A HIGH-ALTITUDE NUCLEAR ENVIRONMENT SIMULATION

by

RYAN D. WHITE

B.S., Kansas State University, 2008

A THESIS

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Mechanical and Nuclear Engineering
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2009

Approved by:

Major Professor
Dr. Kenneth Shultis

Abstract

A program which calculates the radiation dosage to a predetermined set of components inside of a kill vehicle as a result of natural or artificial radiation sources has been developed for use within the confines of a parent external simulation. This dose can then be used to determine if a critical component has malfunctioned or failed completely, thereby rendering the interceptor unable to finish its mission. Knowledge of system and component performance as a function of incident high-energy particles leads to better battle management planning, CONOPS, and potentially a more efficient shielding design to achieve a higher probability of mission success.

Table of Contents

List of Figures

List of Tables

CHAPTER 1 INTRODUCTION

1.1 Motivation for Present Work

The motivation for this investigation is to simulate and accurately model the radiation environment from a high-altitude nuclear explosion and the resulting radiation effects on a ballistic missile interceptor. While working at Raytheon Missile Systems (RMS), I was asked to simulate the radiation effects of a high altitude nuclear burst and to calculate how the increased radiation levels would affect mission success for a missile interceptor scenario. Knowing how an interceptor would function in a given radiation environment would assist with the design process for a new generation of interceptors as well as create a more acceptable Concept of Operations (CONOPS). These actions taken together would maximize the usefulness of interceptors for a given mission.

The Defense Threat Reduction Agency (DTRA) currently has standalone simulations and software packages that can accurately model the individual effects of a nuclear burst, as well as provide some basic estimates of radiation effects to satellites and other space vehicles. The DTRA models include the ability to calculate the desired initial and persistent radiation effects, but each does not meet the necessary requirements for a fast-running simulation model. Specifically, these models cannot be easily integrated within the scope of a much larger simulation. An additional limitation is that each of these simulations calculates a specific initial or persistent radiation phenomenon rather than computing the complete phenomenology. There is currently no single simulation that unifies all of the desired characteristics into a single software package.

Given the computational environment of RMSs internally-developed missile flight simulator, none of the available DTRA programs will suffice alone. A single unified simulation, with a nearly realtime capability to model all of the desired generation and transport phenomena, from a nuclear burst to dosages within an interceptor component as well as easy integration into the flight simulation is needed. This work describes the physics incorporated and describes the workings of a software package that fulfills these requirements for calculating an artificial radiation environment.

Midway through development of the simulation, it was determined that natural radiation from both the trapped radiation held within the Van Allen belts as well as solar events and general space weather should also be modeled. The National Aeronautics and Space Administration (NASA) and the European Space Agency (ESA) provide publicly-available models that can simulate trapped radiation; these models have been adapted for use in this software. Models also exist for solar weather such as storms and flares but due to the model design they are not suitable for calculations over a trajectory, but rather over many tens of orbits. These models are not explored in depth in this document due to their highly unpredictable behavior when predicting a ballistic trajectory, as well as the lack of available data needed to accurately model them.

1.2 Problem Overview

The presence of multiple nuclear powers in the world implies that there exists the possibility of a nuclear attack. One scenario could include of the launch of a nuclear-tipped intercontinental ballistic missile (ICBM) from an aggressor country and the use of a ballistic interceptor by a defending country. This ballistic interceptor would be equipped with one or more kill vehicles (KV), each targeting a re-entry vehicle (RV).

To fully understand the complexity of the phenomena and effects of a nuclear burst, both underground and above ground nuclear tests have been performed in the past. Because current international treaties prevent nuclear testing, this is no longer an option. To help analyze and predict possible scenarios as well as the outcome of each situation without performing a nuclear test, large-scale simulations are used which are composed of many smaller, specific modules. Simulations have been widely used for nuclear applications, and can trace their origins to the use of the ENIAC computer during the development of the first hydrogen bomb. Since that time, computers have become a staple tool in the development and testing of numerous nuclear applications, especially in cases where the phenomena analyzed are unable to be tested extensively. The simulation module described in this document focuses on the radiation effects or an interceptor subject to both natural space radiation and radiation produced by a nuclear device.

Trapped particles surrounding the earth constitute a significant source of natural radiation that can play a role in compromising mission success. The types of radiation particles a space

vehicle can expect to encounter in this environment are energetic protons and electrons. A running tally of total dose and dose rate from these particles is indicative of system performance. Consequently, it is desirable for these values to be calculated, thereby, allowing a computer simulation to determine the performance of critical components as the simulation progresses.

There is a high probability of encountering either a natural or an artificial radiation environment throughout the trajectory of a typical kill vehicle (KV) mission. In the event of multiple KV/RV intercepts, there is a likelihood that one or more of the KVs will encounter a manmade nuclear environment. It is also possible for a KV to experience a natural radiation environment due to the trapped particles surrounding the earth. In either scenario, these KVs are exposed to particles with energy distributions which could potentially damage electronics, sensors, and other onboard equipment, thereby, increasing the likelihood that the KV will fail to complete its mission.

Knowledge of how weapon effects are created and propagated from an exo-atmospheric nuclear explosion allows the flux of radiation incident upon a KV to be estimated. This knowledge, in turn, can be used to compute the individual dose rates and total doses experienced by the components of interest within the interceptor. The computed total dose rate may then be compared to a predetermined threshold value to indicate if the component would survive intact, malfunction, or fail completely if used for a particular engagement geometry.

The types of weapon-induced radiation which a KV in the upper- or exo-atmosphere can reasonably be expected to encounter are X rays, neutrons, negatively charged beta particles (electrons), and gamma rays. When a particle interacts with a medium, it loses some of its energy, which is deposited into the interacting medium. One interaction is usually insignificant but, if enough interactions take place, or occur rapidly enough, malfunctions and failures within critical devices can occur. Thus, it is important to understand the radiation doses that a KV may expect to encounter and to determine how much radiation energy penetrates and is deposited within the components comprising the KV.

1.3 Thesis Organization

This document describes software designed to calculate the effects of natural and artificial radiation incident upon a space-borne interceptor. The strategy is to provide a brief

description of the phenomena involved and their characteristics, and to describe the physics and methods applied by the model when applicable.

Some of the major items considered in this document are identified below:

Chapter 2.1 – The radiation trapped within the Earth's magnetic field is described and a description of the model used to help calculate the effects of geomagnetically trapped radiation is presented.

Chapter 2.2 – The radiation emitted by a nuclear burst is described, and techniques to calculate the temporal flux spectra incident upon an interceptor are developed.

Chapter 2.2.4 – The techniques required to calculate effects of atmospheric attenuation on particle flux as well line of sight calculations are presented.

Chapter 2.2.5 – The creation of artificial Van Allen belts within the Earth's magnetic field as a result of charged particles from a nuclear burst is discussed.

Chapter 2.3 – Key model assumptions, as well as integration requirements for insertion into larger simulation packages, are discussed.

Chapter 3.1 – A predictive software model and path towards development for a simulation calculating the natural and artificial radiation environment is described, as well as its effects upon a space vehicle.

Chapter 3.3 – Inputs to the simulation from a larger parent simulation are presented and discussed.

Chapter 3.6 – Potential methods to validate simulation results against hand calculations as well as against industry-standard models described.

Chapter 4.1 –Potential enhancements to the model for higher fidelity results are presented with discussion.

Appendices – The appendices list the software members and methods for the simulation, and a brief description of function, inputs and outputs for all classes is presented. Sample input and output files are described.

CHAPTER 2 Sources of Radiation

2.1 Natural Space Radiation

The main types of anticipated natural radiation in a space environment consist of galactic cosmic rays (GCRs), radiation associated with solar events such as flares and storms, and the electrons and protons that are trapped within the earth's Van Allen radiation belts. Because the dosage contribution due to GCR's is small compared to the trapped radiation dose, it is neglected in the present work [Jones, 2000]. There are also no publicly-released industry-standard models that can simulate solar events, so these unpredictable sources of natural space radiation are also neglected [Tylka et al, 1997]. In light of these simulation limitations, only the trapped radiation within the Van Allen belts is considered here.

The earth approximates a large spherical dipole magnet with field lines extending from the magnetic poles as shown in Figure 2.1. These field lines trap charged particles, leading to the creation of the trapped radiation belts surrounding the earth. Locations near the poles where the magnetic field lines converge have the highest trapped density of charged particles, while locations near the magnetic equator tend to have a lower trapped density of charged particles for both protons and electrons.

Electron and proton radiation from the sun and other galactic sources become trapped in the Van Allen belts and contribute to the basic toroidal structure. There is an inner belt consisting of both electrons and protons, surrounded by an outer belt consisting of electrons [Bass et al, 1995]. Electron energies in both belts are generally greater than a few hundred keV and proton energies in the inner belt are generally greater than a few MeV. Between these regions there is an empty "slot" which contains a relatively small number of particles. Figure 2.2 shows the basic structure of the two distinct belts, the slot between them, as well as an 11º shift from the magnetic axis of the earth compared to the rotational axis.

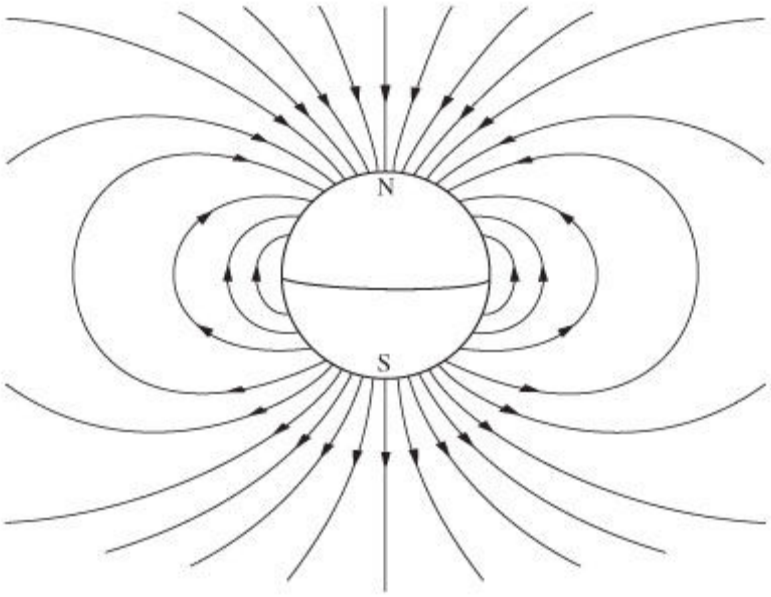

Figure 2.1 Spherical magnetic dipole showing magnetic field lines [Cross, 2007].

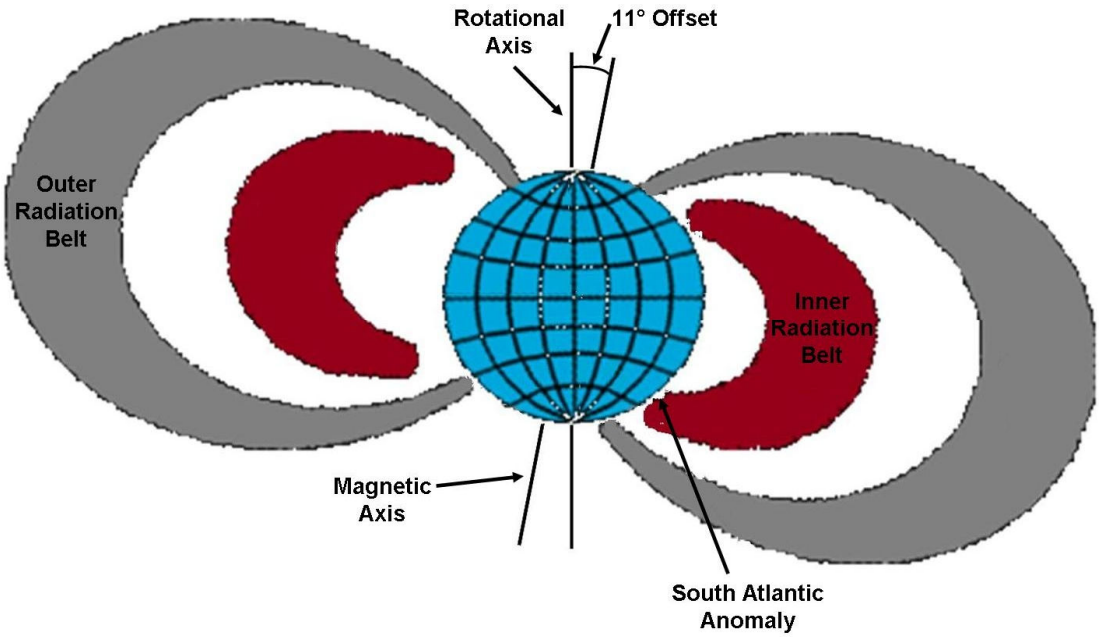

Figure 2.2 Earth depicting inner and outer Van Allen belts and showing location of the South Atlantic Anomaly. The geomagnetic axis is offset by $11^{\mathbf{o}}$ from the rotational axis [taken from http://www.ok4me2.net/wordpress/wp-content/uploads/radiation-belt.jpg].

After electrons and protons are trapped within the geomagnetic field, they travel along the field lines of the earth as shown in Figure 2.3 and also have a "drift" motion perpendicular to the field lines. The perpendicular motion causes electrons to drift eastward and protons to drift westward [Cross, 2007]. As the particles gyrates around the field lines and approaches the earth, the angle which is formed between the gyration moment approaches perpendicularity to the field line. When the gyration becomes perpendicular to the field, conservation of energy causes the particles to reverse direction and travel back along the field line [Tylka et al, 1997]. This phenomenon occurs in a region known as a 'mirror point'.

This reversal of motion leads to bouncing along the field lines between the pairs of magnetic mirror points, with a period of roughly one second. The mirror points are also called the magnetic conjugates. Electrons also gyrate around the field lines with a period on the order of 1 ms, creating "beta tubes". These tubes contain very high electron radiation concentrations.

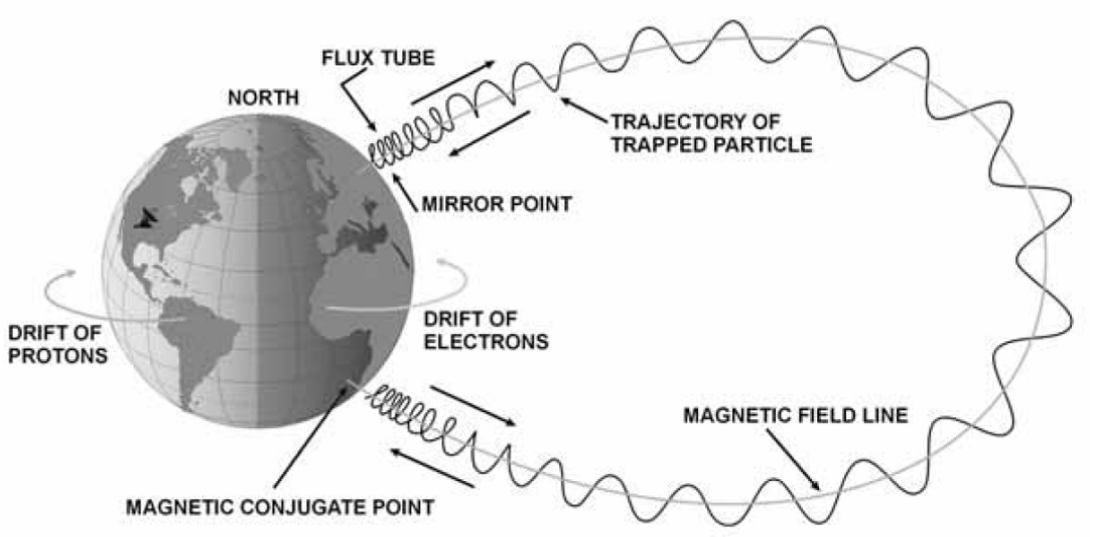

Figure 2.3 Particle motion within magnetic field lines [Cross, 2007]. Once particles are trapped, they gyrate around the field line until they hit a mirror or conjugate point near a magnetic pole.

Because of the influence of solar wind on the trapped radiation belts they are not symmetrical at high altitudes about the magnetic axis, but instead have a bowshock called the magnetopause on the side of the earth facing the sun and a magnetotail region on the opposite side, as shown in Figure 2.4. In this regard, the earth's magnetic field act as a shield protecting the earth and orbiting objects from solar and galactic radiation [Tylka et al, 1997].

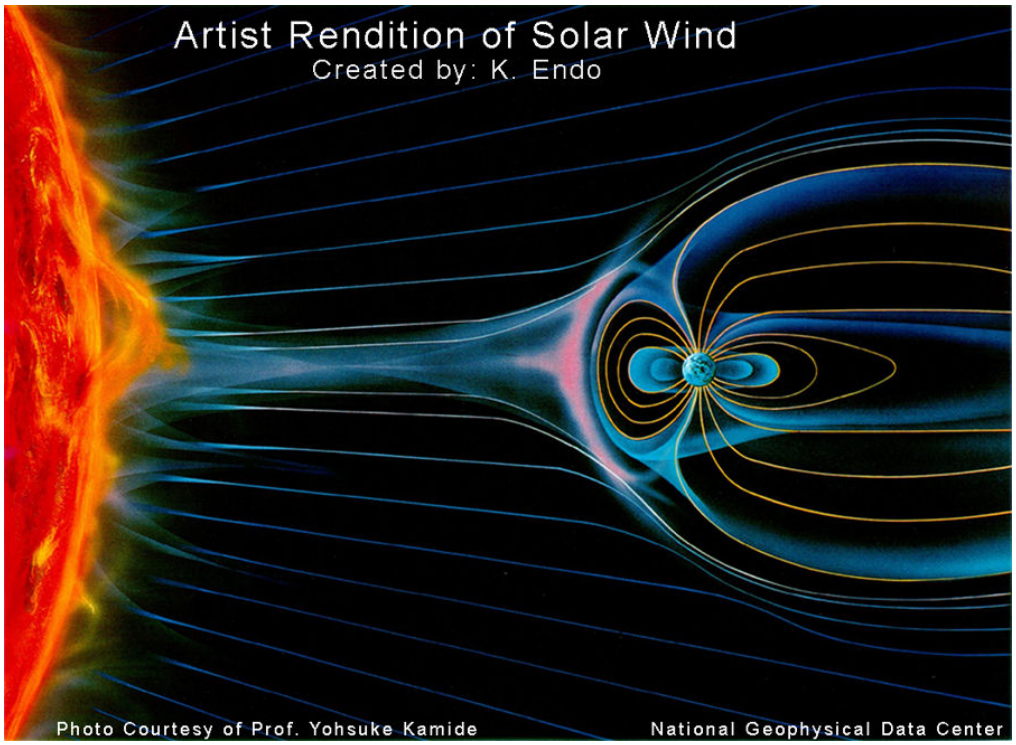

Figure 2.4 Artist rendition of solar wind and its effects on the trapped radiation belts surrounding the Earth. Magnetopause and the magnetotail are depicted. Note that low altitude belts are roughly symmetrical about the geomagnetic axis. Not to scale. [astroprofspage.com]

Because of highly variable solar activity, the radiation belts are highly dynamic [Tylka et al, 1997]. In addition to the bow shock and magnetotail regions changing greatly at high altitudes as the belts rotate around the magnetic axis of the earth, the activity within the shells varies as a function of solar weather. Examples of solar weather are coronal mass ejections, solar flares, and the resulting solar winds. During a solar maximum, the atmosphere is heated as a result of increased solar UV output, and thus, the atmosphere expands to higher altitudes [Sawyer and Vette, 1976]. This leads to additional air mass near the geomagnetic mirror points, thereby increasing the interactions of trapped particles with the air and causing their removal from the flux tubes. This increases solar activity leads to reduced radiation levels in the radiation belts. Similarly, the radiation belts are at their highest levels during a period of solar minimum.

The Van Allen belts reach a low altitude of 200 to 1000 km near the magnetic poles, and extend to 7 R_E near the magnetic equator, where R_E is the radius of the earth For the purposes of this work, only the regions less than 2000 km altitude are considered. The Van Allen belts are

roughly symmetrical about the geomagnetic axis at these low altitudes, and thus the effects of solar wind on the shape of the Van Allen belts need not be considered.

Because of the differences between the magnetic and rotational axis of the earth, there exists a local minimum in the magnetic field centered in the South Atlantic causing the radiation belt to penetrate to a lower altitude [Sawyer and Vette, 1976]. Referred to as the South Atlantic Anomaly (SAA), this region covers a large area and contains a very high proton radiation density because of the increased amount of atmosphere that the radiation belt contains. Figure 2.5 shows the proton radiation levels measured behind 2.06 g/cm^2 of shielding aboard the Russian Mir Space Station in 1996 as well as a longitudinal flux profile for 32º latitude at approximately 325 km altitude [Tylka et al, 1997]. These data show the SAA has a peak dose rate at approximately -43º longitude, and drifts westward at approximately 0.27º per annum when compared with previously taken data. Because the atmospheric density causing the proton interactions varies with solar cycle activity, the SAA is also expected to also follow solar cycle changes.

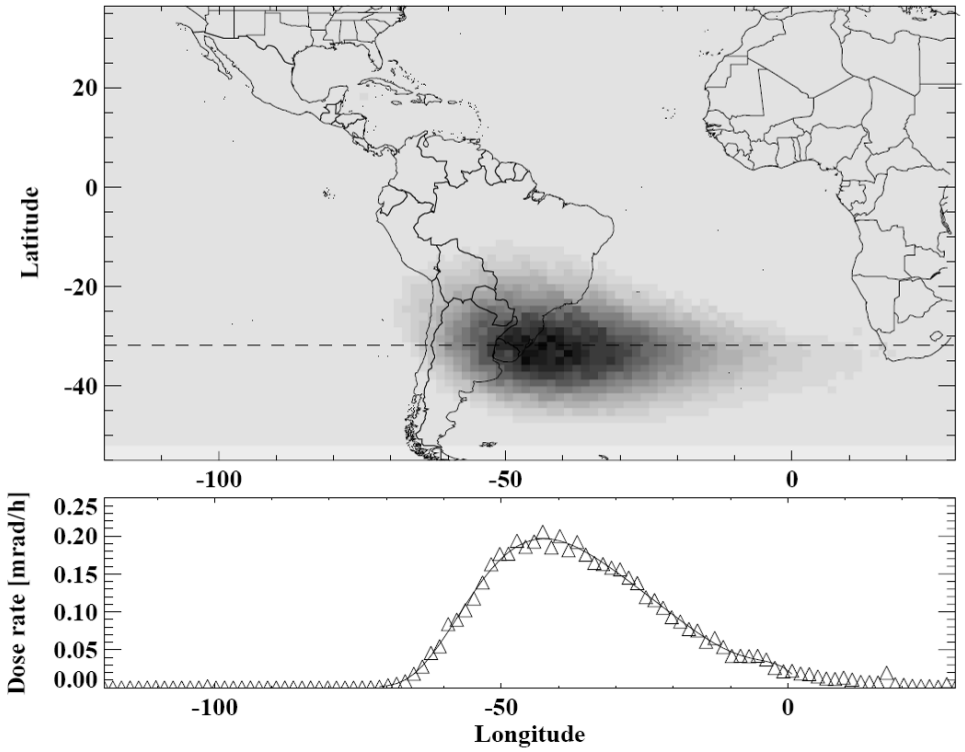

Figure 2.5 Proton radiation levels as measured on the MIR Space Station showing the location and intensity of the South Atlantic Anomaly. Longitudinal proton flux profile is for 32°S (dashed line in upper figure) [Tylka et al, 1997].

