Computational Modeling of Radiation Interactions with Molecular Nitrogen

Tyler Reese

Follow this and additional works at: https://aquila.usm.edu/dissertations

Part of the Atomic, Molecular and Optical Physics Commons, and the Numerical Analysis and Scientific Computing Commons

Recommended Citation

https://aquila.usm.edu/dissertations/1609

This Dissertation is brought to you for free and open access by The Aquila Digital Community. It has been accepted for inclusion in Dissertations by an authorized administrator of The Aquila Digital Community. For more information, please contact Joshua.Cromwell@usm.edu.
Computational Modeling of Radiation Interactions with Molecular Nitrogen

by

Tyler Webster Reese

A Dissertation
Submitted to the Graduate School,
the College of Arts and Sciences
and the School of Mathematics and Natural Sciences
at The University of Southern Mississippi
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

Approved by:

Dr. Chris Winstead, Committee Chair
Dr. Khin Maung Maung
Dr. Jeremy Scott
Dr. Michael Vera

Dr. Chris Winstead
Committee Chair

Dr. Bernd Schroeder
Director of School

Dr. Karen S. Coats
Dean of the Graduate School

December 2018
ABSTRACT

The ability to detect radiation through identifying secondary effects it has on its surrounding medium would extend the range at which detections could be made and would be a valuable asset to many industries. The development of such a detection instrument requires an accurate prediction of these secondary effects. This research aims to improve on existing modeling techniques and help provide a method for predicting results for an affected medium in the presence of radioactive materials. A review of radioactivity and the interactions mechanisms for emitted particles as well as a brief history of the Monte Carlo Method and its application to radiation transport are provided.

Many software packages have been previously developed for the purposes of radiation transport modeling efforts. Geant4 was selected, implemented, and evaluated as a candidate for the inclusion of molecular interaction mechanisms. It was determined that Geant4 requires further customization to accurately account for the prediction of very low energy secondary electrons, and these interaction calculations must still be handled separately. Additionally, it was determined that the distance traveled by low energy secondary electrons is small enough to be considered negligible relative to the path length of the primary. This allowed for a transition away from performing a transport calculation to a separate approach for predicting the effect on the medium. Subsequently, the calculations for the newly developed Averaged Loss Method and its implementation are described. When compared to results obtained through a separate Monte Carlo simulation, the Averaged Loss Method is found to be successful and potential avenues for the further improvement of this method are outlined as well.
ACKNOWLEDGMENTS

I would like to thank Dr. Christopher D. Walker for his correspondence on this project and providing the source code that would ultimately make generating the results necessary for validation possible. I would also like to thank my committee members, Dr. Khin Maung Maung, Dr. Jeremy Scott, and Dr. Michael Vera, for their support and guidance in the preparation of this dissertation. And I would especially like to thank my committee chair and advisor, Dr. Chris Winstead, who has been a remarkable influence on my life, both academic and otherwise, and has been my mentor throughout my entire graduate career.
# TABLE OF CONTENTS

ABSTRACT ........................................................................................................................................ ii

ACKNOWLEDGMENTS ......................................................................................................................... iii

LIST OF TABLES ................................................................................................................................... vi

LIST OF ILLUSTRATIONS .................................................................................................................. vii

CHAPTER I - Introduction .................................................................................................................... 1

CHAPTER II – Review of Radioactivity and Monte Carlo Methods ......................................................... 3
  2.1 Radioactivity .................................................................................................................................... 3
  2.2 Interaction Mechanisms .................................................................................................................... 5
  2.3 Monte Carlo Methods ....................................................................................................................... 18
  2.4 Modeling Radiation Transport ......................................................................................................... 20

CHAPTER III – Implementation of Existing Monte Carlo Software ...................................................... 22
  3.1 Basic Requirements of a Geant4 Simulation ..................................................................................... 23
  3.2 Implementation of Geant4 ................................................................................................................ 24

CHAPTER IV – Assessing Limitations of Existing Software ................................................................. 27

CHAPTER V – A New Approach to Low Energy Interactions .................................................................. 32
  5.1 The Averaged Loss Method ............................................................................................................. 32

CHAPTER VI – Results & Performance Evaluation ............................................................................. 38

CHAPTER VII – Conclusions ................................................................................................................ 43

APPENDIX A – Source Code for the Averaged Loss Method ............................................................... 45
LIST OF TABLES

Table 2.1 Molecular Nitrogen State Term Symbols ................................................................. 8
Table 2.2 Largest Franck-Condon factors for each state and their resulting transition energy ........................................................................................................................................ 18
Table 3.1 Comparison of Geant4 isotope models’ maximum range to the approximate anticipated range based on stopping power and CSDA range calculations ........................................... 25
Table 5.1 Sample calculation of the averaged energy lost due to interactions at 1 keV ........ 35
Table 6.2 Comparison of Monte Carlo and Averaged Loss Method generated W/e values at select values. Monte Carlo values are the result of the full 100,000 particle simulation. ........................................................................................................................................ 41
LIST OF ILLUSTRATIONS

Figure 2.1 Atomic vs Molecular Nitrogen Cross Section Comparison. .......................... 7
Figure 2.2 Molecular Nitrogen Excitation Cross Sections. ............................................. 11
Figure 2.3 Molecular Nitrogen Ionization Cross Sections ............................................. 12
Figure 2.4 Molecular Nitrogen Interaction Probabilities: 0 to 1000 eV ......................... 13
Figure 2.5 Molecular Nitrogen Interaction Probabilities: 0 to 75 eV ............................... 14
Figure 2.6 Example Secondary Electron Spectrum ....................................................... 14
Figure 2.7 Demonstration of Franck-Condon Principle. ................................................ 16
Figure 3.1 Geant4 Isotope Visualizations ................................................................. 25
Figure 4.1 Geant4 Sub-keV electron spectrum prediction .............................................. 28
Figure 4.2 Geant4 and MCAFS predicted Sub-keV electron spectra ............................... 31
Figure 5.1 Demonstration of the 1 keV step in the Averaged Loss Method. Note that product count examples for each state correspond to those in Table 5.1 ...................... 36
Figure 6.1 Average Work Per Ion vs Number of Particles Modeled. ............................... 39
Figure 6.2 Assessment of convergence of Monte Carlo results ..................................... 40
Figure 6.3 Average Work Required Per Ion vs Incident Energy Comparison ................. 41
Figure 6.4 Comparison of Resulting Yield of Molecular Nitrogen State ....................... 42
CHAPTER I - Introduction

Improving detection and modeling capabilities for radiation is of interest to a wide variety of industries ranging from defense to medical to space exploration. Current detection technologies are limited to direct impingement of the radiative particle on the sensor. This requires the sensor to be in close proximity to the source in most cases due to strong attenuation as these particles pass through surrounding materials and atmospheric gases. This also means that any sensor that requires a technician present to operate would potentially be putting that technician in direct contact as well. An improvement to the understanding of secondary effects induced in the medium by radiation could potentially lead to an improved detection method based on indirect detection instead of this direct impingement. A successful indirect detection mechanism would extend the range at which detection could be achieved as well as provide more separation between any technicians in the field and potentially hazardous exposure.

Unfortunately, the current modeling capabilities for predicting secondary effects are significantly lacking. While there are a variety of modeling software packages available to simulate radiation transport, they are largely focused on the effects of the system on the energetic particle (i.e. how far did it go, how quickly or slowly did it lose energy). This focus makes sense when one is interested in determining where best to put a direct impingement sensor or how much shielding is required to prevent over exposure. But the models do not provide predictions regarding how the particle’s existence and propagation through a region changed the materials in that region. The model scheme of treating the medium through which the particle passes as unaffected does improve the performance characteristics of these models by reducing runtime and memory
requirements, but this affected medium information is precisely what is needed to begin modeling secondary effects and allow for predictions of indirect detection mechanisms. Identifying a change in the chemical make up of a gas (such as ions and free atoms that were previously neutral molecules) can allow one to predict chemical reactions and products that were previously unavailable. Detection of such products is one possible avenue to indirect detection of radiation.

While there have been similarly oriented efforts\(^1\), these were limited by the available data and techniques at the time (circa 1974) and have become significantly outdated. This project endeavors to develop a multi stage approach for computational predictions of how radiation affects the medium in which the radiation deposits its energy. The goal is to eventually be able to take advantage of the existing modeling software, with its many benefits and efficiencies, and follow up with a separate calculation to determine the effect on the molecules that occupy the region of interest.

Presented here is a review of pertinent concepts from radioactive decay and radiation interaction mechanisms to the development and application of the Monte Carlo method. An existing modeling toolkit, Geant4, is implemented, and its potential benefits as well as its shortcomings as it applies to the effort are investigated. A new technique for calculating molecular interaction results, instead of the traditional radiation transport approach, is proposed and validated against a separate transport based Monte Carlo based simulation. Finally, these results are presented, and the preliminary success of this new technique is established. From this assessment, key areas for improvement in this new technique are also identified.
Efficient detection of the presence of radiation and determination of radiation levels are of interest to many different industries. Successful detection is dependent on being well informed on what one expects to detect and how that relates to the radiation source itself. The following is a brief review of radioactivity, interaction mechanisms for particles emitted as radiation, the principles behind modeling such interactions computationally, and the basic framework of an implementation of such modeling techniques.

2.1 Radioactivity

Radioactive decay occurs when the nucleus of an unstable isotope emits a particle. If this emission does not yield a stable nucleus, additional decay processes may occur until a stable isotope is reached. These decays are a random process, with each isotope having a particular duration over which half of the atoms present will have decayed, referred to as its half-life. The random nature of this process means that one cannot predict precisely when a decay will occur. However, for a given isotope of a known quantity, the average rate at which decays occur can be predicted. This rate is known as a source’s radioactivity and is measured in curies (Ci), with 1 Ci equal to $3.7 \times 10^{10}$ decays per second.

Ionizing radiation is any radiation that has sufficient energy to ionize the surrounding material. The three main types of ionizing radioactive decay are alpha, beta, and gamma emissions. An alpha particle is effectively a Helium nucleus, comprised of two protons and two neutrons. This means alpha particles carry a +2 charge and a mass of 4 AMU which in turn means an alpha decay results in an isotope with Z-2 and A-4
relative to the original nuclear charge and mass. The kinetic energy spectrum of alpha particle emissions from a given isotope has a very narrow bandwidth meaning they have very nearly the same initial energy each time.

