
65
66                 ## full form factor using expansions
67                 full_ff = 22*(1-(Q/(hbar*c))**2/6*R2+(Q/(hbar*c))**4/120*R4)/Qw
68
69                 ## Fermi Constant is in MeV^(-2)
70                 Gf = 1.1663787e-5/(1000**2)
71
72                 ## cross section and flux
73                 Cross_Section = (Gf**2)/(2*3.1415926)*M*(2-2*t/z+(t/z)**2-
M*t/(z**2))*Qw**2/4*(full_ff**2)
74                 fu = 16/m**4*(3*m*z**2-4*z**3)
75                 fe = 96/m**4*(m*z**2-2*z**3)
76
77                 ## The full integral,multiply by (hbar*c)^2 to normalize.
78                 Nt = 54.63e28/39.962383
79                 C = 10**11
80                 f2 = (hbar*c)**2*C*fe*Cross_Section*(3.154e7)*Nt/1000
81                 return f2
82
83
84             I1 = scipy.integrate.quad(integrand1,(t+sqrt(t**2+2*t*M))/2,m/2,args=
(R2,R4))
85             I2 = scipy.integrate.quad(integrand2,(t+sqrt(t**2+2*t*M))/2,m/2,args=
(R2,R4))
86             sum = I1[0]+I2[0]
87             ssum = ssum + I1[0]+I2[0]
88             #event.append(int(sum))
89         event.append(int(ssum))
90     return event
91
92
93 ## Part3: Plots and Tables
94 ################################################################################
95 energy = np.linspace(5,150,num=30)
96 event1 = dNdt(R2,R4)
97 event2 = dNdt(R2*0.9**2,R4*0.9**4)
98 event3 = dNdt(R2*0.8**2,R4*0.8**4)
99 event4 = dNdt(R2*1.1**2,R4*1.1**4)
100 event5 = dNdt(R2*1.2**2,R4*1.2**4)
101
102 table = pd.DataFrame({ 'Bin Range(keV)': energy, 'Events' : event1, 'Events (-10%)' :
event2, 'Events (-20%)' : event3, 'Events (+10%)' : event4, 'Events (+20%)' :
event5})
103 #print(event1)
104 #table.to_csv('table_Ar_Expanded.csv', index=False, encoding='utf-8')
105
106 # # Plot:
107 # plt.rcParams['font.size'] = 10
108 # plt.rcParams['axes.linewidth'] = 2
109 # fig = plt.figure()
110 # ax = fig.add_axes([0, 0, 1, 1])
111 # # Edit the major and minor ticks of the x and y axes
112 # ax.xaxis.set_tick_params(which='major', size=10, width=2, direction='in', top='on')
113 # ax.xaxis.set_tick_params(which='minor', size=7, width=2, direction='in', top='on')
```

```
# ax.yaxis.set_tick_params(which='major', size=10, width=2, direction='in', right='on')
# ax.yaxis.set_tick_params(which='minor', size=7, width=2, direction='in', right='on')

# ax.plot(energy,event1,label='R')
# ax.plot(energy,event2,label='0.9R')
# ax.plot(energy,event3,label='0.8R')
# ax.plot(energy,event4,label='1.1R')
# ax.plot(energy,event5,label='1.2R')

# # Set the axis limits
# ax.set_xlim(0, 150)
# ax.set_ylim(0, 650)
# # Edit the major and minor tick locations
# ax.xaxis.set_major_locator(MultipleLocator(20))
# ax.xaxis.set_major_formatter(FormatStrFormatter('%d'))
# ax.xaxis.set_minor_locator(AutoMinorLocator(3))
# ax.yaxis.set_major_locator(MultipleLocator(100))
# ax.yaxis.set_major_formatter(FormatStrFormatter('%d'))
# ax.yaxis.set_minor_locator(AutoMinorLocator(3))

# # Add legend to plot
# ax.legend(bbox_to_anchor=(0.95, 0.95),loc=1, frameon=False, fontsize=10)
# ax.set_xlabel('Nucleus Recoil Energy (keV)', labelpad=10)
# ax.set_ylabel('Events/(keV year tonne)', labelpad=10)
# plt.savefig('Scattering_Events_Ar.png', dpi=300, transparent=False, bbox_inches='tight')
# plt.show()

# ## Second Plot: Compare the difference between this and the Amanik's form factor.
# (energy0,event0) = np.loadtxt('table_Ar.csv', unpack=True, delimiter=',', skiprows=1)
# diff = event1-event0
# plt.rcParams['font.size'] = 10
# plt.rcParams['axes.linewidth'] = 2
# fig = plt.figure()
# ax2 = fig.add_axes([0, 0, 1, 0.4])
# ax1 = fig.add_axes([0, 0.6, 1, 0.4])

# # # Set the axis limits
# ax1.set_xlim(0, 150)
# ax1.set_ylim(0, 650)
# ax2.set_xlim(0, 150)
# ax2.set_ylim(-10, 80)
# # # Edit the major and minor ticks of the x and y axes
# ax1.xaxis.set_tick_params(which='major', size=10, width=2, direction='in', top='on')
# ax1.xaxis.set_tick_params(which='minor', size=7, width=2, direction='in', top='on')
# ax1.yaxis.set_tick_params(which='major', size=10, width=2, direction='in', right='on')
# ax1.yaxis.set_tick_params(which='minor', size=7, width=2, direction='in', right='on')
# ax2.xaxis.set_tick_params(which='major', size=10, width=2, direction='in', top='on')
# ax2.xaxis.set_tick_params(which='minor', size=7, width=2, direction='in', top='on')
# ax2.yaxis.set_tick_params(which='major', size=10, width=2, direction='in', right='on')
# ax2.yaxis.set_tick_params(which='minor', size=7, width=2, direction='in', right='on')
```

```
164
165 # ax1.plot(energy,event1,label='With Expansion')
166 # ax1.plot(energy,event0,label='Without Expansion')
167 # ax2.plot(energy,diff)
168
169 # # Edit the major and minor tick locations
170 # ax1.xaxis.set_major_locator(MultipleLocator(20))
171 # ax1.xaxis.set_major_formatter(FormatStrFormatter('%d'))
172 # ax1.xaxis.set_minor_locator(AutoMinorLocator(3))
173 # ax1.yaxis.set_major_locator(MultipleLocator(100))
174 # ax1.yaxis.set_major_formatter(FormatStrFormatter('%d'))
175 # ax1.yaxis.set_minor_locator(AutoMinorLocator(3))
176 # ax2.xaxis.set_major_locator(MultipleLocator(20))
177 # ax2.xaxis.set_major_formatter(FormatStrFormatter('%d'))
178 # ax2.xaxis.set_minor_locator(AutoMinorLocator(3))
179 # ax2.yaxis.set_major_locator(MultipleLocator(10))
180 # ax2.yaxis.set_major_formatter(FormatStrFormatter('%d'))
181 # ax2.yaxis.set_minor_locator(AutoMinorLocator(3))
182 # # Add legend to plot
183 # ax1.legend(bbox_to_anchor=(0.95, 0.95),loc=1, frameon=False, fontsize=10)
184 # ax2.set_xlabel('Nucleus Recoil Energy (keV)', labelpad=10)
185 # ax2.set_ylabel('Difference in Events/(keV year tonne)', labelpad=10)
186 # ax1.set_ylabel('Events/(keV year tonne)', labelpad=10)
187 # plt.savefig('Scattering_Events_Ar_diff.png', dpi=300, transparent=False,
    bbox_inches='tight')
188 # plt.show()
189
190 ## Bar Plot:
191 plt.rcParams['font.size'] = 10
192 plt.rcParams['axes.linewidth'] = 2
193 fig = plt.figure()
194 ax = fig.add_axes([0, 0, 1, 1])
195 # Edit the major and minor ticks of the x and y axes
196 ax.xaxis.set_tick_params(which='major', size=2, width=2, direction='in', top='on')
197 ax.yaxis.set_tick_params(which='major', size=5, width=2, direction='in', right='on')
198 ax.yaxis.set_tick_params(which='minor', size=2, width=2, direction='in', right='on')
199
200 # Set the axis limits
201 ax.set_xlim(0, 150)
202 ax.set_ylim(0, 3200)
203 # Edit the major and minor tick locations
204 ax.yaxis.set_major_locator(MultipleLocator(800))
205 ax.yaxis.set_major_formatter(FormatStrFormatter('%d'))
206 ax.yaxis.set_minor_locator(AutoMinorLocator(2))
207
208 plt.bar(energy,event1,align='center',width=4)
209 plt.xticks(energy)
210 plt.xticks(rotation=90)
211 plt.ylabel("Number of Counts (keV year tonne)")
212 plt.xlabel("Nucleus Recoil Energy (keV)")
213 # for i, v in enumerate(event1):
214 #     plt.text(energy[i] -0.25, v , str(v))
215 plt.savefig('Scattering_Events_Ar_binned.png', dpi=300, transparent=False,
    bbox_inches='tight')
216 plt.show()
217
```

```
## Hongyong Zhang
## Scattering Events for Ge using the first two terms of the form factor expansion and
## only the neutron form factor.
## 05/19/2020
import matplotlib as mpl
import matplotlib.pyplot as plt
from matplotlib import rc
from matplotlib.ticker import (MultipleLocator, FormatStrFormatter,AutoMinorLocator)
import numpy as np
import scipy.integrate
import pandas as pd
from numpy import cos, sin, exp, sqrt

# PART 1: CONSTANTS
#############################################################################################
## Mass are in Mev/c^2
M_bar = (69.9242474*0.2057+71.92207589*0.2745+72.9234589*0.0775+73.9211778*0.3650+75.9214026*0.0773)*931.5
m = 105.6583745
hbar = 6.582119569e-22
c = 3e8
R2 = (4.0495e-15)**2
R4 = (4.3765e-15)**4
M_detector = 1000*5.5866e29
Na = 6.022e23