2.1.1 The AP8 and AE8 Models

Because of the dynamic nature and lack of an analytic model to calculate the locations and magnitudes of the trapped radiation belts, the publicly-available NASA-developed AP8/AE8 empirical model is used to simulate the trapped radiation surrounding the earth. This model has been an industry standard since its inception in 1976. The omnidirectional integral electron and proton fluxes ranging from 1.15 to 8 earth radii can be calculated from the empirical data taken by the AZUR and OV3-3 satellites [Sawyer and Vette, 1976, Vette, 1991]. This model contains data for both a solar minimum (epoch 1964) as well as a solar maximum (epoch 1970) [TRP: AP8MIN/AP8MAX Models].

The AP8/AE8 model is used to determine the flux energy spectrum at a point or in a plane. Figures 2-6 and 2-7 depict the model predicted trapped radiation radially from the earth as well as projected onto the earth's surface [SPENVIS Background].

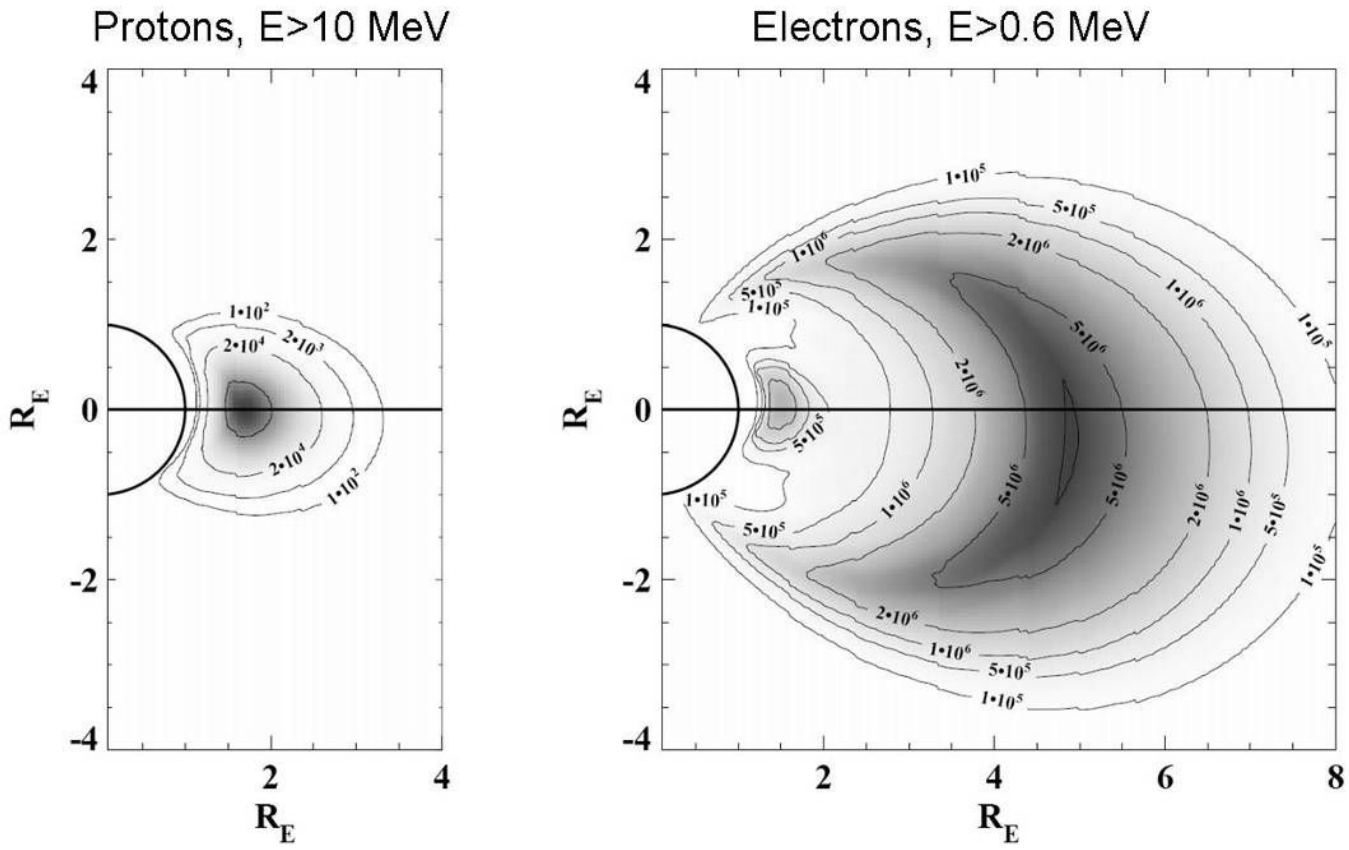

Figure 2.6 Radial flux profiles for protons and electrons calculated using the AP8/AE8 software packages [Tylka et al, 1997]. Note there are one trapped proton belt and two trapped electron belts.

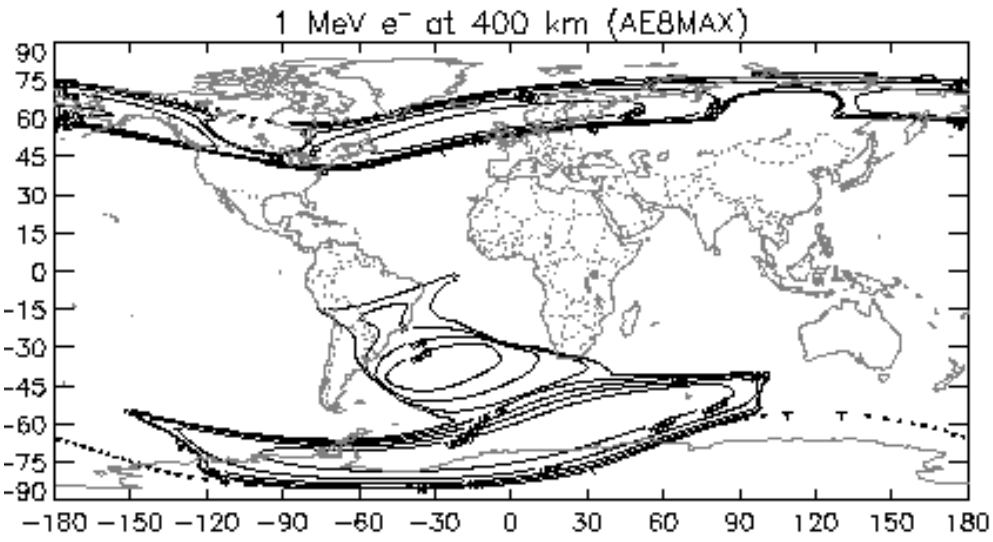

Figure 2.7 AEMAX flux mat at 400 km altitude. Note the higher particle flux densities near the geomagnetic poles as well as the SAA.

Because trapped electron concentrations are correlated with the phase of the solar cycle and trapped proton concentrations are anti-correlated with solar cycle phase, different flux maps designated AE8 and AP8 respectively have been developed for each particle species [Tylka et al, 1997, Sawyer and Vette, 1976, Vette, 1991]. These flux maps were generated using data collected from orbiting satellites. Figures 2.8 and 2.9 show the averaged minimum and maximum trapped proton and electron spectra for one hundred simulated orbits of the Hubble Space Telescope (HST), at an altitude of 600 km and an orbit inclination of 28.5º [Tylka et al, 1997].

Because the AP8/AE8 model is based entirely on empirical data, it does include the effects of the earth shielding and the magnetotail as well as the SAA. SPENVIS online models were used to show the proton flux map in Figure 2.10.

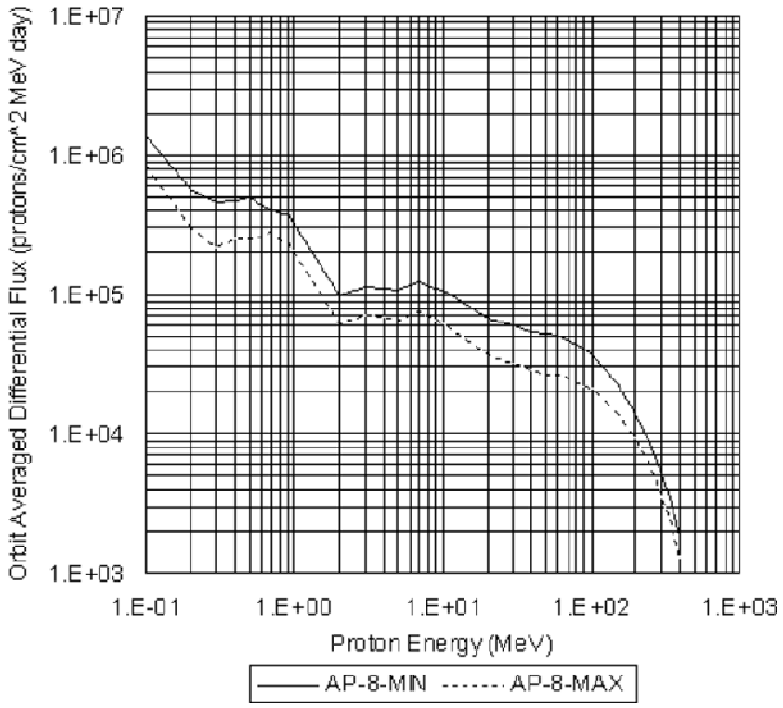

Figure 2.8 Orbit averaged trapped proton spectra for HST showing differences between the AP8MAX and the AP8MIN energy dependent flux densities [Tylka et al, 1997].

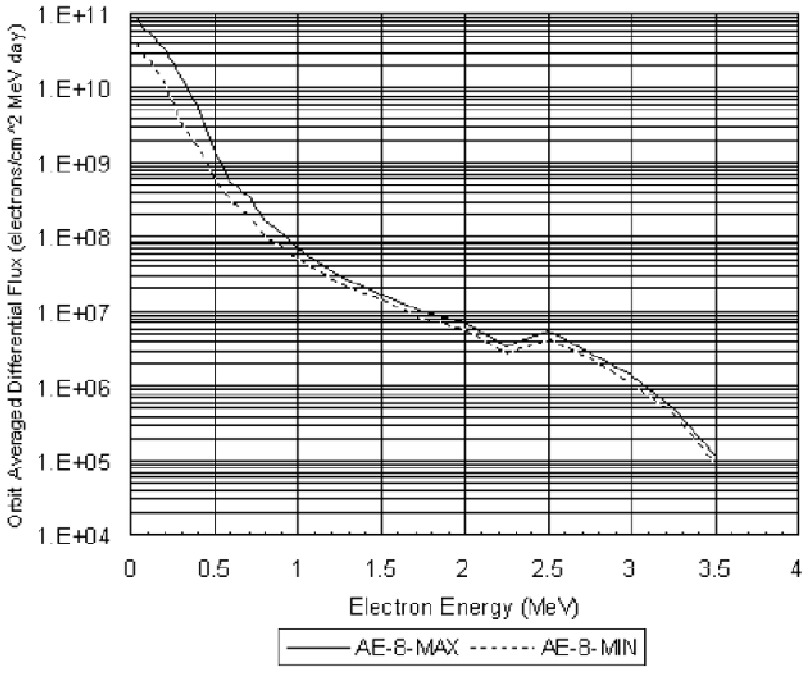

Figure 2.9 Orbit averaged trapped electron spectra for HST showing differences between the AE8MAX and the AE8MIN energy dependent flux densities [Tylka et al, 1997].

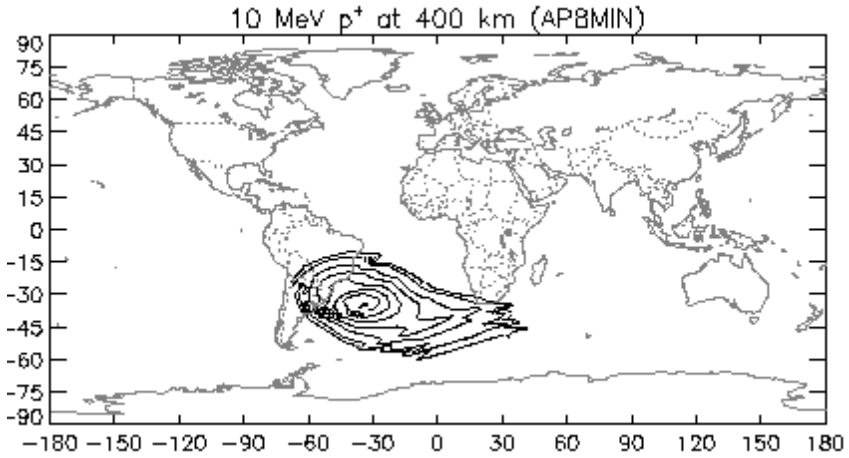

Figure 2.10 APMIN flux map showing location and center of SAA.

2.1.2 Limitations to the The AP8/AE8 Model

The AP8/AE8 model was correct at the time of inception in 1976, but because it is based entirely on empirical data and the fact that the radiation belts are highly dynamic, the model is not entirely correct at the current date. The AP8 model predicts the SAA has a 1970 max flux at -36º longitude, while it was measured to be near -43º longitude in 1996 due to a yearly westerly drift of 0.27º. Others have attempted to update the model to the current epoch, but their models are not publicly available [Pfitzer, 1991].

Another limitation of using two main data sets each correlating to a solar minimum and solar maximum is that there is no way to interpolate between them to determine the trapped flux at an intermediate stage in a solar cycle. For this reason it is recommended that the APMIN and AEMAX flux profiles are used in order to simulate a larger trapped flux, thus leading to a more conservative natural environment model.

The primary focus of the efforts used to create the AP8/AE8 data sets were in the magnetic tail and near magnetosphere boundaries. The inner magnetosphere was not extensively studied at the time of the AP8/AE8 model's inception [Pfitzer, 1991]. For this reason, the trapped flux for both particles is known to have a higher error below 1000 km altitude, with errors greater than a factor of two to three below 250 km altitude [TRP: Limitations].

The usage of data sets to create the model also eliminates the calculation of the influences of space weather on the trapped radiation belts. Solar flares, coronal mass ejections, GCR's and other space weather events and their associated particle injections are not able to be modeled

[Armstrong and Colburn, Trapped Radiation Model Uncertainties, 2000]. Artificial particle injections are not able to be modeled for the same reason.

The AP8/AE8 model is designed to work well when calculating total doses and endurance over long durations in orbit, as anomalies are averaged out of the results. For short missions, the model accuracy degrades. Commonly quoted errors are a "factor of two" [Pfitzer, 1991]. Worst case scenarios for minutes, hours, days, weeks, and the effects of transient peak environments are not accurately modeled [Armstrong and Colborn, Trapped Radiation Model Uncertainties, 2000]. The model over-predicts the dose behind aluminum shielding associated with electrons with energies greater than 1 MeV, and under-predicts the unshielded surface dose associated with a decreased electron flux at energies less than 100 keV [Armstrong and Colborn, Evaluation of Trapped Radiation Model, 2000]. In general, the expected proton spectra is harder than the expected electron spectra, because of the shielding of the instruments used to gather the data sets [Tylka et al, 1997]. Despite these limitations of the AP8/AE8 model, it is incorporated into the natural radiation environment simulation because it is still considered an industry standard and is the best publicly available model.

2.2 Artificial Radiation Sources

To calculate the effects of radiation from a nuclear detonation on a KV, the weapon emissions (consisting primarily of neutrons, prompt and secondary gamma and X rays, and electrons) and their interaction with the earth's magnetic field must be understood. Approximately 85% of the energy released by an endo-atmospheric nuclear detonation is in the form of air blast and shock, thermal radiation, and heat. The remaining 15% of a blast's energy is in the form of various radiation particles, of which 5% is prompt gammas and X rays [Dolan and Glasstone, 1977]. The present work takes only the radiation particles into account, as blast/shock is of much lesser concern at high altitudes than at lower altitudes, and thermal radiation is beyond the scope of this work.

2.2.1 Types of Artificial Radiation Associated with a Nuclear Burst

The radiation created by a nuclear event is divided into two temporal periods: prompt and persistent radiation. Prompt radiation is a result of short term nuclear processes, and is emitted almost instantaneously. Persistent radiation is emitted by the fission products, neutron activation of matter surrounding the explosion, and the debris field.

Immediately after a nuclear event (<1 μs), prompt gamma rays and X rays are emitted [Dolan and Glasstone, 1977]. The initial photon flux is associated with the fission process, where a neutron enters a heavy nucleus, is absorbed and causes the nucleus to split into smaller isotopes thus releasing energy and radioactive species [Dolan and Glasstone, 1977]. Persistent gamma rays are emitted by fission products, as are isotopes that have been neutron activated. Neutron activation is a process by which a stable isotope is changed to an unstable isotope by the addition of a neutron, thus creating a secondary radiation environment [Dolan and Glasstone, 1977].

Prompt photons interact with surrounding materials to produce prompt beta particles in the form of photoelectrons, relativistic Compton electrons, and electron/positron pairs. Compton electrons are stripped off of any nearby matter and give off low energy photons under acceleration, causing the electro-magnetic pulse (EMP) associated with a nuclear burst [Dolan and Glasstone, 1977]. EMP calculations and effects are beyond the scope of this work.

During the fission and decay processes, neutrons are also emitted. A maximum in the neutron flux occurs approximately 10 μs after the nuclear event, and is attributed to the fission reactions. Persistent neutrons come from fission products decaying by neutron emission, and also (γ, n) reactions in which a gamma ray dislocates a neutron from a nucleus.

Photons travel at the speed of light but because neutron particles possess a finite rest mass, their speed is a function of particle energy. A thermal neutron with an energy of .0253eV travels at 2,200 m/s, while a 10 MeV neutron travels at 43,600 km/s. Thus, it is important to track the energy of a group of neutrons as well as their time of creation in order to get an accurate estimate of flux at the KV location. The notional neutron flux of an arbitrary weapon incident as a function of time upon an object 100 km from the point of detonation is shown in Figure 2.11.

While electrons can travel at nearly the speed of light, they exhibit more complex behavior than simply a burst and line-of-sight propagation. These particles can become trapped in the earth's magnetic field creating beta tubes [Dolan and Glasstone, 1977]. Once caught in a magnetic field line, these particles gyrate around the field line with a period of approximately one ms while oscillating between the earth's magnetic poles every second. The belts of negatively-charged electrons rotate from west to east around the earth, with a period of roughly

29 minutes [Dolan and Glasstone, 1977]. This high electron flux has been responsible for rendering unshielded satellites non-functional in a matter of days or weeks [Hess, 1964].

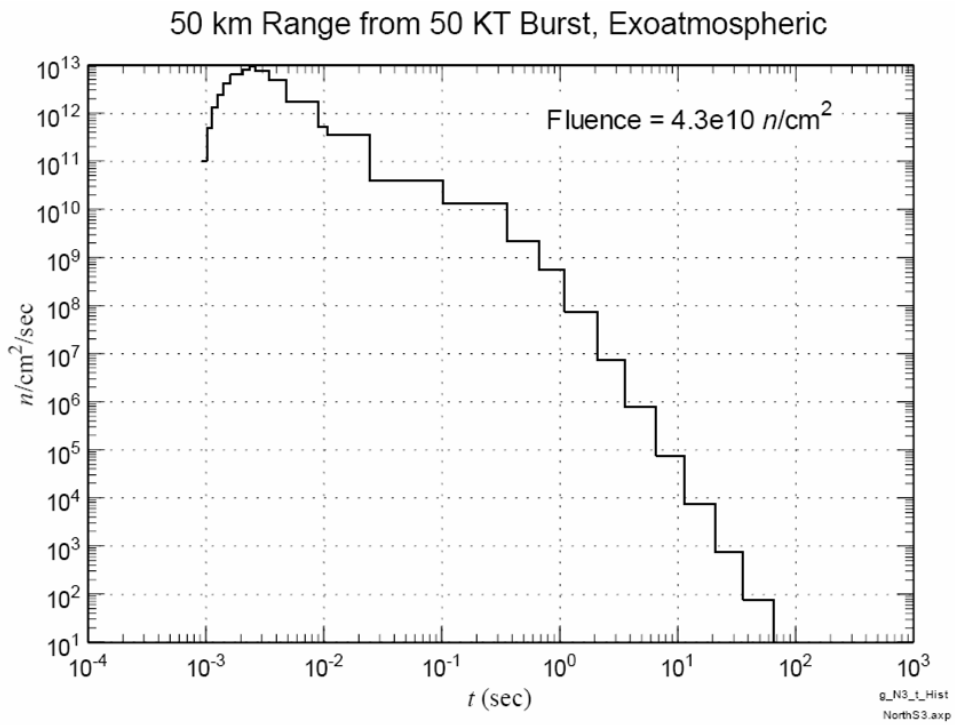

Figure 2.11 Prompt neutron flux as a function of time after a 50 kT exo-atmospheric burst at a distance of 50 km [tpub.com]. Approximate burst neutron spectrum can be determined using time of arrival information shown in chart.

2.2.2 Flux Calculations

To calculate the neutron, gamma ray, or X-ray flux incident upon a KV, a general isotropic propagation equation is used. Because the size of a nuclear weapon is very small in comparison to the blast area within which objects are affected, the nuclear event can be represented as a point source, emitting particles isotropically assuming that the event is in a non-attenuating medium. Equation 2.1 shows the observed flux ($cm^{-2}s^{-1}$) at a KV location as a function of energy at a radius r from the nuclear event, where S_p is the source strength and μ is the attenuation coefficient for the surrounding medium [Shultis and Faw, 2000].

$$\varphi(E) = \frac{S_p(E)}{4\pi r^2} e^{-\mu r} \ . \tag{2.1}$$

To calculate an unattenuated dose rate or exposure from the flux, the reaction rate per incident particle as a function of energy is multiplied by the number of incident particles as shown in Eq. 2.2. Initial calculations are for a non-attenuating medium, therefore $\varphi(E)$ is transformed into an uncollided flux $\varphi^0(E)$ because the attenuation coefficient μ is zero.

$$D^0(E) = \varphi^0(E)\Re(E) \, . \tag{2.2}$$

The fluence-to-dose conversion coefficient $\Re(E)$ measured in (rad cm^2) can be found using either lookup tables for simple geometries and media [Shultis and Faw, 2000], or using advanced Monte Carlo techniques to simulate particles incident upon a complex geometry. The end result of Eq. 2.2 is an unattenuated dose rate measured in (rads s^{-1}).

These two equations are adequate when considering only a point source either emitting instantaneously or when it is emitting a constant particle flux. Because neither of these are the case for a nuclear event exuding prompt and delayed radiation, Eq. 2.2 must be modified to make it a function of time-after-event, resulting in Eq. 2.3.

$$D^0(E,t) = \varphi^0(E,t)\Re(E) \, . \tag{2.3}$$

It is important to keep track of the photon energy spectra, as attenuating mediums have a lower attenuation coefficient for higher energetic photons when compared to lower energy particles. An example of this phenomenon is shown in Figure 2.12 for iron. This means that, the lower the energy of the particle, the higher the chance of an interaction occurring and energy being deposited. Highly energetic particles tend to pass through interacting mediums, often seemingly unaffected by their presence.