Beta emissions occur when a neutron decays into a proton and an electron. The electron is emitted as the beta particle (along with an anti-neutrino). This results in \( Z \) increasing by one but has only a small effect on \( A \). Energy released during a beta decay is shared between the resulting beta particle and the anti-neutrino. Because of this, the energy of the beta particle is not limited to a narrow bandwidth but can vary significantly from one emission to another across an energy distribution that is unique for each beta-emitting isotope.

Gamma emissions are a subsequent process of an isotope that has decayed through some other mechanism and resulted in an excited nucleus. This excess nuclear energy is given off in the form of a photon. This means that neither \( Z \) nor \( A \) are changed following the gamma emission. Similar to the alpha emission spectrum, gammas are emitted with a very narrow linewidth for a given isotope.

Similar to the timing of a single decay or a beta particle’s initial energy, the interactions that any of these emitted particles have as they propagate through a material cannot be accurately predicted on an individual basis. However, these interactions can be observed to follow energy dependent probability distributions. That is to say that a particle’s probability of interacting only depends only on its current energy. Accordingly, once the particle’s energy is changed, so do the probabilities of that particle’s available interactions. Additionally, the energy lost through various inelastic scattering mechanisms is not only dependent on the mechanism but can also depend on the
particle’s incident energy. This accounting for energy loss and particle path is known as radiation transport. Because the interaction probability and the energy lost at each interaction depend on the incident particle’s energy and the type of interactions available at that energy, detailed radiation transport calculations lack a straightforward or analytical solution.

2.2 Interaction Mechanisms

Once a particle or photon has been emitted, it begins to interact with the surrounding material. This interaction is referred as scattering. Particles can scatter elastically, meaning its direction is changed but any change to its energy is negligible, or can scatter inelastically meaning that its direction and kinetic energy are affected. For the particle energies found in most decays of interest for detection of radioactive materials, inelastic scattering is dominated by excitation and ionization interactions.

The likelihood of scattering depends on the particle type, its incident kinetic energy, and the composition of the material it is passing through. With this information, the likelihood of scattering can be expressed in a quantity known as a cross section, with units of area, which can be thought of as an effective area. One common analogy for cross section is to imagine randomly throwing darts at a dart board whose size varies. The larger the dart board, the more likely the dart is to hit it. The same is true for scattering cross sections; larger cross sections represent more likely interactions.

For much of an emitted particle’s trajectory, these scattering mechanisms are dominated by atomic interactions. That is to say that the incident particle interacts with the atoms of the surrounding material as if it were composed of individual atoms, regardless of their being a part of a molecule. However, as energy is reduced, there is an
energy threshold at which interactions with the molecular orbital structure become more prevalent. In his dissertation titled “Monte Carlo Simulation of Electron-Induced Air Fluorescence Utilizing Mobile Agents: A New Paradigm for Collaborative Scientific Simulation,” Walker demonstrated an approach that dealt with the shift in interaction mechanisms and incorporated molecular nitrogen interactions and their cross sections\(^2\).

The model he developed operated as a stand alone simulation and, as the title suggests, used a technique known as Monte Carlo.

In Walker’s dissertation the doubled atomic nitrogen ionization cross section is compared with the molecular ionization cross section as is shown below in Figure 2.1. Above an energy of approximately 1 keV, the molecular cross sections (Tabata Mod) and the double atomic cross sections (EEDL) agree well for ionization.
Figure 2.1 Atomic vs Molecular Nitrogen Cross Section Comparison.

This plot demonstrates the divergence at lower energies of the atomic cross section (EEDL) and the molecular cross section data (Tabata, Tabata Mod, and Itikawa). Reproduced from C.Walker’s dissertation with permission.

As the incident energy drops below 1000 eV, the atomic and molecular cross sections can be seen to diverge more significantly. This 1000 eV threshold is commonly used as the limit at which the accuracy of atomic cross section data starts to break down.

As was discussed in the introduction, the effect radiation has on the surrounding medium is of particular interest, and in this case the focus is on molecular nitrogen. This effect is characterized by transitions to excited neutral and ionized states of the nitrogen molecule during an interaction and quantified by counting how many of each state are
generated. It should also be noted that some excitation and ionization states can also result in dissociation leading to the presence of free neutral atoms and atomic ions. While these will become pertinent to the overall effect on the medium, the main focus at this point is the initial molecular state produced. Below in Table 2.1, one can find a list of all of the interaction states to be accounted for in this project. Each state is given a term symbol (with the exception of the 40eV ionization state). The first letter can be thought of as the name of the state. The remaining portion of the term symbol provides information relating to the highest populated electronic orbital. The numeral superscript indicates spin multiplicity while the remaining notation refers to the symmetry of the orbital configuration projected about various axes\(^3\).

### Table 2.1

**Molecular Nitrogen State Term Symbols**

<table>
<thead>
<tr>
<th>( N_2 ) EXCITATION STATES</th>
<th>( N_2^+ ) IONIZATION STATES</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A^3\Sigma_u^+ )</td>
<td>( X^2\Sigma_g^- )</td>
</tr>
<tr>
<td>( B^1\Pi_g )</td>
<td>( A^2\Pi_u )</td>
</tr>
<tr>
<td>( W^3\Delta_u )</td>
<td>( B^3\Sigma_u )</td>
</tr>
<tr>
<td>( a^1\Pi_g )</td>
<td>( C^3\Sigma_u^- )</td>
</tr>
<tr>
<td>( B^3\Sigma_u )</td>
<td>( D^2\Pi_g )</td>
</tr>
<tr>
<td>( a^1\Sigma_u )</td>
<td>( 40\text{eV} )</td>
</tr>
<tr>
<td>( w^1\Delta_u )</td>
<td></td>
</tr>
<tr>
<td>( C^3\Pi_u )</td>
<td></td>
</tr>
<tr>
<td>( E^3\Sigma_g^- )</td>
<td></td>
</tr>
<tr>
<td>( a^1\Sigma_g )</td>
<td></td>
</tr>
<tr>
<td>( c^3\Pi_u )</td>
<td></td>
</tr>
<tr>
<td>( c^1\Sigma_u )</td>
<td></td>
</tr>
<tr>
<td>( b^1\Pi_u )</td>
<td></td>
</tr>
<tr>
<td>( b^1\Sigma_u )</td>
<td></td>
</tr>
</tbody>
</table>

Cross sections for the various ionization and excitation states are determined by accessing a table of values from which any required intermediate can be found through
interpolation or by utilizing analytic functions to calculate each required value. This will depend on which representations of the data are available. Cross sections are also required for determining the kinetic energy of secondary electrons produced from ionization processes. Walker’s dissertation focuses on energetic electron models and utilizes the work of Itikawa\textsuperscript{4} and analytic forms provided by Tabata, et al.\textsuperscript{5} to calculate the excitation cross sections and the analytic functions found in the work of Jackman, Garvey, and Green\textsuperscript{6} as well as Green and Sawada\textsuperscript{7} to determine the ionization cross section and secondary electron energy respectively. A significant amount of deposited energy will be found to be transferred to secondary electrons, and while accommodation of alpha or gamma emitter simulations would necessitate the incorporation of their respective interaction cross sections, they would still also require the subsequent propagation of secondary electrons. Based on this observation, this research will focus solely on modeling electron interactions.

The basis for determining the expected excitation cross section at each incident energy comes the work of Tabata, et al.\textsuperscript{5} and is a combination of the follow three equations:

\[
f_1(x; c_1, c_2) = \sigma_0 c_1 \left( \frac{x}{E_R} \right)^{c_2}
\]

\[
f_2(x, c_1, c_2, c_3, c_4) = \frac{f_1(x; c_1, c_2)}{1 + \left( \frac{x}{c_3} \right)^{c_2+c_4}}
\]

\[
f_3(x, c_1, c_2, c_3, c_4, c_5, c_6) = \frac{f_1(x; c_1, c_2)}{1 + \left( \frac{x}{c_3} \right)^{c_2+c_4} + \left( \frac{x}{c_5} \right)^{c_2+c_6}}
\]

with \( \sigma_0 = 1 \times 10^{-16} \text{ cm}^2 \) and \( E_R = 1.361 \times 10^{-2} \text{ keV} \).
These three functional forms are then applied in the following seven equations:

\[ \sigma = f_2(E_1; a_1, a_2, a_3, a_4) \]

\[ \sigma = f_2(E_1; a_1, a_2, a_3, a_4) + f_2(E_1; a_5, a_6, a_7, a_4) \]

\[ \sigma = f_2(E_1; a_1, a_2, a_3, a_4) + f_2(E_1; a_5, a_6, a_7, a_8) \]

\[ \sigma = f_2(E_1; a_1, a_2, a_3, a_4) + f_2(E_1; a_5, a_6, a_7, a_8) + f_2(E_1; a_9, a_{10}, a_{11}, a_{12}) \]

\[ \sigma = f_2(E_1; a_1, a_2, a_3, a_4, a_5, a_6) \]

\[ \sigma = f_2(E_1; a_1, a_2, a_3, a_4, a_5, a_6) + f_2(E_1; a_7, a_8, a_9, a_{10}) \]

\[ \sigma = \frac{\sigma_0 a_1 \left( \ln \left( \frac{E}{E_{th}} \right) + a_2 \right)}{E_{th} E \left( 1 + \left( \frac{a_3}{E_1} \right)^{a_4} \right)} \]

where \( E \) is the incident energy, \( E_{th} \) is the threshold energy, \( E_1 \) is the difference between \( E \) and \( E_{th} \), and the remaining values (\( a_1 \) through \( a_{12} \), as needed) in addition to which equation to determine the cross section, \( \sigma \), are all specified for each \( N_2 \) interaction state in that same work\(^5\).

