
Bharat Ratanlal Sagar

Master of Science (120 credits)
Space Engineering - Space Master

Luleå University of Technology
Department of Computer Science, Electrical and Space Engineering
SIMULATION OF TRANSPORT OF GAMMA-RAYS THROUGH ATMOSPHERE USING MONTE CARLO METHOD FOR THE TARANIS MISSION OF CNES

September 18, 2012

Bharat Ratankal Sagar

Master Thesis - SpaceMaster Round 6

Thesis Supervisors: Professor Francois Forme, IRAP, Toulouse, France
Professor Pierre-Louis Blelly, IRAP, Toulouse, France

Coordinator: Christophe Peymirat, IRAP, Toulouse, France

Examiner: Johnny Ejemalm, LTU, Kiruna, Sweden
Acknowledgments

It is really immense pleasure to thank all of the people who made this thesis possible.
First of all I will like to thank my Thesis supervisor Professor François Forme for his help and prolific advise. With his inspiration and great effort of explaining things in a simple and precise manner, really make programming easier for me.

I will also like to thank my another supervisor Professor Pierre-Louis Blelly who helped me for choosing such a nice subject for my master thesis. I am also grateful to my coordinator Professor Christophe Peymirat who helped me not only during my second year of my master studies but also during my thesis as well. I am thankful to Erasmus Mundus for such an excellent SpaceMaster program and for the scholarship I obtained for the educational journey through different universities in Consortium.

Finally, I will like to dedicate this thesis to my Master who always guided me on the correct path and helped me in taking correct decisions in my life.

For any errors or inadequacies that may remain in my master thesis work, the responsibility is entirely my own.
Abstract

Simplified analytical approximation is proposed for determining total cross sections in order to describe gamma ray photon interactions with atomic electron. Interactions such as incoherent scattering\(^1\), photoelectric effect and pair production are taken into consideration. Angular scattering probability is determined using differential cross section (DCS) and by taking cumulative distribution function (CDF) of DCS, scattering angle of photon is determined. Monte Carlo method\(^2\) is used for random sampling of every single scattered photon and their respective direction. Developed a code in FORTRAN 90 software to simulate all different kinds of photon interactions randomly depending on probability of occurrence. Comparison of obtained results of simulation with Ostgaard (2008)\([15]\), to validate developed code.

**Keywords:** Monte Carlo simulation, Radiation transport, Sampling algorithm, Total cross section, Angular scattering probability

Résumé

Une approximation analytique simplifiée est proposée pour déterminer la section efficace totale afin de décrire les interactions entre les électrons atomiques et les photons gamma. Les interactions telles que la diffusion, l’effet photoélectrique et la production de paires sont considérées. Angular scattering probability est déterminée en utilisant les sections efficaces différentielles et grâce à leur fonction de répartition, l’angle de diffusion est déterminé. La méthode de Monte Carlo est utilisée pour un échantillonnage aléatoire de chaque photon diffusé et leur direction respective. Un code en FORTRAN 90 a été développé pour simuler les différentes interactions aléatoirement en fonction de la probabilité d’apparition. La comparaison des résultats obtenus par simulation avec ceux d’Ostgaard (2008)\([16]\) valident le code.

**Mots clés:** Simulation Monte Carlo, Transport radiatif, Algorithme d’échantillonnage, Section efficace totale, angular scattering probability.

---

\(^1\)Considering both bound and unbound electron.

\(^2\)Composition and rejection method.
<table>
<thead>
<tr>
<th>Quantity</th>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avogadro’s number</td>
<td>$N_A$</td>
<td>$6.0221415 \times 10^{23}$ mol$^{-1}$</td>
</tr>
<tr>
<td>Velocity of light in vacuum</td>
<td>$c$</td>
<td>$2.99792458 \times 10^8$ m s$^{-1}$</td>
</tr>
<tr>
<td>Reduced Planck’s constant</td>
<td>$\hbar = \hbar/(2/\pi)$</td>
<td>$6.58211915 \times 10^{-16}$ eV s</td>
</tr>
<tr>
<td>Electron charge</td>
<td>$e$</td>
<td>$1.60217653 \times 10^{-19}$ C</td>
</tr>
<tr>
<td>Electron mass</td>
<td>$m_e$</td>
<td>$9.1093826 \times 10^{-31}$ Kg</td>
</tr>
<tr>
<td>Electron rest energy (ERE)</td>
<td>$m_e c^2$</td>
<td>$510.998918$ KeV</td>
</tr>
<tr>
<td>Classical electron radius</td>
<td>$r_e = e^2/(m_e c^2)$</td>
<td>$2.817940325 \times 10^{-15}$ m</td>
</tr>
<tr>
<td>Fine-structure constant</td>
<td>$\alpha = e^2/(\hbar c)$</td>
<td>$1/137.03599911$</td>
</tr>
<tr>
<td>Bohr radius</td>
<td>$a_0 = \hbar^2/(m_e e^2)$</td>
<td>$0.5291772108 \times 10^{-10}$ m</td>
</tr>
</tbody>
</table>