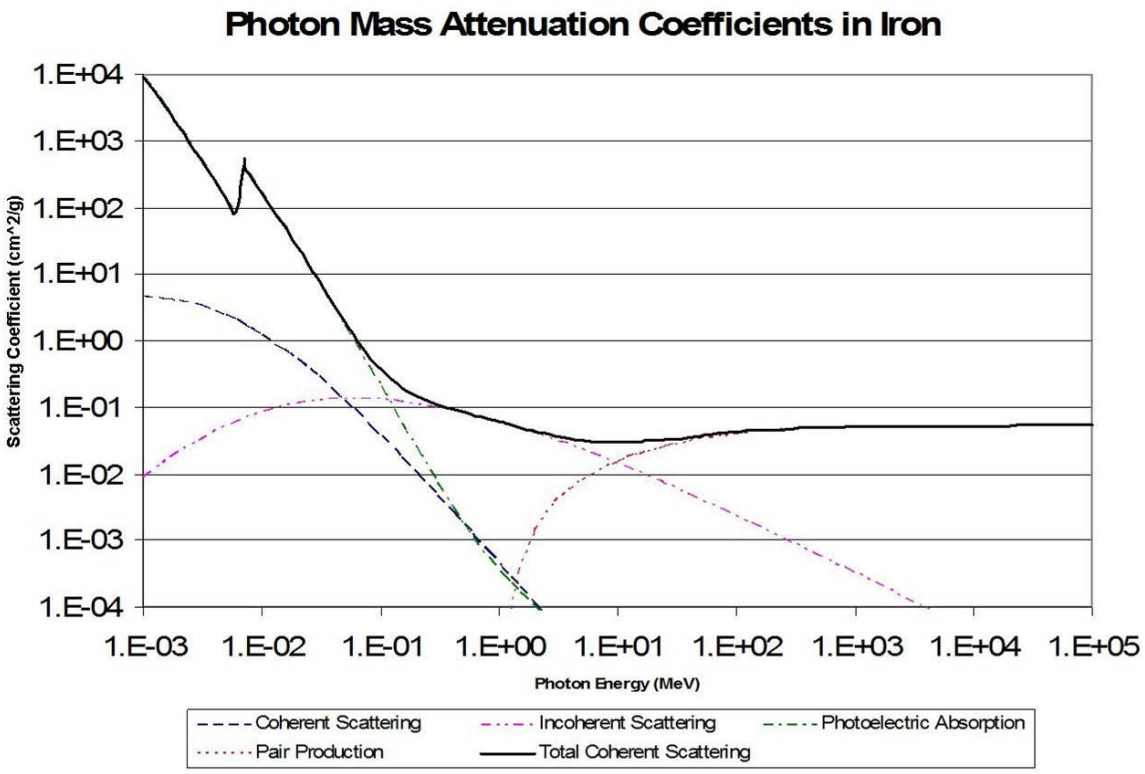

Figure 2.12: Photon Mass Attenuation Coefficients in Iron [Shultis and Faw, 2000]

2.2.3 Angle of Incidence

For the analyses performed by this software, the reaction rates are a function of the angle of incidence (α) from the axis of the interceptor relative to the blast front, as shown in Figure 2.13 [Northrup, 1996]. This information allows the incident radiation at a given angle of incidence upon an interceptor to help calculate an accurate dose measurement based on shielding strengths and weaknesses.

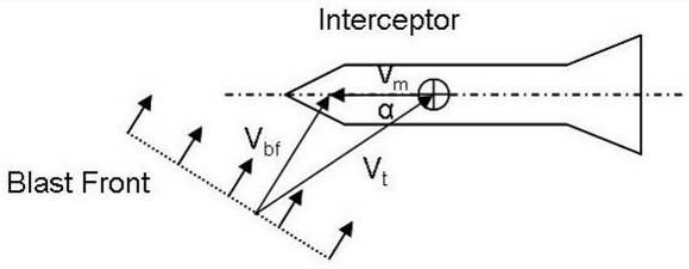

Figure 2.13 Vectors showing angle of incidence between blast front and interceptor axis.

The angle of radiation incidence is calculated using the vector of the interceptor heading V_m as well as the vector from the location of the nuclear event to the interceptor, V_{bf}. The resultant vector V_t is important to calculate because, at different angle of intercepts, a component inside the interceptor may be inherently shielded by the objects (batteries, casing, etc.) surrounding it. This shielding must be taken into account, and is represented as different entries in a dose lookup table with each entry corresponding to a particular angle of incidence.

2.2.4 Atmospheric Attenuation of Particles

To correctly determine the flux at a point away from a burst location, the radiation emitted from a burst event must be attenuated through the atmosphere encountered between the two locations. This allows the present work to calculate a more correct dose than in an unattenuated medium. An exponentially attenuated flux is obtained by first calculating the optical thickness of the air encountered between a radiation source and an interceptor, and then using this optical thickness to correctly scale the unattenuated flux.

The two types of particles for which attenuation effects are calculated are photons and neutrons. While electrons are also attenuated by the atmosphere, much more complex physics are involved when the earth's magnetic field is taken into account. Consequently, these attenuation effects for free electrons are currently neglected in this work. Neutron albedo and bounceback due to the earth's atmosphere are also neglected at this time. Neutron interaction with an attenuating media causes scattering and reflection effects, which reduce the linearity of the line-of-sight calculations due to changes within the energy spectrum. This effect is also neglected at this time.

2.2.4.1 Line of Sight Calculations

Before the attenuated flux at an interceptor location can be determined, there must be a line of sight (LOS) between the interceptor and the source of radiation. This means that the interceptor can "see" the burst event directly without the earth blocking its view. Figure 2.14 shows an example of *Interceptor A* having a direct LOS to a given burst event, while *Interceptor B* does not.

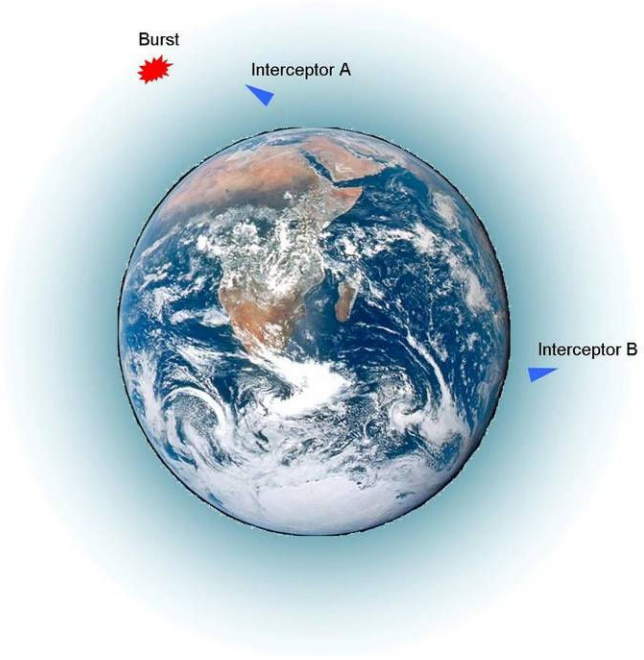

Figure 2.14 Earth showing Interceptor A with a Line of Sight, and interceptor B without a Line of Sight.

Whether or not an interceptor has a direct LOS to a given burst is calculated using the Earth Centered Earth Fixed (ECEF) coordinates of each event and treating each event as a point. Figure 2.15 shows the ECEF coordinate system in relation to the earth. The point (0,0,0) represents the mass center of the earth. The *x*-axis intersects the earth at 0° longitude and 0° latitude. The *z*-axis lies along the earth's rotational axis, pointing north. Using this coordinate system, any point is fixed in relation to the earth.

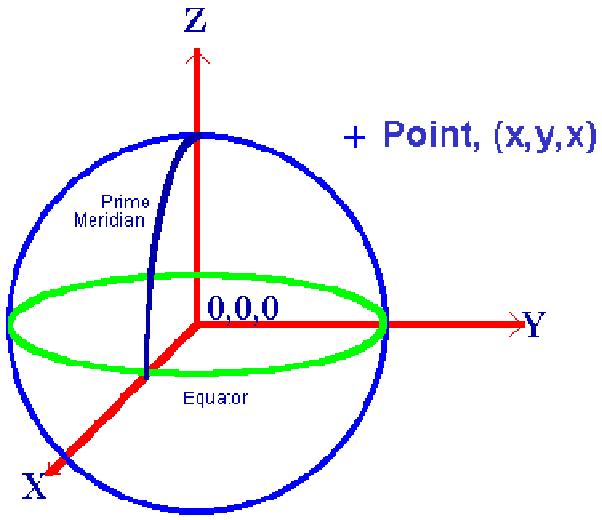

Figure 2.15 Diagram showing ECEF coordinate system in relation to earth. The x-axis contains the point 0° longitude and 0° latitude and the z-axis lies along the rotational axis pointing north [Colorado.edu].

By creating a line linking the points representing an interceptor and an event, the minimum distance along this line to the ECEF origin at the center of the earth is calculated. If this minimum distance is greater than the radius of the earth plus some minimum altitude above it, then the objects are able to "see" each other. Otherwise the earth is blocking their view and any potential particles are not able to transport directly between those two points.

Figure 2.16 shows the three possible scenarios when calculating LOS.

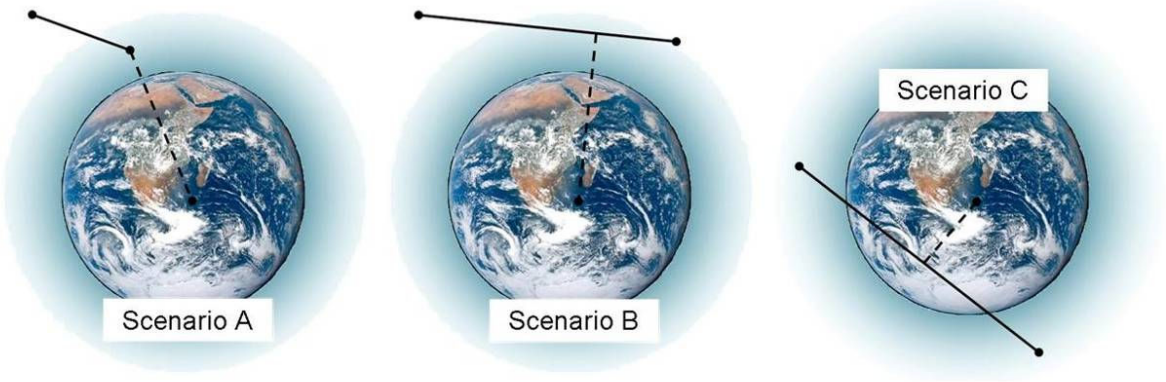

Figure 2.16 Three possible LOS scenarios. Scenario A has a LOS because both points are on the same side of the Earth. Scenario B has a LOS because a line can be drawn between the points even though they are on opposite sides of the earth. Scenario B is an example of a non-LOS situation because the line passing between the two points intersects the Earth.

The three scenarios shown in the above figure represent events on the same side of the earth, different sides of the earth but still maintaining a LOS, and different sides of the earth and not maintaining a LOS. The dashed line represents R_{min}, the shortest distance between any point on the LOS line and the origin. In Scenario A, this is merely the altitude of the lower point.

To find the lowest altitude along the LOS line for Scenarios B and C, the distance D between the two points is calculated using Eq. 2.4. The coordinates R_x, R_y, and R_z correspond to the *(x,y,z)* ECEF coordinates of each point.

$$D = \sqrt{(R_{1x} - R_{2x})^2 + (R_{1y} - R_{2y})^2 + (R_{1z} - R_{2z})^2} \quad . \tag{2.4}$$

The distance D, along with the altitude from origins R_1 and R_2 for each point give the three sides of a triangle as shown in Figure 2.17.

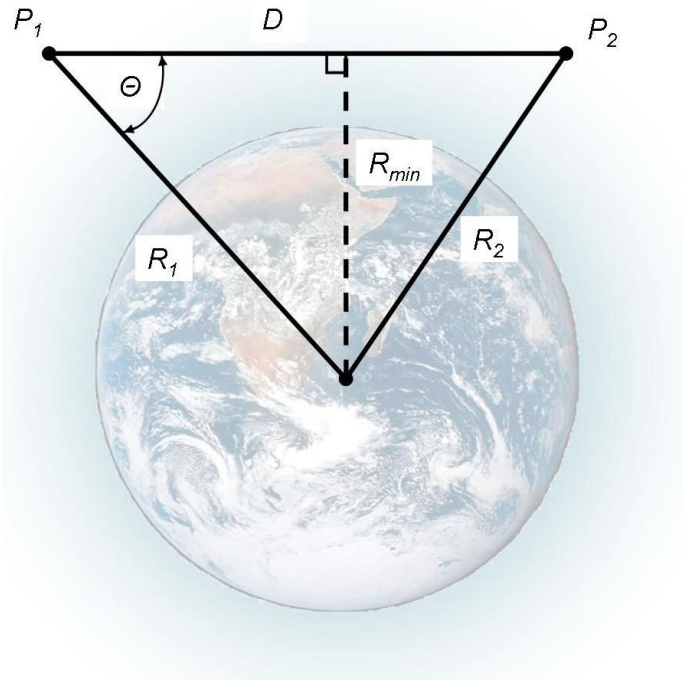

Figure 2.17 Triangle diagram used for calculating Line of Sight.

The interior angle Θ is calculated using Eq. 2.5 which is derived from the law of cosines.

$$\Theta = \cos^{-1}\left(\frac{R_1^2 + D^2 - R_2^2}{2R_1 D}\right). \tag{2.5}$$

The minimum distance from a point along the LOS line to the ECEF origin, R_{min}, is then computed using the length of R_1, shown in Eq. 2.6.

$$R_{\min} = R_1 \cos(90 - \Theta). \tag{2.6}$$

This minimum distance can then be compared to the radius of the earth plus some minimum altitude to account for strongly attenuating air if so desired. If R_{min} is greater than the minimum altitude, then the two points have a LOS, as for Scenarios B. Otherwise as represented by Scenario C, the earth is between the two points and thus would absorb any radiation travelling from one point to the other and the incident flux is zero.

2.2.4.2 Method for Flux Attenuation

The uncollided or unattenuated dose at the interceptor coordinates as a result of a nuclear burst is given in Eq. 2.7 and is represented by D^0. This dose is a result of a flux spectrum and their subsequent reaction rates integrated over all energies.

$$D^0 = \int_0^\infty R(E)\varphi^0(E)BdE. \tag{2.7}$$

Substituting the unattenuated flux φ^0 given in Eq 2.1 gives Eq. 2.8 where the optical thickness of a material as a function of energy, $\ell(E)$, is used to represent the attenuating properties of a medium. From this point forward, a buildup factor, B, is not included in this equation because the altitudes that this work encompasses are above 100 km, and there is not expected to be significant dose buildup due to low air densities at these conditions. However, buildup is expected to be a concern between the altitudes of 20 and 80 km, and must be taken into account if this simulation is to cover lower altitudes.

$$D^0 = \int_0^\infty R(E)\frac{S_p(E)}{4\pi\left|\underline{P_1} - \underline{P_2}\right|^2}e^{-\ell(E)}dE. \tag{2.8}$$

The optical thickness of the air between points P_1 and P_2 designed ℓ is graphically depicted in Figure 2.18, as well as the differential path line segment designated *dl*.

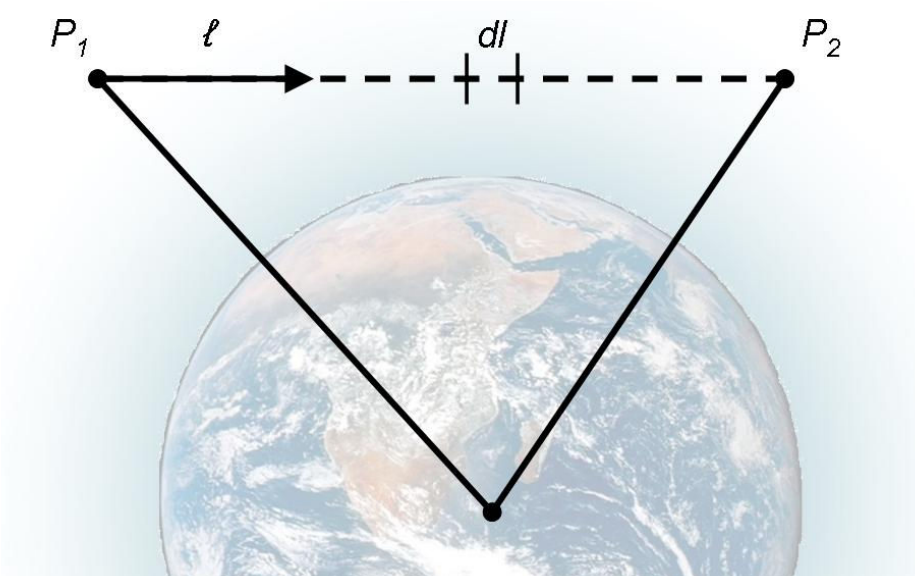

Figure 2.18 Path line between points showing optical thickness and differential line segment.

The optical thickness of the path line between the two points is determined by the number of mean free paths of attenuating material, and is given in Eq. 2.9 where μ is the attenuation coefficient for each line segment as a function of energy [Shultis and Faw, 2000].

$$\ell(E) = \int_{P_1}^{P_2} \mu(l, E)dl \,. \tag{2.9}$$

The density of the air is not assumed to be constant over the length of the path line. The mass attenuation coefficient, μ/ρ, and the density of the air, ρ, over each discrete line segment are used to correct for this as follows:

$$\ell(E) = \left(\frac{\mu}{\rho}(E)\right)\int_{P_1}^{P_2} \rho(l)dl \,. \tag{2.10}$$

The computer simulation encompassed by this work cannot evaluate this integration directly, but can approximate it as a numerical integration. The air mass density over each line

segment, *dl*, is summed over the path line to represent an approximation of the total air density between points P_1 and P_2 as shown in Eq. 2.11.

$$\ell(E) \cong \left(\frac{\mu}{\rho}(E)\right)\sum_{i=1}^{N}\rho(l_i)\Delta l_i \ . \tag{2.11}$$

This provides the optical thickness to be used in Eq. 2.8 allowing this work to calculate the effects of air attenuation on a flux spectrum. How the length of each line segment and its associated density is calculated is presented in the following sections.

It is important to note that these mass attenuation coefficients are a function of energy, with low energy particles attenuating more than highly energetic particles [Shultis and Faw, 2000]. This is one of the reasons that the radiation spectra emitted from a burst event is tracked as a function of energy by this computer simulation and is not normalized. The mass attenuation coefficients for photons used in the model were obtained from Shultis and Faw. The same coefficients for neutrons were obtained using MCNP.

2.2.4.3 Calculating Air Density at a Point

The air density of each line segment along a path line is assumed to be identical to the density of the first point on each line segment. A lookup table of sample densities at varying heights was used to create a trend line allowing the density at any point to be calculated up to 1000 km. This data was calculated using the publicly available Mass Spectrometer Incoherent Scatter Model (MSIS) developed for higher altitude density calculations [Hedin, 1987]. This model is maintained by the Naval Research Lab and used by the U.S. Air Force to determine satellite drag [Drob and Picone, 2000, Marcos et al, 2006]. Other models such as the Jacchia-Bowman 2006 model do exist, but it has been determined that discrepancies averaged over orbits in the models are very small, and "there is no single model which stands out as demonstrably superior over any other" [Akins et al, 2003].

The data presented in Table 2.1 simulates a density averaged over 24 hours on August 6, 1965, but is available for other days as well from NASA's Modelweb [Cross, 2007, Akins et al, 2003]. A trend line was created from this data, and it was determined that two distinct regions

existed based on altitude, with the cutoff being roughly 100 km. Figures 2.19 and 2.20 show the plotted data points as well as the trend line fit.

Table 2.1 Air density as a function of altitude. Data simulating a 24 hour average air density average for August 6, 1965. Taken from the MSIS model as run by Cross [Cross, 2007].

Altitude (km)	Average Density (g*cm^-3)
0	1.19E-03
5	7.18E-04
10	4.17E-04
30	1.93E-05
50	1.18E-06
80	1.92E-08
100	4.50E-10
200	1.77E-13
300	7.73E-15
400	7.10E-16
500	8.98E-17
600	1.43E-17
700	3.31E-18
800	1.28E-18
900	7.25E-19
1000	4.87E-19

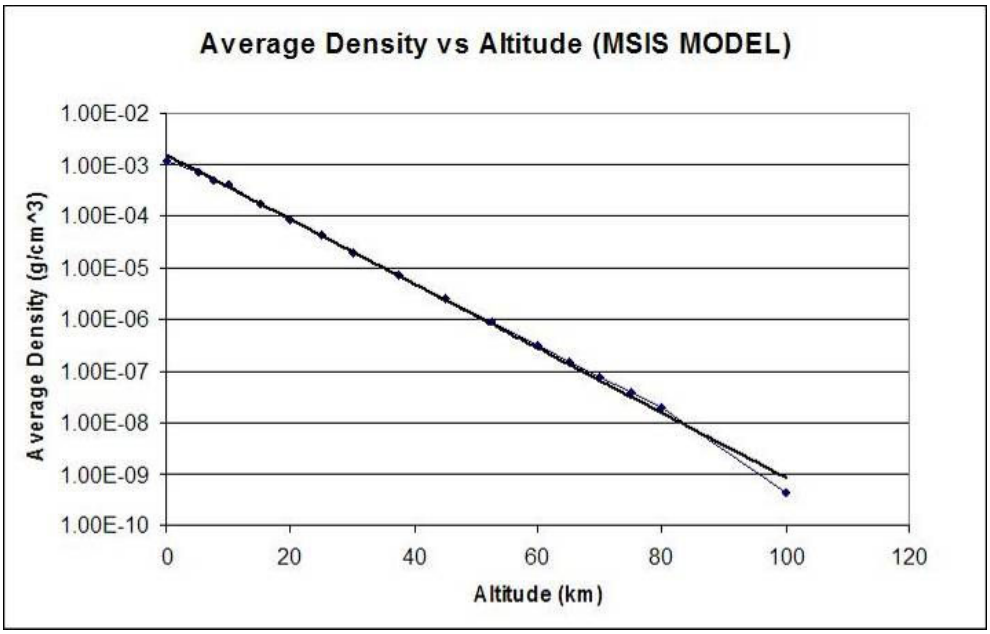

Figure 2.19 Average Density vs. Altitude below 100 km for the data presented in Table 2.1.