Likewise, the calculations for the cross sections of the various ionization states are found by the following calculation outlined in the work of Jackman, et al.\(^6\):

\[ S_i(E, T) = \frac{A(E) \Gamma(E)^2}{[T - T_0(E)]^2 + \Gamma(E)^2} \]

where

\[ A(E) = \left( \frac{K}{E} + K_B \right) \ln \left( \frac{E}{J + J_B + J_C E} \right) \]

\[ \Gamma(E) = \Gamma_S E/(E + \Gamma_B) \]

\[ T_0(E) = T_S - \left[ \frac{T_A}{(E + T_B)} \right] \]
Where \( E \) is the incident energy and the remaining parameters (\( K, K_B, J, J_B, J_C, \Gamma_S, \Gamma_B, T_S, T_A, \) and \( T_B \)) correspond to values provided therein for each ionization state.

Walker includes a detailed explanation of the calculation of the interaction cross sections for each excitation and ionization state in his dissertation. As will be discussed in more detail in \textit{CHAPTER IV}, direct correspondence with Walker has led to his generously providing a set of Python classes for generating these cross sections for this project. Figure 2.2 and Figure 2.3 contain the excitation and ionization cross sections generated from each state’s respective threshold energy up to 1000 eV.

![Molecular Excitation Cross-Sections for \( N_2 \)](image)

**Figure 2.2 Molecular Nitrogen Excitation Cross Sections.**

Displayed are the excitation cross sections from their respective thresholds up to 1000 eV. Inset axes highlights changes in cross section near threshold energies. The horizontal axes of the inset ranges from 6 eV up to 30 eV.
Figure 2.3 Molecular Nitrogen Ionization Cross Sections

Displayed are the excitation cross sections from their respective thresholds up to 1000 eV.

With a complete set of interaction cross sections, one can determine the relative probability of each interaction by dividing the individual interaction cross section at a given energy by the total of all cross sections at that energy. This converts cross sections into fractional probabilities. The fractional probabilities for the combined excitation and ionization interactions for electrons in molecular nitrogen can be seen below in Figure 2.4 as a function of incident electron energy and in Figure 2.5 highlighting the low energy region from 0 to 75 eV. Also shown below in Figure 2.6 is an example secondary electron spectrum. This spectrum corresponds to the probability vs secondary electron kinetic energy in response to an ionization interaction between molecular nitrogen and a 1000 eV incident electron. It should be noted that for any ionization event the higher energy electron leaving the interaction is deemed the incident electron. Therefore, the secondary electron can only have kinetic energy up to half of the remaining incident...
electron energy after it has lost the requisite energy to affect the ionization interaction. This can be seen as the sudden drop off in the secondary electron spectrum just below 500 eV.

Figure 2.4 Molecular Nitrogen Interaction Probabilities: 0 to 1000 eV

Displayed are the effective fractional probabilities of each interaction state up to 1000 eV
Figure 2.5 Molecular Nitrogen Interaction Probabilities: 0 to 75 eV

Displayed are the effective fractional probabilities of each interaction state up to 75 eV.

Figure 2.6 Example Secondary Electron Spectrum

Displayed is the secondary electron energy distribution resulting from a 1000 eV incident electron ionization a nitrogen molecule into the $X^3\Sigma_u^-$ state.
This secondary electron energy distribution is found using the form provided in the work of Green and Sawada\textsuperscript{7} which has also been discussed at length in Walker’s dissertation and is calculated by the following equation:

\[
\sigma_i(E) = A(E)\Gamma(E) \left[ \tan^{-1} \left( \frac{T_{\text{Max}} - T_0(E)}{\Gamma(E)} \right) + \tan^{-1} \left( \frac{T_{\text{Min}} - T_0(E)}{\Gamma(E)} \right) \right]
\]

where \(A(E), \Gamma(E),\) and \(T_0(E)\) are equivalent to those described previously for the determination of the ionization cross sections. \(T_{\text{Min}}\) for the purpose of this research is fixed at 0 eV and \(T_{\text{Max}}\) defined as follows:

\[
T_{\text{Max}} = \frac{1}{2} (E - I)
\]

with \(I\) equal to the energy lost by the incident electron due to the ionization process and as before \(E\) equals the incident electron’s kinetic energy.

The energy lost by the incident particle to a given state is equal to the amount of energy used to excite to the transition into the new state and, in the case of ionizations, the kinetic energy of the secondary electron produced. While there is a minimum threshold energy for each state, this does not always correspond to the most likely transition. The Franck-Condon principle states that electronic transitions happen on a much shorter time scale than changes in nuclear separation, in this case the distance between the two nitrogen atoms’ nuclei\textsuperscript{3}. This means that during a transition to another electronic state, the nuclear separation will still be that of the previous state. When considering the wave functions for the available vibrational levels of the new state, this might lead to a better overlap between wave functions corresponding to an excited vibrational level. Figure 2.7 demonstrates this behavior between two arbitrary electronic states, \(E_0\) and \(E_1\). It can be seen that this effectively means that when looking at such a
representation of electronic state potential energy curves one moves vertically up from the vibrational ground state of the starting orbital to determine the strongest overlapping vibrational state of the new orbital. This does not mandate that a single vibrational state is always selected, but the strength of this overlap determines a value referred to as the Franck-Condon factor for each vibrational state, i.e. the probability of transitioning into each available vibrational state.

![Figure 2.7 Demonstration of Franck-Condon Principle.](https://commons.wikimedia.org/w/index.php?curid=33461268)

At ambient temperature conditions and within statistical significance, all molecules can be assumed to be in their ground vibrational state. For each of the molecular nitrogen excitation and ionization state interactions, the vibrational level of the new states with the highest Franck-Condon factor was used to determine the most likely
transition energy\textsuperscript{8,9}. This is a first order approximation, and future efforts may seek to improve on these values by calculating a weighted average across all available Franck-Condon factors and the respective vibrational levels. The resulting transition energy is found by the following calculation:

\[ T = T_e + w_e \left( v + \frac{1}{2} \right) - w_e x_e \left( v + \frac{1}{2} \right)^2 \]

With \( v \) equal to the selected vibrational energy level and the remaining molecular values \((T_e, w_e, \text{ and } w_e x_e)\) are unique to each state and were taken from the work of Herzberg and Huber\textsuperscript{10}. These values are used in an expansion approximation to better account for the vibrational energy. In case of ionizations, the value of \( T \) for the ground state ion (namely \( \text{X}^2\Sigma_g^+ \)) taken from the calculated value of \( T \) for that particular state, and then the accepted value for the ionization potential (15.5808 eV) is added to the resulting value. This represents the combination energy required to ionize the nitrogen molecule and the subsequent energy required to reach the particular vibrational level specified for the target excited state of the ion. In cases where the Franck-Condon factor data was unavailable, the ground vibrational state is used to determine the expected energy lost due to that interaction. Likewise, in some cases for which the second order molecular coefficient \((w_e x_e)\) was unavailable, zero was used in order to calculate the nearest available approximation.
Table 2.2

Largest Franck-Condon factors for each state and their resulting transition energy

<table>
<thead>
<tr>
<th>New State</th>
<th>Vibrational Level Of New State</th>
<th>Franck-Condon Factor</th>
<th>Resulting Transition Energy</th>
</tr>
</thead>
<tbody>
<tr>
<td>A^3σ_u</td>
<td>8</td>
<td>0.095</td>
<td>7.4938 eV</td>
</tr>
<tr>
<td>B^3Π_g</td>
<td>2</td>
<td>0.195</td>
<td>7.7725 eV</td>
</tr>
<tr>
<td>W^3Δ_u</td>
<td>7</td>
<td>0.102</td>
<td>8.5847 eV</td>
</tr>
<tr>
<td>a^3Π_g</td>
<td>3</td>
<td>0.183</td>
<td>9.1583 eV</td>
</tr>
<tr>
<td>B^3Σ_u</td>
<td>7</td>
<td>0.105</td>
<td>9.3966 eV</td>
</tr>
<tr>
<td>a^1Σ_u</td>
<td>7</td>
<td>0.108</td>
<td>9.6429 eV</td>
</tr>
<tr>
<td>w^1Δ_u</td>
<td>6</td>
<td>0.170</td>
<td>9.9889 eV</td>
</tr>
<tr>
<td>C^1Π_u</td>
<td>0</td>
<td>0.545</td>
<td>11.0319 eV</td>
</tr>
<tr>
<td>E^3Σ_g</td>
<td>0</td>
<td>0.929</td>
<td>11.8746 eV</td>
</tr>
<tr>
<td>a^1Σ_g</td>
<td>--</td>
<td>--</td>
<td>12.2547 eV</td>
</tr>
<tr>
<td>c^3Π_u</td>
<td>--</td>
<td>--</td>
<td>12.9431 eV</td>
</tr>
<tr>
<td>c^1Σ_u</td>
<td>--</td>
<td>--</td>
<td>12.9487 eV</td>
</tr>
<tr>
<td>b^1Π_u</td>
<td>--</td>
<td>--</td>
<td>12.4997 eV</td>
</tr>
<tr>
<td>b^3Σ_u</td>
<td>--</td>
<td>--</td>
<td>12.8574 eV</td>
</tr>
<tr>
<td>X^3Σ_G</td>
<td>0</td>
<td>0.917</td>
<td>15.5808 eV</td>
</tr>
<tr>
<td>A^2Π_U</td>
<td>1</td>
<td>0.318</td>
<td>16.9309 eV</td>
</tr>
<tr>
<td>B^3Σ_U</td>
<td>0</td>
<td>0.883</td>
<td>18.7506 eV</td>
</tr>
<tr>
<td>C^3Σ_U</td>
<td>6</td>
<td>0.125</td>
<td>25.0757 eV</td>
</tr>
<tr>
<td>D^2Π_G</td>
<td>--</td>
<td>--</td>
<td>21.9870 eV</td>
</tr>
</tbody>
</table>

2.3 Monte Carlo Methods

Stanislaw Marcin Ulam is credited with inventing the Monte Carlo method in 1946 while pondering the odds of a particular game of solitaire resulting in success. His idea came from struggling to calculate every possible permutation and then going on to calculate the subsequent chances of success from each of those possible games. He speculated that if one were able to play a large enough number of games, then the probability of winning could be calculated based on the observed success rate. Ulam happened to be working at Los Alamos with John von Neumann. Sharing this idea of using repeated trials to
determine average behavior with his colleague led to the extension of this concept to a computational approach to their work on predicting neutron diffusion. Also significant was the fact that the Electronic Numerical Integrator and Computer (ENIAC) had recently been constructed. This meant that not only did the concept of this new approach have theoretical merit, but it was now a much more realistic endeavor to achieve. Subsequent collaboration with R.D. Richtmyer and eventually Nicholas Metropolis saw this proposed new method come to fruition and branch out beyond neutron diffusion\textsuperscript{11}.