#PART2: Integration
#############################################################################################
def dNdt(R2,R4):
    event = []
    event_bin =[]
    for p in range(1,17):

        sum_bin = 0
        for i in range(1,6):
            ssum = 0
            t = ((p-1)*5+i)*0.001
            def integrand1(z,R2,R4):
                # Calculate dN/dt numerically, variable z is the energy E.
                s = 0.5e-15
                Qw = 22 - 18*(1-4*0.231)
                ## Fermi Constant is in MeV^(-2)
                Gf = 1.1663787e-5/(1000**2)
                # Calculate the cross section for Ge. Summing over different isoptopes.
                M1 = 69.9242474*931.5
                M2 = 71.92207589*931.5
                M3 = 72.9234589*931.5
                M4 = 73.9211778*931.5
                M5 =75.9214026*931.5
                X1 = 0.2057
                X2 = 0.2745
                X3 = 0.0775
                X4 = 0.3650
                X5 = 0.0773
                N1 = 38
                N2 = 40
```

```
57                N3 = 41
58                N4 = 42
59                N5 = 44
60                Q = (2*(z**2)*t*M_bar/(z**2-z*t))**0.5/(hbar*c)
61                R2_eff = R2
62                R4_eff = R4
63
64                cs1 = Gf**2/(8*3.1415926)*(X1*N1**2*(2-2*t/z+(t/z)**2)-X1*N1**2*M1*
(t/z**2)-R2_eff*X1*N1**2*M1*(2-2*t/z+(t/z)**2)*Q**2/(3*M_bar)+R2_eff*X1*N1**2*M1**2*
(t/z**2)*Q**2/(3*M_bar)+R2_eff**2*X1*N1**2*M1**2*(2-2*t/z+
(t/z)**2)*Q**4/(36.0*M_bar**2)-R2_eff**2*X1*N1**2*M1**3*
(t/z**2)*Q**4/(36.0*M_bar**2)+R4_eff*X1*N1**2*M1**2*(2-2*t/z+
(t/z)**2)*Q**4/(60.0*M_bar**2)-R4_eff*X1*N1**2*M1**3*(t/z**2)*Q**4/(60.0*M_bar**2)-
R2_eff*R4_eff*X1*N1**2*M1**3*(2-2*t/z+
(t/z)**2)*Q**6/(360.0*M_bar**3)+R2_eff*R4_eff*X1*N1**2*M1**4*
(t/z**2)*Q**6/(360.0*M_bar**3))
65                cs2 = Gf**2/(8*3.1415926)*(X2*N2**2*(2-2*t/z+(t/z)**2)-X2*N2**2*M2*
(t/z**2)-R2_eff*X2*N2**2*M2*(2-2*t/z+(t/z)**2)*Q**2/(3*M_bar)+R2_eff*X2*N2**2*M2**2*
(t/z**2)*Q**2/(3*M_bar)+R2_eff**2*X2*N2**2*M2**2*(2-2*t/z+
(t/z)**2)*Q**4/(36.0*M_bar**2)-R2_eff**2*X2*N2**2*M2**3*
(t/z**2)*Q**4/(36.0*M_bar**2)+R4_eff*X2*N2**2*M2**2*(2-2*t/z+
(t/z)**2)*Q**4/(60.0*M_bar**2)-R4_eff*X2*N2**2*M2**3*(t/z**2)*Q**4/(60.0*M_bar**2)-
R2_eff*R4_eff*X2*N2**2*M2**3*(2-2*t/z+
(t/z)**2)*Q**6/(360.0*M_bar**3)+R2_eff*R4_eff*X2*N2**2*M2**4*
(t/z**2)*Q**6/(360.0*M_bar**3))
66                cs3 = Gf**2/(8*3.1415926)*(X3*N3**2*(2-2*t/z+(t/z)**2)-X3*N3**2*M3*
(t/z**2)-R2_eff*X3*N3**2*M3*(2-2*t/z+(t/z)**2)*Q**2/(3*M_bar)+R2_eff*X3*N3**2*M3**2*
(t/z**2)*Q**2/(3*M_bar)+R2_eff**2*X3*N3**2*M3**2*(2-2*t/z+
(t/z)**2)*Q**4/(36.0*M_bar**2)-R2_eff**2*X3*N3**2*M3**3*
(t/z**2)*Q**4/(36.0*M_bar**2)+R4_eff*X3*N3**2*M3**2*(2-2*t/z+
(t/z)**2)*Q**4/(60.0*M_bar**2)-R4_eff*X3*N3**2*M3**3*(t/z**2)*Q**4/(60.0*M_bar**2)-
R2_eff*R4_eff*X3*N3**2*M3**3*(2-2*t/z+
(t/z)**2)*Q**6/(360.0*M_bar**3)+R2_eff*R4_eff*X3*N3**2*M3**4*
(t/z**2)*Q**6/(360.0*M_bar**3))
67                cs4 = Gf**2/(8*3.1415926)*(X4*N4**2*(2-2*t/z+(t/z)**2)-X4*N4**2*M4*
(t/z**2)-R2_eff*X4*N4**2*M4*(2-2*t/z+(t/z)**2)*Q**2/(3*M_bar)+R2_eff*X4*N4**2*M4**2*
(t/z**2)*Q**2/(3*M_bar)+R2_eff**2*X4*N4**2*M4**2*(2-2*t/z+
(t/z)**2)*Q**4/(36.0*M_bar**2)-R2_eff**2*X4*N4**2*M4**3*
(t/z**2)*Q**4/(36.0*M_bar**2)+R4_eff*X4*N4**2*M4**2*(2-2*t/z+
(t/z)**2)*Q**4/(60.0*M_bar**2)-R4_eff*X4*N4**2*M4**3*(t/z**2)*Q**4/(60.0*M_bar**2)-
R2_eff*R4_eff*X4*N4**2*M4**3*(2-2*t/z+
(t/z)**2)*Q**6/(360.0*M_bar**3)+R2_eff*R4_eff*X4*N4**2*M4**4*
(t/z**2)*Q**6/(360.0*M_bar**3))
68                cs5 = Gf**2/(8*3.1415926)*(X5*N5**2*(2-2*t/z+(t/z)**2)-X5*N5**2*M5*
(t/z**2)-R2_eff*X5*N5**2*M5*(2-2*t/z+(t/z)**2)*Q**2/(3*M_bar)+R2_eff*X5*N5**2*M5**2*
(t/z**2)*Q**2/(3*M_bar)+R2_eff**2*X5*N5**2*M5**2*(2-2*t/z+
(t/z)**2)*Q**4/(36.0*M_bar**2)-R2_eff**2*X5*N5**2*M5**3*
(t/z**2)*Q**4/(36.0*M_bar**2)+R4_eff*X5*N5**2*M5**2*(2-2*t/z+
(t/z)**2)*Q**4/(60.0*M_bar**2)-R4_eff*X5*N5**2*M5**3*(t/z**2)*Q**4/(60.0*M_bar**2)-
R2_eff*R4_eff*X5*N5**2*M5**3*(2-2*t/z+
(t/z)**2)*Q**6/(360.0*M_bar**3)+R2_eff*R4_eff*X5*N5**2*M5**4*
(t/z**2)*Q**6/(360.0*M_bar**3))
69                cross_section = cs1 + cs2 + cs3 + cs4 + cs5
70
71                ## flux
72                fu = 16/m**4*(3*m*z**2-4*z**3)
73                fe = 96/m**4*(m*z**2-2*z**3)
74
75                ## The full integral,multiply by (hbar*c)^2 to normalize.
76                C = 10**11
```

```
77                f1 = (hbar*c)**2*C*fu*cross_section*(3.156e7)*M_detector/1000
78                return f1
79
80            def integrand2(z,R2,R4):
81                # Calculate dN/dt numerically, variable z is the energy E.
82                s = 0.5e-15
83                Qw = 22 - 18*(1-4*0.231)
84                ## Fermi Constant is in MeV^(-2)
85                Gf = 1.1663787e-5/(1000**2)
86                # Calculate the cross section for Ge. Summing over different
isoptopes.
87                M1 = 69.9242474*931.5
88                M2 = 71.92207589*931.5
89                M3 = 72.9234589*931.5
90                M4 = 73.9211778*931.5
91                M5 =75.9214026*931.5
92                X1 = 0.2057
93                X2 = 0.2745
94                X3 = 0.0775
95                X4 = 0.3650
96                X5 = 0.0773
97                N1 = 38
98                N2 = 40
99                N3 = 41
100               N4 = 42
101               N5 = 44
102               M_bar =
(69.9242474*0.2057+71.92207589*0.2745+72.9234589*0.0775+73.9211778*0.3650+75.9214026*
0.0773)*931.5
103               Q = (2*(z**2)*t*M_bar/(z**2-z*t))**0.5/(hbar*c)
104               R2_eff = R2
105               R4_eff = R4
106
107               cs1 = Gf**2/(8*3.1415926)*(X1*N1**2*(2-2*t/z+(t/z)**2)-X1*N1**2*M1*
(t/z**2)-R2_eff*X1*N1**2*M1*(2-2*t/z+(t/z)**2)*Q**2/(3*M_bar)+R2_eff*X1*N1**2*M1**2*
(t/z**2)*Q**2/(3*M_bar)+R2_eff**2*X1*N1**2*M1**2*(2-2*t/z+
(t/z)**2)*Q**4/(36.0*M_bar**2)-R2_eff**2*X1*N1**2*M1**3*
(t/z**2)*Q**4/(36.0*M_bar**2)+R4_eff*X1*N1**2*M1**2*(2-2*t/z+
(t/z)**2)*Q**4/(60.0*M_bar**2)-R4_eff*X1*N1**2*M1**3*(t/z**2)*Q**4/(60.0*M_bar**2)-
R2_eff*R4_eff*X1*N1**2*M1**3*(2-2*t/z+
(t/z)**2)*Q**6/(360.0*M_bar**3)+R2_eff*R4_eff*X1*N1**2*M1**4*
(t/z**2)*Q**6/(360.0*M_bar**3))
108               cs2 = Gf**2/(8*3.1415926)*(X2*N2**2*(2-2*t/z+(t/z)**2)-X2*N2**2*M2*
(t/z**2)-R2_eff*X2*N2**2*M2*(2-2*t/z+(t/z)**2)*Q**2/(3*M_bar)+R2_eff*X2*N2**2*M2**2*
(t/z**2)*Q**2/(3*M_bar)+R2_eff**2*X2*N2**2*M2**2*(2-2*t/z+
(t/z)**2)*Q**4/(36.0*M_bar**2)-R2_eff**2*X2*N2**2*M2**3*
(t/z**2)*Q**4/(36.0*M_bar**2)+R4_eff*X2*N2**2*M2**2*(2-2*t/z+
(t/z)**2)*Q**4/(60.0*M_bar**2)-R4_eff*X2*N2**2*M2**3*(t/z**2)*Q**4/(60.0*M_bar**2)-
R2_eff*R4_eff*X2*N2**2*M2**3*(2-2*t/z+
(t/z)**2)*Q**6/(360.0*M_bar**3)+R2_eff*R4_eff*X2*N2**2*M2**4*
(t/z**2)*Q**6/(360.0*M_bar**3))
109               cs3 = Gf**2/(8*3.1415926)*(X3*N3**2*(2-2*t/z+(t/z)**2)-X3*N3**2*M3*
(t/z**2)-R2_eff*X3*N3**2*M3*(2-2*t/z+(t/z)**2)*Q**2/(3*M_bar)+R2_eff*X3*N3**2*M3**2*
(t/z**2)*Q**2/(3*M_bar)+R2_eff**2*X3*N3**2*M3**2*(2-2*t/z+
(t/z)**2)*Q**4/(36.0*M_bar**2)-R2_eff**2*X3*N3**2*M3**3*
(t/z**2)*Q**4/(36.0*M_bar**2)+R4_eff*X3*N3**2*M3**2*(2-2*t/z+
(t/z)**2)*Q**4/(60.0*M_bar**2)-R4_eff*X3*N3**2*M3**3*(t/z**2)*Q**4/(60.0*M_bar**2)-
R2_eff*R4_eff*X3*N3**2*M3**3*(2-2*t/z+
(t/z)**2)*Q**6/(360.0*M_bar**3)+R2_eff*R4_eff*X3*N3**2*M3**4*
(t/z**2)*Q**6/(360.0*M_bar**3))
```