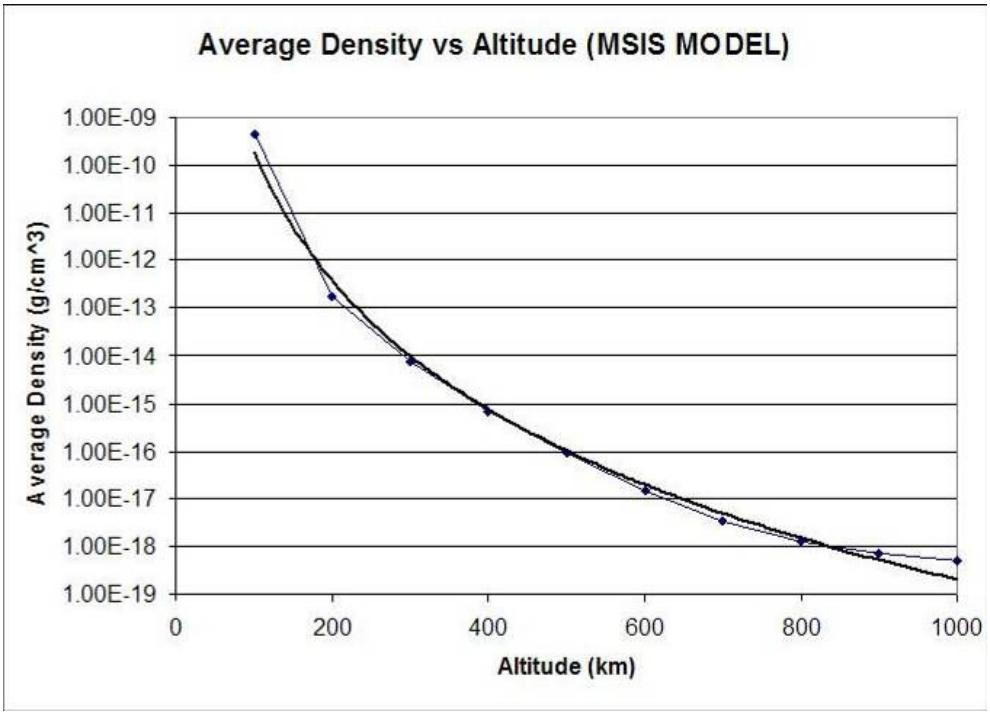

Figure 2.20 Average Density vs. Altitude above 100 km for the data presented in Table 2.1.

The curve of best fit for the data is presented as an exponential in Eq. 2.12 for altitudes below 100 km, and as a power law for altitudes greater than or equal to 100 km. The altitude

variable H should be expressed in km. The coefficients of determination for each equation are 0.9997 and 0.9934 respectively. Eq. 2.12 can be extrapolated up to altitudes of 2000 km. This is allowed because of the minute amounts of air above 1000 km, which is assumed to not have a significant part in the overall density calculation.

$$\rho(H) = \begin{cases} 0.00138e^{-0.14H} & \text{H} < 100\text{km} \\ 1.71\text{E}8 * \text{H}^{-8.98} & \text{H} > 100\text{km} \end{cases}. \tag{2.12}$$

2.2.4.4 Integrating Air Density along a Line of Sight

After it has been determined that two points have a direct line of sight, then the total density of the air along the LOS path must be calculated using the approximation given in Eq. 2.11. For Scenario A mentioned previously, this is done by designating the points with the higher altitude from origin P_1, and the lower P_2. Starting at the higher altitude, the density is integrated numerically along the path line, with a higher fidelity model at the lower altitudes as shown in Figure 2.21. Ideally this process would involve a large number of points to create a high fidelity numerical integration, but due to the processing power and computational time this would require, a small number of points are chosen to maximize their added value to the overall density calculation.

The distinct points at which to find a density along the LOS path line are calculated using a logarithmic function. This allows a higher point concentration at lower altitudes where the air has a higher density and thus plays a more significant role in the overall integration.

The first step is to find the vector, $\underline{\boldsymbol{V}}$, with endpoints P_1 and P_2 and pointing in the direction of P_2, as shown compactly in Eq. 2.13. The *(x,y,z)* Cartesian components of this vector are represented as V_x, V_y, V_z.

$$\underline{V} = \underline{P_2} - \underline{P_1}. \tag{2.13}$$

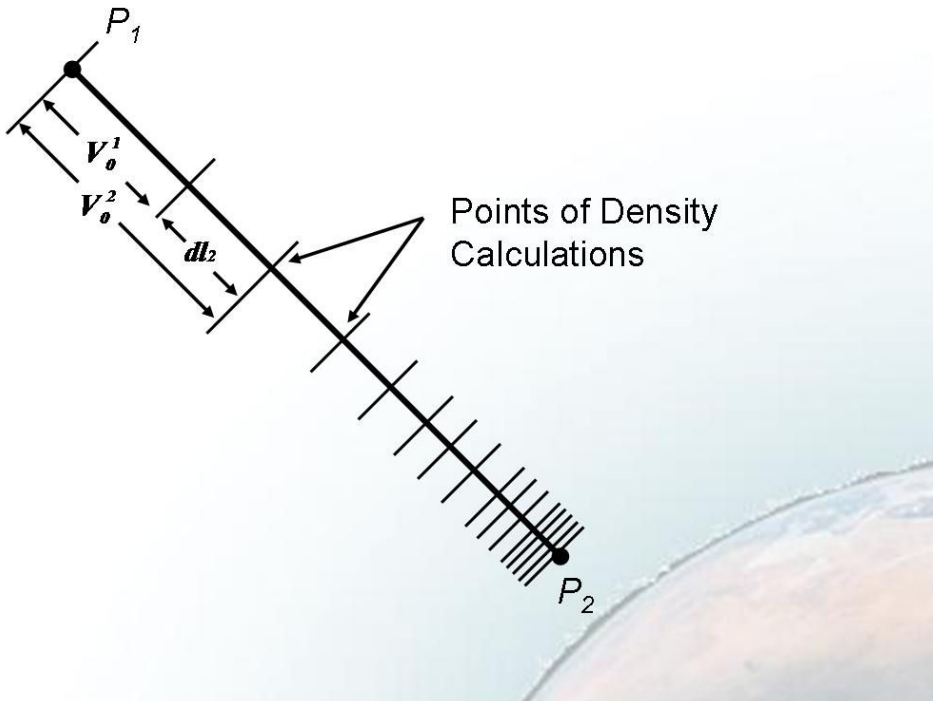

Figure 2.21 Points of density calculations along LOS for Scenario A. This diagram also shows how each line differential line segment, dl_n, is calculated using other line segments.

To determine the length of each line segment used in the numerical integrations, a formula is used that is based on the percentage along the path line, designated by V_0^n. This is shown in Eq. 2.14 where N is the number of points along the path line for which to calculate density. Using this system and referring back to Figure 2.21, V_0^1 represents the length of the first line 3segment and V_0^N represents the length of the entire line having endpoints P_1 and P_2. A higher number of points will increase model fidelity and thus yield a more correct result, but will take more processing time.

$$V_0^n = 100^{n/N} \qquad n = 1,2,...N\,. \tag{2.14}$$

The differential length along the path line for each line segment, dl_n, is found using Eq. 2.15 where n is the current position along the path line and D is the total distance between the two points.

$$dl_n = D*(V_0^n - V_0^{n-1}) \qquad n = 1, 2, \ldots N \,. \tag{2.15}$$

Each vector component along the line of sight is updated to represent the new point along the path line, and a density is calculated at each point and added to the running total density. Eq. 2.16 shows the how the new point is calculated.

$$\begin{aligned} P_{x,n} &= P_{1x} - \frac{V_x}{V_0^n} \qquad n = 1,2,\ldots N \\ P_{y,n} &= P_{1y} - \frac{V_y}{V_0^n} \qquad n = 1,2,\ldots N \,. \\ P_{z,n} &= P_{1z} - \frac{V_z}{V_0^n} \qquad n = 1,2,\ldots N \end{aligned} \tag{2.16}$$

The altitude above sea level at each point, H_i, is calculated using Eq. 2.17, where R_e is the radius of the earth in meters.

$$H_i = \sqrt{P_{x,i}^2 + P_{y,i}^2 + P_{z,i}^2} - R_e \tag{2.17}$$

This altitude can then be used to determine the atmospheric density, $\rho(H_i)$, using Eq. 2.12. When $\rho(H_i)$ is substituted for $\rho(l_i)$ in Eq. 2.11, along with the length of each differential line segment, dl_i, substituted for Δl_i, this yields the optical thickness for the amount of air encountered over each differential line segment and thus the total optical thickness for the entire path line when each segment is summed to create the whole.

This process for using the optical thickness of the air between two points to determine the amount of atmospheric attenuation yields best results for Scenario A referred to in Figure 2.16, yielding the density profile shown in Figure 2.22.

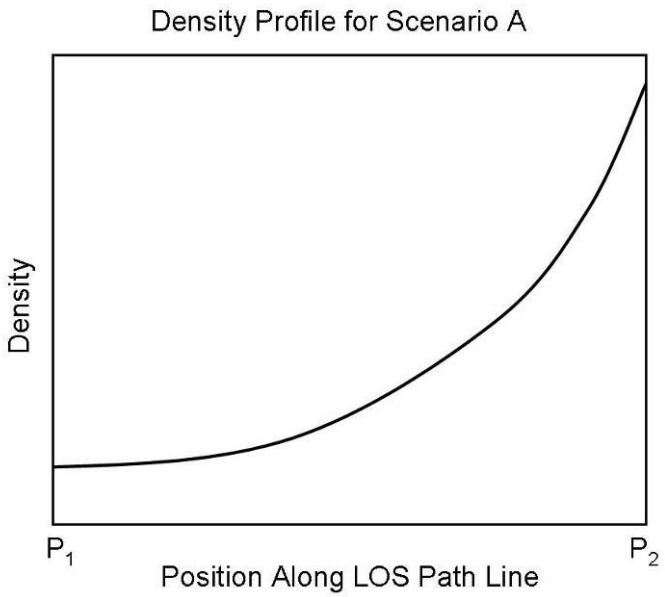

Figure 2.22 Potential density profile as a function of position along LOS path line for Scenario A.

For Scenario B in Figure 2.16 the lower of the two points, P2, is not the actual lowest point on the path line connecting P1 and P2. This results in the density profile presented in Figure 2.23.

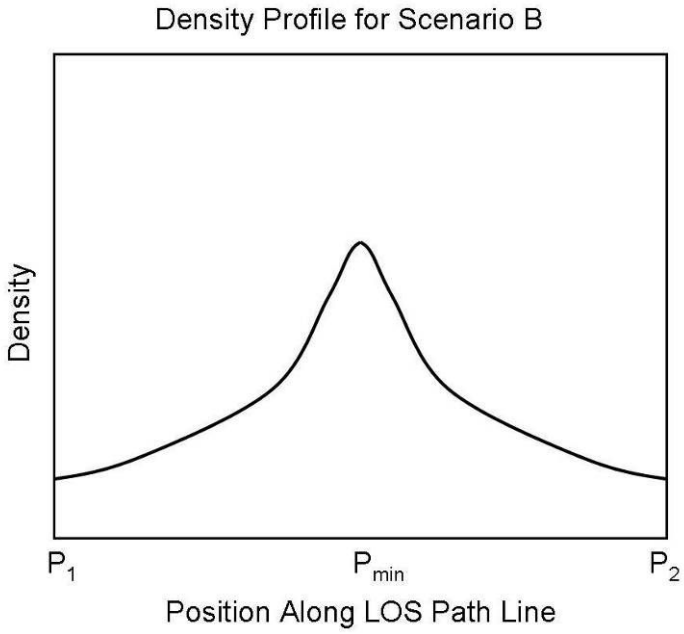

Figure 2.23 Potential density profile as a function of position along LOS path line for Scenario B

Because of this, a different approach is used to accurately model the total air density. The density is calculated treating each endpoint of the LOS path line as the higher altitude point, and calculating the density for each working towards P_{min} as shown in Figure 2.24. The same process used for Scenario A is applied twice in this instance, and the results are summed to obtain the total density of air encountered.

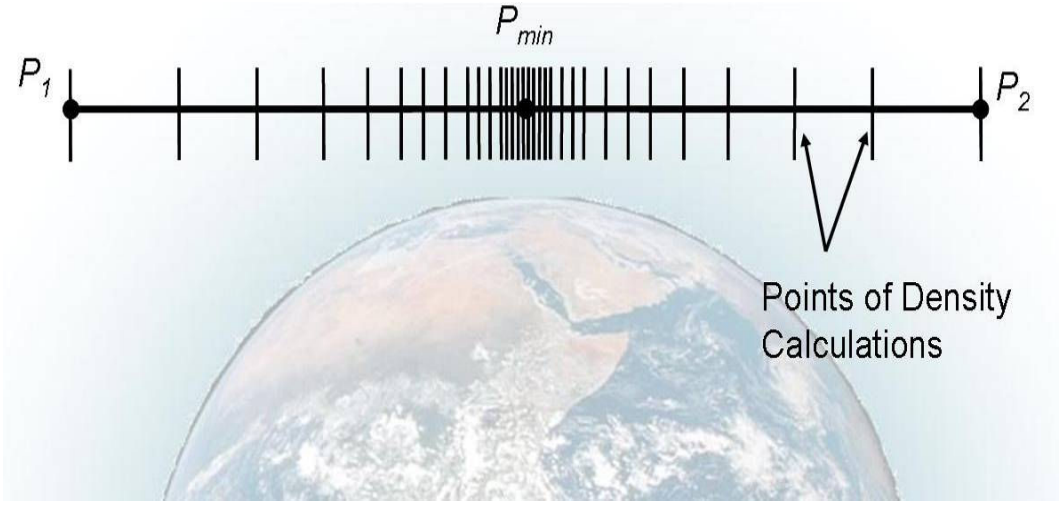

Figure 2.24 Points of density calculations along LOS for Scenario B. Note that two separate numerical integrations take place, starting at P_1 and P_2 and heading in the direction of P_{min}.

2.2.4.5 Limitations to the Atmospheric Models

This method to calculate an attenuation scalar for an unattenuated flux in this simulation does not take atmospheric heave, atmospheric bounce back, secondary radiation, or buildup into account. Atmospheric heave occurs when a nuclear device is detonated in the atmosphere; ionizing and heating the air around it [Rabinowitz et al, 1992]. This heated air now has buoyant forces acting on it, rises, and changes the density composition for a large area. Figure 2.25 shows a sample atmospheric heave effect in the atmosphere.

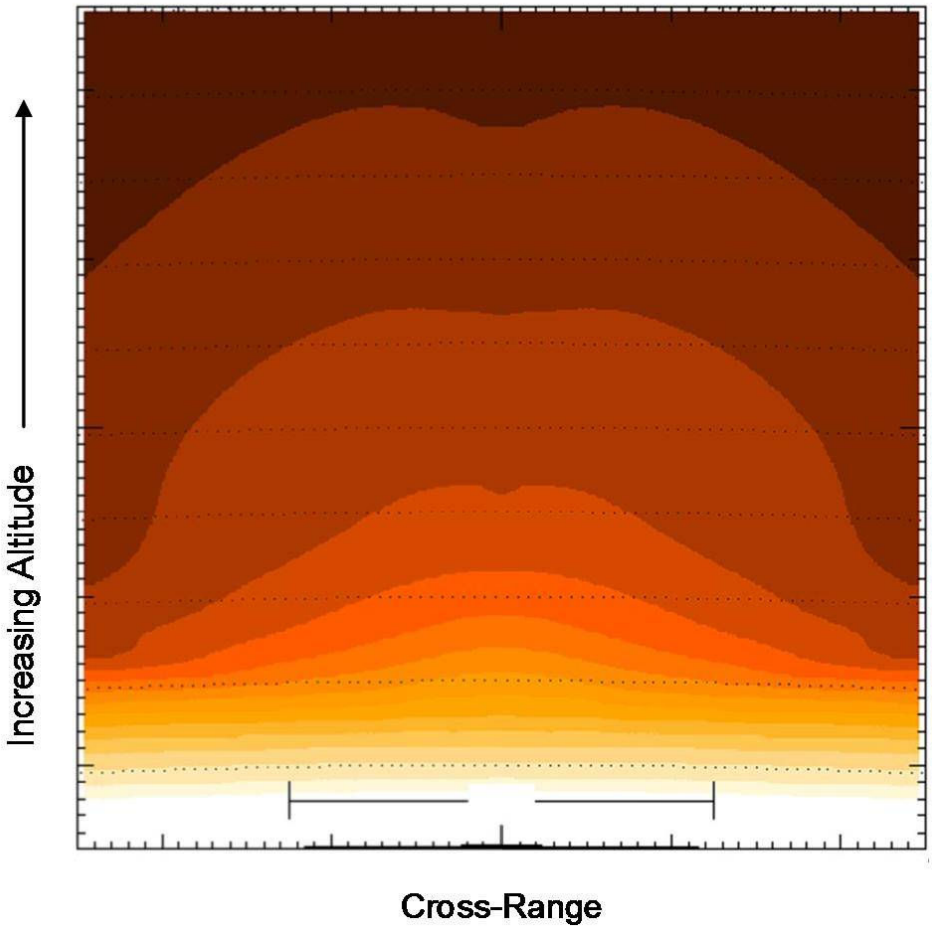

Figure 2.25 Atmospheric density after a low altitude nuclear burst showing atmospheric heave. Air mass densities at higher altitudes are orders of magnitude larger than unaffected air at the same altitude.

This burst effect on atmospheric densities is not included in the model, as the computational requirements are prohibitively large. Another benefit to not including atmospheric heave is that without it the calculated atmospheric densities would actually be lower in the immediate vicinity of any subsequent bursts, which would result in a larger attenuated flux at an interceptor and thus a more conservative estimate of the total doses received.

The primary buildup of radiation particles in the atmosphere that this model should include are the X rays and electrons that create a debris ionization region roughly between 20 and 80 km altitude [Byrd, 1995]. Since the air is much thinner above 80 km compared to lower altitudes, particles do not significantly interact with the air. At lower altitudes, the air is thick enough to have a significant number of particle interactions which leads to a highly ionized region [Dolan and Glasstone, 1977]. MCNP simulations using representative air densities above

100 km altitude show a change between the attenuated and unattenuated flux of less than 5%. This shows that neglecting buildup is a reasonable assumption for this work.

Because electrons travel along the geomagnetic field line to their conjugate point, there are dual beta patches associated with a burst. Witnesses in Samoa to the TEAK and ORANGE high altitude burst events, some 2000 miles to the south of the burst locations of Johnston Island and lacking a direct line of sight, reported seeing the beta patch's associated aurora fractions of seconds after the burst [Dolan and Glasstone, 1977]. Figure 2.26 shows the conjugate beta patch associated with a burst as well as the X-ray patch and magnetic field lines. Because the primary buildup locations for all radiation species in the atmosphere are below the simulation's 100 km operating requirement, they are ignored in the construction of this model.

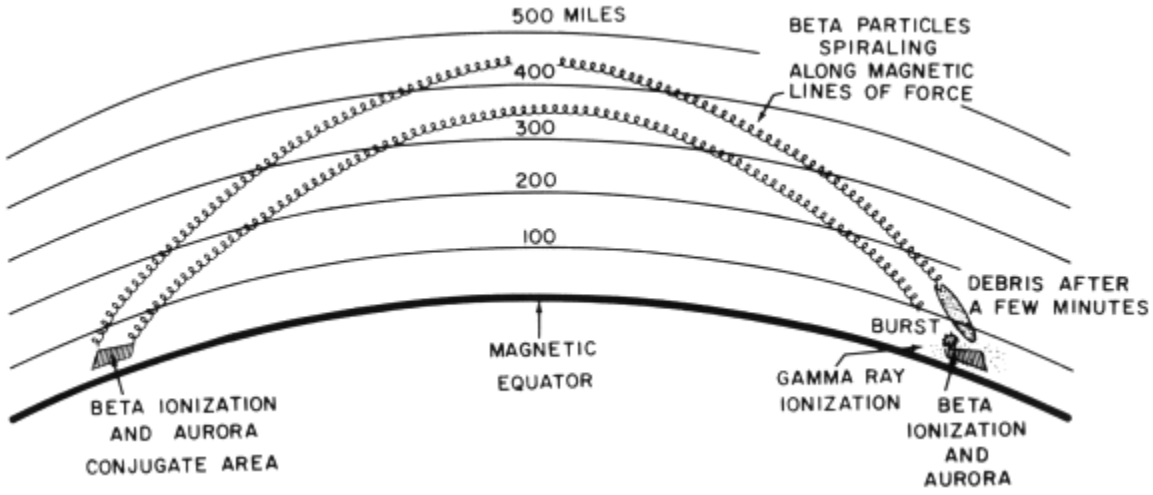

Figure 2.26 Burst scenario showing dual beta patches near the conjugate and mirror points for trapped particles [Dolan and Glasstone, 1977].

The intense radiation field created by a burst leads to secondary particle generation in the atmosphere. To accurately simulate secondary radiation, MCNP or similar software would be required. This would require large computational resources and time, and thus is not suitable for a fast running dynamic scenario. This phenomenon is still included in the model however, as a parameter for the temporal spectra. The temporal spectra model is designed to include debris radiation, and thus the debris parameter could be scaled accordingly by radiation species to include secondary radiation.

2.2.4.6 Comments on the Atmospheric Attenuation Model

In order to verify the atmospheric attenuation effects calculated by the present work, MCNP was used to create points of comparison. An MCNP input file was created that calculated atmospheric attenuation of photons as well as neutrons. Spheres representing altitude layers in the atmosphere having the same size and air densities as presented in Table 2.1 were created in an attempt to accurately and consistently model the variations in atmospheric densities due to altitude. This MCNP model will actually cause more attenuation than the calculations within the present work, because of the uniformity of the density of each atmospheric layer instead of interpolating by altitude.

A monodirectional source with an energy spectrum was placed at 2000 km, emitting radiation directly at the origin. A fluence tally detector was placed directly below the source, at altitudes ranging from 30 km to 200 km. Placing the source at the upper altitude threshold of 2000 km and detectors near the lower altitude limits of this simulation should model the maximum possible amount of air encountered by radiation between two points with the same latitude and longitude. Atmospheric bounceback was not modeled. A sample input file is presented in Appendix A.

The results of these MCNP simulations are presented in Table 2.2 and prove correct the assumptions made by this work that significant atmospheric attenuation does not exist above 100 km altitude. Significant attenuation was not present until the detector was placed at an altitude of 50 km, well below the simulation threshold.

Table 2.2 Attenuated flux as a function of detector altitudes from MCNP study.

Detector Altitude	Original Flux	Flux Attenuated
200	100.00%	0.00%
100	99.99%	0.01%
80	99.81%	0.19%
50	89.86%	10.14%
30	0.00%	100.00%

It is possible that radiation modeled by the present work could encounter larger amounts of air if the source and the detector were at different geographic locations and altitudes. As long as the minimum point on the line between the source and the detector were at or above 100 km, this is not expected to become a factor due to extremely low atmospheric densities.

Attenuation results calculated by the present work under similar conditions match the MCNP study within 0.5% for altitudes above 80 km. In light of this fact, and the fact that there is no significant attenuation above 80 km, the atmospheric attenuation module of the simulation could be neglected. However, if the altitude threshold for this simulation were to be lowered below the current threshold, atmospheric attenuation may need to be accounted for.

2.2.5 Effects of Nuclear Bursts on Radiation Belts and Geomagnetic Fields

One of the major effects of a nuclear burst at high altitudes is the effect it has on the earth's local magnetic field as well as the injection of charged particles into the trapped radiation belts. These effects can last momentarily as in the case of a magnetic bubble, to years after a burst event for trapped radiation.