The premise of the Monte Carlo method is to randomly sample a given probability distribution a sufficient number of times to be able to draw conclusions about its behavior without having to explicitly calculate all possible outcomes. A Markov Chain Monte Carlo simulation is one whose next step depends solely on the state of the system at the conclusion of the previous step\textsuperscript{12}. While this does lend itself directly to simulating radiation transport behavior, the applications of the Monte Carlo method and its many variations are vast and extend into other fields such as mathematics, economics, and even weather prediction models\textsuperscript{13}.

Nicholas Metropolis is the namesake for the well known Metropolis Algorithm. Metropolis published the development of this modified Monte Carlo algorithm with Arianna Rosenbluth, Marshall Rosenbluth, Augusta Teller, and Edward Teller\textsuperscript{14}. The basic premise is to start by generating an initial value or set of values that is governed by a known relation with a given distribution and then randomly sample variations of this value or set of values. For each variation, the effect on the system based on the governing relation is evaluated to decide whether or not to accept the new value as an improvement or reject it and keep the prior value. For example, Metropolis, et al. used it to minimize
the energy of a system of particles based on the potential between the particles. Also included in this algorithm is the option for a separate mechanism for acceptance outside of a strict improvement relative to the governing relation. In their example of minimizing energy, the Boltzmann distribution is sampled to decide whether or not to allow for increases in energy. In this manner, the value or set of values is sampled until ideally there is no more room for improvement. The allowance for variations that are not strictly improvements in a single iteration allow for the sample space probed to extend further which can help in avoiding getting trapped in areas of local minima.

2.4 Modeling Radiation Transport

If one has the means for determining the scattering cross sections for all possible interactions of a particle emitted by radioactive decay, as well as a mechanism for determining how those interactions affect the particles energy and direction, then one can construct a Monte Carlo simulation to transport that particle through its surrounding material. A basic outline of the Monte Carlo method for radiation transport can be described in 5 steps:

1. The incident particle’s energy is used to evaluate the probability of each of the possible interactions.
2. A random number is generated to determine how far the particle travels before interacting based on the total probability of interacting.
3. A random number is generated and used to determine which of the interactions is selected based on the relative probabilities of each interaction.
4. The incident particle’s direction and energy are used to evaluate the selected interaction’s probability of affecting changes in direction and, if applicable, changes in the energy of the incident particle,

5. A random number is generated for each applicable attribute and is used to determine the particle’s direction and energy following the interaction. If a secondary particle is generated, its attributes are added to a queue to be handled in similar fashion.

For each particle introduced in the simulation these steps are repeated until the particle has traveled a user defined maximum distance (i.e. leaves the region of interest) or it runs out of energy (or more likely falls below a user specified nonzero minimum energy), recording along the way whatever data the user specified. This is repeated for a large number of particles until whatever behavior the user is concerned with can be assessed, assuming assessment of such behaviors are available.

Recording unnecessary information throughout a simulation can lead to excessively large file sizes. Recording too little information nullifies the purpose of the simulation all together. Notice that the above steps assume a uniform homogeneous medium through which the particle propagates and does not include checking to see if the particle has entered a new material or volume. Large or complicated geometries can lead to prohibitive runtime issues if not handled appropriately. Optimizing this balance and expanding the capabilities of Monte Carlo simulations as it pertains to radiation transport has led to several large scale efforts at various research institutions resulting in the availability of software packages and toolkits tailored to this sole purpose.
CHAPTER III – Implementation of Existing Monte Carlo Software

Developing an in-house Monte Carlo model becomes exceedingly labor intensive if one ever anticipates extending it to incorporate a wide variety of materials or geometries. Anticipating the future of this research effort will be to model more and more realistic scenarios, finding an existing software package that would better handle these complexities was prioritized. There are many research fields that require sophisticated transport model capabilities such as defense programs, medical physics, radiation safety, and accelerator/collider research. Across these applications a variety of transport packages have been developed, and while there are many options available, two were considered as the most viable candidates: MCNP and Geant4.

Monte Carlo N-Particle (MCNP) transport code is a package developed at Los Alamos National Lab and is funded and maintained by the US government\textsuperscript{15}. MCNP uses atomic interaction cross sections and is a well established Fortran-based package that can be released solely as an executable or with the source code depending on citizenship and application. This conditional release is because it is an export controlled technology. This means that controlling access could become a liability, especially in an academic research setting.

Geant4 is an open source toolkit developed by a collaboration of researchers based at CERN\textsuperscript{16}. Originally released in 1998, it is currently on version 10.4 with a beta release of 10.5 currently available. Geant4 is a C++ based toolkit that can handle very complicated geometries and installs with a vast collection of standard materials and interaction databases. It is also based on atomic interactions but does advertise the ability to incorporate new processes programmatically (i.e. without requiring modification of the
Geant4 source code). Because it is a toolkit, as opposed to an out of the box executable, means that the user has complete freedom and control over most aspects of any given implementation. The open source nature, capability of incorporating complex geometries with built in material databases, and the inherent flexibility of the toolkit format led to the selection of Geant4 for the initial modeling efforts of this project.

3.1 Basic Requirements of a Geant4 Simulation

When creating a simulation in Geant4, there are three user specified classes that must always be present: a primary action generator class, a detector construction class, and a physics list class\textsuperscript{17}. The primary action generator class handles the introduction of energetic particles to the simulation. Without this class, there would be no interactions to model. These primary particles can be a generated by a radioisotope decay or arbitrarily generated to the user’s requirements (i.e. particle type and energy). The user can also specify a point source or a distributed source as well as dictate the particle’s initial momentum vector as needed.

The second required class, a detector construction class, establishes the world in which the simulation is taking place (size, shape, etc.). It also handles any additional volumes and materials to be included in the scenario (i.e. shielding, sensors, etc.). This class is also used to specify which volumes are to be used as sensitive detectors in the simulation. Sensitive detector volumes are those in which user specified results are recorded. For example, a scenario may call for a sensitive detector region to record absorbed dose behind a lead shield. If interactions inside the lead shield are not of interest, then it is not specified a sensitive detector volume. Interactions will still be modeled throughout the lead shield to account for the appropriate behaviors, but
computational resources will not be expended to record these interactions. Without a
detector construction class, not only would there be no world within which interactions
can take place, but there would also be no record of any interactions.

The physics list class is where the user specifies which set of physics models are
to be included. Different energy regimes are characterized by different interactions.
Extremely high energy interactions allow for fragmentation events that are not available
to lower energy particles. Making sure that the appropriate physics lists are active in a
given simulation ensures that it runs efficiently and as accurately as possible. If an
appropriate physics list class is not present, the simulation would not return any valid or
useful results.

3.2 Implementation of Geant4

The initial implementation of Geant4 for this project was a straight forward
simulation of four different radioisotopes modeled within a representation of an 18”
stainless steel vacuum chamber. This served two main purposes: demonstrate an
appropriate choice for the radioactive source selected for experimental measurements in
the laboratory and to demonstrate successful modeling results against something that can
be calculated directly. The latter refers to the stopping power/continuous slowing down
approximation (CSDA) based calculations of the expected ranges in dry air at sea level
for particles of known energy\textsuperscript{18}. In the panels of Figure 3.1 are visualizations of these
four different isotopes, and the values in Table 3.1 demonstrate an acceptable agreement
with expected range calculation results. The CSDA range calculation is representative of
the average expected range based on linear attenuation. Slight deviations from this are to
be expected.
Figure 3.1 Geant4 Isotope Visualizations

Top Left, Po-210 (alpha). Top Right, Ni-63 (beta). Bottom Left, Sr-90 (beta). Bottom Right, Co-60 (gamma) Reproduced from prior poster presentation

Table 3.1

Comparison of Geant4 isotope models’ maximum range to the approximate anticipated range based on stopping power and CSDA range calculations.

<table>
<thead>
<tr>
<th>SOURCE</th>
<th>PARTICLE ENERGY\textsuperscript{20} (MeV)</th>
<th>CSDA RANGE\textsuperscript{18} (cm)</th>
<th>MAX RANGE IN MODEL (cm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Po-210, Alpha</td>
<td>5.304</td>
<td>3.94</td>
<td>4.16</td>
</tr>
<tr>
<td>Ni-63 Beta</td>
<td>0.065</td>
<td>6.46</td>
<td>5.46</td>
</tr>
<tr>
<td>Sr-90, Beta</td>
<td>0.546</td>
<td>188.75</td>
<td>1002.7</td>
</tr>
<tr>
<td>Ly-90, Beta</td>
<td>2.28</td>
<td>1034.9</td>
<td></td>
</tr>
</tbody>
</table>

In the case of beta decay, where the beta emission is accompanied by an antineutrino and the decay energy is distributed between the two particles, particle energy refers to the maximum possible kinetic energy of the beta particle. Additionally, the Sr-90 beta emission results in the unstable Y-90 which also undergoes a beta decay. The model implementation recorded only the maximum range, thus the Y-90 emission dominated this result. Reproduced from prior poster presentation\textsuperscript{19}
Such models are appropriate for simulations that are not negatively affected by taking advantage of the approximation of materials as collections of atoms, but in the case of this research, molecular interactions are needed as well. Upon further investigation into adding user specified interactions, it was decided that this would not yield the results initially sought. The main reason for this is that like many other existing transport packages Geant4 is a model based on an unaffected medium. Typical values of interest are particle range and where a given particle lost its energy (often collected as dose deposited). The desired quantities for this project are how energy deposition affected the material, not simply that the energy was deposited. Instead of continuing to pursue complicated adaptations of the Geant4 source code, the focus of the Geant4 implementation was shifted to modeling particle behavior above 1000 eV and handling molecular interactions below this threshold (demonstrated in Figure 2.1) with an external calculation. As has been demonstrated\textsuperscript{21} and can be calculated based on values provided by NIST\textsuperscript{18}, electrons with energy less than 1000 eV can be conservatively calculated to travel distances well under 1 mm in air at ambient conditions. Therefore, modeling down to a 1 keV in Geant4 maintains the spatial fidelity of the simulation.
CHAPTER IV – Assessing Limitations of Existing Software