```
110                 cs4 = Gf**2/(8*3.1415926)*(X4*N4**2*(2-2*t/z+(t/z)**2)-X4*N4**2*M4*
(t/z**2)-R2_eff*X4*N4**2*M4*(2-2*t/z+(t/z)**2)*Q**2/(3*M_bar)+R2_eff*X4*N4**2*M4**2*
(t/z**2)*Q**2/(3*M_bar)+R2_eff**2*X4*N4**2*M4**2*(2-2*t/z+
(t/z)**2)*Q**4/(36.0*M_bar**2)-R2_eff**2*X4*N4**2*M4**3*
(t/z**2)*Q**4/(36.0*M_bar**2)+R4_eff*X4*N4**2*M4**2*(2-2*t/z+
(t/z)**2)*Q**4/(60.0*M_bar**2)-R4_eff*X4*N4**2*M4**3*(t/z**2)*Q**4/(60.0*M_bar**2)-
R2_eff*R4_eff*X4*N4**2*M4**3*(2-2*t/z+
(t/z)**2)*Q**6/(360.0*M_bar**3)+R2_eff*R4_eff*X4*N4**2*M4**4*
(t/z**2)*Q**6/(360.0*M_bar**3))
111                 cs5 = Gf**2/(8*3.1415926)*(X5*N5**2*(2-2*t/z+(t/z)**2)-X5*N5**2*M5*
(t/z**2)-R2_eff*X5*N5**2*M5*(2-2*t/z+(t/z)**2)*Q**2/(3*M_bar)+R2_eff*X5*N5**2*M5**2*
(t/z**2)*Q**2/(3*M_bar)+R2_eff**2*X5*N5**2*M5**2*(2-2*t/z+
(t/z)**2)*Q**4/(36.0*M_bar**2)-R2_eff**2*X5*N5**2*M5**3*
(t/z**2)*Q**4/(36.0*M_bar**2)+R4_eff*X5*N5**2*M5**2*(2-2*t/z+
(t/z)**2)*Q**4/(60.0*M_bar**2)-R4_eff*X5*N5**2*M5**3*(t/z**2)*Q**4/(60.0*M_bar**2)-
R2_eff*R4_eff*X5*N5**2*M5**3*(2-2*t/z+
(t/z)**2)*Q**6/(360.0*M_bar**3)+R2_eff*R4_eff*X5*N5**2*M5**4*
(t/z**2)*Q**6/(360.0*M_bar**3))
112                 cross_section = cs1 + cs2 + cs3 + cs4 + cs5
113
114                 ## flux
115                 fu = 16/m**4*(3*m*z**2-4*z**3)
116                 fe = 96/m**4*(m*z**2-2*z**3)
117
118                 ## The full integral,multiply by (hbar*c)^2 to normalize.
119                 C = 10**11
120                 f2 = (hbar*c)**2*C*fe*cross_section*(3.156e7)*M_detector/1000
121                 return f2
122
123
124             I1 = scipy.integrate.quad(integrand1,(t+sqrt(t**2+2*t*M_bar))/2,m/2,args=
(R2,R4))
125             I2 = scipy.integrate.quad(integrand2,(t+sqrt(t**2+2*t*M_bar))/2,m/2,args=
(R2,R4))
126             ssum = ssum + I1[0]+I2[0]
127             #print(ssum)
128             sum_bin = sum_bin + ssum
129             event.append(int(ssum))
130
131         event_bin.append(int(sum_bin))
132     return (event,event_bin)
133
134
135 ## Part3: Plots and Tables
136 ################################################################################
137 energy = np.linspace(1,80,num=80)
138 energy2 = np.linspace(5,80,num=16)
139 (event1,event_bin1) = dNdt(R2,R4)
140     # (event2,event_bin2) = dNdt(R2*0.9**2,R4*0.9**4)
141     # (event3,event_bin3) = dNdt(R2*0.8**2,R4*0.8**4)
142     # (event4,event_bin4) = dNdt(R2*1.1**2,R4*1.1**4)
143     # (event5,event_bin5) = dNdt(R2*1.2**2,R4*1.2**4)
144 #print("event=",event1)
145 #print("binned event=",event_bin1)
146 table = pd.DataFrame({ 'Bin Range(keV)': energy, 'Events' : event1})
147 #table = pd.DataFrame({ 'Bin Range(keV)': energy, 'Events' : event1, 'Events (-10%)'
: event2, 'Events (-20%)' : event3, 'Events (+10%)' : event4, 'Events (+20%)' :
event5})
148 table.to_csv('table_Ge_data.csv', index=False, encoding='utf-8')
149
```

```
150
151     # #plt.plot(energy,event1)
152     # plt.bar(energy2,event_bin1,align='center',width=4)
153     # plt.xticks(energy2)
154     # plt.ylabel("Number of Counts")
155     # plt.xlabel("Energy/keV")
156     # plt.title("# of Events per (keV year tonne) for Ge")
157     # for i, v in enumerate(event_bin1):
158     #     plt.text(energy2[i] -0.25, v , str(v))
159     # plt.show()
160
161
162 ## Bar Plot:
163 plt.rcParams['font.size'] = 10
164 plt.rcParams['axes.linewidth'] = 2
165 fig = plt.figure()
166 ax = fig.add_axes([0, 0, 1, 1])
167 # Edit the major and minor ticks of the x and y axes
168 ax.xaxis.set_tick_params(which='major', size=2, width=2, direction='in', top='on')
169 ax.yaxis.set_tick_params(which='major', size=5, width=2, direction='in', right='on')
170 ax.yaxis.set_tick_params(which='minor', size=2, width=2, direction='in', right='on')
171
172 # Set the axis limits
173 ax.set_xlim(0, 80)
174 ax.set_ylim(0, 11000)
175 # Edit the major and minor tick locations
176 ax.yaxis.set_major_locator(MultipleLocator(1000))
177 ax.yaxis.set_major_formatter(FormatStrFormatter('%d'))
178 ax.yaxis.set_minor_locator(AutoMinorLocator(2))
179
180 plt.bar(energy2,event_bin1,align='center',width=4)
181 plt.xticks(energy2)
182 plt.xticks(rotation=90)
183 plt.ylabel("Number of Counts (keV year tonne)")
184 plt.xlabel("Nucleus Recoil Energy (keV)")
185 for i, v in enumerate(event_bin1):
186     plt.text(energy2[i] -0.25, v , str(v))
187 plt.savefig('Scattering_Events_Ge_binned.png', dpi=300, transparent=False,
    bbox_inches='tight')
188 plt.show()
```

```python
## Hongyong Zhang
## Scattering Events for Ge including the proton and cross terms.
## 05/19/2020
import matplotlib as mpl
import matplotlib.pyplot as plt
from matplotlib import rc
from matplotlib.ticker import (MultipleLocator, FormatStrFormatter,AutoMinorLocator)
import numpy as np
import scipy.integrate
import pandas as pd
from numpy import cos, sin, exp, sqrt

##### Calculate the Contribution from the proton part.

# PART 1: CONSTANTS
#############################################################################################
# Some constants that we need.
## Mass are in Mev/c^2
M_bar = (69.9242474*0.2057+71.92207589*0.2745+72.9234589*0.0775+73.9211778*0.3650+75.9214026*0.0773)*931.5
m = 105.6583745
hbar = 6.582119569e-22
c = 3e8
R2 = (4.0495e-15)**2
R4 = (4.3765e-15)**4
M_detector = 1000*5.5866e29
Na = 6.022e23

s = 0.5e-15
#Qw = 22 - 18*(1-4*0.231)
## Fermi Constant is in MeV^(-2)
Gf = 1.1663787e-5/(1000**2)
# Calculate the cross section for Ge. Summing over different isoptopes.
M1 = 69.9242474*931.5
M2 = 71.92207589*931.5
M3 = 72.9234589*931.5
M4 = 73.9211778*931.5
M5 =75.9214026*931.5
X1 = 0.2057
X2 = 0.2745
X3 = 0.0775
X4 = 0.3650
X5 = 0.0773
N1 = 38
N2 = 40
N3 = 41
N4 = 42
N5 = 44
Z = 32
s2thw = 2.31e-1
alpha = 1-4*s2thw

R2_eff = R2
R4_eff = R4

## Calculate the R2_eff and R4_eff for proton
R2_eff_p = (X1*M1*(15.4282334386e-30)+X2*M2*(15.8950606333e-30)+X3*M3*(15.6649486626e-30)+X4*M4*(15.9736268768e-30)+X5*M5*(16.0256582784e-30))/(X1*M1+X2*M2+X3*M3+X4*M4+X5*M5)
```

```
R4_eff_p = (X1*M1*(323.6887827706e-60)+X2*M2*(344.1084360256e-60)+X3*M3*(330.9393314606e-60)+X4*M4*(345.718316624e-60)+X5*M5*(345.8995154439e-60))/(X1*M1+X2*M2+X3*M3+X4*M4+X5*M5)