Table 1: Symbols and numerical values often used in Thesis
# Contents

1 Introduction ................................. 1
   1.1 TARANIS Mission and Objectives ................................. 1
   1.2 Mechanisms of generation of TLEs and TGFs ......................... 2
   1.3 Simulation of Gamma Rays: Radiation Transport ..................... 3
   1.4 Structure of Thesis .................................. 4

2 Monte Carlo: Random Sampling Methods .......... 6
   2.1 Inverse-Transform Method .................................. 6
   2.2 Rejection Method .................................... 7
   2.3 Composition Method .................................. 9

3 Photon Interactions ............................ 11
   3.1 Path Length And Interacting Medium .......................... 12
   3.2 Compton (Incoherent) Scattering .............................. 12
      3.2.1 Free Electron At Rest ................................ 13
      3.2.1.1 Algorithm For Random Sampling ...................... 17
      3.2.1.2 Steps for sampling random values of “Fractional Energy” \( \tau \) .......................... 18
      3.2.2 Bound Electron .................................... 19
      3.2.2.1 Algorithm For Random Sampling ...................... 20
      3.2.2.2 Steps for sampling random values of “Fractional Energy” \( \tau \) .......................... 20
   3.3 Photoelectric Effect .................................. 21
      3.3.1 Algorithm for Photoelectric Simulation ................... 21
      3.3.2 Steps for simulation of photo-electron ................... 22
   3.4 Electron-positron pair production ......................... 24
      3.4.1 Algorithm for sampling of “electron reduced energy” \( \epsilon \) .................. 26
      3.4.2 Steps for random sampling of “electron reduced energy” \( \epsilon \) ............... 27
      3.4.3 Angular distribution of produced particles .................. 28

4 Conclusion and future work ..................... 30

5 Appendix .................................... 35
   5.1 Tracing A Path .................................... 35
   5.2 Schematic for Monte Carlo Simulation for Photon Transport ........ 37
Chapter 1

Introduction

1.1 TARANIS Mission and Objectives

TARANIS is the mission from French Space Agency (CNES) and it is scheduled to be launched by the end of year, 2015. Currently its in development phase. TARANIS stands for Tools for the Analysis of RAdiation from lightNIing and Sprites. The main aim of TARANIS mission is to study coupling of magnetosphere-ionospheresphere-atmosphere via transient processes (sprites and their associated phenomena). Transient processes such as blue jets, red sprites, halos, elves, etc observed at altitudes from 20 to 100 Km, are known as Transient Luminous Events (TLEs) shown in Figure 1.1.

Figure 1.1: Diagram of different TLEs between 20 Km to 100 Km
Moreover, the detection and the study of gamma-ray and X-ray flashes probably linked to TLEs, known as Terrestrial Gamma Flashes (TGFs), are part of the mission objectives. Study’s reach of this mission has been extended to include the transient precipitations and acceleration of electrons, regardless of whether they are directly linked to TLEs. The satellite’s orbit is mostly emphasis on medium and low altitudes, that has been studied very few times in past.[19]

Four main objectives of TARANIS mission:

- Estimate the rate of occurrence, highlight trigger factors and associated emissions of TLEs and TGFs.
- Characterize TGFs and runaway electrons that accelerate upwards in the atmosphere towards the magnetosphere.
- Identify the effects of TLEs and TGFs on coupling between ionosphere and magnetosphere.
- Specify the role of precipitated electrons in coupling between the magnetosphere and the atmosphere.

1.2 Mechanisms of generation of TLEs and TGFs

TGFs were accidently discovered by an astrophysics experiment called BASTE, designed to probe the cosmos for gamma-rays bursts. Normally TGFs were observed in upper atmosphere and would appear to be related to solar activity.