Because Geant4 uses atomic rather than molecular cross sections, a Geant4 simulation was designed to produce the sub-keV electron spectrum from a 1 MeV electron source in pure nitrogen. This uses Geant4’s capabilities at energies above 1 keV while allowing for the sub-keV particles to be processed separately. The world volume was filled with nitrogen and the sensitive detector volume size was set to extend well beyond the range of the 1 MeV primary particle. To collect the sub-keV spectrum, two main adjustment were made. First, electrons (the primary and secondaries) with kinetic energy above 1 keV were propagated through model until their energy fell below the 1 keV threshold. This was accomplished by setting the lowest allowed electron energy equal to 1 keV in the model. Geant4 continues discrete interaction modeling until an interaction drops an electron below this threshold and afterwards uses a CSDA approach. The energy is recorded in 1 eV bins to create a sub-keV electron spectrum. The sensitive detector is checking for particles with energy below this threshold as well, and when one is discovered, the appropriate bin is incremented. Second, sub-keV secondaries must be allowed to be generated. This is accomplished in the physics list class by forcing the Geant4 production cuts threshold down to an energy of 1 eV. As discussed later, this is well beyond the recommended limit. Geant4 documentation suggest a limit of 1 keV with exceptions down to 250 eV (with loss of accuracy for modeling interactions as this limit is approached). An additional step was taken to reduce runtime. The stacking action class handles the generation of particle tracks by adding them to the stack of particles to be subsequently propagated through the simulation. In the stacking action class function that handles creation of new tracks, a case was added to check to see if the proposed new
particle track would instantiate with an energy below 1 keV. If that was the case, it would be recorded and terminated before ever being introduced to the simulation. The hope was that this would avoid modeling any interactions so far outside the suggested energy range and accordingly mitigate any issues arising from the inappropriate production cut energy. Additional adjustments to the available interaction mechanisms included turning on atomic de-excitation. For high Z atoms, an inner shell ionization often results in a higher orbital electron falling to fill the opening, resulting in an x-ray emission. However, for a lower Z atom such as nitrogen, the de-excitation process is almost always an Auger electron emission\textsuperscript{22}. Figure 4.1 contains the resulting sub-keV spectrum which includes Auger emissions at the appropriate energies. This spectrum is the average number produced per incident particle and was produced by modeling 10000 1 MeV electrons.

![Sub-keV Electron Spectrum](image-url)

**Figure 4.1 Geant4 Sub-keV electron spectrum prediction**

While the simulation ran well and produced what seemed like appropriate results (high energy electrons trailing in on right, appropriate Auger peaks, and an increasing...
number of lower energy secondaries), there is a significant problem this this spectrum. Namely, there are far too many secondary electrons. The average work per ion pair, or W/e, is a well established quantity for a variety of scenarios, this one included\textsuperscript{23}. The accepted value for W/e in nitrogen at this density (i.e. pressure) is 34.8 eV. Dividing the average number of secondaries predicted here, approximately 52,336.5 secondary electrons, into the original 1 MeV incident particle energy gives a value of approximately 19.1 eV for W/e. While this value seems low at first glance considering the eventual target of 34.8 eV, it is even lower when one considers that over half of the primary electron’s energy is stored in the kinetic energy of these secondaries and many more secondaries have yet to be generated through subsequent interactions. As a first order approximation, one could assume that the more appropriate target for W/e would be double the eventual value, meaning something in the vicinity of 70 eV. Removing the first six energy bins from this spectrum (corresponding to secondaries with energies up six eV), brings the total number of particles down to approximately 13,258.7 which brings W/e up to 75.4. Understanding this is a completely arbitrary truncation compared to an approximate target, and not a definitive proof that the steep increase at the low energy end is the problem, a method to externally generate a separate sub-keV spectrum to compare against is needed.

Direct correspondence with Walker was initiated in hopes of using his existing Monte Carlo model to generate a separate sub-keV spectrum. His model is a full Monte Carlo simulation incorporating the molecular cross sections and is expected to be more accurate at low energies. This resulted in a collaboration that significantly aided this project. Walker provided access to source code for a Python based adaptation of his
simulation for Monte Carlo for air fluorescence simulation (MCAFS)\textsuperscript{24}. The goal at this point being to confirm that the Geant4 prediction is currently over producing very low energy secondaries, this Monte Carlo model does not need to be a full radiation transport code. Instead, it can be focused entirely on being able to calculate an accurate sub-keV spectrum. Realizing this, Walker removed the positional information and tracking from the model which significantly improves runtime. The MCAFS model is equipped with cutoff parameters to be specified by the user. In addition to providing a cutoff for interactions, any particle that falls below this cutoff is tallied in the appropriate energy bin, effectively generating a sub-cutoff spectrum by default. Setting the cutoffs to 1 keV then generates the desired sub-keV spectrum. This spectrum, as well as the geant4 based spectrum, can be seen in Figure 4.2. It should be noted that Auger emission electrons were not considered a priority for this level of performance evaluation and as a result were not included in the MCAFS simulation. This obviously results in the lack of these peaks in the predicted sub-keV spectrum. The MCAFS model generated an average of 14,681.9 secondaries which corresponds to a value of 68.1 eV for W/e which is very close to the 70 eV target. While this is again only an approximation due to the fact that there is still more than half the initial kinetic energy still remaining in the system, this is still considered a much more appropriate value than the Geant4 result. This also confirms suspicions that the error in the Geant4 spectrum lies in the very low energy region and that this particular implementation in Geant4 is inappropriate. At this point, Geant4 development is being set aside for the time being, and a new method for determining molecular products is taking priority.
Figure 4.2 Geant4 and MCAFS predicted Sub-keV electron spectra
CHAPTER V – A New Approach to Low Energy Interactions

Adjustments to the MCAFS source code allowed for the code to output which state interaction was selected for each step. Recording a tally for each state allows for an assessment of the incident particle’s effect on the medium, in this case how much of the nitrogen present was left in a different state. Acknowledging that sub-keV particles traverse a very small range\textsuperscript{18,21}, this establishes a lack of the need for positional information in the molecular energy regime. Spatial considerations for higher energy particles will be addressed with a modified Geant4 model, allowing for this separate processing of low energy processes. The lack of any spatial dependence turns the desired calculation into a fully averaged result. This raised the questions about the potential using deterministic calculations of this average result and whether this might offer an improvement in run time over a sub-keV Monte Carlo transport calculation. With the end goal still being to develop a two stage method that lets a well established package like Geant4 handle higher energy transport and spatial dependence (which depending on scenario can still lead to lengthy run times), avoiding the need to follow this with an additional series of Monte Carlo simulations to continue the transport of low energy particles would be a significant benefit. This proposed deterministic approach has been termed the Averaged Loss Method.

5.1 The Averaged Loss Method

In order to be sure that comparisons with molecular products from the MCAFS model and the new method are valid, it is important to ensure that both are calculating the interaction cross sections identically. The definition of an N2_Model class is contained within the MCAFS model source code. Of particular interest here is that this N2_Model
class in turn calls on N2_Ionization_Model and N2_Excitation_model classes from which calculated cross section values for molecular nitrogen can be returned. The N2_Ionization_Model can also return secondary spectra information for nitrogen molecules. Taking advantage of these existing class definitions, the new method described below was made to incorporate this N2_Model class as well. Adjustments were made to ensure that both methods were using the first order approximation of the Frank-Condon transition energies.

The Averaged Loss Method is split into two distinct parts. The first part uses the cross sections, secondary spectra, and Franck-Condon energies to calculate an average energy lost at each incident energy from 1 to 1000 eV. That calculation is accomplished as follows:

- Use N2_Model to create cross section dictionary. For each state, at each incident energy, the cross section is calculated.
- Use N2_Model to create secondary spectra dictionary. For each ionization state, for each incident energy, for each possible secondary energy, the cross section for a secondary of that energy is calculated.
- Using the cross section dictionary, another dictionary of interaction probabilities is created. For each state, for each incident energy, the probability is calculated based on the cross section divided by the sum of all cross sections at that energy.
- Using the secondary spectra dictionary, another dictionary of normalized secondary spectra is created. For each ionization state, for each incident energy, for each possible secondary energy, the cross section for that
secondary energy is divided by the sum of all the secondary energy cross sections corresponding to that incident energy to calculate the probability of each secondary energy.

- These normalized spectra are used to determine the corresponding average secondary kinetic energy dictionary. For each ionization state, for each incident energy, the probability of each secondary energy is multiplied by that secondary energy. These products are summed to create a weighted average secondary kinetic energy.