#PART2: Integration
################################################################################
def dNdt(R2,R4):
    event = []
    event_bin =[]
    for p in range(1,17):

        sum_bin = 0
        for i in range(1,6):
            ssum = 0
            t = ((p-1)*5+i)*0.001
            ## The neutron part for flux fu.
            def integrand1(z,R2,R4):
                R2_eff = R2
                R4_eff = R4
                # Calculate dN/dt numerically, variable z is the energy E.
                Q = (2*(z**2)*t*M_bar/(z**2-z*t))**0.5/(hbar*c)
                # Five parts of cross section.
                cs1 = Gf**2/(8*3.1415926)*(X1*N1**2*(2-2*t/z+(t/z)**2)-X1*N1**2*M1*(t/z**2)-R2_eff*X1*N1**2*M1*(2-2*t/z+(t/z)**2)*Q**2/(3*M_bar)+R2_eff*X1*N1**2*M1**2*(t/z**2)*Q**2/(3*M_bar)+R2_eff**2*X1*N1**2*M1**2*(2-2*t/z+(t/z)**2)*Q**4/(36.0*M_bar**2)-R2_eff**2*X1*N1**2*M1**3*(t/z**2)*Q**4/(36.0*M_bar**2)+R4_eff*X1*N1**2*M1**2*(2-2*t/z+(t/z)**2)*Q**4/(60.0*M_bar**2)-R4_eff*X1*N1**2*M1**3*(t/z**2)*Q**4/(60.0*M_bar**2)-R2_eff*R4_eff*X1*N1**2*M1**3*(2-2*t/z+(t/z)**2)*Q**6/(360.0*M_bar**3)+R2_eff*R4_eff*X1*N1**2*M1**4*(t/z**2)*Q**6/(360.0*M_bar**3))
                cs2 = Gf**2/(8*3.1415926)*(X2*N2**2*(2-2*t/z+(t/z)**2)-X2*N2**2*M2*(t/z**2)-R2_eff*X2*N2**2*M2*(2-2*t/z+(t/z)**2)*Q**2/(3*M_bar)+R2_eff*X2*N2**2*M2**2*(t/z**2)*Q**2/(3*M_bar)+R2_eff**2*X2*N2**2*M2**2*(2-2*t/z+(t/z)**2)*Q**4/(36.0*M_bar**2)-R2_eff**2*X2*N2**2*M2**3*(t/z**2)*Q**4/(36.0*M_bar**2)+R4_eff*X2*N2**2*M2**2*(2-2*t/z+(t/z)**2)*Q**4/(60.0*M_bar**2)-R4_eff*X2*N2**2*M2**3*(t/z**2)*Q**4/(60.0*M_bar**2)-R2_eff*R4_eff*X2*N2**2*M2**3*(2-2*t/z+(t/z)**2)*Q**6/(360.0*M_bar**3)+R2_eff*R4_eff*X2*N2**2*M2**4*(t/z**2)*Q**6/(360.0*M_bar**3))
                cs3 = Gf**2/(8*3.1415926)*(X3*N3**2*(2-2*t/z+(t/z)**2)-X3*N3**2*M3*(t/z**2)-R2_eff*X3*N3**2*M3*(2-2*t/z+(t/z)**2)*Q**2/(3*M_bar)+R2_eff*X3*N3**2*M3**2*(t/z**2)*Q**2/(3*M_bar)+R2_eff**2*X3*N3**2*M3**2*(2-2*t/z+(t/z)**2)*Q**4/(36.0*M_bar**2)-R2_eff**2*X3*N3**2*M3**3*(t/z**2)*Q**4/(36.0*M_bar**2)+R4_eff*X3*N3**2*M3**2*(2-2*t/z+(t/z)**2)*Q**4/(60.0*M_bar**2)-R4_eff*X3*N3**2*M3**3*(t/z**2)*Q**4/(60.0*M_bar**2)-R2_eff*R4_eff*X3*N3**2*M3**3*(2-2*t/z+(t/z)**2)*Q**6/(360.0*M_bar**3)+R2_eff*R4_eff*X3*N3**2*M3**4*(t/z**2)*Q**6/(360.0*M_bar**3))
                cs4 = Gf**2/(8*3.1415926)*(X4*N4**2*(2-2*t/z+(t/z)**2)-X4*N4**2*M4*(t/z**2)-R2_eff*X4*N4**2*M4*(2-2*t/z+(t/z)**2)*Q**2/(3*M_bar)+R2_eff*X4*N4**2*M4**2*(t/z**2)*Q**2/(3*M_bar)+R2_eff**2*X4*N4**2*M4**2*(2-2*t/z+(t/z)**2)*Q**4/(36.0*M_bar**2)-R2_eff**2*X4*N4**2*M4**3*(t/z**2)*Q**4/(36.0*M_bar**2)+R4_eff*X4*N4**2*M4**2*(2-2*t/z+(t/z)**2)*Q**4/(60.0*M_bar**2)-R4_eff*X4*N4**2*M4**3*(t/z**2)*Q**4/(60.0*M_bar**2)-R2_eff*R4_eff*X4*N4**2*M4**3*(2-2*t/z+(t/z)**2)*Q**6/(360.0*M_bar**3)+R2_eff*R4_eff*X4*N4**2*M4**4*(t/z**2)*Q**6/(360.0*M_bar**3))
```

```
82                cs5 = Gf**2/(8*3.1415926)*(X5*N5**2*(2-2*t/z+(t/z)**2)-X5*N5**2*M5*
(t/z**2)-R2_eff*X5*N5**2*M5*(2-2*t/z+(t/z)**2)*Q**2/(3*M_bar)+R2_eff*X5*N5**2*M5**2*
(t/z**2)*Q**2/(3*M_bar)+R2_eff**2*X5*N5**2*M5**2*(2-2*t/z+
(t/z)**2)*Q**4/(36.0*M_bar**2)-R2_eff**2*X5*N5**2*M5**3*
(t/z**2)*Q**4/(36.0*M_bar**2)+R4_eff*X5*N5**2*M5**2*(2-2*t/z+
(t/z)**2)*Q**4/(60.0*M_bar**2)-R4_eff*X5*N5**2*M5**3*(t/z**2)*Q**4/(60.0*M_bar**2)-
R2_eff*R4_eff*X5*N5**2*M5**3*(2-2*t/z+
(t/z)**2)*Q**6/(360.0*M_bar**3)+R2_eff*R4_eff*X5*N5**2*M5**4*
(t/z**2)*Q**6/(360.0*M_bar**3))
83                cross_section = cs1 + cs2 + cs3 + cs4 + cs5
84
85                ## flux
86                fu = 16/m**4*(3*m*z**2-4*z**3)
87                fe = 96/m**4*(m*z**2-2*z**3)
88
89                ## The full integral,multiply by (hbar*c)^2 to normalize.
90                C = 10**11
91                f1 = (hbar*c)**2*C*fu*cross_section*(3.156e7)*M_detector/1000
92                return f1
93            ## The neutron part for flux fe.
94            def integrand2(z,R2,R4):
95                R2_eff = R2
96                R4_eff = R4
97                # Calculate dN/dt numerically, variable z is the energy E.
98                Q = (2*(z**2)*t*M_bar/(z**2-z*t))**0.5/(hbar*c)
99                #Five parts of cross section.
100               cs1 = Gf**2/(8*3.1415926)*(X1*N1**2*(2-2*t/z+(t/z)**2)-X1*N1**2*M1*
(t/z**2)-R2_eff*X1*N1**2*M1*(2-2*t/z+(t/z)**2)*Q**2/(3*M_bar)+R2_eff*X1*N1**2*M1**2*
(t/z**2)*Q**2/(3*M_bar)+R2_eff**2*X1*N1**2*M1**2*(2-2*t/z+
(t/z)**2)*Q**4/(36.0*M_bar**2)-R2_eff**2*X1*N1**2*M1**3*
(t/z**2)*Q**4/(36.0*M_bar**2)+R4_eff*X1*N1**2*M1**2*(2-2*t/z+
(t/z)**2)*Q**4/(60.0*M_bar**2)-R4_eff*X1*N1**2*M1**3*(t/z**2)*Q**4/(60.0*M_bar**2)-
R2_eff*R4_eff*X1*N1**2*M1**3*(2-2*t/z+
(t/z)**2)*Q**6/(360.0*M_bar**3)+R2_eff*R4_eff*X1*N1**2*M1**4*
(t/z**2)*Q**6/(360.0*M_bar**3))
101               cs2 = Gf**2/(8*3.1415926)*(X2*N2**2*(2-2*t/z+(t/z)**2)-X2*N2**2*M2*
(t/z**2)-R2_eff*X2*N2**2*M2*(2-2*t/z+(t/z)**2)*Q**2/(3*M_bar)+R2_eff*X2*N2**2*M2**2*
(t/z**2)*Q**2/(3*M_bar)+R2_eff**2*X2*N2**2*M2**2*(2-2*t/z+
(t/z)**2)*Q**4/(36.0*M_bar**2)-R2_eff**2*X2*N2**2*M2**3*
(t/z**2)*Q**4/(36.0*M_bar**2)+R4_eff*X2*N2**2*M2**2*(2-2*t/z+
(t/z)**2)*Q**4/(60.0*M_bar**2)-R4_eff*X2*N2**2*M2**3*(t/z**2)*Q**4/(60.0*M_bar**2)-
R2_eff*R4_eff*X2*N2**2*M2**3*(2-2*t/z+
(t/z)**2)*Q**6/(360.0*M_bar**3)+R2_eff*R4_eff*X2*N2**2*M2**4*
(t/z**2)*Q**6/(360.0*M_bar**3))
102               cs3 = Gf**2/(8*3.1415926)*(X3*N3**2*(2-2*t/z+(t/z)**2)-X3*N3**2*M3*
(t/z**2)-R2_eff*X3*N3**2*M3*(2-2*t/z+(t/z)**2)*Q**2/(3*M_bar)+R2_eff*X3*N3**2*M3**2*
(t/z**2)*Q**2/(3*M_bar)+R2_eff**2*X3*N3**2*M3**2*(2-2*t/z+
(t/z)**2)*Q**4/(36.0*M_bar**2)-R2_eff**2*X3*N3**2*M3**3*
(t/z**2)*Q**4/(36.0*M_bar**2)+R4_eff*X3*N3**2*M3**2*(2-2*t/z+
(t/z)**2)*Q**4/(60.0*M_bar**2)-R4_eff*X3*N3**2*M3**3*(t/z**2)*Q**4/(60.0*M_bar**2)-
R2_eff*R4_eff*X3*N3**2*M3**3*(2-2*t/z+
(t/z)**2)*Q**6/(360.0*M_bar**3)+R2_eff*R4_eff*X3*N3**2*M3**4*
(t/z**2)*Q**6/(360.0*M_bar**3))
103               cs4 = Gf**2/(8*3.1415926)*(X4*N4**2*(2-2*t/z+(t/z)**2)-X4*N4**2*M4*
(t/z**2)-R2_eff*X4*N4**2*M4*(2-2*t/z+(t/z)**2)*Q**2/(3*M_bar)+R2_eff*X4*N4**2*M4**2*
(t/z**2)*Q**2/(3*M_bar)+R2_eff**2*X4*N4**2*M4**2*(2-2*t/z+
(t/z)**2)*Q**4/(36.0*M_bar**2)-R2_eff**2*X4*N4**2*M4**3*
(t/z**2)*Q**4/(36.0*M_bar**2)+R4_eff*X4*N4**2*M4**2*(2-2*t/z+
(t/z)**2)*Q**4/(60.0*M_bar**2)-R4_eff*X4*N4**2*M4**3*(t/z**2)*Q**4/(60.0*M_bar**2)-
R2_eff*R4_eff*X4*N4**2*M4**3*(2-2*t/z+
```