The United States has detonated 10 high altitude bursts, as part of the HARDTACK, ARGUS, and FISHBOWL test series [United States Nuclear Tests, 2000]. The various high altitude test series started out as a way of testing the feasibility of using nuclear weapons as part of the intercontinental ballistic defense system, and later expanded the tests to focus on high altitude effects [Hess, 1964]. These tests were conducted at night above Johnston Island, in the South Pacific [United States Nuclear Tests, 2000]. Of primary importance to this model, are the results from the Starfish Prime burst.

2.2.5.1 The Starfish Prime Event

The Starfish Prime event as part of the FISHBOWL series of tests was primarily designed to test the effects on the geomagnetic field. Secondary goals were testing effects on electronics and long range communication. A 1.8 MT device was detonated at a 400 km altitude over Johnston Island on July 6 1962. Within milliseconds after the burst, visible electron striations following the earth's magnetic field lines were visible at both the northern (400 km distance) and southern (2000 km distance) geomagnetic conjugates [Narin and Dumas, 1962]. Figure 2.27 demonstrates why the high altitude tests subjects were often referred to as "Rainbow Bombs", due to their accompanied plasma striations as seen from the ground. Figure 2.28 shows the striations as seen from an aircraft 400 km away, 3 minutes after burst. These visible effects lasted for several minutes.

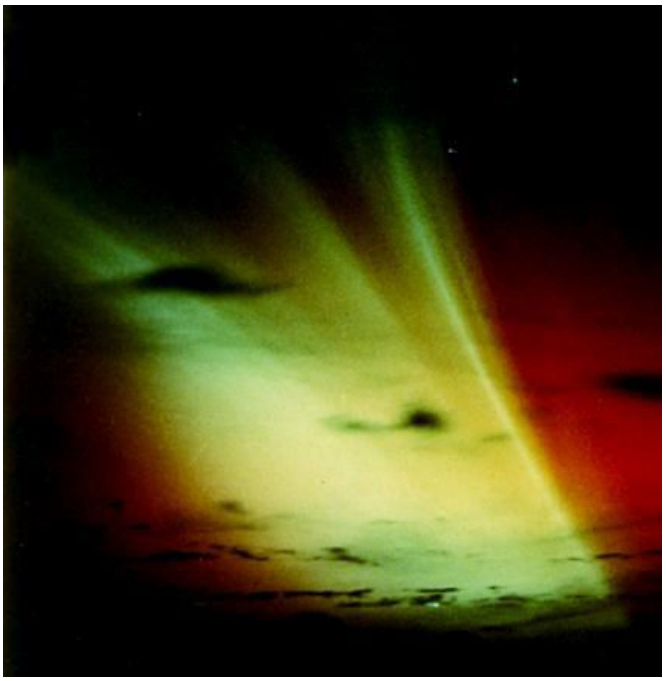

Figure 2.27 Starfish Prime plasma striations seen from ground, 60 seconds after burst event. The plasma striations follow the magnetic field lines of the Earth [Wikipedia, Starfish-Prime_nuclear_test_from_ground.jpg].

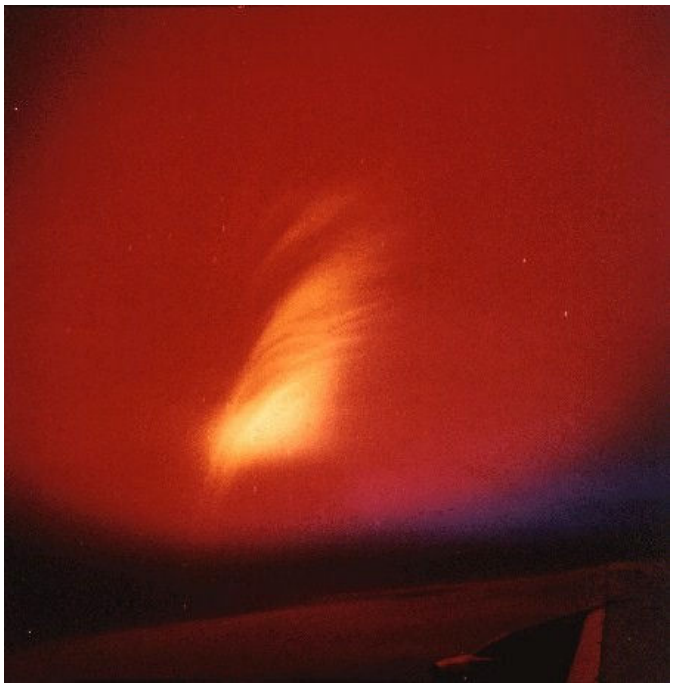

Figure 2.28 Starfish Prime plasma striations seen from airplane, 3 minutes after burst event [Wikipedia, Operation_Dominic_Starfish-Prime_nuclear_test_from_plane.jpg].

Although nothing but the burst flash was visible from Johnston Island due to cloud cover, the fireball was visible in Honolulu, some 1400 km away [Narin and Dumas, 1962]. Figure 2.29 shows the burst from a Honolulu rooftop. The islands of Hawaii also experienced a disruption of radio communication, brief power outages, and disruptions on other electronics lasting three hours [Longmire, 1985]. The significance of the Starfish burst and the data generated from it cannot be underestimated, as it is the basis for most of the high altitude burst models and theories that are currently employed.

Figure 2.29 Starfish Prime as seen from Honolulu rooftop seconds after burst [Wikipedia, Starfish5.JPG].

2.2.5.2 The Magnetic Bubble

A magnetic bubble is created by a high altitude burst event. Debris particles traveling at sub-Alfvén velocities push the geomagnetic field lines out of their original locations, creating a "bubble" void of natural field lines [Pavel et al, 1977]. The size of the magnetic bubble formed from expanding debris is a function of total energy emitted from a nuclear burst, with higher yield devices creating a larger void in the magnetic field [Hess, 1964].

A large number of electrons are confined and amplified within this region, with beta levels typically exceeding that of trapped beta radiation. The total ionizing dose and the effects caused by the beta radiation in this region can be devastating to a sensor or vehicle. Starfish Prime measurements showed the bubble lasted "a minimum of 7 seconds", with models predicting up to 10 second life spans [Dyal, 2006]. Due to the complexity of this model and lack of publicly available data, this is not incorporated into the simulation at this time. For vehicles within the magnetic bubble, this lack of simulation will result in an underestimate of total ionizing dose.

2.2.5.3 Pumped Radiation Belts

Synchrotron radiation is given off when electrons are accelerated in a circle, such as the gyrations within the beta tubes that follow the geomagnetic field lines. Measuring stations in Peru observed increases in synchrotron radiation indicating the presence of artificially created radiation belts due to the Starfish Prime event at +6 and +35 minutes. Further measurements were stable due to diffusion of charged particles within the radiation belts [Hess, 1964]. This fact indicates that the Van Allen belts for electrons move eastward, and it takes roughly 29 minutes to completely circle the earth. Figure 2.30 shows a typical simulation highlighting the drift of the trapped radiation belts and diffusion within the belts by depicting the trapped electron density at 1300 km altitude.

This artificially created radiation belt could last from months to years. Measurements taken after the Starfish event indicated that its effects were noticed for four years [Sawyer and Vette, 1976]. The measured electron decay constant for this event was on the order of 115 days for low altitude belts [Hess, 1964]. This increase in encountered radiation significantly shortened the life spans of unshielded objects in space.

Starfish caused many satellites to have disrupted operation and eventually rendered 7 satellites in low earth orbit non-functional within weeks to months after the event due to increased levels of persistent radiation trapped within the Van Allen belts [Hess, 1964, Dupont, 2004]. These 7 satellites represented approximately one third of the satellites orbiting the Earth at the time [Dupont, 2004]. A Defense Threat Reduction Agency (DTRA) analysis estimated that many if not all currently operational satellites in LEO would be damaged or would fail catastrophically due to persistent radiation from an exo-atmospheric burst [Dolan and Glasstone,

1977, Cross, 2007]. This estimate was not based on analysis, but merely a blanket estimate giving possible and worst case scenarios.

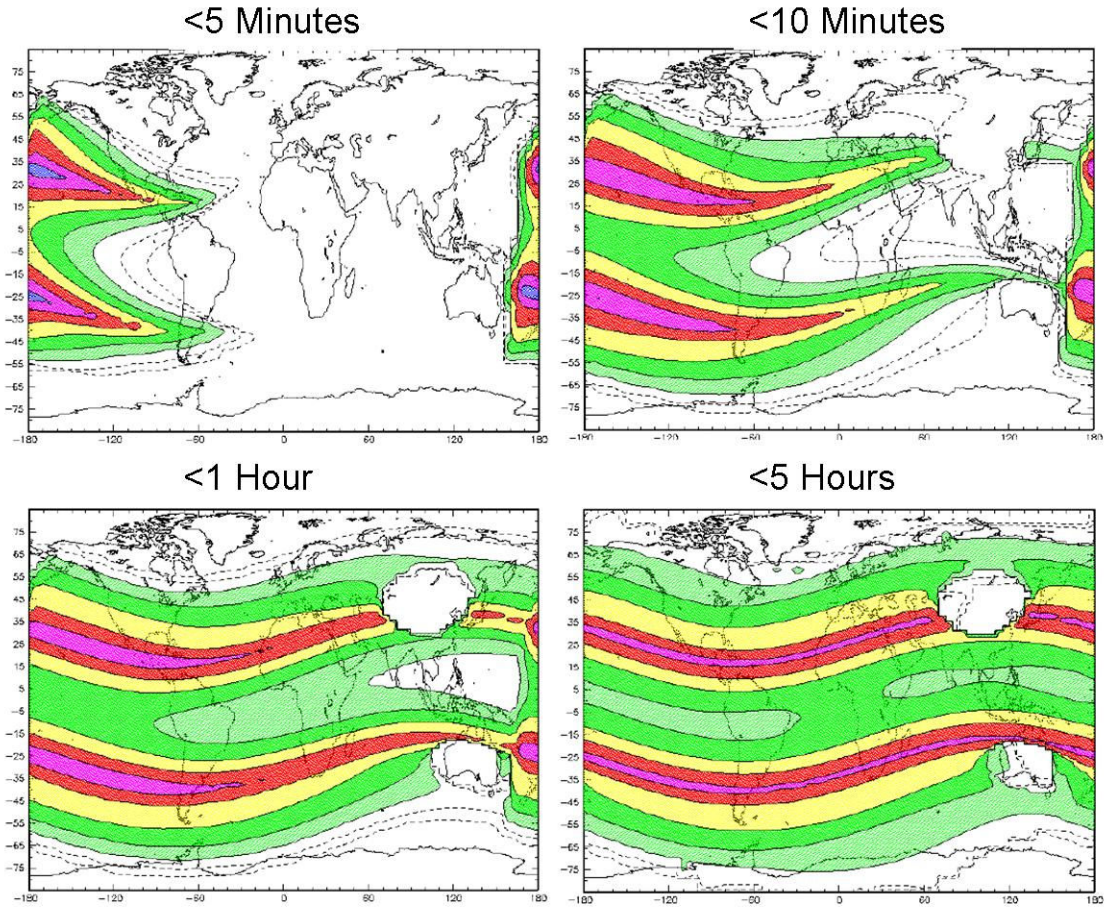

Figure 2.30 Trapped electron radiation at 1300 km altitude after a mid-latitude burst. Sequential pictures show formation of the pumped radiation belts and diffusion of electrons within the belts.

Initial Starfish electron decay within the low altitude belts is fairly slow when compared with higher altitudes. Below altitudes of roughly 1.7 earth radii, the decay constant has been measured as months [McIlwain, 1963]. Electrons with lower energies scatter first, leaving a hardened spectra with a peak around 2 MeV [Hess, 1964]. Because an altitude greater than 1.7 earth radii is beyond the altitude scope of this model, it is not further explored.

Starfish proton belt pumping led to a sevenfold increase in 55 MeV trapped proton flux measurements at 350 ± 50 km [Sawyer and Vette, 1976]. The observed flux decayed to a constant within one year, as the solar cycle increase offset the Starfish decay. Starfish-induced

protons contributed significantly to the trapped proton levels for four years after the burst [Sawyer and Vette, 1976].

2.2.5.4 A Model for Radiation Belt Pumping

There is currently no publicly available analytical model to simulate a persistent environment after a high altitude nuclear detonation. Lt. Col. Christopher Cross of the Naval Postgraduate School is developing a model that is designed to allow the DTRA as well as others to simulate the burst injection of charged particles into the earth's geomagnetic field and its persistence and decay. The model would then determine the effects on satellites due to the artificially increased levels of radiation [Dolan and Glasstone, 1977].

Dr. Cross' high fidelity model takes into account the complex physics involved with the trapping of charged particles leading to artificial radiation belts as well as their decay over a period of months and years. Particle infusion, drift around the magnetic axis due to the geomagnetic gradient, and ionization energy loss are the basis' for the model. This model is currently under development, and is expected to be available with limited distribution in late 2009 [Cross, email with the author, 2008]. Once available, it is expected to be inserted into the simulation in order to calculate the magnetic bubble and radiation belt pumping that could not be computed otherwise.

2.3 Simulation Modeling Assumptions

The program described by this paper assumes that all burst and interceptor instances are above 100km in altitude. Given the amount of atmosphere at this altitude and the fact that a typical interceptor flight is not expected to last an extended amount of time, has led to these additional assumptions:

- These calculations do not take the rotation of the earth into consideration.
- Secondary effects such as attenuation from the atmosphere, neutron bounceback, etc. are not computed at this time. To accurately model these phenomena, the computational resources required would render the simulation too slow to meet the design requirements.
- No electron generation or associated phenomena are computed at this time.
- A given detonation will eventually have a "time of no importance" after the event has occurred based on temporal data contained in input files.

- Thermal and shock characteristics of a detonation are neglected as there is a negligible amount of atmosphere at the altitudes the program is expecting to encounter.
- The orientation of an interceptor is the same as its velocity. This simplifies the calculations when determining the angle that radiation impacts an interceptor.
- This simulation is not a Monte Carlo simulation. For this reason, the reaction rate data contained in the input files for specific interceptor components of interest should be made using a Monte Carlo radiation transport simulation. It is assumed that the use of a Monte Carlo radiation transport simulation will ensure more accurate results than analytical transport equations.
- Energy bins for initial spectra, spectral time dependencies, and interceptor component reaction rates are likely not identical. This simulation assumes that enough energy bins for each data set exist to create a roughly smooth curve to represent the data, as a linear average of bins is utilized when energies from multiple sources when they overlap. The end result could be a product of two or more averages.
- Values per energy bin for time dependencies of a flux and reaction rates will have discrete increments. The data for a time dependence of the flux and a reaction rate as a function of angle of incidence is an interpolative function of time and angle of incidence respectively. The end result could be a product of two interpolations.
- The reaction rates for components of interest inside the interceptor are a function of the azimuthal angle along the axis of the interceptor and averaged over all roll angles. The roll angle of the interceptor when a flux is incident is immaterial; to include this in the simulation would require that the reaction rate lookup tables include additional variables which could increase their size dramatically.
- The mass attenuation coefficients in air for photons and neutrons do not take energy losses due to atmospheric scatter into account. If desired, the user may create an input file consisting of custom attenuation coefficients that better model energy peak shifting.

CHAPTER 3 Nature of the Program

There are simulations and models that incorporate many of the effects and phenomena previously discussed, but there is not a single simulation that includes all of them. The majority of existing simulations also have processing times that are prohibitively large and cannot produce the quality and types of details that this model desires [Sawyer and Vette, 1976, Vette, 1991, CREME96 Homepage].

Given the trajectory of a KV by a driver program, this model calculates the total ionizing dose and dose rate experienced by not only the interceptor as a whole, but individual components within the interceptor. The dose is a result of encountering both the natural radiation trapped within the Van Allen belts, as well as any resultant dose due to encountering a nuclear burst at high altitudes. This simulation is designed to run as a function of time, with discrete time steps over the duration of the flight. Existing simulations cannot distinguish between the whole KV and components and cannot calculate radiation effects as a function of time and trajectory. This simulation is designed to handle multiple KV's and multiple RV's simultaneously throughout the course of a larger simulation.

This program is designed to utilize unclassified code. The classified portions of the program consist of the data that are found in the input files; unclassified data files may be used if so desired.

3.1 Interface Assumption

Because this model is designed to act as a module within a larger simulation, specific guidelines about the interface between the driver software and this model must be made. The following are assumed about the external software:

- Earth is assumed to be spherical; positions are specified in Earth-Centered Earth-Fixed (ECEF) coordinates in meters *(x,y,z)*.
- The shock front and atmospheric heave effects do not affect the trajectory of interceptor.
- An externally supplied function will interpret total doses by component to determine if a problem is occurring, rather than this judgement being performed by the dose calculating program.

- One time dynamic memory allocations are made in order to make the program's matrices reflect the data contained in the input files
- The code is assumed to be compiled using a compiler compatible with the 2005 Visual C++ Express Edition compiler, running in Windows XP

3.2 How the Program Interacts with a Larger Simulation

The routines that calculate the dose or dose rate for the components of interest inside of an interceptor consist of multiple classes, defined as a group as 'Weapon Effects and Probability of Nuclear Survival' (WEAPONS). Examples of the software architecture are: a class for a KV instantiation, a wrapper class enveloping the AP8/AE8 model, a nuclear event instantiation, an atmospheric attenuation class, a linkage class that associates the KV with the nuclear event, and a class that integrates the dose seen by a component. This configuration allows multiple instances of KV's and nuclear events to exist simultaneously with full interaction between them while still maintaining their unique identities and properties.

The *Interceptor* class contains the earth-centered earth-fixed (ECEF) coordinates and heading of an interceptor instance as well as radiation susceptibility and position ("angular reaction rate") information for a predetermined set of components inside of the interceptor. The attenuation rate as a function of incident angle along the interceptor's central axis for the various radiation species of a component is multiplied by the incident flux computed at the interceptor skin to find the component dose during that time step. This instantaneous dose is integrated by the *Dosemath* class in order to find the total dose. After initialization, this class only requires periodic updates of the interceptor coordinates and velocity vectors from the main program.

The *VanAllen* class is an interface wrapper for NASA's publicly available AP8/AE8 code. It takes an ECEF location in space and desired energies for flux profiles, and calculates the particle fluxes as a function of energy at that point. This flux information is returned to the calling methods, where the dose due to trapped electron and proton radiation is calculated. This class should not be called by the main program.

The *Fission* class, which creates a nuclear event instance, contains information about the initial and temporal particle flux as a function of weapon configuration and yield. This information is used to create a temporally varying flux at the interceptor location, thereby

allowing the component dosage to be calculated. This class should not be called by the main program for any reason other than initialization.

The *Atmosphere* class attenuates a particle flux through air. Given the ECEF locations of a burst and an interceptor for which to calculate a dose, it scales the flux to account for the scattering and absorption of the particle flux as a function of energy. Any energy loss due to scattering which resulted in a peak shift would have to be accounted for with the correct attenuation coefficients. This class should not be called by the main program.

The *Dosemath* class integrates doses received by each interceptor component during a given time step. This class keeps track of the average, max, and total dose seen by a component during the length of the runtime. The main program is able to call *Dosemath* class instances as a variable within an interceptor in order to return doses per interceptor component.

The *Fluxtodose* linkage class is the main interface between the calling program, the interceptor instantiations, the natural radiation model, and the event instantiations. For natural radiation this class calls the *VanAllen* class wrapper, receives a flux, and calculates the various interceptor component doses accordingly. For artificial radiation, this class calculates the distance and the angle of incidence for each interceptor and event instantiation passed as a reference. It calls the flux-at-interceptor calculating function, *Fluxm()*, and multiplies the flux per particle by its respective angular reaction rate to determine the component dose per unit time. This dose rate is integrated over the last time step to find the total dose in a given component.

An example block diagram of how the various classes interact with the main program and with each other is shown in Figure 3.1. A complete list with descriptions of methods and members for all classes, including inputs and outputs are presented in Appendices B-G.

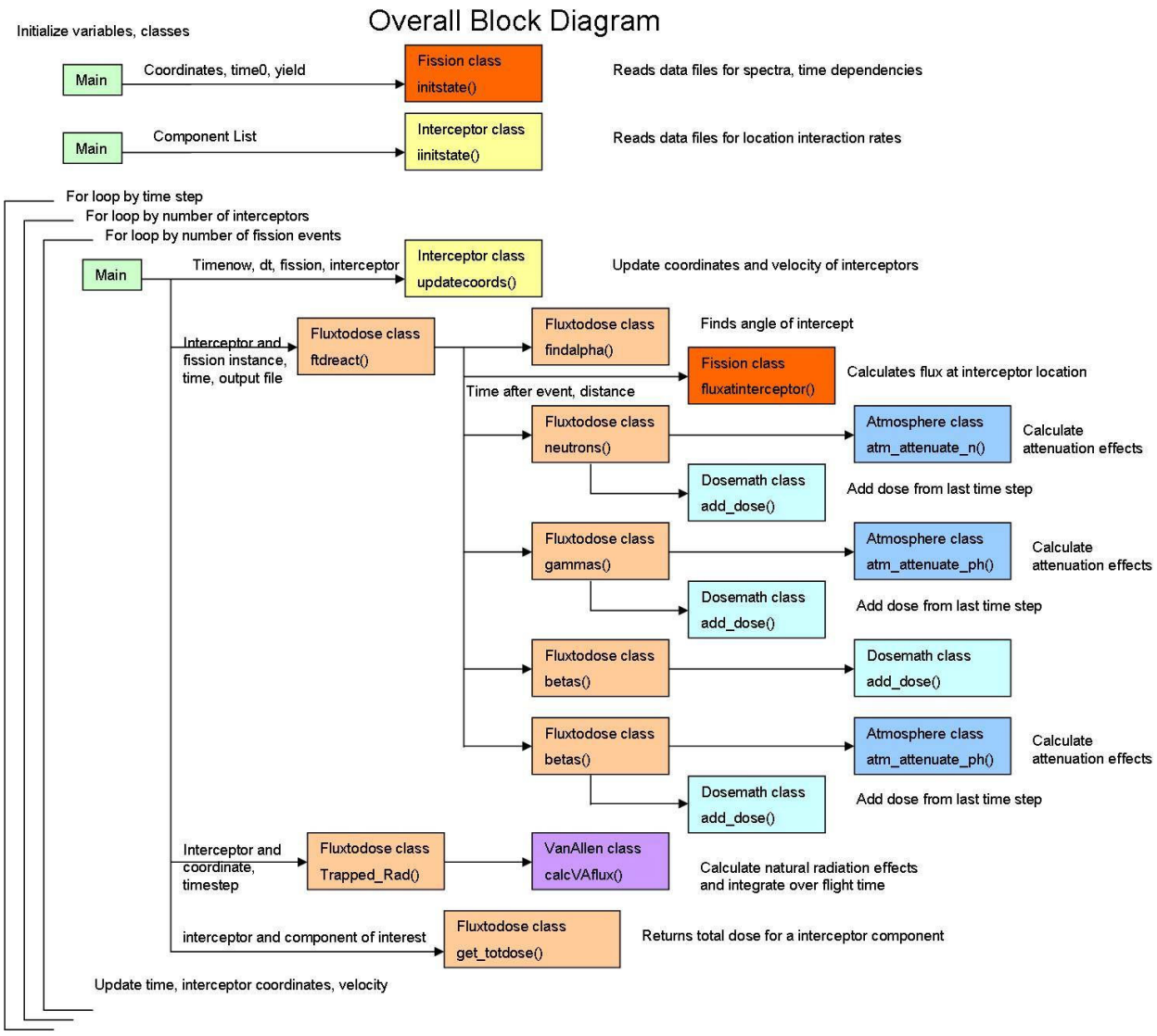

Figure 3.1 Sample overall block diagram for simulation. This shows the flow of information and interactions between the various classes, and the outside driver program designated "main".