- Create Franck-Condon energy dictionary entry for each state that is the first order approximation of the Franck-Condon energy

- Create average loss list. For each incident energy the average loss is equal to the sum over all available states of the following:

  \[(\text{Franck-Condon Energy} + \text{Average Secondary Energy}) \times \text{Cross Section Probability}\]

  In this way, the probability for each state acts as a weighting factor. For excitation states, the Average Secondary Energy term is equal to zero and can be dropped. An example of this calculation can be seen in Table 5.1.
Table 5.1

Sample calculation of the averaged energy lost due to interactions at 1 keV

<table>
<thead>
<tr>
<th>N2 State</th>
<th>Cross Section (cm²)</th>
<th>Probability (%)</th>
<th>Expected Energy Loss (eV)</th>
<th>Ave. Secondary Energy (eV)</th>
<th>Combined Energy Scaled by Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>A³Σ_u</td>
<td>7.98x10^{-22}</td>
<td>8.03x10^{-4}</td>
<td>7.4938</td>
<td>--</td>
<td>6.02x10^{-5}</td>
</tr>
<tr>
<td>B³Π_g</td>
<td>8.45x10^{-23}</td>
<td>8.51x10^{-5}</td>
<td>7.7725</td>
<td>--</td>
<td>6.61x10^{-6}</td>
</tr>
<tr>
<td>W³Δ_u</td>
<td>4.45x10^{-23}</td>
<td>4.48x10^{-5}</td>
<td>8.5847</td>
<td>--</td>
<td>3.84x10^{-6}</td>
</tr>
<tr>
<td>a¹Π_u</td>
<td>5.76x10^{-19}</td>
<td>5.80x10^{-1}</td>
<td>9.1583</td>
<td>--</td>
<td>5.31x10^{-2}</td>
</tr>
<tr>
<td>B³Σ_u</td>
<td>1.78x10^{-22}</td>
<td>1.79x10^{-4}</td>
<td>9.3966</td>
<td>--</td>
<td>1.68x10^{-5}</td>
</tr>
<tr>
<td>a¹Σ_u</td>
<td>7.80x10^{-20}</td>
<td>7.85x10^{-2}</td>
<td>9.6429</td>
<td>--</td>
<td>7.57x10^{-3}</td>
</tr>
<tr>
<td>w¹Δ_u</td>
<td>6.79x10^{-24}</td>
<td>6.83x10^{-6}</td>
<td>9.9889</td>
<td>--</td>
<td>6.83x10^{-7}</td>
</tr>
<tr>
<td>C³Π_u</td>
<td>1.63x10^{-20}</td>
<td>1.64x10^{-2}</td>
<td>11.0319</td>
<td>--</td>
<td>1.81x10^{-3}</td>
</tr>
<tr>
<td>E³Σ_g</td>
<td>1.69x10^{-22}</td>
<td>1.71x10^{-4}</td>
<td>11.8746</td>
<td>--</td>
<td>2.03x10^{-5}</td>
</tr>
<tr>
<td>a¹Σ_g</td>
<td>7.25x10^{-20}</td>
<td>7.30x10^{-2}</td>
<td>12.2547</td>
<td>--</td>
<td>8.95x10^{-3}</td>
</tr>
<tr>
<td>c¹Π_u</td>
<td>4.64x10^{-18}</td>
<td>4.67</td>
<td>12.9431</td>
<td>--</td>
<td>6.04x10^{-1}</td>
</tr>
<tr>
<td>c¹Σ_u</td>
<td>3.99x10^{-18}</td>
<td>4.02</td>
<td>12.9487</td>
<td>--</td>
<td>5.20x10^{-1}</td>
</tr>
<tr>
<td>b¹Π_u</td>
<td>3.06x10^{-18}</td>
<td>3.08</td>
<td>12.4997</td>
<td>--</td>
<td>3.85x10^{-1}</td>
</tr>
<tr>
<td>b¹Σ_u</td>
<td>4.25x10^{-18}</td>
<td>4.28</td>
<td>12.8574</td>
<td>--</td>
<td>5.50x10^{-1}</td>
</tr>
<tr>
<td>X²Σ_g</td>
<td>3.79x10^{-17}</td>
<td>38.1</td>
<td>15.5808</td>
<td>29.78</td>
<td>17.28</td>
</tr>
<tr>
<td>A²Π_u</td>
<td>1.66x10^{-17}</td>
<td>16.7</td>
<td>16.9309</td>
<td>29.74</td>
<td>7.78</td>
</tr>
<tr>
<td>B²Σ_u</td>
<td>8.60x10^{-18}</td>
<td>8.65</td>
<td>18.7506</td>
<td>29.68</td>
<td>4.19</td>
</tr>
<tr>
<td>C²Σ_u</td>
<td>5.77x10^{-18}</td>
<td>5.80</td>
<td>25.0757</td>
<td>29.52</td>
<td>3.17</td>
</tr>
<tr>
<td>D²Π_g</td>
<td>5.77x10^{-18}</td>
<td>5.81</td>
<td>21.9870</td>
<td>29.58</td>
<td>3.00</td>
</tr>
<tr>
<td>40eV</td>
<td>8.14x10^{-18}</td>
<td>8.19</td>
<td>40</td>
<td>29.05</td>
<td>5.65</td>
</tr>
</tbody>
</table>

Average Energy Loss: 43.19 eV

The second part is the application of this averaged loss to the incident particle. It requires a dictionary to accumulate an interaction state tally for each state and a single set of bins for new secondaries to be accumulated into throughout:

- For a particle with a given incident energy, the incident energy is decremented by the averaged loss value at the incident energy, the interaction state counter is incremented by the cross section probability for
each state at the incident energy, and appropriate new secondary bins are
incremented by the normalized spectrum values for each state multiplied
by the cross section probability of that state at the incident energy. A
visual representation of one step in this process can be seen below in
Figure 5.1.

- This process is iterated until the incident particle's energy is reduced to a
  value below the lowest threshold energy.
- Then this process is repeated for all nonzero entries in the new secondaries
  bins. This is iterated from high to low such that subsequent new
  secondaries can still be added to lower energy bins that have not yet been
  processed.

Figure 5.1 Demonstration of the 1 keV step in the Averaged Loss Method.
Note that product count examples for each state correspond to those in Table
5.1
The only difference between the new secondaries loop and the primary particle loop is that the value added to the state counter is also multiplied by the value in the corresponding secondary energy bin. This allows for accumulation of results from a fractional quantity of secondaries.

After all of the new secondaries have been processed, the resulting accumulation of state occurrences is the number of each state that occurred per incident particle at the initial incident energy. Repeating this whole process across the entire sub-keV energy range allows one to characterize the expected average results for a particle of each incident sub-keV energy.
CHAPTER VI – Results & Performance Evaluation

The first set of results to consider is the W/e predictions as a function of incident energy for the MCSFS code. Longer than expected to be necessary simulations for incident energies of 250, 500, 750, and 1000 eV were conducted to provide insight into the necessary runtimes for convergence of the Monte Carlo model. Figure 6.1 depicts the W/e results for 100 simulations of 1,000 particles each as well as their cumulative average throughout for the four different incident electron energies. Figure 6.2 demonstrates the convergence behavior of each of these datasets by dividing the cumulative average up to any given point by the cumulative average for the total run of 100,000 incident particles. While this will by definition be equal to one for the final point, the time it takes to approach a value near one and its subsequent stability around a value of one indicate convergence with some random noise as expected. For all of these energies, the Monte Carlo model’s cumulative average appears to reach a relatively stable result well within the 100,000 particle range. Noise at the simulation level results could be mitigated by running fewer simulations of a larger number of incident particles, but this would have no benefit on the cumulative average and thus would not improve runtime to reach convergence. As all of these energies fall within 0.1% of the total average by 20,000 particles and in most cases dropping to within 0.05% shortly thereafter, 20000 will be considered a reasonable benchmark for assessing a lower limit on runtimes at this point. 20,000 particles with incident energy of 1 keV modeled in the Monte Carlo simulation required 1 hour, 28 minutes, and 54.30 seconds to complete. 20,000 incident particles at 250, 500, and 750 eV required 23 minutes 22.4 seconds, 45 minutes 5.72 seconds, and 1 hour 6 minutes 31.07 seconds respectively. The times required to
complete the full 100,000 particles were, as expected, found to be approximately five times the time required for the 20,000 particles.

Figure 6.1 Average Work Per Ion vs Number of Particles Modeled.

Demonstrating the effect of increasing number of particles included in average W/e calculation. Colored curves are comprised of results from individual 1,000 particle simulations while the dark curves represent the running average corresponding to those values.
Figure 6.2 Assessment of convergence of Monte Carlo results.

To determine a basis for a sufficient number of particles simulated, the cumulative average of the particles simulated up to a given point on the horizontal axis is divided by the total average after simulating the entire 100000 particles.

For determining $W/\varepsilon$ behaviors across the entire energy range with the Monte Carlo method, particles are simulated with 20, 40, 60, 80, 100, 150, 200, 250, 300, 350, 400, 450, 500, 750, and 1000 eV incident kinetic energy. For each of these, 20,000 particles are simulated. For the Averaged Loss Method, predicted $W/\varepsilon$ values are calculated at each integer energy from just above the ionization threshold at 16 eV up to 1000 eV. These two sets of data can be seen below in Figure 6.3. The steep climb as incident energies decrease and approach the ionization threshold of molecular nitrogen is a result of excitation processes becoming more and more likely compared to ionization process until the ionization threshold is reached. Below which excitations are the only available interaction mechanisms, and $W/\varepsilon$ is no longer an applicable value.
Table 6.2

Comparison of Monte Carlo and Averaged Loss Method generated W/e values at select values. Monte Carlo values are the result of the full 100,000 particle simulation.

<table>
<thead>
<tr>
<th>Energy (eV)</th>
<th>Monte Carlo W/e (eV)</th>
<th>Averaged Loss W/e (eV)</th>
<th>Percent Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>250 eV</td>
<td>33.826</td>
<td>33.831</td>
<td>0.01%</td>
</tr>
<tr>
<td>500 eV</td>
<td>33.016</td>
<td>33.252</td>
<td>0.71%</td>
</tr>
<tr>
<td>750 eV</td>
<td>32.796</td>
<td>33.090</td>
<td>0.90%</td>
</tr>
<tr>
<td>1000 eV</td>
<td>32.712</td>
<td>33.052</td>
<td>1.04%</td>
</tr>
</tbody>
</table>

The runtime to calculate the averaged loss values (includes cross section, probability, and secondary spectra calculations) took 21.26 seconds. Because this set of values is determined over the entire range of 1 to 1000 eV and then applied to prediction calculations as needed, this calculation is only required once. Following the preparation of the averaged loss values, a calculation of the results for an incident electron with 1000 eV of kinetic energy takes 1.37 seconds. Calculating the results for all integer incident
electron energies from 8 to 1000 eV only requires 7 minutes and 55.19 seconds. Adding the initial average loss values calculation to this means that the total runtime is 8 minutes and 16.45 seconds.

The final behavior to compare is the ability of the Averaged Loss Method to predict the effect the incident electrons have on the nitrogen molecules. Below in Figure 6.4 are the molecular product values as a result of a 1000 eV incident particle for both the Monte Carlo simulation and the Averaged Loss Method. In the case of the Monte Carlo results, these are the average results after 20,000 incident particles. The averaged loss values are the direct result from the calculation described in Section 5.1 and have not been scaled or adjusted in any way.