The mechanisms explained for generation of TLEs and TGFs have been contradictory. Some studies suggest that generation of TGFs and TLEs are related to quasi-static electric field that
results in breakdown of air (Figure 1.2 LHS); other studies suggest that TLEs and TGFs were related with avalanche of relativistic electrons that are triggered by cosmic radiation (Figure 1.2 RHS) and can extent as far as the ionosphere and magnetosphere. Deceleration radiation by relativistic electron can also produce X-rays and gamma rays. Observation of X-rays and gamma rays shows the probable existence of runaway electrons travelling upwards as TLEs occurs. Some studies suggest that these runaway electrons crosses the ionosphere and can spread into the magnetosphere.

Further studies on highly energetic gamma rays and how these gamma rays interact with the medium through which it passes, will enable us to understand more deeply about the generation mechanisms of TGFs and TLEs.

1.3 Simulation of Gamma Rays: Radiation Transport

The transport of Gamma rays radiation through atmosphere is a complicated process that is amenable to Monte Carlo simulation method. These Gamma rays photons are highly energetic and can go up-to GeV but due to the fact that photons from Transient Luminous Events (TLEs) studied in TARANIS mission has specific energy distribution that varies from some lower limit to highest energy range, i.e. why we are considering photons energy range from 1 KeV to 20 MeV. TARANIS (Tool for the Analysis of Radiation from lightNING and Sprites) is a mission to study coupling of magnetosphere-ionosphere-atmosphere via transient processes. TLEs covers all the optical phenomenon (such as blue jets, red sprites, halos elves, etc), as shown in Figure 1.1, observed at an altitude ranging from 20 Km to 100 Km. Most dominant photons interactions of TLEs with atmosphere are compton scattering, photoelectric effect and pair production. Interaction of photons with atomic electron will reduce the part of photons energy and get scattered in different direction, as shown in Figure 1.3. The atomic electron may take total energy of photon and terminates photon’s history as it is considered to be absorb by electron. Based on the idea of relative rates a relative probability is assigned to each scattering or absorption event. A gamma-ray photon travels in straight lines until it interacts with an atom, at which point it may disappear or scatter at random. The path lengths for next interaction and the amount of energy that atomic electron takes from photons are both random process.

The path of photons is traced until it get absorb or went out of specified boundary limit, in our case boundary limits are from 20 Km to 100 Km. In summary, the simulation of gamma ray photon can be performed by tracing life history of every single photon originate at a random
position, direction, and energy, which travels in straight-line between successive interaction whose path lengths are random. The photon interacts with the atoms in atmosphere at random, and its life history is concluded at random.

1.4 Structure of Thesis

Aim of my master thesis work at IRAP laboratory is to build a code for simulation of transport of gamma rays through atmosphere using Monte Carlo method for the TARANIS mission. Chapter 1 of my master thesis introduces the TARANIS mission and its objectives, generation mechanisms of TLEs and TGFs with some insights of Photon simulation. In chapter 2 different Monte carlo methods are explained, that will be used for random sampling. Chapter 3 contains detailed discription of photon interactions, their cross sections, scattering angle, scattered energy, etc. and algorithm for the simulation for all different kinds of photon interactions by using Monte Carlo method discussed in chapter 2. Chapter 4 contains the comparisons of simulation results and final conclusion followed by appendix showing derivation for path length for next interaction and detail schimetic of Monte Carlo simulation in order to understand the way in which the developed code works.
Chapter 2

Monte Carlo: Random Sampling Methods

In the year 1940, Monte Carlo methods was used for the first time by the scientists working on the development of nuclear weapon in Las Almos. Monte Carlo methods belongs to one of the branch of mathematical method in which various computational problem are solved by using random numbers. Nowadays, Monte Carlo methods are widely used to solve complex physics problems, including multivariate independent variables problems, radiation transport and statistical physics where conventional methods would takes much more computer time and memory[17].

In Monte Carlo simulation of radiation transport, track of each particle is viewed as a random sequence that ends with particular interaction and as a result of this interaction, particles changes its direction, losses its energy and give rise to another secondary particle. To simulate these interaction events, an interaction model is required i.e. set of differential cross sections (DCSs) to describe these interaction events. The probability distributions functions (PDFs) of random variables are determined by their DCSs to characterize their track; 1) distance between two successive interaction events, 2) kind of interactions that particle undergoes randomly and 3) scattering angle and loss of energy by particle (and if any secondary particle is emitted then its initial states is also determined). Once PDFs are determined, Random histories are generated using appropriate sampling methods, discussed later in this chapter.