```
(t/z)**2)*Q**6/(360.0*M_bar**3)+R2_eff*R4_eff*X4*N4**2*M4**4*
(t/z**2)*Q**6/(360.0*M_bar**3))
104                 cs5 = Gf**2/(8*3.1415926)*(X5*N5**2*(2-2*t/z+(t/z)**2)-X5*N5**2*M5*
(t/z**2)-R2_eff*X5*N5**2*M5*(2-2*t/z+(t/z)**2)*Q**2/(3*M_bar)+R2_eff*X5*N5**2*M5**2*
(t/z**2)*Q**2/(3*M_bar)+R2_eff**2*X5*N5**2*M5**2*(2-2*t/z+
(t/z)**2)*Q**4/(36.0*M_bar**2)-R2_eff**2*X5*N5**2*M5**3*
(t/z**2)*Q**4/(36.0*M_bar**2)+R4_eff*X5*N5**2*M5**2*(2-2*t/z+
(t/z)**2)*Q**4/(60.0*M_bar**2)-R4_eff*X5*N5**2*M5**3*(t/z**2)*Q**4/(60.0*M_bar**2)-
R2_eff*R4_eff*X5*N5**2*M5**3*(2-2*t/z+
(t/z)**2)*Q**6/(360.0*M_bar**3)+R2_eff*R4_eff*X5*N5**2*M5**4*
(t/z**2)*Q**6/(360.0*M_bar**3))
105                 cross_section = cs1 + cs2 + cs3 + cs4 + cs5
106
107                 ## flux
108                 fu = 16/m**4*(3*m*z**2-4*z**3)
109                 fe = 96/m**4*(m*z**2-2*z**3)
110
111                 ## The full integral,multiply by (hbar*c)^2 to normalize.
112                 C = 10**11
113                 f2 = (hbar*c)**2*C*fe*cross_section*(3.156e7)*M_detector/1000
114                 return f2
115
116             ## The proton part for flux fu.
117             def integrand3(z):
118                 # Calculate dN/dt numerically, variable z is the energy E.
119                 Q = (2*(z**2)*t*M_bar/(z**2-z*t))**0.5/(hbar*c)
120                 # Five parts of cross section.
121                 cs1 = Gf**2/(8*3.1415926)*(X1*Z**2*(2-2*t/z+(t/z)**2)-X1*Z**2*M1*
(t/z**2)-R2_eff_p*X1*Z**2*Z*(2-2*t/z+(t/z)**2)*Q**2/(3*M_bar)+R2_eff_p*X1*Z**2*M1**2*
(t/z**2)*Q**2/(3*M_bar)+R2_eff_p**2*X1*Z**2*M1**2*(2-2*t/z+
(t/z)**2)*Q**4/(36.0*M_bar**2)-R2_eff_p**2*X1*Z**2*M1**3*
(t/z**2)*Q**4/(36.0*M_bar**2)+R4_eff_p*X1*Z**2*M1**2*(2-2*t/z+
(t/z)**2)*Q**4/(60.0*M_bar**2)-R4_eff_p*X1*Z**2*M1**3*(t/z**2)*Q**4/(60.0*M_bar**2)-
R2_eff_p*R4_eff_p*X1*Z**2*M1**3*(2-2*t/z+
(t/z)**2)*Q**6/(360.0*M_bar**3)+R2_eff_p*R4_eff_p*X1*Z**2*M1**4*
(t/z**2)*Q**6/(360.0*M_bar**3))
122                 cs2 = Gf**2/(8*3.1415926)*(X2*Z**2*(2-2*t/z+(t/z)**2)-X2*Z**2*M2*
(t/z**2)-R2_eff_p*X2*Z**2*Z*(2-2*t/z+(t/z)**2)*Q**2/(3*M_bar)+R2_eff_p*X2*Z**2*M2**2*
(t/z**2)*Q**2/(3*M_bar)+R2_eff_p**2*X2*Z**2*M2**2*(2-2*t/z+
(t/z)**2)*Q**4/(36.0*M_bar**2)-R2_eff_p**2*X2*Z**2*M2**3*
(t/z**2)*Q**4/(36.0*M_bar**2)+R4_eff_p*X2*Z**2*M2**2*(2-2*t/z+
(t/z)**2)*Q**4/(60.0*M_bar**2)-R4_eff_p*X2*Z**2*M2**3*(t/z**2)*Q**4/(60.0*M_bar**2)-
R2_eff_p*R4_eff_p*X2*Z**2*M2**3*(2-2*t/z+
(t/z)**2)*Q**6/(360.0*M_bar**3)+R2_eff_p*R4_eff_p*X2*Z**2*M2**4*
(t/z**2)*Q**6/(360.0*M_bar**3))
123                 cs3 = Gf**2/(8*3.1415926)*(X3*Z**2*(2-2*t/z+(t/z)**2)-X3*Z**2*M3*
(t/z**2)-R2_eff_p*X3*Z**2*Z*(2-2*t/z+(t/z)**2)*Q**2/(3*M_bar)+R2_eff_p*X3*Z**2*M3**2*
(t/z**2)*Q**2/(3*M_bar)+R2_eff_p**2*X3*Z**2*M3**2*(2-2*t/z+
(t/z)**2)*Q**4/(36.0*M_bar**2)-R2_eff_p**2*X3*Z**2*M3**3*
(t/z**2)*Q**4/(36.0*M_bar**2)+R4_eff_p*X3*Z**2*M3**2*(2-2*t/z+
(t/z)**2)*Q**4/(60.0*M_bar**2)-R4_eff_p*X3*Z**2*M3**3*(t/z**2)*Q**4/(60.0*M_bar**2)-
R2_eff_p*R4_eff_p*X3*Z**2*M3**3*(2-2*t/z+
(t/z)**2)*Q**6/(360.0*M_bar**3)+R2_eff_p*R4_eff_p*X3*Z**2*M3**4*
(t/z**2)*Q**6/(360.0*M_bar**3))
124                 cs4 = Gf**2/(8*3.1415926)*(X4*Z**2*(2-2*t/z+(t/z)**2)-X4*Z**2*M4*
(t/z**2)-R2_eff_p*X4*Z**2*Z*(2-2*t/z+(t/z)**2)*Q**2/(3*M_bar)+R2_eff_p*X4*Z**2*M4**2*
(t/z**2)*Q**2/(3*M_bar)+R2_eff_p**2*X4*Z**2*M4**2*(2-2*t/z+
(t/z)**2)*Q**4/(36.0*M_bar**2)-R2_eff_p**2*X4*Z**2*M4**3*
(t/z**2)*Q**4/(36.0*M_bar**2)+R4_eff_p*X4*Z**2*M4**2*(2-2*t/z+
(t/z)**2)*Q**4/(60.0*M_bar**2)-R4_eff_p*X4*Z**2*M4**3*(t/z**2)*Q**4/(60.0*M_bar**2)-
```