Figure 6.4 Comparison of Resulting Yield of Molecular Nitrogen State.
CHAPTER VII – Conclusions

Clearly the Geant4 implementation requires an investigation to see if it is capable of being adjusted to generate an appropriate sub-keV electron spectrum. If this proves to be unattainable directly, one proposed method of maintaining the benefit of Geant4’s inherent ability to handle complex geometries and incorporate built in materials databases is to test the dependence of the sub-keV spectrum on the incident energy of above keV particles. If a relationship between energy deposited in a volume to the distribution of sub-keV particle can be determined and is unaffected by incident energy within some range, then a calibration scheme could be determined allowing for such energy deposition results to convert in to sub-keV spectra results.

While there is a slight separation between the two curves in Figure 6.3, the overall behavior matches throughout the energy range and the separation is greatest at 1000 eV and only reaches 1.04% difference. As an initial evaluation of this new technique, these results for W/e are considered to be a reasonable success. One likely candidate for improvement of the W/e value is expanding the Franck-Condon factor calculation to include several more factors in a weighted average. This would help to more accurately account for energy loss throughout the incident particle’s lifetime.

On the evaluation of runtime benefits of this new technique, these results demonstrate a drastic improvement. The four incident energies detailed above using the Monte Carlo method combine for a total of 3 hours 43 minutes 53.49 seconds of runtime and the simulated 20000 particles at the fifteen different energies used to determine the W/e curve in Figure 6.3 required a cumulative runtime of 7 hours 27 minutes and 16.36 seconds while the Averaged Loss Method only took 8 minutes and 16.45 seconds to
process nearly 1000 energies. Clearly, the Averaged Loss Method demonstrates a distinct advantage in runtime over the Monte Carlo method. One of the main benefits of this improvement in runtime would be the ability to adjust input parameters and rapidly be able to evaluate the predicted effect on system response.

The Averaged Loss Method shows good agreement across the molecular product results with the possible exception of the $A^3\Sigma_u$ state. This state shows the largest deviation between the two techniques. This state also has the lowest threshold energy. It is suspected that the fixed 1 eV bin width across the entire energy range is the most likely reason for this deviation. As electron energies approach this threshold, the bin width becomes a significant fraction of their energy. Future improvements to the method will include the implementation of variable bin width to account for this issue.

Once these issues have been remedied or accounted for, the next natural step is to extend this process to calculations for molecular oxygen and eventually molecular water vapor as well. From that point composite models of air like mixtures could be calculated.
APPENDIX A – Source Code for the Averaged Loss Method

The follow are the various Python files developed for this project. Not included are the files for the software developed by C.Walker for MCAFS. However, it should be noted that this code as it is written does require the ability to import the N2_Model class definition and its subsequent dependencies from the MCAFS software. Additionally, the Averaged Loss Method software was developed in Python3 while the MCAFS software was developed in Python2. Accordingly, the Python tool 2to3 was used to convert the necessary files from Python2 syntax to Python3.

A.1 AveragedLossMethod.py

# This project uses an approach involving applying the average energy lost from an # interaction at a given incident energy as the energy lost for all particles with that energy. # Significant contributions were made by Dr. Christopher Walker in the separate development of the # N2_Model class definition (as part of the MCAFS) and the pertinent cross section calculations handled within.

import numpy as np
import copy
import json

from getStateEnergies import getVerticalEnergies
from getCrossSections import getCS, getCSP
from getNewSecondaries import getSecondarySpectra, getNormalizedSpectra, getAveKinEnergy

from timeit import default_timer as timer
start = timer()

binEnergies = [float(i) for i in range(1001)]

names, crossSections = getCS(binEnergies)

binEnergyCSP = getCSP(len(binEnergies), crossSections)
# retrieve most likely energy loss per process
processDeltaE = getVerticalEnergies()

# define list of ion state names
ionStates = ['x2sg', 'd2pg', 'c2su', 'b2su', 'a2pu', 'ev40']

# retrieve the expected resulting secondary electron energy distributions
# for each incident energy
secondarySpectra = getSecondarySpectra(binEnergies, ionStates)

# normalize this spectrum such that each entry becomes the fraction
# of new secondaries with corresponding kinetic energy
normalizedSpectra = getNormalizedSpectra(secondarySpectra)

# determine the average new secondary kinetic energy for each incident energy
aveKinEnergy = getAveKinEnergy(normalizedSpectra)

aveIncidentEnergyLoss = []
for i in range(0, len(binEnergies)):
    currentTotalLoss = 0.0
    for processName in processDeltaE:
        if processName in ('x2sg', 'd2pg', 'c2su', 'b2su', 'a2pu', 'ev40'):
            lossContribution = processDeltaE[processName][0] + aveKinEnergy[processName][i, 1]
        else:
            lossContribution = processDeltaE[processName][0]
        currentTotalLoss += lossContribution * binEnergyCSP[processName][i]
    aveIncidentEnergyLoss.append(currentTotalLoss)

end = timer()

elapsed = end-start
m, s = divmod(elapsed, 60)
h, m = divmod(m, 60)

print("%.2f seconds" % elapsed)
print("%d:%02d:%.2f" % (h, m, s))

# start new timer to aid in assessing the performance of the
# iterative portion of this method evaluated below
start = timer()

# specify the energies of interest. In this case, 1 to 1000 eV in integer steps.
Energies = range(1, 1001)

# create empty lists to hold incident energy dependent cumulative values
totalIonCount = []
workPerIon = []

# create empty dictionary with the process names as keys
numTotalProducts = copy.deepcopy(names)

# Loop through this calculation for each entry in Energies
for e in range(len(Energies)):
    # indicates progress through Energies
    if e%25 == 0:
        print(e)

    # define current energy based on corresponding entry in Energies
    # and initialize the index holder. EnergySteps is a separate behavior
    # analysis tool used to record how many steps a given particle goes
    # through until it falls below the specified threshold energy
    CurrentEnergy = Energies[e]
    CurrentIndex = e
    EnergySteps = []
    EnergySteps.append(CurrentEnergy)
    StepLoss = []

    # ionCount is used to record what fraction of each interaction corresponds
    # to an ionization state. This becomes the scaling factor when accumulating
    # new secondary electrons based on these ionization interactions.
    # numTotalProducts is a dictionary used to accumulate the number of each
    # interaction process throughout the entire calculation.
    ionCount = {}
    for processName in numTotalProducts:
        ionCount[processName] = np.zeros(len(Energies))
        numTotalProducts[processName].append(0.0)

    # for each incident energy, the newSecondaries list is reinitialized
    newSecondaries = np.zeros(1000)

    # this calculation currently operates over bins defined with 1 eV bin width
    # 8 eV is the last integer value energy that is above the excitation threshold
    while CurrentEnergy > 8:

        # for each interaction process, increment by the interaction probability.
        # in the case of ionizations, also increment the appropriate ionCount value
        for processName in binEnergyCSP:
            if processName in ionStates:
                numTotalProducts[processName][e] += binEnergyCSP[processName][CurrentIndex]
                ionCount[processName][CurrentIndex] += binEnergyCSP[processName][CurrentIndex]
            else:
                numTotalProducts[processName][e] += binEnergyCSP[processName][CurrentIndex]

        # reduced the current energy by the corresponding value in the average loss list
        CurrentEnergy = CurrentEnergy - aveIncidentEnergyLoss[CurrentIndex]
# determine the appropriate integer bin index
CurrentIndex = int(CurrentEnergy)

# append value to the StepLoss and EnergySteps analysis tools
StepLoss.append(aveIncidentEnergyLoss[int(CurrentIndex)])
EnergySteps.append(CurrentEnergy)

# this loops through all energies to check for ionization events.
# for each energy at which an ionization occured, add the appropriately scaled
# secondary spectrum to the newSecondaries list
for i in range(len(Energies)):
    for processName in ionStates:
        if ionCount[processName][i] != 0:
            for j in range(int(i/2)+1):
                newSecondaries[j] +=
                normalizedSpectra[processName][Energies[i]][j+1]*ionCount[processName][i]

# starting from the upper end of the newSecondaries list (this allows
# for subsequent secondaries to be added into the same list onto entries
# that have not yet been processed), loop through the nonzero entries in the
# newSecondaries list.
for i in reversed(range(len(newSecondaries))):
    if newSecondaries[i] != 0:
        # for each nonzero entry, this process is handled identically to that
        # of the primary particle with the exception of a scaling factor (the
        # values stored in the newSecondaries list) to allow for non-integer
        # weighting factors resulting from partial interactions at higher energies
        # and the accumulation of multiple partial secondaries at lower energies
        CurrentEnergy = Energies[i]
        CurrentIndex = i
        EnergySteps.append(CurrentEnergy)

        while CurrentEnergy > 8:
            for processName in binEnergyCSP:
                if processName in ionStates:
                    numTotalProducts[processName][e] +=
                    binEnergyCSP[processName][CurrentIndex]*newSecondaries[i]
                    ionCounter = binEnergyCSP[processName][CurrentIndex]*newSecondaries[i]

                    for j in range(int(CurrentIndex/2)+1):
                        newSecondaries[j] +=
                        normalizedSpectra[processName][CurrentIndex][j+1]*ionCounter
                else:
                    # do something

numTotalProducts[processName][e] +=
binEnergyCSP[processName][CurrentIndex]*newSecondaries[i]

CurrentEnergy = CurrentEnergy - aveIncidentEnergyLoss[CurrentIndex]
CurrentIndex = int(CurrentEnergy)
StepLoss.append(aveIncidentEnergyLoss[int(CurrentIndex)])
EnergySteps.append(CurrentEnergy)

# create a tally for all ion state products
ionTotal = 0
for name in ionStates:
    ionTotal += numTotalProducts[name][-1]

# append this ionTotal to the totalIonCount list
totalIonCount.append(ionTotal)

# define the next value in the workPerIon list as the
# current energy divided by the total number of ions created
if totalIonCount[-1] != 0:
    workPerIon.append(Energies[e] / totalIonCount[-1])

# end timer
end = timer()
elapsed = end-start
m, s = divmod(elapsed, 60)
h, m = divmod(m, 60)
# display elapsed time in both seconds and hours,minutes,seconds
print("%.2f seconds" % (elapsed))
print("%d:%02d:%.2f" % (h,m,s))