The main disadvantage of the Monte Carlo method lies in its random nature: statistical uncertainties affects the obtained results, only way to reduce these uncertainties is to increase the sampled population and, hence, the computation time. But in some special cases, the statistical uncertainties may be reduced by using variance-reduction techniques [6].

2.1 Inverse-Transform Method

Inverse-transform method is the method in which the cumulative distribution function of given probability distribution function (PDF) $p(x)$, is a non-decreasing function of $x$ and therefore, it has an inverse function $P^{-1}(\xi)$. Here $\xi$ is random number uniformly distributed in the interval $(0, 1)$ and the variable $x$ defined by $x = P^{-1}(\xi)$ is randomly distributed in the interval $(x_{\text{min}}, x_{\text{max}})$ with PDF $p(x)$. The variable $x$ is randomly sampled by using the sampling (Eq. 2.1) for any value of $\xi$ in the interval $(0, 1)$. This procedure for random sampling is known as the inverse-transform method [18].
The inverse-transform technique can be used to sample from uniform, exponential, and Wentzel distribution. Although inverse-transform is not the one of the fastest method for sampling from a distribution because it requires number of comparisons and a close form of equation \( p(x) \) but on the other hand it has some important features that makes it attractive:

- Inverse-transform techniques can be used for sampling from conditional distribution
- Inverse transform method preserves monotonicity and correlation which helps variance reduction methods, generating truncated distributions and getting ordered statistics.

### 2.2 Rejection Method

Rejection method is another kind of method in which random sampling from a certain distribution function [other than \( p(x) \)] and then subjecting that function to a random test to decide whether it will be rejected or accepted for use. This method is one of the general techniques for sampling from any PDF.
Figure 2.2 is the graphical explanation of rejection method algorithms. Consider that random values of $x$ are generated from PDF $\pi(x)$ using inverse transform method or any other sampling method. Now we sample a random values of $y$ for each sampled value of $x$ distributed in the interval $(0, C\pi(x))$, where $C$ is positive constant. Now, consider that the distribution $\pi(x)$ is such that $C\pi(x) \geq p(x)$ for some $C > 0$ and the generated random points $(x, y)$ uniformly distributed in region $A$ of the plane limited by $x$-axis and the curve $y = C\pi(x)$. If we accept points with $y \leq p(x)$ and reject points with $y > p(x)$ then we can get points uniformly distributed in the region between the curve $y = p(x)$ and $x$-axis and hence, the resulting $x$-coordinate is distributed according to $p(x)$.[18]

The rejection method can be represented by PDF $p(x)$ as

$$p(x) = C\pi(x)r(x)$$

(2.2)

where $C$ is a positive constant, $r(x)$ is confined as $0 \leq r(x) \leq 1$ and the PDF $\pi(x)$ can be sampled by inverse-transform method.

The algorithm for using rejection method for sampling from $p(x)$ proceeds as follows:

- Step 1 - Generate a random value of $x$ from $\pi(x)$.
- Step 2 - Generate a random number $\xi$.
- Step 3 - If $\xi > r(x)$, go to Step 1.
- Step 4 - Deliver $x$.

Advantage of using rejection method is based on the fact that a certain loss of efficiency is largely compensated because it enable us to sample $x$ from $\pi(x)$ instead of $p(x)$. The rejection method
requires the generation of several random number \(\xi\) to sample each \(x\) value, i.e. the one of the major disadvantage of using this method.

2.3 Composition Method

Composition method is a technique in which two-dimensional random variable can be generated by using single-variable sampling method. This technique is also true for multivariate distribution because \(n\)-dimensional PDF can be expressed as product of single-variable marginal distribution and a conditional PDF of \((n - 1)\) dimension.

Let’s consider a two-dimensional \((x, y)\) random variable with joint probability distribution function \(p(x, y)\) having marginal PDF \(q(y)\) and Conditional PDF \(p(x \mid y)\). The bivariate distribution of \(p(x, y)\) can be written as

\[
p(x, y) = q(y)p(x \mid y)
\]  

(2.3)

From the definition of marginal PDF of \(x\),

\[
q(x) \equiv \int p(x, y)dy = \int q(y)p(x \mid y)dy
\]  

(2.4)

Now we can sample \(y\) from \(q(y)\) and \(x\) from \(p(x \mid y)\) in order to generate random point \((x, y)\) from \(p(x, y)\). Note that generated values of \(x\) are distributed according to \(q(x)\). This is the basic idea used for sampling in composition method, which are applicable when \(p(x)\) (distribution to be simulated), is a mixture of several PDFs.