```
R2_eff_p*R4_eff_p*X4*Z**2*M4**3*(2-2*t/z+
(t/z)**2)*Q**6/(360.0*M_bar**3)+R2_eff_p*R4_eff_p*X4*Z**2*M4**4*
(t/z**2)*Q**6/(360.0*M_bar**3))
125                 cs5 = Gf**2/(8*3.1415926)*(X5*Z**2*(2-2*t/z+(t/z)**2)-X5*Z**2*M5*
(t/z**2)-R2_eff_p*X5*Z**2*Z*(2-2*t/z+(t/z)**2)*Q**2/(3*M_bar)+R2_eff_p*X5*Z**2*M5**2*
(t/z**2)*Q**2/(3*M_bar)+R2_eff_p**2*X5*Z**2*M5**2*(2-2*t/z+
(t/z)**2)*Q**4/(36.0*M_bar**2)-R2_eff_p**2*X5*Z**2*M5**3*
(t/z**2)*Q**4/(36.0*M_bar**2)+R4_eff_p*X5*Z**2*M5**2*(2-2*t/z+
(t/z)**2)*Q**4/(60.0*M_bar**2)-R4_eff_p*X5*Z**2*M5**3*(t/z**2)*Q**4/(60.0*M_bar**2)-
R2_eff_p*R4_eff_p*X5*Z**2*M5**3*(2-2*t/z+
(t/z)**2)*Q**6/(360.0*M_bar**3)+R2_eff_p*R4_eff_p*X5*Z**2*M5**4*
(t/z**2)*Q**6/(360.0*M_bar**3))
126                 cross_section = alpha**2*(cs1 + cs2 + cs3 + cs4 + cs5)
127
128                 ## flux
129                 fu = 16/m**4*(3*m*z**2-4*z**3)
130                 fe = 96/m**4*(m*z**2-2*z**3)
131
132                 ## The full integral,multiply by (hbar*c)^2 to normalize.
133                 C = 10**11
134                 f3 = (hbar*c)**2*C*fu*cross_section*(3.156e7)*M_detector/1000
135                 return f3
136
137             ## The proton part for flux fe.
138             def integrand4(z):
139                 # Calculate dN/dt numerically, variable z is the energy E.
140                 Q = (2*(z**2)*t*M_bar/(z**2-z*t))**0.5/(hbar*c)
141                 # Five parts of cross section.
142                 cs1 = Gf**2/(8*3.1415926)*(X1*Z**2*(2-2*t/z+(t/z)**2)-X1*Z**2*M1*
(t/z**2)-R2_eff_p*X1*Z**2*Z*(2-2*t/z+(t/z)**2)*Q**2/(3*M_bar)+R2_eff_p*X1*Z**2*M1**2*
(t/z**2)*Q**2/(3*M_bar)+R2_eff_p**2*X1*Z**2*M1**2*(2-2*t/z+
(t/z)**2)*Q**4/(36.0*M_bar**2)-R2_eff_p**2*X1*Z**2*M1**3*
(t/z**2)*Q**4/(36.0*M_bar**2)+R4_eff_p*X1*Z**2*M1**2*(2-2*t/z+
(t/z)**2)*Q**4/(60.0*M_bar**2)-R4_eff_p*X1*Z**2*M1**3*(t/z**2)*Q**4/(60.0*M_bar**2)-
R2_eff_p*R4_eff_p*X1*Z**2*M1**3*(2-2*t/z+
(t/z)**2)*Q**6/(360.0*M_bar**3)+R2_eff_p*R4_eff_p*X1*Z**2*M1**4*
(t/z**2)*Q**6/(360.0*M_bar**3))
143                 cs2 = Gf**2/(8*3.1415926)*(X2*Z**2*(2-2*t/z+(t/z)**2)-X2*Z**2*M2*
(t/z**2)-R2_eff_p*X2*Z**2*Z*(2-2*t/z+(t/z)**2)*Q**2/(3*M_bar)+R2_eff_p*X2*Z**2*M2**2*
(t/z**2)*Q**2/(3*M_bar)+R2_eff_p**2*X2*Z**2*M2**2*(2-2*t/z+
(t/z)**2)*Q**4/(36.0*M_bar**2)-R2_eff_p**2*X2*Z**2*M2**3*
(t/z**2)*Q**4/(36.0*M_bar**2)+R4_eff_p*X2*Z**2*M2**2*(2-2*t/z+
(t/z)**2)*Q**4/(60.0*M_bar**2)-R4_eff_p*X2*Z**2*M2**3*(t/z**2)*Q**4/(60.0*M_bar**2)-
R2_eff_p*R4_eff_p*X2*Z**2*M2**3*(2-2*t/z+
(t/z)**2)*Q**6/(360.0*M_bar**3)+R2_eff_p*R4_eff_p*X2*Z**2*M2**4*
(t/z**2)*Q**6/(360.0*M_bar**3))
144                 cs3 = Gf**2/(8*3.1415926)*(X3*Z**2*(2-2*t/z+(t/z)**2)-X3*Z**2*M3*
(t/z**2)-R2_eff_p*X3*Z**2*Z*(2-2*t/z+(t/z)**2)*Q**2/(3*M_bar)+R2_eff_p*X3*Z**2*M3**2*
(t/z**2)*Q**2/(3*M_bar)+R2_eff_p**2*X3*Z**2*M3**2*(2-2*t/z+
(t/z)**2)*Q**4/(36.0*M_bar**2)-R2_eff_p**2*X3*Z**2*M3**3*
(t/z**2)*Q**4/(36.0*M_bar**2)+R4_eff_p*X3*Z**2*M3**2*(2-2*t/z+
(t/z)**2)*Q**4/(60.0*M_bar**2)-R4_eff_p*X3*Z**2*M3**3*(t/z**2)*Q**4/(60.0*M_bar**2)-
R2_eff_p*R4_eff_p*X3*Z**2*M3**3*(2-2*t/z+
(t/z)**2)*Q**6/(360.0*M_bar**3)+R2_eff_p*R4_eff_p*X3*Z**2*M3**4*
(t/z**2)*Q**6/(360.0*M_bar**3))
145                 cs4 = Gf**2/(8*3.1415926)*(X4*Z**2*(2-2*t/z+(t/z)**2)-X4*Z**2*M4*
(t/z**2)-R2_eff_p*X4*Z**2*Z*(2-2*t/z+(t/z)**2)*Q**2/(3*M_bar)+R2_eff_p*X4*Z**2*M4**2*
(t/z**2)*Q**2/(3*M_bar)+R2_eff_p**2*X4*Z**2*M4**2*(2-2*t/z+
(t/z)**2)*Q**4/(36.0*M_bar**2)-R2_eff_p**2*X4*Z**2*M4**3*
(t/z**2)*Q**4/(36.0*M_bar**2)+R4_eff_p*X4*Z**2*M4**2*(2-2*t/z+
```

```
(t/z)**2)*Q**4/(60.0*M_bar**2)-R4_eff_p*X4*Z**2*M4**3*(t/z**2)*Q**4/(60.0*M_bar**2)-
R2_eff_p*R4_eff_p*X4*Z**2*M4**3*(2-2*t/z+
(t/z)**2)*Q**6/(360.0*M_bar**3)+R2_eff_p*R4_eff_p*X4*Z**2*M4**4*
(t/z**2)*Q**6/(360.0*M_bar**3))
                cs5 = Gf**2/(8*3.1415926)*(X5*Z**2*(2-2*t/z+(t/z)**2)-X5*Z**2*M5*
(t/z**2)-R2_eff_p*X5*Z**2*Z*(2-2*t/z+(t/z)**2)*Q**2/(3*M_bar)+R2_eff_p*X5*Z**2*M5**2*
(t/z**2)*Q**2/(3*M_bar)+R2_eff_p**2*X5*Z**2*M5**2*(2-2*t/z+
(t/z)**2)*Q**4/(36.0*M_bar**2)-R2_eff_p**2*X5*Z**2*M5**3*
(t/z**2)*Q**4/(36.0*M_bar**2)+R4_eff_p*X5*Z**2*M5**2*(2-2*t/z+
(t/z)**2)*Q**4/(60.0*M_bar**2)-R4_eff_p*X5*Z**2*M5**3*(t/z**2)*Q**4/(60.0*M_bar**2)-
R2_eff_p*R4_eff_p*X5*Z**2*M5**3*(2-2*t/z+
(t/z)**2)*Q**6/(360.0*M_bar**3)+R2_eff_p*R4_eff_p*X5*Z**2*M5**4*
(t/z**2)*Q**6/(360.0*M_bar**3))
                cross_section = alpha**2*(cs1 + cs2 + cs3 + cs4 + cs5)

                ## flux
                fu = 16/m**4*(3*m*z**2-4*z**3)
                fe = 96/m**4*(m*z**2-2*z**3)

                ## The full integral,multiply by (hbar*c)^2 to normalize.
                C = 10**11
                f4 = (hbar*c)**2*C*fe*cross_section*(3.156e7)*M_detector/1000
                return f4
            ## The Cross term for flux fu and fe.
            def integrand5(z,R2,R4):
                R2_eff = R2
                R4_eff = R4
                # Calculate dN/dt numerically, variable z is the energy E.
                Q = (2*(z**2)*t*M_bar/(z**2-z*t))**0.5/(hbar*c)
                # Five parts of cross section.
                cs1 = Gf**2/(8*3.1415926)*(X1*N1*Z*(2-2*t/z+(t/z)**2)-X1*N1*Z*M1*
(t/z**2)-R2_eff*X1*N1**2*M1*(2-2*t/z+(t/z)**2)*Q**2/(6*M_bar)+R2_eff*X1*N1**2*M1**2*
(t/z**2)*Q**2/(6*M_bar)-R2_eff_p*X1*Z**2*M1*(2-2*t/z+
(t/z)**2)*Q**2/(6*M_bar)+R2_eff_p*X1*Z**2*M1**2*
(t/z**2)*Q**2/(6*M_bar)+R2_eff*R2_eff_p*X1*N1*Z*M1**2*(2-2*t/z+
(t/z)**2)*Q**4/(36.0*M_bar**2)-R2_eff*R2_eff_p*X1*N1*Z*M1**3*
(t/z**2)*Q**4/(36.0*M_bar**2)+R4_eff*X1*N1**2*M1**2*(2-2*t/z+
(t/z)**2)*Q**4/(120.0*M_bar**2)-R4_eff*X1*N1**2*M1**3*
(t/z**2)*Q**4/(120.0*M_bar**2)+R4_eff_p*X1*Z**2*M1**2*(2-2*t/z+
(t/z)**2)*Q**4/(120.0*M_bar**2)-R4_eff_p*X1*Z**2*M1**3*
(t/z**2)*Q**4/(120.0*M_bar**2)-R2_eff_p*R4_eff*X1*N1*Z*M1**3*(2-2*t/z+
(t/z)**2)*Q**6/(720.0*M_bar**3)+R2_eff_p*R4_eff*X1*N1*Z*M1**4*
(t/z**2)*Q**6/(720.0*M_bar**3)-R2_eff*R4_eff_p*X1*N1*Z*M1**3*(2-2*t/z+
(t/z)**2)*Q**6/(720.0*M_bar**3)+R2_eff*R4_eff_p*X1*N1*Z*M1**4*
(t/z**2)*Q**6/(720.0*M_bar**3))
                cs2 = Gf**2/(8*3.1415926)*(X2*N2*Z*(2-2*t/z+(t/z)**2)-X2*N2*Z*M2*
(t/z**2)-R2_eff*X2*N2**2*M2*(2-2*t/z+(t/z)**2)*Q**2/(6*M_bar)+R2_eff*X2*N2**2*M2**2*
(t/z**2)*Q**2/(6*M_bar)-R2_eff_p*X2*Z**2*M2*(2-2*t/z+
(t/z)**2)*Q**2/(6*M_bar)+R2_eff_p*X2*Z**2*M2**2*
(t/z**2)*Q**2/(6*M_bar)+R2_eff*R2_eff_p*X2*N2*Z*M2**2*(2-2*t/z+
(t/z)**2)*Q**4/(36.0*M_bar**2)-R2_eff*R2_eff_p*X2*N2*Z*M2**3*
(t/z**2)*Q**4/(36.0*M_bar**2)+R4_eff*X2*N2**2*M2**2*(2-2*t/z+
(t/z)**2)*Q**4/(120.0*M_bar**2)-R4_eff*X2*N2**2*M2**3*
(t/z**2)*Q**4/(120.0*M_bar**2)+R4_eff_p*X2*Z**2*M2**2*(2-2*t/z+
(t/z)**2)*Q**4/(120.0*M_bar**2)-R4_eff_p*X2*Z**2*M2**3*
(t/z**2)*Q**4/(120.0*M_bar**2)-R2_eff_p*R4_eff*X2*N2*Z*M2**3*(2-2*t/z+
(t/z)**2)*Q**6/(720.0*M_bar**3)+R2_eff_p*R4_eff*X2*N2*Z*M2**4*
(t/z**2)*Q**6/(720.0*M_bar**3)-R2_eff*R4_eff_p*X2*N2*Z*M2**3*(2-2*t/z+
(t/z)**2)*Q**6/(720.0*M_bar**3)+R2_eff*R4_eff_p*X2*N2*Z*M2**4*
(t/z**2)*Q**6/(720.0*M_bar**3))
```