# save results in json files
json.dump(workPerIon, open('workPerIon_1to1000eV.txt', 'w'))
json.dump(numTotalProducts, open('numTotalProducts_1to1000eV.txt','w'))

A.2 getCrossSection.py

# This code is based largely on code provided by Dr. Christopher Walker
# and is used to take care of the retrieval, handling, and subsequent
# manipulations of the N2 cross sections

import N2_Model
from numpy import zeros
import copy
# This function is also directly based on code supplied by Dr. Walker, N2_Model.
# This function reads in a range of energy values and uses the N2_Model class definition to retrieve the corresponding cross sections. Both the list containing containing the names of all available N2 states (names) and the dictionary containing the cross sections (results) are returned.

def getCS(electron_energies):
    model = N2_Model.N2_Model()

    results = dict()

    excitations = model.Excitation_Model.electron_excitation_states

    ionizations = model.Ionization_Model.electron_ionization_states

    for state in excitations:
        results[state.__name__.lower()] = []
    for state in ionizations:
        results[state.lower()] = []

    names = copy.deepcopy(results)

    for value in electron_energies:
        for state in excitations:
            results[state.__name__.lower()].append(state(value))

        for state in ionizations:
            results[state.lower()].append(ionizations[state][value])

    return names, results

# This function converts the cross sections into fractional probabilities.
def getCSP(num_entries,CS):
    totalCSAtEnergy = zeros((num_entries))

    for i in range(0,num_entries):
        currentSum = 0
        for processName in CS:
            currentSum += CS[processName][i]
        totalCSAtEnergy[i] = currentSum

    # create the cross section probabilities (CSP) for each entry in integratedCS
and set its values by divided each integratedCS value by the appropriate value in totalICSAtEnergy

```python
CSP = copy.deepcopy(CS)
for i in range(0, num_entries):
    for processName in CSP:
        if totalICSAtEnergy[i] != 0:
            CSP[processName][i] = CS[processName][i] / totalICSAtEnergy[i]

return CSP
```

### A.3 `getNewSecondaries.py`

This code is used to calculate the number and energy distributions for new secondaries resulting from ionization events that occur below 1 keV

```python
import N2_Model
import numpy as np
import copy
from getStateEnergies import getVerticalEnergies

# This function calculates the predicted new secondaries based on the N2_Model implementation of the work by Green and Sawada
def getSecondarySpectra(binEnergies, ionStates):
    model = N2_Model.N2_Model()

    processDeltaE = getVerticalEnergies()

    # create dictionary to hold the spectrum data for each ionization process and each incident particle energy
    secondarySpectra = {}
    for name in ionStates:

        # create an empty dictionary entry for each process in ionStates
        secondarySpectra[name] = {}

        parameters = model.Ionization_Model.JGG.params[name.upper()]

        I = parameters[0]
        J = parameters[1]
        K = parameters[2]
        Gamma_s = parameters[3]
        Gamma_b = parameters[4]
        T_s = parameters[5]
        T_b = parameters[6]

        T_a = 1000
```
# in each process in ionStates, create an empty list at each incident energy
for energy in binEnergies:
    secondarySpectra[name][energy] = []

# the following loops create a spectrum of new secondaries produced for each
# incident particle energy
for energy in secondarySpectra[name]:
    for i in range(len(binEnergies)):
        E = energy  # admitted redundant but keeps with notation in publication
        T_m = (E-processDeltaE[name][0]) / 2.0
        Gamma = Gamma_s*(E/(E+Gamma_b))
        T_0 = T_s-(T_a/(E+T_b))

        if energy < I:
            # no secondaries produced below ionization threshold
            secondarySpectra[name][energy].append(0)
        elif binEnergies[i] <= T_m:
            # new secondaries may have energy up to half the
            # remaining energy of the incident particle
            A = (1e-16)*K/E*np.log(E/J)
            T = binEnergies[i]
            secondarySpectra[name][energy].append(A*(Gamma**2 / ((T-T_0)**2 + Gamma**2)))
        else:
            # no new secondaries above this limit
            secondarySpectra[name][energy].append(0)

return secondarySpectra

# This function normalizes the new secondary spectra such that each value
# becomes the fraction of new secondaries that have that kinetic energy
# for each incident energy and ionization process
def getNormalizedSpectra(secondarySpectra):
    normalizedSpectra = copy.deepcopy(secondarySpectra)

    for name in normalizedSpectra:
        for incidentEnergy in normalizedSpectra[name]:
            total = sum(secondarySpectra[name][incidentEnergy])

            for i in range(len(secondarySpectra[name][incidentEnergy])):
                if total != 0:
                    normalizedSpectra[name][incidentEnergy][i] = secondarySpectra[name][incidentEnergy][i]/total

    return normalizedSpectra

# This function finds the average kinetic energy of the new secondaries
# resulting from each incident particle energy and each ionization state
def getAveKinEnergy(normalizedSpectra):
    aveKinEnergy = {}
    for name in normalizedSpectra:
        aveKinEnergy[name] = np.zeros(((len(normalizedSpectra[name]), 2)))
        for energy in normalizedSpectra[name]:
            weightedAverage = 0
            for i in range(len(normalizedSpectra[name][energy])):
                # i is the kinetic energy and the corresponding entry in normalizedSpectra
                # is the fraction of resulting new secondaries that have this energy
                weightedAverage += i * normalizedSpectra[name][energy][i]
            aveKinEnergy[name][int(energy), 0] = i
            aveKinEnergy[name][int(energy), 1] = weightedAverage
    return aveKinEnergy

A.4 getStateEnergies.py

# These are functions used in determining the energy lost in taking a ground state N2 molecule
# to a given excited/ionization state. These ionization state energy values are solely those
# required to get the molecule to that particular state and make no account for the kinetic
# energy of any secondary electrons.

# calculation of the energy used in determining the energy difference between two states
def stateEnergy(Te, we, wexe, v):
    T = Te + we*(v + 0.5) - wexe*(v + 0.5)**2
    return T

# Returns the single most probable energy difference between a given state and the ground state.
# These are based on the transition with the highest Franck-Condon factor and have been calculated
# manually using the equation above. A weighted average of each vibrational state energy by its
# corresponding factor would be more accurate. Cross section values are rounded to the nearest 0.0001
# This corresponds to the level of precision maintained when manually transferring these values to
# the MCAFS model to assure consistent account for energy loss.

def getVerticalEnergies():
    x1sg_0energy = stateEnergy(0, 2358.57, 14.324, 0)
    x2sg_0energy = stateEnergy(0, 2207, 16.10, 0)
    eVConversion = 8065.5
    E = {
        'a3su': [(stateEnergy(50203.63, 1460.64, 13.872, 8) - x1sg_0energy) / eVConversion],
        'b3pg': [(stateEnergy(59619.35, 1733.39, 14.122, 2) - x1sg_0energy) / eVConversion],
        'w3du': [(stateEnergy(59808, 1501.4, 11.6, 7) - x1sg_0energy) / eVConversion],
        'a1pg': [(stateEnergy(69283.06, 1694.208, 13.9491, 3) - x1sg_0energy) / eVConversion],
        'b3su': [(stateEnergy(66272.47, 1516.88, 12.181, 7) - x1sg_0energy) / eVConversion],
        'a1su': [(stateEnergy(68152.66, 1530.254, 12.0747, 7) - x1sg_0energy) / eVConversion],
        'w1du': [(stateEnergy(72097.4, 1559.26, 11.63, 6) - x1sg_0energy) / eVConversion],
        'c3pu': [(stateEnergy(89136.88, 2047.178, 28.4450, 0) - x1sg_0energy) / eVConversion],
    }
'e3sg':[(stateEnergy[95858,2185,0,0]-x1sg_0energy)/eVConversion],
'a1sg':[(stateEnergy[100016,0,0,0]-x1sg_0energy)/eVConversion],
'c1pu':[(stateEnergy[104476,2192.20,14.70,0]-x1sg_0energy)/eVConversion],
'c1su':[(stateEnergy[104519,2201.78,25.199,0]-x1sg_0energy)/eVConversion],
'b1pu':[(stateEnergy[101675,634.8,0,0]-x1sg_0energy)/eVConversion],
'b1su':[(stateEnergy[104498,760.08,4.418,0]-x1sg_0energy)/eVConversion],

'x2sg':[15.5808],
'a2pu':[15.5808+((stateEnergy[9166.95,1903.7,15.02,1]-x2sg_0energy)/eVConversion)],
'b2su':[15.5808+((stateEnergy[25461.46,2419.84,23.189,0]-x2sg_0energy)/eVConversion)],
'c2su':[15.5808+((stateEnergy[64608.1,2071.5,9.29,6]-x2sg_0energy)/eVConversion)],
'd2pg':[15.5808+((stateEnergy[52318.2,907.71,11.91,0]-x2sg_0energy)/eVConversion)],
'ev40':[40],}

for name in E:
    E[name][0] = round(E[name][0],4)

return E
APPENDIX B – Hardware & Software Specifications

B.1 Hardware

For the evaluation of the required runtime presented in CHAPTER VI, both methods were executed on the same machine using single thread implementation. The following is a description of the specifications of that machine:

Model: Dell Precision M4700 mobile workstation
Processor: Intel Core i7-3840QM CPU @ 2.80GHz x 8
Memory: 16 GB
Storage: 256 GB SSD

Operating System: CentOS 7
Kernel: Linux 3.10.0-514.21.1el7.x86_64
Desktop Environment: MATE 1.16.2

B.2 Software

The Geant4 results presented in CHAPTER III were obtained using Geant4 version 10.1.2 on a desktop computer running a Scientific Linux Cern 6 distribution of Linux OS. The Geant4 results presented in CHAPTER IV were obtained using Geant4 version 10.2.1 installed on a desktop computer running a Fedora23 distribution of Linux OS.

Implementation of the MCAFS software was achieved using Python 2.7.13 via Anaconda 4.3.1, and the Averaged Loss Method was achieved using Python 3.5.2 via a separate installation of Anaconda 4.3.1. Both of these were run on the mobile workstation described above.
REFERENCES


(20) Haynes, W. CRC Handbook of Chemistry and Physics, 91st ed.; CRC Press, Inc.: Boca Raton, FL.