Nowadays combination of composition, rejection and inverse-transform methods is used to enhance the ability of simulation code in order to do sampling faster and to devise exact sampling algorithms for virtually any single or multivariate PDF[18].
Chapter 3

Photon Interactions

Photons of energy $E$ interact with atoms of element ($Z$ – atomic number) and depending on the level of energy, photons undergo different kinds of interactions. We will consider Photons of energy range starting from 1 KeV to 20 MeV, where most dominant interactions are Photoelectric absorption, Compton scattering, Pair-production and Rayleigh scattering. Other interactions such as Photo-nuclear absorption is not taken under consideration because its probability of occurrence is very small and it contributes only $(5 - 10\%)$ of total DCS and occurs in energy range from 5 MeV to 40 MeV [13].

We assume that the considered photons are non-polarized and molecular cross section for a interaction is approximated by the sum of atomic cross sections of all the atoms in molecule (additive rule), this approximation is valid as long as an atom is not appreciably distorted by molecular binding.

![Figure 3.1: Basic Photon Interactions](image)

Figure 3.1: Basic Photon Interactions
Moreover, we will consider only Photoelectric absorption, Compton scattering and Pair production for simulation purpose and in order to describe these interactions, we will determine their Total Cross Sections.

### 3.1 Path Length And Interacting Medium

Gamma ray photons originates from TGFs and TLEs passes through composite medium i.e. atmosphere with variable density depending on the the altitude and the different kinds of species like nitrogen, oxygen, argon, etc. interacts with photons. Using Total Cross Sections of different interactions, we can determine the path length $s$ for next interaction, given by (Ostgaard, 2008)[15]

$$s = -\frac{1}{a \cos(\alpha)} \log \left( 1 + \frac{\log(1 - \xi) a \cos(\alpha)}{\sigma \rho_0} \right)$$

(3.1)

Where path length $s$ in upward direction is symbolize plus symbol and downward direction is symbolize by minus, $\alpha$ is defined in Figure 3.2, $\xi$ is the random number, $\sigma$ is the total cross section, $\rho_z$ is the atmospheric density at starting altitude $Z_0$ given by Eq. [3.2] and $a$ is the exponential fit to atmospheric density shown in Figure 3.3.

$$\rho_z = \rho_0 e^{-aZ_0}$$

(3.2)

### 3.2 Compton (Incoherent) Scattering

Compton (Incoherent) scattering is a type of scattering in which a (depolarized) photon of energy $E$ collides in-elastically with an atomic electron. As a result of this interaction electron absorbs the incident photon and re-emits secondary photon of energy $E_s$ (lower than incident photon energy $E$) in the direction $\Omega = (\theta, \phi)$ relative to the direction of incident photon. After this interaction, electron itself recoils with kinetic energy $E_e$ in the direction $\Omega_e = (\theta_e, \phi_e)$.  

---

1. Appendix 4.1 Tracing A Path
Compton scattering events describe by total cross section that can be obtained by two different consideration. In first consideration, total cross section obtained by considering electrons are free at rest given by Klein and Nishina [1]. In second consideration, total cross section obtained by considering bound atomic electron with binding energy $U_i$ given by Hubbel [2].

3.2.1 Free Electron At Rest

In first consideration in which scattering caused by free electrons at rest, the conservation of energy and momentum will lead to following relation between “fractional energy” $\tau$ ($= E_s/E$) of secondary photon and emission angle of secondary photon and electron:

$$\cos \theta = \frac{1}{\kappa} \left( \kappa + 1 - \frac{1}{\tau} \right)$$  \hspace{1cm} (3.3)

$$E_s = \frac{E}{1 + \kappa (1 - \cos \theta)}$$  \hspace{1cm} (3.4)

$$E_e = E - E_s$$ \hspace{1cm} (3.5)

$$\cos \theta_e = (\kappa + 1) \left( \frac{1 - \tau}{\kappa [2 + \kappa (1 - \tau)]} \right)^{\frac{1}{2}}$$  \hspace{1cm} (3.6)