3.3 Simulation Model Inputs

In order for the dose calculating functions to work, they must first be initialized with the correct state variables, and updated during runtime to track interceptors as they traverse the upper atmosphere. The two specific classes that need to be initialized before use are *Interceptor* and *Fission*. Initializing these classes defines the initial coordinates, type, yield, and other parameters required for operation. During this initialization process, the input files are read for

the corresponding interceptor components and nuclear event types. The data values contained in the input files are placed in dynamically allocated arrays for use during the simulation.

During runtime, the *Fission* class instance does not need to be updated by the external simulation. The *Fluxtodose* class calls the *Fission* class instance and passes the time after event. Conversely, the *Interceptor* class needs to have its ECEF coordinates and velocity vector updated continuously.

3.3.1 Initial Inputs from a Larger Simulation

- *Interceptor* class
    - Type of interceptor component (must have input file to match each component)
- *VanAllen* class
    - *None*
- *Fission* class
    - Time of nuclear event
    - Yield
    - ECEF coordinates of event
    - Temporal spectra for event (must have input file to match)
- *Atmosphere* class
    - *None*
- *Dosemath* class
    - *None*
- *Fluxtodose* class
    - *None*

3.3.2 Running Inputs from a Larger Simulation

- *Interceptor* class
    - Updated coordinates and velocity vector of interceptor (ECEF coordinates)
- *VanAllen* class
    - *None*
- *Fission* class
    - *None*

- *Atmosphere* class
    - *None*
- *Dosemath* class
    - *None*
- *Fluxtodose* class
    - Interceptor instance to calculate dose for
    - Desire to calculate natural and/or artificial radiation effects
    - Event instance that is affecting an interceptor
    - Current time
    - Time elapsed since last call

3.3.3 List of Input Variables

The following is a list of input variables that the calling program may use when invoking one of the three classes with which it will interact. The variables that are not meant to be updated during runtime serve to initialize each instance of the class.

Table 3.1 Variables sent to the Various Classes from Driver Program

Class Sent to	Variable Name	Variable Type	Description	Units	Updated during Runtime by Driver
Interceptor	complist	vector<mloclist> &	Contains a list of driver type location nodes	-	No
Interceptor	X	double	y ECEF coordinate of the interceptor	meters	Yes
Interceptor	Y	double	y ECEF coordinate of the interceptor	meters	Yes
Interceptor	Z	double	z ECEF coordinate of the interceptor	meters	Yes
Interceptor	Vx	double	Velocity in the x direction	meters/second	Yes
Interceptor	Vy	double	Velocity in the z direction	meters/second	Yes
Interceptor	Vz	double	Velocity in the z direction	meters/second	Yes
Fission	X	double	x ECEF coordinate of the nuclear event	meters	No
Fission	Y	double	y ECEF coordinate of the nuclear event	meters	No
Fission	Z	double	z ECEF coordinate of the nuclear event	meters	No
Fission	yld	double	Yield of the nuclear event	kT	No
Fission	time0	double	Time of nuclear event	seconds	No
Fission	ID	int	Identification number of event	-	No
Fission	filename	string	Contains the file to read a spectra from	-	No
Fission	timenow	double	Time of simulation	seconds	No
Fission	timeafter	double	Time after nuclear event	seconds	No
Fission	distance	double	Distance interceptor is from nuclear event	meters	No
Fission	fout	ofstream &	Output file to generate report	-	No
FluxtoDose	fiss	*fission	An instance of the fission class	-	Yes

Class Sent to	Variable Name	Variable Type	Description	Units	Updated during Runtime by Driver
FluxtoDose	inter	*interceptor	An instance of the interceptor class	-	Yes
FluxtoDose	timenow	double	Current time	seconds	Yes
FluxtoDose	deltat	double	Time step increment	seconds	Yes
FluxtoDose	fileout	ofstream &	Output file to generate report	seconds	Yes
Dosemath	dose	double	Dose for current timestep	Rads (Si)	No
Dosemath	timestep	double	Change in time	seconds	No
VanAllen	X	double	x ECEF coordinate of interest	meters	No
VanAllen	Y	double	y ECEF coordinate of interest	meters	No
VanAllen	Z	double	z ECEF coordinate of interest	meters	No
VanAllen	eflux	double **	Array for flux, first column is energy, second is number of particles	MeV, # particles	No
VanAllen	eflux_size	int	Size of flux spectra array	integer	No
VanAllen	pflux	double **	Array for flux, first column is energy, second is number of particles	MeV, # particles	No
VanAllen	pflux_size	int	Size of flux spectra array	integer	No
Atmosphere	x1	double	x ECEF coordinate of interest for point 1	meters	No
Atmosphere	y1	double	y ECEF coordinate of interest for point 1	meters	No
Atmosphere	z1	double	z ECEF coordinate of interest for point 1	meters	No
Atmosphere	x2	double	x ECEF coordinate of interest for point 2	meters	No
Atmosphere	y2	double	y ECEF coordinate of interest for point 2	meters	No
Atmosphere	z2	double	z ECEF coordinate of interest for point 2	meters	No
Atmosphere	coeff	double **	Spectral array, attenuation factor per energy	MeV, %	No
Atmosphere	num_coeff	int	Size of spectral array	integer	No
Atmosphere	filein	string &	Input file name of mass attenuation coefficients	-	No

3.4 Input File Formats

Because this program relies almost entirely upon the data files which it calls upon, it is imperative that care be taken in creating this data. This program is also designed to allow additional interceptor locations and nuclear event types to be added with minimal changes to the actual code so long as the proper corresponding input files exist.

In order for the program to read the input files correctly, the files must follow a specific data format. The size of an input file is immaterial as long as it follows the correct format, as the program is designed to store the data in dynamically sized arrays. The specific input files to be read are also to be determined during runtime in the case of reaction rate data for a given interceptor component and the energy and temporal spectra for a nuclear event.

3.4.1 Interceptor Class Input Files

The interceptor class needs a set of input files which define the following information for each component of interest:

- Reaction rate per particle type as a function of energy,
- The angular dependence of the reaction rate, and the
- Component location within the interceptor.

The name of this file is expected to be of the form *XXX_YYY_ZZZ_RRR*.dat where *XXX* corresponds to an interceptor designator, *YYY* corresponds to the interceptor type, *ZZZ* corresponds to a unique interceptor identification number, and *RRR* denotes the component described by the file.

Each component reaction rate input file has the format shown in Appendix H. The first line should contain a location ID in the same format as the input file name, the number of X ray, neutron, gamma, electron and proton energy bins, and the number of azimuthal angles for that component.

This simulation is not a Monte Carlo type simulation, but is capable of Monte Carlo type results if the component input files were formed using the results of Monte Carlo simulations. It is recommended that MCNP or similar software with a high fidelity interceptor model be used to create the reaction rate data for an energy spectra as well as azimuthal angle along the interceptor's axis.

To create the data contained in the input file, the geometry of the interceptor being modeled must be known. It is recommended that an average reaction rate for each discrete "angular bin" be calculated as the average dose received by a flux spectra incident circumferentially around the axis of the interceptor. The angular bins could be uniformly spaced, or could have a higher fidelity around angles with particularly important shielding characteristics.

See Appendix H for a more detailed description and sample input file with comments.

3.4.2 Fission Class Input Files

The event class input file should be of the format shown in Appendix I. Again, the first line contains data on the number of energy bins for the initial spectra for each particle type, and similar data for the time dependencies of the flux. The following lines contain an upper energy bin limit, a lower bin limit, and the initial flux. The end line of each particle species section of the input file is a number representing the number of MeV output per kT of explosive yield. This process is repeated for neutrons, gammas, electrons, and X rays in that order.

For temporal dependence, the first line contains the times during which a flux as a function of time is tallied. The following lines contain an upper and lower energy bin limit, and then a flux as a percentage of the original for every time increment. Again, this format is repeated for each of the radiation species in the order listed above.

See Appendix I for a more detailed description and sample input file with comments.

3.5 Outputs to External Simulation

In order to receive the component dosage calculated by the various classes, the calling program invokes a function in the linkage class which returns the dosage for each component of interest inside of the interceptor. The dosage is returned as a neutron, gamma, X ray, electron and proton total dose and max dose rate by using the referenced variables passed to the external simulation. This total dosage and maximum dose rate may then be compared by the calling function to component-specific threshold values in order to determine if the component is either working normally, encountering errors, or has failed completely. This comparison is not within the scope of the dose calculating functions, and is expected to be performed elsewhere.

3.6 Verification and Validation

To validate the results from calculations, hand calculations based on a test set of input files must be performed and compared with actual results from the simulation. These hand calculations will include comparisons of quantities such as distance to event from interceptor, flux by particle and energy at interceptor, and correct dosage per time step given a flux and blast front angle of intercept. Test data has been made to simulate an unclassified blast. As no factual reaction rates per interceptor component have been created yet, representative test files were

used. Also, certain components of DTRA provided software packages will be used to help correlate dosages and flux components associated with a nuclear burst of given parameters.

CHAPTER 4 Recommended Enhancements

To further increase the ability to calculate the radiation environment experienced by an interceptor during a typical mission, the following enhancements should be addressed.

4.1 Natural Radiation

The AP8/AE8 model, while still an industry standard, is decades old and many limitations in the results have been noted. NASA is currently developing a newer trapped radiation model, designated AP9/AE9. This model is expected to be beta-released around 2010. Assuming similar functionality and input parameters, the incorporation of this model could greatly increase the accuracy of natural radiation effects on interceptors.

Galactic Cosmic Rays and the effects of solar activities such as solar flares and solar storms are currently not modeled due to a lack of publicly available source code and model compatibility. The CREME96 model is widely regarded as the best model for simulating GCR's and solar effects. This model has publicly available services online, but no publicly available source code or executables [CREME Homepage]. This model is also designed to calculate the average effects over multiple orbits, and does not allow trajectory modeling. If this or a similar code were to become available and able to calculate effects over a trajectory, it is recommended it become part of the larger natural environment model.

4.2 Artificial Radiation

Further enhancements to the fidelity of the simulation would be to model the dependence of neutron speed as a function of energy: higher energy neutrons travel much faster than lower energy neutrons. A function could be written to store neutrons as a function of energy, and track them as they arrive at an interceptor location thus adding to its flux as a function of time and distance away from event. A linked list might be a good way to store neutrons in flight, and then delete them once they have had a chance to interact with all interceptors in a given scenario. This feature has not been added because of the dynamic memory size restrictions currently placed on the model. Neutron bounce from the atmosphere would also have to be incorporated if

higher fidelity neutrons physics were desired. A free neutron decay half life of roughly 14 minutes could also be incorporated.

The program is currently designed to determine reaction rates of a particular interceptor component of interest as a function of the 1-D azimuthal or pitch angle along the axis of the KV. This value is based on an average over all roll angles associated with that pitch angle. To make this more correct, the reaction rate could be a function of the roll angle as well as the angle of intercept relative to the axis of the KV for prompt radiation. For persistent radiation, a circumferential average would be sufficient.

The axis of the interceptor is designed to be oriented in the same direction as the interceptor's velocity vector. This assumes that the interceptor is not skidding or pitching in any direction during flight. This assumption could be removed if a separate variable were introduced accounting for the interceptors heading as well as its velocity vector.

Artificial electrons and their interaction within the magnetosphere surrounding the earth could be taken into account to more correctly model the corresponding effects. This would include their interaction with the Van Allen belts and also beta tube drift effects. A separate class which models electrons would be a good way to simulate this effect, as it would also allow long-term persistent beta fields to be modeled. The model currently under development by Lt. Col. Cross is expected to perform these calculations, and should be implemented upon availability.

To make the WEAPONS model more accurate without changing any actual programming, the various input files may be given a higher fidelity. This can include particle interaction rates tabulated for more interceptor components. Also, the energy and angular resolution of existing reaction tables can be refined to give a better representation of what effect incident particles are going to have on the total dosage of a given component.

Currently this code is not a Monte Carlo-type of simulation, and therefore does not have explicit statistics built into it. The statistics involved are all implicit in nature and based on the input files. The spectra input files have an associated error given this information's availability, and the interaction rate data as a function of azimuthal angle is created by an external simulation and therefore may have statistics associated with it that are not currently included in the present work. If the input files were appended to include this error, it could be included into the simulation to help determine how accurate the end results may be.

The current model is set to a fixed earth coordinate system. This could be enhanced by moving from an ECEF to an ECI coordinate system. The advantage to this modification would allow WEAPONS to be more seamlessly integrated into a wider variety of simulation platforms.

References

Akins, Keith A., Liam M. Healy, Shannon L. Coffey and J. Michael Picone, 2003, "Comparison of MSIS and Jacchia Atmospheric Density Models for Orbit Determination and Propagation," Paper AAS 03-165, *Spaceflight Mechanics 2003,* Advances in the Astronautical Sciences, Volume 114, pp. 951-970.

Armstrong, T. W. and B. L. Colborn, 2000, "Evaluation of Trapped Radiation Model Uncertainties for Spacecraft Design", NASA/CR-2000-210072, NASA, Marshall Space Flight Center, Alabama 358120.

Armstrong, T. W. and B. L. Colborn, 2000, "Trapped Radiation Model Uncertainties, Model-Data and Model-Model Comparisons", NASA/CR-2000-210071, NASA, Marshall Space Flight Center, Alabama 35812.

"Background: Trapped particle radiation models." SPENVIS - Space Environment, Effects, and Education System. 25 Feb. 2009 <http://www.spenvis.oma.be/spenvis/help/background/traprad/traprad.html#VAMP>.

Bass, J. N., S. M. Ayer, N. A. Bonito, R. G. Caton, and C. U. Cook. 1995. "Radiation Belt Analysis and Modeling." NASA STI/Recon Technical Report 96.

Byrd, R.C., 1995, 'Atmospheric transport of neutrons and gamma rays from a high-altitude nuclear detonation.' United States. Los Alamos National Lab. Department of Energy, LA--12962-MS.

CREME96 Homepage. 05 Mar. 2009 <https://creme96.nrl.navy.mil/>.

Cross, Christopher G. 2007. Computational Modeling of the Spatial Distribution and Temporal Decay of Geomagnetically Trapped Debris of a High Altitude Nuclear Detonation. Diss. Naval Postgraduate School.

Cross, Lt. Col. Christopher. E-mail to the author. 15 Oct. 2008.

Dolan, Philip J., and Samuel Glasstone, 1977, The Effects of Nuclear Weapons. Washington D.C.: The United States Department of Defense.

Drob, D. and J. Picone, 2000, Statistical Performance Measures of the HWM-93 and MSISE-90 Empirical Atmospheric Models and the Relation to Infrasonic CTBT Monitoring, *Proceedings of the 22nd Annual Seismic Research Symposium,* New Orleans, Sept. 12-15.

Dupont, D. G., 2004, "Nuclear Explosions in Orbit," *Scientific American*, vol. 290, iss. 6.

Dyal, P. 2006, Particle and field measurements of the Starfish diamagnetic cavity, *J. Geophys. Res.*, *111*, A12211, doi:10.1029/2006JA011827.

Hedin, A.E., 1987, "MSIS-86 Thermospheric Model", Journal of Geophysical Research, Vol. 92, No. A5, pp 4649-4662.

Hess, Wilmot N., 1964, *The Effects of High Altitude Explosions*, NASA TN D-2402, NASA Goddard, MD.

Jones, M. R., 2000, "ACD WFC CCD Radiation Test: The Radiation. Environment", ACS Instrument Science Report 00-09, STScI. Baltimore.

Leonard, Dan. 1998, "Radbelt.c." MIT. 09 Mar. 2009 <http://web.mit.edu/nbshah/Public/MSDO/rad_code/radbelt.c>.

Longmire, C.L., 1985, “EMP on Honolulu from the Starfish Event,” U.S. Air Force Weapons Laboratory Theoretical Note 353, 1985.

Marcos, F.A, B.R. Bowman and R.E. Sheehan, 2006. "Accuracy of Earth’s Thermospheric Neutral Density Models," AIAA 1006-6167.

McIlwain, C. E., 1963. "The Radiation Belts, Natural and Artificial," *Science* 142:355.

Narin, Francis and Maj. Walter Dumas, 1962,*A Quick Look at the Technical Results of Starfish Prime*.

"NEUTRON FLUX SPECTRUM." Nuclear Fundamentals. <http://www.tpub.com/content/doe/h1019v1/css/h1019v1_137.htm>.

Northrop, John. 1996, Handbook of Nuclear Weapon Effects. 1st Edition. Alexandria Virginia: Defense Special Weapons Agency. (Distribution Limited)

"Operation Dominic Starfish-Prime nuclear test from plane.jpg -." Wikipedia, the free encyclopedia. 25 Feb. 2009 <http://en.wikipedia.org/wiki/File:Operation_Dominic_Starfish-Prime_nuclear_test_from_plane.jpg >.

OK4me2 - Interest based News and support, for one’s desire to learn. 25 Feb. 2009 <http://www.ok4me2.net/wordpress/wp-content/uploads/radiation-belt.jpg>.

Pavel, A. L., K. I. Golden and M. B. Silevitch, 24 February 1977, “Nuclear Burst Plasma Injection into the Magnetosphere with Resulting Spacecraft Charging”, Proceedings of the Spacecraft Charging Technology Conference Air Force Surveys in Geophysics No. 364 (AFGL-TR-77-0051, C. P. Pike and R. R. Lovell, Eds.

Pfitzer, K. A., 1991, *Improved models of the inner and outer radiation belts*, MA: Phillips Laboratory, U.S. Air Force Systems Command, PLTR-91-2187.

Rabinowitz, M., A. P. Meliopoulos, E. N. Glytsis, and G. J. Cokkinides, 1992 'Nuclear Magnetohydrodynamic EMP, Solar Storms, and Substorms,' *International Journal of Modern Physics B,* Vol 6, No 20, pp 3353-3380.

Sawyer, D.M. and J.I. Vette, 1976. "Trapped Particle Environment for Solar Maximum and Solar Minimum (AP8)", NSSDC Report 76-06.

Shultis J. K. and R. E. Faw, 2000, *Radiation Shielding*. La Grange Park, IL: American Nuclear Society.

"Starfish5.JPG -." Wikipedia, the free encyclopedia. 09 Mar. 2009 <http://en.wikipedia.org/wiki/File:Starfish5.JPG>.

“The Aurora (Part 2: Origin of the Aurora).” 25 Feb. 2009 <http://astroprofspage.com/archives/421>

"TRP: AP8MIN/AP8MAX Models." CREME96 Homepage. 25 Feb. 2009 <https://creme96.nrl.navy.mil/cm/AP8.htm>.

"TRP: Limitations." CREME96 Homepage. 25 Feb. 2009 <https://creme96.nrl.navy.mil/cm/trplimits.htm>.

Tylka, A.J. Adams, J.H., Jr. Boberg, P.R. Brownstein, B. Dietrich and W.F. Flueckiger, 1997, "CREME96: A Revision of the Cosmic Ray Effects on Micro-Electronics Code", IEEE Trans. Nuclear Sci., 44, 2150-2160.

United States Nuclear Tests, July 1945 through September 1992 (DOE/NV--209-REV 15). Las Vegas, NV: Nevada Operations Office, Department of Energy, 2000.

University of Colorado at Boulder. 17 Feb. 2009
<http://www.colorado.edu/geography/gcraft/notes/gps/gif/ecefxyz.gif>.

Vette, J. I., 1991, "The AE-8 trapped electron model environment," *NSSDC WDC-A-R\&S 91-24*: NASA-GSFC.

APPENDIX A MCNP Input File for Atmospheric Attenuation Testing

The following is a sample MCNP atmospheric attenuation input file that was used to generate the results presented in Table 2.2. Atmospheric densities are the same as those used in the program and are presented in Table 2.1. The file presents a collimated source at the upper simulation altitude limit aimed directly at the origin. Detectors are placed at altitudes presented in Table 2.2.

```
Atmospheric Testing
c  ------- Project definition --------
c   This is designed to test atmospheric attenuation of particles at high altitudes
c
c @@@@@@@@@@@@@@@@@@@@@@@@@@@  BLOCK 1 -- cell cards @@@@@@@@@@@@@@@@@@@@@@@@@@@
c Atmospheric densities created from MSIS model, for average day 1963
1   0 -10 110                    IMP:P=0      $ inside of the half earth, graveyard
2   1 -1.19E-03 1 -5 110         IMP:P=1      $ 0 - 5      km altitude
3   1 -7.18E-04 5 -10 110        IMP:P=1      $ 5 - 10     km altitude
4   1 -4.17E-04 10 -30 110       IMP:P=1      $ 10 - 30    km altitude
5   1 -1.93E-05 30 -50 110       IMP:P=1      $ 30 - 50    km altitude
6   1 -1.18E-06 50 -80 110       IMP:P=1      $ 50 - 80    km altitude
7   1 -1.92E-08 80 -100 110      IMP:P=1      $ 80 - 100   km altitude
8   1 -4.50E-10 100 -200 110 111 IMP:P=1      $ 100 - 200  km altitude
9   1 -1.77E-13 200 -300 110     IMP:P=1      $ 200 - 300  km altitude
10  1 -7.73E-15 300 -400 110     IMP:P=1      $ 300 - 400  km altitude
11  1 -7.10E-16 400 -500 110     IMP:P=1      $ 400 - 500  km altitude
12  1 -8.98E-17 500 -600 110     IMP:P=1      $ 500 - 600  km altitude
13  1 -1.43E-17 600 -700 110     IMP:P=1      $ 600 - 700  km altitude
14  1 -3.31E-18 700 -800 110     IMP:P=1      $ 700 - 800  km altitude
15  1 -1.28E-18 800 -900 110     IMP:P=1      $ 800 - 900  km altitude
16  1 -7.25E-19 900 -1000 110    IMP:P=1      $ 900 - 1000  km altitude
17  1 -4.87E-19 1000 -999 110    IMP:P=1      $ 1000 - 3000  km altitude
c  Detector sphere, void to protect against double counts
55  0 -111                       IMP:P=0
98  0 -110                       IMP:P=0      $ Graveyard
99  0 999                        IMP:P=0      $ Graveyard

c @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  BLOCK 2 -- Surfaces @@@@@@@@@@@@@@@@@@@@@@@@@@@
c
1    so 637813700           $0 km altitude
5    so 638313700           $5 km altitude
10   so 638813700           $10 km altitude
30   so 640813700           $30 km altitude
50   so 642813700           $50 km altitude
80   so 645813700           $80 km altitude
100  so 647813700           $100 km altitude
200  so 657813700           $200 km altitude
300  so 667813700           $300 km altitude
400  so 677813700           $400 km altitude
500  so 687813700           $500 km altitude
600  so 697813700           $600 km altitude
700  so 707813700           $700 km altitude
800  so 717813700           $800 km altitude
900  so 727813700           $900 km altitude
1000 so 737813700           $1000 km altitude
```

```
999  so 937813700           $ outer radius of air (problem boundary) at 3000km
altitude
110  px 0                   $ cut earth in half at equator
33   sx 642823700 100       $ 1m radius detector sphere at 30km altitude
55   sx 642823700 100       $ 1m radius detector sphere at 50km altitude
88   sx 645933700 100       $ 1m radius detector sphere at 80km altitude
111  sx 647913700 100       $ 1m radius detector sphere at 100km altitude
c 222  sx 659813700 100     $ 1m radius detector sphere at 200km altitude

c @@@@@@@@@@@@@@@@@@@@@@@@  BLOCK 3 -- Data cards @@@@@@@@@@@@@@@@@@@@@@@@
c
c ----- Collimated Disc Gamma Source at x=2000 along x axis ------
SDEF  POS 837813700 0 0  AXS= 1 0 0  EXT = 0  RAD = d1  ERG=d2  PAR=2
       VEC = -1 0 0  Dir = 1    ARA =1
SI1 0 50
SP1 -21 1
c
c ----- Energy bins and frequency for source -----
SI2 H 1 2 3 4 5 6 7 8 9 10    $ energy bins created, H=histogram
SP2 D 0 1 1 1 1 1 1 1 1 1     $ normalized to emit all energies at the same frequency
c
PRINT -85 -86 -100 -110 -130 -140
NPS  100000          $-- particle histories
c
c ----- Detectors
F1:p 111             $ output fluence over detector sphere surface
c
MODE P
PHYS:P 10.0  0 1     $ Photons, 10 MeV limit, with Brehm, with coherent scattering
c
CUT:p  j  0.02       $ Kill all particles less than 20 keV
c
c  ------------------------------------------------------------
c  composition of air (nominal density 0.01205 g/cm^3 at sea level)
c  Composition changes at high altitudes, ignoring this fact due to low densities
c  ------------------------------------------------------------
M1    6000 -0.000124 $ C
      7000 -0.755268 $ N
      8000 -0.231781 $ O
     18000 -0.012827 $ Ar
```

APPENDIX B *Interceptor* Class Definition

The *Interceptor* class represents an instance of an interceptor, including its local coordinates, velocity, heading, reaction rate data per component of interest, dosage information per component of interest by particle species.