```
166                 cs3 = Gf**2/(8*3.1415926)*(X3*N3*Z*(2-2*t/z+(t/z)**2)-X3*N3*Z*M3*
(t/z**2)-R2_eff*X3*N3**2*M3*(2-2*t/z+(t/z)**2)*Q**2/(6*M_bar)+R2_eff*X3*N3**2*M3**2*
(t/z**2)*Q**2/(6*M_bar)-R2_eff_p*X3*Z**2*M3*(2-2*t/z+
(t/z)**2)*Q**2/(6*M_bar)+R2_eff_p*X3*Z**2*M3**2*
(t/z**2)*Q**2/(6*M_bar)+R2_eff*R2_eff_p*X3*N3*Z*M3**2*(2-2*t/z+
(t/z)**2)*Q**4/(36.0*M_bar**2)-R2_eff*R2_eff_p*X3*N3*Z*M3**3*
(t/z**2)*Q**4/(36.0*M_bar**2)+R4_eff*X3*N3**2*M3**2*(2-2*t/z+
(t/z)**2)*Q**4/(120.0*M_bar**2)-R4_eff*X3*N3**2*M3**3*
(t/z**2)*Q**4/(120.0*M_bar**2)+R4_eff_p*X3*Z**2*M3**2*(2-2*t/z+
(t/z)**2)*Q**4/(120.0*M_bar**2)-R4_eff_p*X3*Z**2*M3**3*
(t/z**2)*Q**4/(120.0*M_bar**2)-R2_eff_p*R4_eff*X3*N3*Z*M3**3*(2-2*t/z+
(t/z)**2)*Q**6/(720.0*M_bar**3)+R2_eff_p*R4_eff*X3*N3*Z*M3**4*
(t/z**2)*Q**6/(720.0*M_bar**3)-R2_eff*R4_eff_p*X3*N3*Z*M3**3*(2-2*t/z+
(t/z)**2)*Q**6/(720.0*M_bar**3)+R2_eff*R4_eff_p*X3*N3*Z*M3**4*
(t/z**2)*Q**6/(720.0*M_bar**3))
167                 cs4 = Gf**2/(8*3.1415926)*(X4*N4*Z*(2-2*t/z+(t/z)**2)-X4*N4*Z*M4*
(t/z**2)-R2_eff*X4*N4**2*M4*(2-2*t/z+(t/z)**2)*Q**2/(6*M_bar)+R2_eff*X4*N4**2*M4**2*
(t/z**2)*Q**2/(6*M_bar)-R2_eff_p*X4*Z**2*M4*(2-2*t/z+
(t/z)**2)*Q**2/(6*M_bar)+R2_eff_p*X4*Z**2*M4**2*
(t/z**2)*Q**2/(6*M_bar)+R2_eff*R2_eff_p*X4*N4*Z*M4**2*(2-2*t/z+
(t/z)**2)*Q**4/(36.0*M_bar**2)-R2_eff*R2_eff_p*X4*N4*Z*M4**3*
(t/z**2)*Q**4/(36.0*M_bar**2)+R4_eff*X4*N4**2*M4**2*(2-2*t/z+
(t/z)**2)*Q**4/(120.0*M_bar**2)-R4_eff*X4*N4**2*M4**3*
(t/z**2)*Q**4/(120.0*M_bar**2)+R4_eff_p*X4*Z**2*M4**2*(2-2*t/z+
(t/z)**2)*Q**4/(120.0*M_bar**2)-R4_eff_p*X4*Z**2*M4**3*
(t/z**2)*Q**4/(120.0*M_bar**2)-R2_eff_p*R4_eff*X4*N4*Z*M4**3*(2-2*t/z+
(t/z)**2)*Q**6/(720.0*M_bar**3)+R2_eff_p*R4_eff*X4*N4*Z*M4**4*
(t/z**2)*Q**6/(720.0*M_bar**3)-R2_eff*R4_eff_p*X4*N4*Z*M4**3*(2-2*t/z+
(t/z)**2)*Q**6/(720.0*M_bar**3)+R2_eff*R4_eff_p*X4*N4*Z*M4**4*
(t/z**2)*Q**6/(720.0*M_bar**3))
168                 cs5 = Gf**2/(8*3.1415926)*(X5*N5*Z*(2-2*t/z+(t/z)**2)-X5*N5*Z*M5*
(t/z**2)-R2_eff*X5*N5**2*M5*(2-2*t/z+(t/z)**2)*Q**2/(6*M_bar)+R2_eff*X5*N5**2*M5**2*
(t/z**2)*Q**2/(6*M_bar)-R2_eff_p*X5*Z**2*M5*(2-2*t/z+
(t/z)**2)*Q**2/(6*M_bar)+R2_eff_p*X5*Z**2*M5**2*
(t/z**2)*Q**2/(6*M_bar)+R2_eff*R2_eff_p*X5*N5*Z*M5**2*(2-2*t/z+
(t/z)**2)*Q**4/(36.0*M_bar**2)-R2_eff*R2_eff_p*X5*N5*Z*M5**3*
(t/z**2)*Q**4/(36.0*M_bar**2)+R4_eff*X5*N5**2*M5**2*(2-2*t/z+
(t/z)**2)*Q**4/(120.0*M_bar**2)-R4_eff*X5*N5**2*M5**3*
(t/z**2)*Q**4/(120.0*M_bar**2)+R4_eff_p*X5*Z**2*M5**2*(2-2*t/z+
(t/z)**2)*Q**4/(120.0*M_bar**2)-R4_eff_p*X5*Z**2*M5**3*
(t/z**2)*Q**4/(120.0*M_bar**2)-R2_eff_p*R4_eff*X5*N5*Z*M5**3*(2-2*t/z+
(t/z)**2)*Q**6/(720.0*M_bar**3)+R2_eff_p*R4_eff*X5*N5*Z*M5**4*
(t/z**2)*Q**6/(720.0*M_bar**3)-R2_eff*R4_eff_p*X5*N5*Z*M5**3*(2-2*t/z+
(t/z)**2)*Q**6/(720.0*M_bar**3)+R2_eff*R4_eff_p*X5*N5*Z*M5**4*
(t/z**2)*Q**6/(720.0*M_bar**3))
169                 cross_section =-1*alpha*2*(cs1 + cs2 + cs3 + cs4 + cs5)
170
171                 ## flux
172                 fu = 16/m**4*(3*m*z**2-4*z**3)
173                 fe = 96/m**4*(m*z**2-2*z**3)
174
175                 ## The full integral,multiply by (hbar*c)^2 to normalize.
176                 C = 10**11
177                 f5 = (hbar*c)**2*C*(fu+fe)*cross_section*(3.156e7)*M_detector/1000
178                 return f5
179
180             I1 = scipy.integrate.quad(integrand1,(t+sqrt(t**2+2*t*M_bar))/2,m/2,args=
(R2,R4))
181             I2 = scipy.integrate.quad(integrand2,(t+sqrt(t**2+2*t*M_bar))/2,m/2,args=
(R2,R4))
```

```
            I3 = scipy.integrate.quad(integrand3,(t+sqrt(t**2+2*t*M_bar))/2,m/2)
            I4 = scipy.integrate.quad(integrand4,(t+sqrt(t**2+2*t*M_bar))/2,m/2)
            I5 = scipy.integrate.quad(integrand5,(t+sqrt(t**2+2*t*M_bar))/2,m/2,args=(R2,R4))
            ssum = I1[0]+I2[0]+I3[0]+I4[0]+I5[0]
            #print(ssum)
            sum_bin = sum_bin + ssum
            event.append(int(ssum))

        event_bin.append(int(sum_bin))
    #print(event)
    return (event,event_bin)


# PART3: Plots and Exporting data.
#################################################################################
energy = np.linspace(1,80,num=80)
energy2 = np.linspace(5,80,num=16)
(event1,event_bin1) = dNdt(R2,R4)
# (event2,event_bin2) = dNdt(R2*0.9**2,R4*0.9**4)
# (event3,event_bin3) = dNdt(R2*0.8**2,R4*0.8**4)
# (event4,event_bin4) = dNdt(R2*1.1**2,R4*1.1**4)
# (event5,event_bin5) = dNdt(R2*1.2**2,R4*1.2**4)

# print("event=",event1)
# print("binned event=",event_bin1)

# table = pd.DataFrame({ 'Bin Range(keV)': energy, 'Events' : event1, 'Events (-10%)' : event2, 'Events (-20%)' : event3, 'Events (+10%)' : event4, 'Events (+20%)' : event5})
# table.to_csv('table_Ge_with_proton.csv', index=False, encoding='utf-8')


# #plt.plot(energy,event1)
# plt.bar(energy2,event_bin1,align='center',width=4)
# plt.xticks(energy2)
# plt.ylabel("Number of Counts")
# plt.xlabel("Energy/keV")
# plt.title("# of Events per (keV year tonne) for Ge")
# for i, v in enumerate(event_bin1):
#     plt.text(energy2[i] -0.25, v , str(v))
# plt.show()

# Bar Plot:
plt.rcParams['font.size'] = 10
plt.rcParams['axes.linewidth'] = 2
fig = plt.figure()
ax = fig.add_axes([0, 0, 1, 1])
# Edit the major and minor ticks of the x and y axes
ax.xaxis.set_tick_params(which='major', size=2, width=2, direction='in', top='on')
ax.yaxis.set_tick_params(which='major', size=5, width=2, direction='in', right='on')
ax.yaxis.set_tick_params(which='minor', size=2, width=2, direction='in', right='on')