Where $\kappa = E/m_e c^2$ is photon energy in terms of electron rest energy and

$E = \text{initial photon energy (eV)}$

$m_e = \text{electron rest mass (Kg)}$
Figure 3.4: Nitrogen angular probability for Compton scattering for energies from 15 KeV to 18 MeV obtained by using expression (3.9). Resulting curve shows that photons having high energy has very high probability to get scattered in forward direction and photons having low energy can almost equal probability of getting scattered in forward or backward direction.

\[ c = \text{speed of light (m/s)} \]

Notice that minimum and maximum value of fraction energy \( \tau \) of secondary photon will appear at \( \theta = 0 \) (forward scattered) and \( \theta = \pi \) (backward scatter):

\[ \tau_{\text{min}} = \frac{1}{1 + 2\kappa} \quad \tau_{\text{max}} = 1 \quad (3.7) \]

The angular Differential Cross-Section (DCS) per electron, considering target electron to be free and at rest is given by Klein-Nishina formula [1]

\[ \frac{d\sigma_{KN}(\theta)}{d\Omega} = \frac{r_e^2}{2} \left[ 1 + \kappa(1 - \cos \theta) \right]^{-2} \times \left[ 1 + \cos^2 \theta + \frac{\kappa^2 (1 - \cos \theta)^2}{1 + \kappa(1 - \cos \theta)} \right] \quad (3.8) \]

Integrating expression (3.8) over solid angle (\( \Omega \)) will give us DCS per electron in terms of scattered angle (\( \theta \)),

\[ \frac{d\sigma_{KN}(\theta)}{d\theta} = \pi r_e^2 \left[ 1 + \kappa(1 - \cos \theta) \right]^{-2} \times \left[ 1 + \cos^2 \theta + \frac{\kappa^2 (1 - \cos \theta)^2}{1 + \kappa(1 - \cos \theta)} \right] \quad (3.9) \]

For the simulation purposes it will be convenient to express expression (3.8) in terms of secondary photon “fractional energy” (\( \tau \)) and scattered energy \( E_s \).
Figure 3.5: Oxygen angular probability for Compton scattering for energies from 15 KeV to 18 MeV obtained by using expression (3.9).

\[
\frac{d\sigma_{KN}}{d\tau} = 2\pi \frac{d\sigma(\theta) d(cos \theta)}{d\Omega} \frac{d\tau}{d\sigma_{KN}} = \pi r_e^2 \kappa^3 \left( \frac{1}{\tau^2} + \frac{\kappa^2 - 2\kappa - 2}{\tau} + (2\kappa + 1) + \kappa^2 \tau \right) \]

(3.11)

\[
\frac{d\sigma_{KN}}{dE_s} = \frac{\pi r_e^2}{\kappa^3} E \left( \frac{1}{\tau^2} + \frac{\kappa^2 - 2\kappa - 2}{\tau} + (2\kappa + 1) + \kappa^2 \tau \right) \]

(3.12)
Figure 3.6: Energy Differential cross section for Incoherent (Compton) scattering of 1 keV, 100 keV and 1000 keV photons of nitrogen and oxygen atoms. Blue (upper) curve obtained for free electron assumption by using expression (3.12) and red (lower) curve is obtained for bound electron assumption by using expression (3.29) with analytical scattering function given by expression (3.31). It shows energy DCS (differential cross sections) for different energies as a function of “fractional energy”. It clearly indicated that energy differential cross sections have comparable difference at low energy but this difference decreases with increasing energy. when “fractional energy” $\tau$ tends to unity Energy DCS for free electron assumptions reaches a finite value but on the other hand energy DCS for bound electron assumption tends to zero due to effect of binding [5].

Integrating expression (3.11) over fractional energy $\tau$ will give us famous Klein-Nishina total cross section per target electron:

$$\sigma_{KN} = \int_{\tau_{min}}^{1} \frac{d\sigma_{KN}}{d\tau} d\tau$$  \hspace{1cm} (3.13)

$$\sigma_{KN} = \pi r_e^2 \left( \frac{1}{\kappa^2} + \frac{\kappa^2 - 2\kappa - \frac{2}{\kappa^3}}{\kappa^3 \ln (2\kappa + 1) + \frac{2(\kappa + 1)}{(1 + 2\kappa)^2}} \right)$$  \hspace{1cm} (3.14)
Figure 3.7: Total cross section for Compton scattering of photons by nitrogen and oxygen atoms as a function of energy $E$. Dotted and dash-dot curve represents total cross sections for oxygen and nitrogen, considering electron is free and at rest given by expression (3.13). Continuous and dashed curve represents total cross section for oxygen and nitrogen, considering electron is bound to an atom given by expression (3.32) with analytical scattering function (3.33). Cross sections obtained for bound electron are compared to Ref [11]. Difference between total cross section obtained for bound and unbound electron can be seen at low energy level because of effect of binding energy. At high energy this difference is almost negligible because effect of binding energy is very less in comparison to high energy photon.