The *Interceptor* class is described in the files interceptor.h and interceptor.cpp, which are supplied with the source code.

B.1 Private Methods

```
bool testfileread(string XXX, string YYY, string ZZZ, string
RRR)
```

Outputs the data read from input files to the screen for verification.

Inputs are strings that correspond to interceptor identifiers in the XXX_YYY_ZZZ_RRR.dat format mentioned in section 3.4.1.

The outputs are the contents of the reaction matrices outputted to the screen.

```
bool addlocation(string ID)
```

Takes an input file name as a string and attempts to read the file for the reaction rate data per particle type as a function of azimuthal angle along the interceptors axis and energy bin. This information is stored within the nodes of the linked list of components within the *interceptor* class.

Input is the file name as a string in the XXX_YYY_ZZZ_RRR format as described in Chapter 3.4.1.

The outputs of this method are a boolean indicating whether the file read correctly and the created state variables for the interceptor component added.

B.2 Public Methods

interceptor()

Constructor, initializes a class instance, sets class initialized to false.

No inputs or outputs.

bool iinitstate(mloclist& complist)

Cycles through the external program's list of interceptor components and calls the addlocation(string) method for each node inside the list. It initializes the nodes added correctly to true.

Inputs are the linked list of interceptor components passed by the external program.

Output is the creation of a node within the linked list of components of the reaction rate data, as well as a boolean verifying that the method read the file correctly.

bool updatecoords(double xxm, double yym, double zzm, double vx, double vy, double vz)

Updates the state variable coordinates and velocity of the interceptor during runtime.

Inputs are the (x,y,z) ECEF coordinates of the interceptor and its <x,y,z> velocity vector.

No outputs from this method.

void Set_Calc_TrappedE(bool toggle)

Changes the toggle for calculating trapped electrons for this interceptor instance.

Input is a bool indication whether to calculate trapped electrons.

Output is changing the internal toggle variable.

void Set_Calc_TrappedP(bool toggle)

Changes the toggle for calculating trapped protons for this interceptor instance.

Input is a bool indication whether to calculate trapped protons.

Output is changing the internal toggle variable.

bool Create_Trapped_Arrays()

Creates the array holding the energies for which to calculate trapped proton and electron fluxes.

No inputs to this method.

Output is the creation of the internal array holding all of the proton and electron spectra needed to calculate trapped fluxes.

```
void Quick_Sort(**numbers, int left, int right)
```

Sorts the array of proton or electron spectra in increasing order by energy.

Inputs are the array to be sorted, first and last point in array to sort.

Output is returning a sorted array of pointers.

```
~interceptor()
```

Destructor, deallocates pointers.

No inputs or outputs.

B.3 Private Members

No private members.

B.4 Private Members

Table B.1 Public Members of *Interceptor* Class

Variable Name	Variable Type	Description	Units
initialized	bool	Is the instance properly initialized with a component location	-
Xm	double	ECEF x coordinate of interceptor	meters
ym	double	ECEF y coordinate of interceptor	meters
zm	double	ECEF z coordinate of interceptor	meters
vxm	double	ECEF x velocity vector of interceptor	meters/second
vym	double	ECEF y velocity vector of interceptor	meters/second
vzm	double	ECEF z velocity vector of interceptor	meters/second
ID	double	ID for interceptor instance	-
loclist	vector<compstruct>	Contains a list of interceptor components	-
trapped_elec_size	int	Size of trapped electron list	-
trapped_prot_size	int	Size of trapped proton list	-
tot_trapped_elec	double **	Array of trapped electrons	MeV
tot_trapped_prot	double **	Array of trapped protons	MeV
calc_trappedE	bool	Toggle to calculate trapped electrons	-
calc_trappedP	bool	Toggle to calculate trapped protons	-

APPENDIX C *Fission* Class Definition

The *Fission* class represents a particular instance of a nuclear event. It contains information on the yield, time of burst, location of burst, and initial and time dependant particle spectra.

The *Fission* class is described in the files fission.h and fission.cpp, which are supplied with the source code.

C.1 Private Methods

```
void breadfiles(string filename)
```

Reads an input data file as specified during runtime, creates spectra arrays.

Input is a string containing the data file to read a spectra from.

Outputs are source spectra information for all particle species by energy, as well as a time dependence of the spectra. Also included is the number of MeV per kT of yield for each of the particle species.

```
void testfileread(string filename)
```

Outputs the data read from input files to the screen for verification

Input is a string containing the file to read a spectra from.

The output is the contents of the reaction matrices outputted to the screen.

void findnflux(double timenow, double timeafter, double deltat, double distance)

Finds the neutron flux at the interceptor.

Inputs are time after burst and distance from burst to interceptor.

Outputs are updating the flux array by energy and time to determine the neutron flux at the interceptor location.

void findgflux(double timeafter, double distance)

Finds the gamma flux at the interceptor.

Inputs are time after burst and distance from burst to interceptor.

Outputs are updating the flux array by energy and time to determine the gamma flux at the interceptor location.

void findbflux(double timeafter, double distance)

Finds the beta flux at the interceptor.

Inputs are time after burst and distance from burst to interceptor.

Outputs are updating the flux array by energy and time to determine the beta flux at the interceptor location.

void findxflux(double timeafter, double distance)

Finds the X-ray flux at the interceptor.

Inputs are time after burst and distance from burst to interceptor.

Outputs are updating the flux array by energy and time to determine the X-ray flux at the interceptor location.

C.2 Public Methods

fission()

Constructor, initialized variables to 0

No inputs or outputs.

void initstate(double xxe, double yye, double zze, double yld, double t0, string fname)

Intializes the fission event by setting coordinates, time of burst, and yield. Sets up spectra tables as a method of time and energy.

Inputs are (x,y,z) ECEF coordinates, time of burst, yield of weapon, and file to read from.

No outputs from this method.

void fluxatinterceptor(double timenow, double deltat, double distance)

Calls the methods that determine the temporal flux at the interceptor by spectra.

Inputs are time to get flux, change in time, and distance from burst to interceptor.

No outputs from this method.

void set_calcneutrons(bool toggle)

Allows the user to set whether the fission event includes neutrons.

Input is a boolean indicating the calculation of neutrons.

Output is the changed state variable corresponding to the calculation of neutrons.

void set_calcgammas(bool toggle)

Allows the user to set whether the fission event includes gamma rays.

Input is a boolean indicating the calculation of gamma rays.

Output is the changed state variable corresponding to the calculation of gamma rays.

void set_calcbetas(bool toggle)

Allows the user to set whether the fission event includes beta rays.

Input is a boolean indicating the calculation of beta rays.

Output is the changed state variable corresponding to the calculation of beta rays.

void set_calcxrays(bool toggle)

Allows the user to set whether the fission event includes X rays.

Input is a boolean indicating the calculation of X rays.

Output is the changed state variable corresponding to the calculation of X rays.

bool outputnflux(ofstream &fout)

Appends the current neutron spectra flux matrix to the specified output file.

Input is the output file to append.

Output is the flux matrix written to the file. Returns true if the output file is already open, method will not execute and return false otherwise.

bool outputgflux(ofstream &fout)

Appends the current gamma spectra flux matrix to the specified output file.

Input is the output file to append.

Output is the flux matrix written to the file. Returns true if the output file is already open, method will not execute and return false otherwise.

bool outputbflux(ofstream &fout)

Appends the current beta spectra flux matrix to the specified output file.

Input is the output file to append.

Output is the flux matrix written to the file. Returns true if the output file is already open, method will not execute and return false otherwise.

bool outputxflux(ofstream &fout)

Appends the current X-ray spectra flux matrix to the specified output file.

Input is the output file to append.

Output is the flux matrix written to the file. Returns true if the output file is already open, method will not execute and return false otherwise.

~fission()

Destructor, deallocates memory used by pointers

No inputs or outputs.

C.3 Private Members

Table C.1 Private Members of *Fission* Class

Variable Name	Variable Type	Description	Units
minyield	double	Min yield from input file	kT
maxyield	double	Max yield from input file	kT
solidangle	double	Used for solid angle calculations, converts meters to cm	-
nMeVperkt	double	Number of MeV released per kT for neutrons	MeV/kT
gMeVperkt	double	Number of MeV released per kT for gammas	MeV/kT
bMeVperkt	double	Number of MeV released per kT for betas	MeV/kT
xMeVperkt	double	Number of MeV released per kT for X rays	MeV/kT
nMeV	double	MeV per kT multiplied by kT for neutrons	MeV
gMeV	double	MeV per kT multiplied by kT for gammas	MeV
bMeV	double	MeV per kT multiplied by kT for betas	MeV
xMeV	double	MeV per kT multiplied by kT for X rays	MeV
ntimeindex	int	Time index for neutrons for temporal spectra	-
gtimeindex	int	Time index for gammas for temporal spectra	-
btimeindex	int	Time index for betas for temporal spectra	-
xtimeindex	int	Time index for X rays for temporal spectra	-
nEindex	int	Energy index for neutrons	-
gEindex	int	Energy index for gammas	-
bEindex	int	Energy index for betas	-
xEindex	int	Energy index for X rays	-
ntEbins	int	Number of neutron energy bins for time temporal spectra	-
gtEbins	int	Number of gamma energy bins for time temporal spectra	-
btEbins	int	Number of beta energy bins for time temporal spectra	-
xtEbins	int	Number of X-ray energy bins for time temporal spectra	-
nTimes	int	Number of times in neutron temporal spectra	-
gTimes	int	Number of times in gamma temporal spectra	-
bTimes	int	Number of times in beta temporal spectra	-
xTimes	int	Number of times in X-ray temporal spectra	-
Dt	double	Time increment for program	seconds
Tscalar	double	Scalar used for time interpolations in temporal spectra	-
timedepn	double **	Array holding neutron temporal spectra	-
timedepg	double **	Array holding gamma temporal spectra	-
timedepb	double **	Array holding beta temporal spectra	-
timedepx	double **	Array holding X-ray temporal spectra	-

C.4 Public Members

Table C.2 Public Members of *Fission* Class

Variable Name	Variable Type	Description	Units
finitialized	bool	Is the fission instance properly initialized	-
Yield	double	Yield of fission instance	kT
yielderror	bool	Does the user yield fall within limits of input file temporal spectra	-
Xe	double	ECEF x coordinate of fission event	meters
Ye	double	ECEF y coordinate of fission event	meters
Ze	double	ECEF z coordinate of fission event	meters
time0	double	Simulation time of burst	seconds
nEbins	int	Number of energy bins for neutron temporal spectra	-
gEbins	int	Number of energy bins for gamma temporal spectra	-
bEbins	int	Number of energy bins for beta temporal spectra	-
xEbins	int	Number of energy bins for X-ray temporal spectra	-
Fluxn	double **	Neutron flux by energy at current simulation time	See Below
Fluxg	double **	Gamma flux by energy at current simulation time	See Below
Fluxb	double **	Beta flux by energy at current simulation time	See Below
Flux	double **	X-ray flux by energy at current simulation time	See Below
calcneutrons	bool	Allows user to toggle calculating neutrons from this event	-
calcgammas	bool	Allows user to toggle calculating neutrons from this event	-
Calcbetas	bool	Allows user to toggle calculating neutrons from this event	-
calcxrays	bool	Allows user to toggle calculating neutrons from this event	-
fissionID	int	ID for fission instance	-

APPENDIX D *FluxtoDose* Class Definition

The *FluxtoDose* class is the main interface between external simulations and instances of interceptors and nuclear bursts. It is used to calculate distance from burst, angle of intercept, and flux at the missile by species, and uses the flux and reaction data to update the dosage information per interceptor.

The *FluxtoDose* class is described in the files fluxtodose.h and fluxtodose.cpp, which are supplied with the source code.

D.1 Private Methods

```
bool findalpha()

     Finds the angle of incidence of blast front relative to
     axis of missile.

     Inputs are (x,y,z) coordinates of missile.

     Ouptut is the updated class variable alpha.


bool neutrons(fission &b, interceptor &m, ofstream &fileout)

     Calculates the neutron dose for the specified interceptor
     and nuclear event.

     Inputs are a fission and interceptor class instance as well
     as a file name to generate a report.

     Outputs are updating the neutron dose information struct
     within the linked list of components.
```

bool gammas(fission &b, interceptor &m, ofstream &fileout)

Calculates the gamma dose for the specified interceptor and nuclear event.

Inputs are a fission and interceptor class instance as well as a file name to generate a report.

Outputs are updating the gamma dose information struct within the linked list of components.

bool betas(fission &b, interceptor &m, ofstream &fileout)

Calculates the beta dose for the specified interceptor and nuclear event.

Inputs are a fission and interceptor class instance as well as a file name to generate a report.

Outputs are the updated beta dose information struct within the linked list of components.

bool xrays(fission &b, interceptor &m, ofstream &fileout)

Calculates the X-ray dose for the specified interceptor and nuclear event.

Inputs are a fission and interceptor class instance as well as a file name to generate a report.

Outputs are updating the X-ray dose information struct within the linked list of components.

D.2 Public Methods

FluxtoDose()

Constructor, intializes variables.

No inputs or outputs.

bool ftdreact(fission *f, interceptor *m, double timenow, double deltat, ofstream &fileout)

Reacts a fission instance with an interceptor instance. Calls methods that calculate angle of incidence and the various fluxes at the interceptor location and turns this information into a dose.

Inputs are fission and interceptor instances, current time and time increment, and an output file to send data to.

Outputs are a boolean verifying completion, and updates to the class variable for angle of incidence.

bool Trapped_Rad(interceptor *m, double dt, ofstream &fileout)

Reacts an interceptor instance with the natural trapped protons and electrons. Calls methods that calculate trapped fluxes at the interceptor location for the given energies and then turns this information into a dose.

Inputs are interceptor instance, time increment, and an output file to send data to.

Outputs are the updated dose information within the interceptor class, as well as updates to the report file.

```
~FluxtoDose()
```

Destructor, deallocates memory used by pointers.

No inputs or outputs.

D.3 Private Members

Table D.1 Private Members of *FluxtoDose* Class

Variable Name	Variable Type	Description	Units
Xe	double	ECEF x coordinate of fission event	meters
Ye	double	ECEF y coordinate of fission event	meters
Ze	double	ECEF z coordinate of fission event	meters
Xm	double	ECEF x coordinate of interceptor	meters
Ym	double	ECEF y coordinate of interceptor	meters
Zm	double	ECEF z coordinate of interceptor	meters
distance	double	Distance from interceptor to event	meters
Dkm	double	Distance from interceptor to event in km	kilometers
Vxe	double	ECEF x vector of fission event blast front	meters/second
Vye	double	ECEF y vector of fission event blast front	meters/second
Vze	double	ECEF z vector of fission event blast front	meters/second
Vxm	double	ECEF x velocity vector of interceptor	meters/second
Vym	double	ECEF y velocity vector of interceptor	meters/second
Vzm	double	ECEF z velocity vector of interceptor	meters/second
Alpha	double	Angle of incidence	degrees
Dt	double	Change in time	seconds
timeact	double	Time in external simulation	seconds
timeafter	double	Time after burst	seconds
my_vanallen	vanallen	Instance of Van Allen class to cacluate trapped particle fluxes	-

D.4 Public Members

No public members.

APPENDIX E *DoseMath* Class Definition

The *DoseMath* class is the integrator when calculating a dose and adding to a total dose per location within an interceptor.

The *DoseMath* class is described in the files dosemath.h and dosemath.cpp, which are supplied with the source code.

E.1 Private Methods

No private methods.

E.2 Public Methods

```
DoseMath()
```

Constructor, initializes dose variables to 0.

No inputs or outputs.

```
bool add_dose(double dose, double timest)
```

Add dose from current time step.

Inputs are the dose during last time step, and time step information.

Outputs are updating the class variables representing total, max, and average dose information, as well a boolean verifying that a dose was added.

```
double get_totdose()
```

Allows the user to obtain values for total dose.

No inputs.

Output is the returned value of total dose.

```
double get_maxdose()
```

Allows the user to obtain values for maximum dose rate.

No inputs.

Output is the returned value of maximum dose.

```
double get_avedose()
```

Allows the user to obtain values for average dose rate.

No inputs.

Output is the returned value of average dose.

```
~DoseMath()
```

Destructor.

No inputs or outputs.

E.3 Private Members

Table E.1 Private Members for *DoseMath* Class

Variable Name	Variable Type	Description	Units
totaldose	double	Total dose for interceptor component	Depends on Input
tottime	double	Total time for dose calculations for component	seconds
dmax	double	Max dose rate for component	Depends on Input
dave	double	Average dose rate for component	Depends on Input

E.4 Public Members

No public members.

APPENDIX F *VanAllen* Class Definition

The *VanAllen* class is a portable, user friendly software bundle encompassing NASA's AP8/AE8 model describing the Van Allen radiation belts surrounding the earth. All of the private methods come primarily from the NASA model, while the public method is specific to the interface between WEAPONS or a similar program.

The *VanAllen* class is described in the files vanallen.h and vanallen.cpp, which are supplied with the source code.

F.1 Private Methods

```
static void TRARA1(double FL, double BB0, double *E,double
*F,int N, int* DESCR)
```

```
The following description is per Dan Leonard, the author of
the modified AP8/AE8 model adapted to this simulation
[Leonard, 1998].
****************** TRARA1, TRARA2 **********************************
************************************************************************
************************************************************************
*** TRARA1 FINDS PARTICLE FLUXES FOR GIVEN ENERGIES, MAGNETIC FIELD ***
*** STRENGTH AND L-VALUE. FUNCTION TRARA2 IS USED TO INTERPOLATE IN ***
*** B-L-SPACE.                                                      ***
***   INPUT: DESCR(8)   HEADER OF SPECIFIED TRAPPED RADITION MODEL  ***
***          MAP(...)   MAP OF TRAPPED RADITION MODEL               ***
***                     (DESCR AND MAP ARE EXPLAINED AT THE BEGIN   ***
***                     OF THE MAIN PROGRAM MODEL)                  ***
 (these are currectly in the big static array MAP and DESCR)
***          N          NUMBER OF ENERGIES                         ***
***          Energ(N)   ARRAY OF ENERGIES IN MEV                   ***
***          FL         L-VALUE                                    ***
***          BB0        =B/B0  MAGNETIC FIELD STRENGTH NORMALIZED  ***
***                     TO FIELD STRENGTH AT MAGNETIC EQUATOR      ***
***  OUTPUT: Flux(N)    DECADIC LOGARITHM OF INTEGRAL FLUXES IN    ***
***                     PARTICLES/(CM*CM*SEC)                      ***
************************************************************************
```

```
static double TRARA2(int *SUBMAP, double IL,double IB)
```

The following description is per Dan Leonard, the author of the modified AP8/AE8 model adapted to this simulation.

```
************************************************************
***  TRARA2 INTERPOLATES LINEARLY IN L-B/B0-MAP TO OBTAIN     ***
***  THE LOGARITHM OF INTEGRAL FLUX AT GIVEN L AND B/B0.      ***
***    INPUT: MAP[] IS SUB-MAP (FOR SPECIFIC ENERGY) OF       ***
***                    TRAPPED RADIATION MODEL MAP            ***
***           IL       SCALED L-VALUE                         ***
***           IB       SCALED B/B0-1                          ***
***   OUTPUT: TRARA2   SCALED LOGARITHM OF PARTICLE FLUX       ***
************************************************************
```

```
static double trara3(int *SUBMAP, int position)

static double trara4(int *SUBMAP, int start_psn)

static double trara5(void)
```

The methods trara3 through trara5 were created when the model was brought from its original Fortran form into C++ by Dan Leonard. They perform the goto commands from the original code.

```
static void PopulateArrays(void)
```

This method reads the correct files for the lookup table data corresponding to the flux model at a given position.

No inputs to this method.

The outputs are populated arrays within the class instance.

F.2 Public Methods

VanAllen()

Constructor. This initializes the class variables and calls the PopulateArrays() method.

No inputs to this method.

The outputs are the populated lookup table arrays within the class instance.

bool calcflux (double x, double y, double z, double **eeflux, int eflux_size, double **ppflux, int pflux_size)

This method is meant to be the interface between the external program and the radiation belt model. The external passes the ECEF coordinates of interest, energy spectra data for electrons and protons and recieves a spectral flux upon return.

The inputs are the ECEF coordinates, a half filled [num energies][2] matrix for electrons and protons where the [energy][1] column corresponds to the energy and the [energy][2] column corresponds to the flux per second at that energy, and the number of energies for both electrons and protons.

The outputs are the completed matrix of energies and fluxes for both protons and electrons, as well as a boolean verifying the method worked correctly.

static void Output_VAbeta_flux(double **eflux, int eflux_size)

This method is used to output the electron flux for the given energy and flux matrix following the same format as the matrix in the calcflux() method.

Inputs are the matrix with electron spectra and flux column, and the size of the matrix.

Outputs are the contents of the matrix to the screen.

static void Output_VAproton_flux(double **pflux, int pflux_size)

This method is used to output the proton flux for the given energy and flux matrix following the same format as the matrix in the calcflux() method.