# Set the axis limits
ax.set_xlim(0, 80)
ax.set_ylim(0, 10000)
# Edit the major and minor tick locations
ax.yaxis.set_major_locator(MultipleLocator(1000))
ax.yaxis.set_major_formatter(FormatStrFormatter('%d'))
ax.yaxis.set_minor_locator(AutoMinorLocator(2))
```

```
239
240 plt.bar(energy2,event_bin1,align='center',width=4)
241 plt.xticks(energy2)
242 plt.xticks(rotation=90)
243 plt.ylabel("Number of Counts (keV year tonne)")
244 plt.xlabel("Nucleus Recoil Energy (keV)")
245 for i, v in enumerate(event_bin1):
246     plt.text(energy2[i] -0.25, v , str(v))
247 plt.savefig('Scattering_Events_Ge_with_proton_binned.png', dpi=300,
    transparent=False, bbox_inches='tight')
248 plt.show()
249
250
251 # ## Second Plot: Compare the difference between this and the one without proton.
252 # (energy0,event0) = np.loadtxt('table_Ge_data.csv', unpack=True, delimiter=',',
    skiprows=1)
253 # diff = event1-event0
254 # plt.rcParams['font.size'] = 10
255 # plt.rcParams['axes.linewidth'] = 2
256 # fig = plt.figure()
257 # ax2 = fig.add_axes([0, 0, 1, 0.4])
258 # ax1 = fig.add_axes([0, 0.6, 1, 0.4])
259
260 # # # Set the axis limits
261 # ax1.set_xlim(0, 80)
262 # ax1.set_ylim(0, 2500)
263 # ax2.set_xlim(0, 80)
264 # ax2.set_ylim(-300, 80)
265 # # Edit the major and minor ticks of the x and y axes
266 # ax1.xaxis.set_tick_params(which='major', size=10, width=2, direction='in',
    top='on')
267 # ax1.xaxis.set_tick_params(which='minor', size=7, width=2, direction='in', top='on')
268 # ax1.yaxis.set_tick_params(which='major', size=10, width=2, direction='in',
    right='on')
269 # ax1.yaxis.set_tick_params(which='minor', size=7, width=2, direction='in',
    right='on')
270 # ax2.xaxis.set_tick_params(which='major', size=10, width=2, direction='in',
    top='on')
271 # ax2.xaxis.set_tick_params(which='minor', size=7, width=2, direction='in', top='on')
272 # ax2.yaxis.set_tick_params(which='major', size=10, width=2, direction='in',
    right='on')
273 # ax2.yaxis.set_tick_params(which='minor', size=7, width=2, direction='in',
    right='on')
274
275 # ax1.plot(energy,event1,label='With Proton and Cross Terms')
276 # ax1.plot(energy,event0,label='Without Proton and Cross Terms')
277 # ax2.plot(energy,diff)
278
279 # # Edit the major and minor tick locations
280 # ax1.xaxis.set_major_locator(MultipleLocator(20))
281 # ax1.xaxis.set_major_formatter(FormatStrFormatter('%d'))
282 # ax1.xaxis.set_minor_locator(AutoMinorLocator(5))
283 # ax1.yaxis.set_major_locator(MultipleLocator(500))
284 # ax1.yaxis.set_major_formatter(FormatStrFormatter('%d'))
285 # ax1.yaxis.set_minor_locator(AutoMinorLocator(5))
286 # ax2.xaxis.set_major_locator(MultipleLocator(20))
287 # ax2.xaxis.set_major_formatter(FormatStrFormatter('%d'))
288 # ax2.xaxis.set_minor_locator(AutoMinorLocator(5))
289 # ax2.yaxis.set_major_locator(MultipleLocator(50))
290 # ax2.yaxis.set_major_formatter(FormatStrFormatter('%d'))
```

```
# ax2.yaxis.set_minor_locator(AutoMinorLocator(5))
# # Add legend to plot
# ax1.legend(bbox_to_anchor=(0.95, 0.95),loc=1, frameon=False, fontsize=10)
# ax2.set_xlabel('Nucleus Recoil Energy (keV)', labelpad=10)
# ax2.set_ylabel('Difference in Events/(keV year tonne)', labelpad=10)
# ax1.set_ylabel('Events/(keV year tonne)', labelpad=10)
# plt.savefig('Scattering_Events_Ge_diff.png', dpi=300, transparent=False,
bbox_inches='tight')
# plt.show()


```

```python
## Hongyong Zhang
## Run Monte Carlo Simulations.
## 05/19/2020
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.pyplot as plt
import scipy.integrate
import pandas as pd
from numpy import cos, sin, exp, sqrt
from dNdt_Ge_binned import dNdt
import random
from scipy.stats import chisquare

## Part1: Generate the database.
#############################################################################################
R2 = (4.0495e-15)**2
R4 = (4.3765e-15)**4
events = []
## Generate the standard events for reference first
standard_event = [R2,R4]
(result,result_binned) = dNdt(R2,R4)
standard_event = standard_event + result_binned
#print(standard_event)

## Create database of events using R2 +/- 15% and R4 +/- 30% with 100 values for
each.
for i in range(100):
    R2 = (sqrt(R2)*0.3/100*(i+1)+sqrt(R2)*(1-0.15))**2
    #print(i)
    for p in range(100):
        #print(p)
        R4 = (R4**0.25*0.6/100*(i+1)+R4**0.25*(1-0.3))**4
        event = [R2,R4]
        (result,result_binned) = dNdt(R2,R4)
        event = event + result_binned
        events.append(event)
        #print(event)
np.save("database.npy",events)
# To load the data: Use database = np.load("database.npy")
print(events)



## Part2: Run the Monte Carlo simulation.
#############################################################################################
R2 = (4.0495e-15)**2
R4 = (4.3765e-15)**4
minimal_R2=[]
minimal_R4=[]
for i in range(5000):

    event_with_noise=[R2,R4]
    ## generate noise for each term
    for p in range(16):
        noise = random.uniform(-1,1)*sqrt(standard_event[p+2]) + standard_event[p+2]
        event_with_noise.append(noise)
    chi_square=[]
    ## chi square test
    for q in range(len(events)):
        expected = events[q][2:17]
```

```
        obs = event_with_noise[2:17]
        (chisq,p) = chisquare(obs,f_exp=expected)
        chi_square.append(chisq)
    ## find the minimum value and the corresponding R2 and R4
    minimal_value_R2R4 = events[np.argmin(chi_square)][0:2]
    minimal_R2.append(minimal_value_R2R4[0])
    minimal_R4.append(minimal_value_R2R4[1])

    print(minimal_value_R2R4)
    ##print(event_with_noise)


## Part3: Plots and Tables
#################################################################################################
## Scatter Plot
min_R2 = min(minimal_R2)
max_R2 = max(minimal_R2)
maxlim_R2 = max_R2 + abs((max_R2 - min_R2)*0.2)
minlim_R2 = min_R2 - abs((max_R2 - min_R2)*0.2)
min_R4 = min(minimal_R4)
max_R4 = max(minimal_R4)
maxlim_R4 = max_R4 + abs((max_R4 - min_R4)*0.2)
minlim_R4 = min_R4 - abs((max_R4 - min_R4)*0.2)
plt.scatter(minimal_R2,minimal_R4)
plt.xlim(minlim_R2,maxlim_R2)
plt.ylim(minlim_R4,maxlim_R4)
plt.show()

table = pd.DataFrame({ 'R2': minimal_R2, 'R4' : minimal_R4})
#table.to_csv('Monte_Carlo_new.csv', index=False, encoding='utf-8')
```

```python
## Hongyong Zhang
## Plot the result of the Monte Carlo simulation.
## 05/19/2020
import matplotlib as mpl
import matplotlib.pyplot as plt
from matplotlib import rc
from matplotlib.ticker import (MultipleLocator, FormatStrFormatter,AutoMinorLocator)
import numpy as np
import scipy.integrate
import pandas as pd
from numpy import cos, sin, exp, sqrt

(R2,R4) = np.loadtxt('Monte_Carlo_Ge.csv', unpack=True, delimiter=',', skiprows=1)

min_R2 = min(R2)
max_R2 = max(R2)
maxlim_R2 = max_R2 + abs((max_R2 - min_R2)*0.2)
minlim_R2 = min_R2 - abs((max_R2 - min_R2)*0.2)
min_R4 = min(R4)
max_R4 = max(R4)
maxlim_R4 = max_R4 + abs((max_R4 - min_R4)*0.2)
minlim_R4 = min_R4 - abs((max_R4 - min_R4)*0.2)
# Plot:
plt.rcParams['font.size'] = 10
plt.rcParams['axes.linewidth'] = 2
fig = plt.figure()
ax = fig.add_axes([0, 0, 1, 1])
# Edit the major and minor ticks of the x and y axes
ax.xaxis.set_tick_params(which='major', size=10, width=2, direction='in', top='on')
ax.xaxis.set_tick_params(which='minor', size=7, width=2, direction='in', top='on')
ax.yaxis.set_tick_params(which='major', size=10, width=2, direction='in', right='on')
ax.yaxis.set_tick_params(which='minor', size=7, width=2, direction='in', right='on')

ax.scatter(R2,R4)

# Set the axis limits
ax.set_xlim(minlim_R2, maxlim_R2)
ax.set_ylim(minlim_R4, maxlim_R4)
# Edit the major and minor tick locations
ax.xaxis.set_major_locator(MultipleLocator(0.1))
ax.xaxis.set_major_formatter(FormatStrFormatter('%.2f'))
ax.xaxis.set_minor_locator(AutoMinorLocator(5))
ax.yaxis.set_major_locator(MultipleLocator(0.4))
ax.yaxis.set_major_formatter(FormatStrFormatter('%.2f'))
ax.yaxis.set_minor_locator(AutoMinorLocator(5))

# Add legend to plot
ax.set_xlabel(r'$\langle{R^2_n}\rangle^{0.5}_{eff} (fm)$', labelpad=10)
ax.set_ylabel(r'$\langle{R^4_n}\rangle^{0.25}_{eff} (fm)$', labelpad=10)
plt.savefig('Monte_Carlo_Ge.png', dpi=300, transparent=False, bbox_inches='tight')
plt.show()

```