### 3.2.1.1 Algorithm For Random Sampling

After some manipulation, according to Butcher and Messel [9], total differential cross-section given by expression (3.11) can be rewritten as:

$$
\frac{d\sigma_{KN}}{d\tau} = \frac{X_0 n \pi r_e^2}{\kappa} \left[ \frac{1}{\tau + \tilde{\tau}} \right] \left[ 1 - \frac{\tau \sin^2 \theta}{1 + \tau^2} \right] \propto f(\tau) g(\tau) \tag{3.15}
$$

$$
= \frac{X_0 n \pi r_e^2}{\kappa} [t_1 \pi_1 (\tau) + t_2 \pi_2 (\tau)] [T(\tau)] \tag{3.16}
$$

where $X_0$ [m] is radiation length and $n$ [electron/m$^3$] is electron density,

$$
t_1 = \ln (1 + 2\kappa), \quad t_2 = \frac{2\kappa (1 + 2\kappa)}{(1 + 2\kappa)^2} \tag{3.17}
$$
\[ \pi_1 = \frac{1}{\ln(1 + 2\kappa)} \frac{1}{\tau}, \quad \pi_2 = \frac{(1 + 2\kappa)^2}{2\kappa(1 + \kappa)} \]  

(3.18)

and

\[ T(\tau) = g(\tau) = \left[1 - \frac{\tau \sin^2 \theta}{1 + \tau^2}\right] \]  

(3.19)

In expression (3.15), \( f(\tau) \) can be randomly sampled in interval \([\tau_{min}, 1]\), using \( g(\tau) \) as a valid rejection function.

In order to determine rejection function \( T(\tau) \), we need to get \( \sin^2 \theta \). Let

\[ L = \frac{(1 - \tau)}{\kappa \tau} \]  

(3.20)

Then using expression (3.3), we have

\[ \cos \theta = \frac{1}{\kappa} \left(\kappa + 1 - \frac{1}{\tau}\right) = 1 - L \]  

(3.21)

Therefore,

\[ \sin^2 \theta = 1 - \cos^2 \theta = (1 - \cos \theta)(1 + \cos \theta) = L(2 - L) \]  

(3.22)

3.2.1.2 Steps for sampling random values of “Fractional Energy” \( \tau \)

- Step 1 - Calculate all the parameters \((\kappa, \tau_{min}, t_1, t_2)\) depends on initial photon energy \( E \), but do not calculate “fractional energy” \( \tau \).

- Step 2 - Generate three random numbers \( r_1, r_2, r_3 \).

- Step 3 - According to point probability value of integer \( i \) \((1, 2)\) is selected as

\[ p(1) = \frac{t_1}{t_1 + t_2}, \quad p(2) = \frac{t_1}{t_1 + t_2} \]  

(3.23)

If \( r_1 < p(1) \) then \( i = 1 \) and

\[ \tau = \tau_{min} r_2 \]  

(3.24)

otherwise \( i = 2 \) and

\[ \tau = \tau_{min} + (1 - \tau_{min}) \tau' \]  

(3.25)

Where \( \tau' \) is determined

\[ i f \kappa \geq (1 + \kappa)r_2 \quad \tau' = \max(r_3, r_4) \]  

(3.26)

\[ o t h e r w i s e \quad \tau' = r_3 \]  

(3.27)

Here in expression (3.24) and (3.25) values of \( \tau \) determined using inverse transform method and \( \tau_{min} \) is defined by equation (3.3).

- Step 4 - Determine \( L \) from the expression (3.20) and the rejection function \( g(\tau) \) from the expression (3.19) and (3.22). If \( r_4 < g(\tau) \), then reject the value of \( \tau \) and go to Step 2.
• Step 5 - The value of $\tau$ is accepted.

After sampling the value of “fractional energy” $\tau$, the energy of secondary photon, energy of recoiled electron, scattering angle of secondary photon and electron can be determined using expression (3.4), (3.5), (3.3) and (3.6).