Inputs are the matrix with proton energy and flux column, and the size of the matrix.

Outputs are the contents of the matrix to the screen.

~VanAllen()

Destructor. This method deallocates the dynamic memory assigned to the populated lookup table arrays.

No inputs or outputs for this method.

F.3 Private Members

Table F.1 Private Members of *VanAllen* Class

Variable Name	Variable Type	Description	Units
arrayspopulated	bool	Status of arrays read from input file	-
MAP	int *	Global variable from FORTRAN code moved to class variable	-
MAPPRTNS	int *	Global variable from FORTRAN code moved to class variable	-
MAPELTNS	int *	Global variable from FORTRAN code moved to class variable	-
FISTEP	int	Global variable from FORTRAN code moved to class variable	-
FKB	double	Global variable from FORTRAN code moved to class variable	-
FLOG	double	Global variable from FORTRAN code moved to class variable	-
FKB1	double	Global variable from FORTRAN code moved to class variable	-
FKB2	double	Global variable from FORTRAN code moved to class variable	-
FINCR1	double	Global variable from FORTRAN code moved to class variable	-
FINCR2	double	Global variable from FORTRAN code moved to class variable	-
FKBM	double	Global variable from FORTRAN code moved to class variable	-
FLOGM	double	Global variable from FORTRAN code moved to class variable	-
FNB	double	Global variable from FORTRAN code moved to class variable	-
DFL	double	Global variable from FORTRAN code moved to class variable	-
J1	double	Global variable from FORTRAN code moved to class variable	-
J2	double	Global variable from FORTRAN code moved to class variable	-
ITIME	int	Global variable from FORTRAN code moved to class variable	-
L1	int	Global variable from FORTRAN code moved to class variable	-
L2	int	Global variable from FORTRAN code moved to class variable	-
I1	int	Global variable from FORTRAN code moved to class variable	-
I2	int	Global variable from FORTRAN code moved to class variable	-
FLOG1	int	Global variable from FORTRAN code moved to class variable	-
FLOG2	int	Global variable from FORTRAN code moved to class variable	-
FKBJ1	double	Global variable from FORTRAN code moved to class variable	-
FKBJ2	double	Global variable from FORTRAN code moved to class variable	-
SL1	double	Global variable from FORTRAN code moved to class variable	-
SL2	double	Global variable from FORTRAN code moved to class variable	-

F.4 Public Members

No public members.

APPENDIX G *Atmosphere* Class Definition

The *Atmosphere* class calculates air attenuation coefficients to apply to an unattenuated flux. Given two ECEF coordinates and an energy spectra, the atmospheric model first computes whether a line-of-sight exists between the two endpoints and, if so, integrates the air mass density at various points along the line of sight to determine the total density required to scale air attenuation coefficients.

The *Atmosphere* class is described in the files atmosphere.h and atmosphere.cpp, which are supplied with the source code.

G.1 Private Methods

```
bool populate_coeff()
```

```
Creates and populates the dynamic arrays containing an
energy flux and the corresponding total air attenuation
coefficient for neutrons, photons, and betas.  Uses stock
mass attenuation coefficients.

There are no inputs to this method.

Outputs are the populating of the class matrices storing a
spectra and attenuation coefficients.  Returns false if
insufficient memory available to create dynamic arrays.
```

bool read_n_file(string &filein)

Creates and populates the dynamic arrays containing an energy flux and the corresponding total air attenuation coefficient for neutrons. Uses mass attenuation coefficients read from the passed input file.

Input is the name of the file containing the mass attenuation coefficients for neutrons in air.

Output is the populating of the class matrix storing a spectra and attenuation coefficients for neutrons. Returns false if insufficient memory available to create dynamic arrays or if there was an error reading the file.

bool read_ph_file(string &filein)

Creates and populates the dynamic arrays containing an energy flux and the corresponding total air attenuation coefficient for photons. Uses mass attenuation coefficients read from the passed input file.

Input is the name of the file containing the mass attenuation coefficients for photons in air.

Output is the populating of the class matrix storing a spectra and attenuation coefficients for photons. Returns false if insufficient memory available to create dynamic arrays or if there was an error reading the file.

bool read_b_file(string &filein)

Creates and populates the dynamic arrays containing an energy flux and the corresponding total air attenuation coefficient for betas. Uses mass attenuation coefficients read from the passed input file.

Input is the name of the file containing the mass attenuation coefficients for betas in air.

Output is the populating of the class matrix storing a spectra and attenuation coefficients for betas. Returns false if insufficient memory available to create dynamic arrays or if there was an error reading the file.

bool line_of_sight ()

Given two points, calculates whether they can see each other unobstructed by the earth. Method can be modified so the "earth" includes a set altitude above it, with air densities below that are sufficient to shield a flux spectra.

There are no inputs to this method.

Output is a Boolean indicating an unobscured path line between two points.

double calc_tot_density()

Numerically integrates along a path line to approximate the total air mass density between the two points. The two points are processed from the higher altitude to the lower, and the location of density integration points increase in resolution towards lower altitudes. This allows greater accuracy where the higher densities occur.

There are no inputs to this method.

Output is a total density returned as a double.

G.2 Public Methods

atmosphere()

Constructor. Initializes state variables and calls the array populating methods.

No inputs to this method.

Outputs are the class variables holding attenuation coefficients for neutrons, photons and betas.

bool populate_n_coeff()

Creates and populates the dynamic arrays containing an energy flux and the corresponding total air attenuation coefficient for neutrons using stock data.

No inputs to this method.

Output is the populating of the class matrix storing a spectra and attenuation coefficients for neutrons. Returns false if insufficient memory available to create dynamic arrays.

bool populate_ph_coeff()

Creates and populates the dynamic arrays containing an energy flux and the corresponding total air attenuation coefficient for photons using stock data.

No inputs to this method.

Output is the populating of the class matrix storing a spectra and attenuation coefficients for photons. Returns false if insufficient memory available to create dynamic arrays.

bool populate_b_coeff()

Creates and populates the dynamic arrays containing an energy flux and the corresponding total air attenuation coefficient for betas using stock data.

No inputs to this method.

Output is the populating of the class matrix storing a spectra and attenuation coefficients for neutrons. Returns false if insufficient memory available to create dynamic arrays.

double find_density(double x, double y, double z, double vx, double vy, double vz)

Calculates the total air mass density along the LOS pathline.

Inputs are the ECEF coordinates of a starting point on the line, as well as the <x,y,z> component vector describing the line.

Output is the total air mass density along the path line.

bool atm_attenuate_n(double x1, double y1, double z1, double x2, double y2, double z2, double **coeff, double num_coeff)

Receives two points in ECEF coordinates as well as an (number of spectra energies) x 2 array and calculates the total air attenuation coefficient for neutrons at each spectral energy.

Inputs are two points in ECEF coordinates. Order of the two points is immaterial. Also a (number of spectra energies)x2 dynamic array and the size of the array are passed.

Outputs are the attenuation coefficients in second column of the passed array. If the line of sight between the two points intersects the earth or a fixed altitude above it, the attenuation coefficients are set to 0 indicating that no particles will reach the detector. False is returned if the neutron spectra and attenuation coefficient class array is not populated.

bool atm_attenuate_ph(double x1, double y1, double z1, double x2, double y2, double z2, double **coeff, double num_coeff)

Receives two points in ECEF coordinates as well as an (number of spectra energies) x 2 array and calculates the total air attenuation coefficient for photons (includes gamma rays and X rays) at each spectral energy.

Inputs are two points in ECEF coordinates. Order of the two points is immaterial. Also a (number of spectra energies)x2 dynamic array and the size of the array are passed.

Outputs are the attenuation coefficients in second column of the passed array. If the line of sight between the two points intersects the earth or a fixed altitude above it, the attenuation coefficients are set to 0 indicating that no particles will reach the detector. False is returned if the photon spectra and attenuation coefficient class array is not populated.

bool atm_attenuate_b(double x1, double y1, double z1, double x2, double y2, double z2, double **coeff, double num_coeff)

Receives two points in ECEF coordinates as well as an (number of spectra energies) x 2 array and calculates the total air attenuation coefficient for betas at each spectral energy.

Inputs are two points in ECEF coordinates. Order of the two points is immaterial. Also a (number of spectra energies)x2 dynamic array and the size of the array are passed.

Outputs are the attenuation coefficients in second column of the passed array. If the line of sight between the two points intersects the earth or a fixed altitude above it, the attenuation coefficients are set to 0 indicating that no particles will reach the detector. False is returned if the beta spectra and attenuation coefficient class array is not populated.

~atmosphere()

Destructor. Deallocates dynamic memory,

No inputs or outputs to this method.

G.3 Private Members

Table G.1 Private Members of *Atmosphere* Class

Variable Name	Variable Type	Description	Units
n_initialized	bool	Initialization status of neutron attenuation tables	-
ph_initialized	bool	Initialization status of photon attenuation tables	-
b_initialized	bool	Initialization status of beta attenuation tables	-
x1	double	ECEF x coordinate of first point to calculate from	meters
y1	double	ECEF y coordinate of first point to calculate from	meters
z1	double	ECEF z coordinate of first point to calculate from	meters
x2	double	ECEF x coordinate of second point to calculate from	meters
y2	double	ECEF y coordinate of second point to calculate from	meters
z2	double	ECEF z coordinate of second point to calculate from	meters
vx	double	ECEF x vector from high altitude point to low	meters
vy	double	ECEF y vector from high altitude point to low	meters
vz	double	ECEF z vector from high altitude point to low	meters
lx	double	ECEF x coordinate of minimum point to calculate from	meters
ly	double	ECEF y coordinate of minimum point to calculate from	meters
lz	double	ECEF z coordinate of minimum point to calculate from	meters
Rearth	double	Radius of the Earth in meters	meters
altitude1	double	Altitude of first point	meters
altitude2	double	Altitude of second point	meters
distance_away	double	Distance between two points	meters
n_coeff	double **	Holds energy bins and air attenuation coefficients for neutrons	-
ph_coeff	double **	Holds energy bins and air attenuation coefficients for photons	-
b_coeff	double **	Holds energy bins and air attenuation coefficients for betas	-
num_integrations	int	Number of points along LOS to calculate density for	-
nEnergy	int	Number of spectra points from neutron attenuation file	-
phEnergy	int	Number of spectra points from photon attenuation file	-
bEnergy	int	Number of spectra points from beta attenuation file	-
both_points	bool	Indicates whether the the lowest point along the LOS is not an endpoint	-

G.4 Public Members

No public members.

APPENDIX H Sample *Interceptor* Class Input File

This is a sample *Interceptor* class input file with descriptions in quotes. Leave comments with quotes out of actual input files. This is a rough spectrum in number of both energy bins and angular bins for demonstration purposes, and actual input file should be much more detailed to ensure accuracy.

This data is based on arbitrary values. It is recommended that MCNP or similar software be used with high fidelity geometry to create a detailed reaction rate table by angle of incidence. Recommended units for reaction rate data are Rad(Si) for neutrons, beta rays and gamma rays, and calories for X rays.

```
001001001001                    "component ID in XXX_YYY_ZZZ_RRR.dat format"
 9   10   9   12   6             "Number of energy bins for neutrons, gammas, betas, X rays, protons"
4   5   4   5                    "Number of angles for neutrons, gammas, betas, X rays"

0   60   120   180               "Neutron angular bins (degrees)"
"Upper energy limit, lower energy limit, reaction data by angle (rad(Si)/neutron)"
14.9      10        1.50E-09   1.02E-08   1.02E-08   1.50E-09
10        6.38      1.50E-09   1.02E-08   1.02E-08   1.50E-09
6.38      4.07      1.50E-09   1.02E-08   1.02E-08   1.50E-09
4.07      2.31      1.50E-09   1.02E-08   1.02E-08   1.50E-09
2.31      1.11      1.50E-09   1.02E-08   1.02E-08   1.50E-09
1.11      0.158     1.50E-09   1.02E-08   1.02E-08   1.50E-09
0.158     0.0219    1.50E-09   1.02E-08   1.02E-08   1.50E-09
0.0219   0.000101   1.50E-09   1.02E-08   1.02E-08   1.50E-09
0.000101   1.07E-05   1.50E-09   1.02E-08   1.02E-08   1.50E-09

0   60   90   120   180           "Gamma angular bins (degrees)"
"Upper energy limit, lower energy limit, reaction data by angle (rad(Si)/gamma)"
12        8         1.50E-09   1.02E-08   1.53E-08   1.02E-08   1.50E-09
8         6         1.50E-09   1.02E-08   1.53E-08   1.02E-08   1.50E-09
6         4         1.50E-09   1.02E-08   1.53E-08   1.02E-08   1.50E-09
4         2.05      1.50E-09   1.02E-08   1.53E-08   1.02E-08   1.50E-09
2.05      1.2       1.50E-09   1.02E-08   1.53E-08   1.02E-08   1.50E-09
1.2       0.8       1.50E-09   1.02E-08   1.53E-08   1.02E-08   1.50E-09
0.8       0.1       1.50E-09   1.02E-08   1.53E-08   1.02E-08   1.50E-09
0.1        0.07     1.50E-09   1.02E-08   1.53E-08   1.02E-08   1.50E-09
0.07      0.01      1.50E-09   1.02E-08   1.53E-08   1.02E-08   1.50E-09
0.01      0   1.50E-09   1.02E-08   1.53E-08   1.02E-08   1.50E-09
```

0 60 120 180 “Beta angular bins (degrees)”
“Upper energy limit, lower energy limit, reaction data by angle (rad(Si)/beta)”
14.9 10 1.50E-09 1.02E-08 1.02E-08 1.50E-09
10 6.38 1.50E-09 1.02E-08 1.02E-08 1.50E-09
6.38 4.07 1.50E-09 1.02E-08 1.02E-08 1.50E-09
4.07 2.31 1.50E-09 1.02E-08 1.02E-08 1.50E-09
2.31 1.11 1.50E-09 1.02E-08 1.02E-08 1.50E-09
1.11 0.158 1.50E-09 1.02E-08 1.02E-08 1.50E-09
0.158 0.0219 1.50E-09 1.02E-08 1.02E-08 1.50E-09
0.0219 0.000101 1.50E-09 1.02E-08 1.02E-08 1.50E-09
0.000101 1.07E-05 1.50E-09 1.02E-08 1.02E-08 1.50E-09

0 60 90 120 180 “X-ray angular bins (degrees)”
“Upper energy limit, lower energy limit, reaction data by angle (calories/X ray)”
0.3 0.26 1.50E-09 1.02E-08 1.53E-08 1.02E-08 1.50E-09
0.26 0.19 1.50E-09 1.02E-08 1.53E-08 1.02E-08 1.50E-09
0.19 0.14 1.50E-09 1.02E-08 1.53E-08 1.02E-08 1.50E-09
0.14 0.105 1.50E-09 1.02E-08 1.53E-08 1.02E-08 1.50E-09
0.105 0.075 1.50E-09 1.02E-08 1.53E-08 1.02E-08 1.50E-09
0.075 0.065 1.50E-09 1.02E-08 1.53E-08 1.02E-08 1.50E-09
0.065 0.045 1.50E-09 1.02E-08 1.53E-08 1.02E-08 1.50E-09
0.045 0.02 1.50E-09 1.02E-08 1.53E-08 1.02E-08 1.50E-09
0.02 0.01 1.50E-09 1.02E-08 1.53E-08 1.02E-08 1.50E-09
0.01 0.0025 1.50E-09 1.02E-08 1.53E-08 1.02E-08 1.50E-09
0.0025 0.0008 1.50E-09 1.02E-08 1.53E-08 1.02E-08 1.50E-09
0.0008 0 1.50E-09 1.02E-08 1.53E-08 1.02E-08 1.50E-09

“Energy to calculate trapped protons, and omnidirectional reaction rate at that energy”
50 1.50E-09
30 2.37E-09
20 3.24E-09
15 4.09E-09
5 5.73E-09
1 6.50E-09

APPENDIX I Sample *Fission* Class Input File

This is a sample *Fission* class input file with comments in quotes; these comments should be left out of actual input files. This is a rough spectrum, and actual input file should be much more detailed to ensure accuracy. The program will size its matrices storing the data in the inputs files dynamically, so that is handled automatically. The times involved in the temporal spectra should highlight important characteristics in the time profile such as flux spikes, as linear interpolation is used to scale the spectrum between times. Note that the energy bins for the spectrum and time dependencies do not have to be congruent, the program can handle overlaps by averaging a spectrum over energy and interpolating between times and angles.

This is a sample spectrum partially representing an unclassified burst. A classified spectrum must be created in this format if an actual weapon type were to be simulated.

```
8 10 9 9                "Number of neutron, gamma, beta, and X-ray spectrum energy bins"
8 10 9 8                "Number of neutron, gamma, beta, and X-ray time dependent energy bins"
6 6 5 5                  "Number of neutron, gamma, beta, and X ray number of times with dependencies"
10 50                   "Min and Max yield for current fission model, in kT"

14.9    10      5.63E-04    "Upper and lower energy limit (MeV) and number of particles (/MeV) for neutrons"
10      6.38    5.26E-03
6.38    4.07    2.81E-02
4.07    2.31    8.11E-02
2.31    0.55    2.88E-01
0.55    0.158   4.37E-01
0.158   0.111   4.56E-01
0.111   0.00123 3.50E+00
0.00123         1.07E-05        3.15E+02
4.3462E+20                      "MeV per kT to help scale spectra by yield"

12      10      7.50E-02    "Upper and lower energy limit (MeV) and number of particles (/MeV) for gammas"
10      7       9.00E-03
7       5       6.00E-02
5       2.5     3.00E-02
2.5     1.6     5.00E-02
1.6     1       8.00E-02
1       0.5     3.00E-01
0.5     0.09    1.10E+00
0.09    0.05    6.00E+00
0.05    0       6.00E-03
9.80E+22                        "MeV per kT to help scale spectra by yield"
```

13 10 7.50E-02 “Upper and lower energy limit (MeV) and number of particles (/MeV) for betas”
10 7 9.00E-03
7 5 6.00E-02
5 2.5 3.00E-02
2.5 1 8.00E-02
1 0.5 3.00E-01
0.5 0.09 1.10E+00
0.09 0.05 6.00E+00
0.05 0 6.00E-03
9.80E+22 “MeV per kT to help scale spectra by yield”

0.3 0.26 3.08E-107 “Upper and lower energy limit (MeV) and number of calories (/MeV) for X rays”
0.26 0.19 3.04E-77
0.19 0.105 4.27E-41
0.105 0.045 3.99E-16
0.045 0.02 2.71E-06
0.02 0.01 8.69E-03
0.01 0.0025 4.37E-01
0.0025 0.0008 5.82E-02
0.0008 0 2.09E-02
9.80E+22 “MeV per kT to help scale spectra by yield”

0 0.001 0.01 0.1 1 100 “Times (sec) of relevance for scaling for neutrons”
“Upper and lower energy limit (MeV) for neutrons, then percent of original flux depending on time”
14.9 10 1.00E+00 9.00E-01 8.00E-01 7.00E-01 6.00E-01 4.00E-01
10 4.97 1.00E+00 9.00E-01 8.00E-01 7.00E-01 6.00E-01 4.00E-01
4.97 3.01 1.00E+00 9.00E-01 8.00E-01 7.00E-01 6.00E-01 4.00E-01
3.01 1.83 1.00E+00 9.00E-01 8.00E-01 7.00E-01 6.00E-01 4.00E-01
1.83 0.55 1.00E+00 9.00E-01 8.00E-01 7.00E-01 6.00E-01 4.00E-01
0.55 0.158 1.00E+00 9.00E-01 8.00E-01 7.00E-01 6.00E-01 4.00E-01
0.158 0.0219 1.00E+00 9.00E-01 8.00E-01 7.00E-01 6.00E-01 4.00E-01
0.0219 0.000101 1.00E+00 9.00E-01 8.00E-01 7.00E-01 6.00E--01 4.00E-01

0 0.001 0.01 0.1 1 10 “Times (sec) of relevance for scaling for gammas”
“Upper and lower energy limit (MeV) for gammas, then percent of original flux depending on time”
13 9 1.00E+00 9.00E-01 8.00E-01 7.00E-01 6.00E-01 3.00E-01
9 6.5 1.00E+00 9.00E-01 8.00E-01 7.00E-01 6.00E-01 3.00E-01
6.5 4.5 1.00E+00 9.00E-01 8.00E-01 7.00E-01 6.00E-01 3.00E-01
4.75 2.275 1.00E+00 9.00E-01 8.00E-01 7.00E-01 6.00E-01 3.00E-01
2.275 1.4 1.00E+00 9.00E-01 8.00E-01 7.00E-01 6.00E-01 3.00E-01
1.4 0.65 1.00E+00 9.00E-01 8.00E-01 7.00E-01 6.00E-01 3.00E-01
0.65 0.08 1.00E+00 9.00E-01 8.00E-01 7.00E-01 6.00E-01 3.00E-01
0.08 0.04 1.00E+00 9.00E-01 8.00E-01 7.00E-01 6.00E-01 3.00E-01
0.04 0.02 1.00E+00 9.00E-01 8.00E-01 7.00E-01 6.00E-01 3.00E-01
0.015 0.05 1.00E+00 9.00E-01 8.00E-01 7.00E-01 6.00E-01 3.00E-01

0 0.01 0.1 1 10 "Times (sec) of relevance for scaling for betas"
"Upper and lower energy limit (MeV) for betas, then percent of original flux depending on time"
13 9 1.00E+00 9.00E-01 7.00E-01 6.00E-01 3.00E-01
9 6.5 1.00E+00 9.00E-01 7.00E-01 6.00E-01 3.00E-01
6.5 4.5 1.00E+00 9.00E-01 7.00E-01 6.00E-01 3.00E-01
4.75 2.275 1.00E+00 9.00E-01 7.00E-01 6.00E-01 3.00E-01
2.275 1.4 1.00E+00 9.00E-01 7.00E-01 6.00E-01 3.00E-01
1.4 0.65 1.00E+00 9.00E-01 7.00E-01 6.00E-01 3.00E-01
0.65 0.04 1.00E+00 9.00E-01 7.00E-01 6.00E-01 3.00E-01
0.04 0.02 1.00E+00 9.00E-01 7.00E-01 6.00E-01 3.00E-01
0.015 0.05 1.00E+00 9.00E-01 7.00E-01 6.00E-01 3.00E-01

0 0.02 0.05 0.075 0.1 "Times (sec) of relevance for scaling for X rays"
"Upper and lower energy limit (MeV) for X rays, then percent of original flux depending on time"
0.3 0.22 7.50E-01 1.00E+00 7.00E-01 2.00E-01 0.00E+00
0.22 0.055 7.50E-01 1.00E+00 7.00E-01 2.00E-01 0.00E+00
0.055 0.025 7.50E-01 1.00E+00 7.00E-01 2.00E-01 0.00E+00
0.025 0.015 7.50E-01 1.00E+00 7.00E-01 2.00E-01 0.00E+00
0.015 0.005 7.50E-01 1.00E+00 7.00E-01 2.00E-01 0.00E+00
0.005 0.00125 7.50E-01 1.00E+00 7.00E-01 2.00E-01 0.00E+00
0.00125 0.0004 7.50E-01 1.00E+00 7.00E-01 2.00E-01 0.00E+00