### 3.2.2 Bound Electron

In second consideration scattering caused by electron bound to an atom with binding energy $U_i$ and in this case differential cross-section (DCS) can be obtained by introducing effect of binding effect by means of incoherent scattering function $S(q, Z)$ [2]. In this consideration scattering can be only possible if the energy transfer $E - E_s$ is larger than $U_i$ ionization energy of shell under consideration. The Compton DCS in this case can be expressed in terms of “fractional energy” $\tau$ and scattered secondary photon energy $E_s$ as:

$$
\frac{d\sigma_{in}}{d\tau} = \frac{d\sigma_{KN}}{d\tau} S(q, Z) \tag{3.28}
$$

$$
\frac{d\sigma_{in}}{dE_s} = \frac{\pi \kappa^2}{\kappa^2} E \left( \frac{1}{\tau^2} + \frac{\kappa^2 - 2\kappa - 2}{\tau} + (2\kappa + 1) + \kappa^2 \tau \right) S(q, Z) \tag{3.29}
$$

Where $Z$ is atomic number of element and $q$ is momentum transfer to the electron, expressed as

$$
q^2 = \left( E^2 + E_s^2 - 2EE_s \cos(\theta) \right) / c^2 \tag{3.30}
$$

$$
= (m_e c)^2 \kappa \left[ 2 + \kappa - 2\kappa \left( 1 + \kappa \right) + \tau^2 \kappa \right] \tag{3.31}
$$

Finally, integrating expression (3.28) over fractional energy $\tau$ will lead to expression for total incoherent scattering cross-section per atom:

$$
\sigma_{in} = \int_{\tau_{min}}^{1} \frac{d\sigma_{KN}}{d\tau} S(q, Z) d\tau \tag{3.32}
$$

For the purpose of simulation it will be convenient to use approximate analytical incoherent scattering function $S(x, Z)$ given in J.Boro [4] instead of using incoherent scattering function $S(q, Z)$. The analytical scattering function can be expressed as:

$$
S(x, Z) = Z \left( 1 + \frac{b_1 x^2 + b_2 x^3 + b_3 x^4}{1 + b_4 x^2 + b_5 x^4} \right) \tag{3.33}
$$

where $b_1, b_2, b_3, b_4$ and $b_5$ are the $Z$ dependent parameters given in table by J.Boro [4] and $x$ is dimensionless variable and can be expressed as:

$$
x = \frac{q 10^{-8} [cm]}{4\pi h} = 20.6074 \frac{q}{m_e c} \tag{3.34}
$$

Where $q$ is magnitude of momentum transferred vector defined by equation (3.30)

In this case, energy of scattered photon, scattering angle of photon and electron can be determined by expression (3.4), (3.3) and (3.6) but due to binding effect, scattered energy of electron will decrease and can be expressed as:

$$
E_e = E - E_s - U_i \tag{3.35}
$$
3.2.2.1 Algorithm For Random Sampling

In case 2, as we are considering binding effect by taking into account incoherent scattering function in Klein-Vanishing DCS. It will be simpler to use rejection method for random sampling in comparison of other sampling methods (Zerb, 1963[7]; Nelson, 1985[8]; Ozmuthlu, 1992[10]) because it is easily applicable for binding effects.

The probability distribution function of ratio of scattered photon energy and initial photon energy, i.e. “fractional energy” \( \tau \) is

\[
P_{in}(\tau) = \frac{1}{\sigma_{in}} \frac{d\sigma_{in}}{d\tau}
\]

\[
= \frac{1}{\sigma_{in}} \frac{d\sigma_{KN}}{d\tau} S(x, Z) \Theta(\tau - \tau_{min}) \Theta(1 - \tau)
\]

(3.36)

where \( x \) is dimensionless variable defined by expression (3.34) and \( \Theta(i) \) is Heaviside function (= 0 if \( i \leq 0 \), = 1, otherwise).

According to Nelson [8] after some manipulation equation (3.36) can be expressed as

\[
P_{in}(\tau) = [t_1 \pi_1(\tau) + t_2 \pi_2(\tau)]T(\tau)
\]

(3.37)

where

\[
t_1 = \ln(1 + 2\kappa), \quad t_2 = \frac{2\kappa(1 + 2\kappa)}{(1 + 2\kappa)^2}
\]

(3.38)

\[
\pi_1 = \frac{1}{\ln(1 + 2\kappa)}, \quad \pi_2 = \frac{(1 + 2\kappa)^2}{2\kappa(1 + \kappa)}
\]

(3.39)

and

\[
T(\tau) = \left\{ \frac{1}{\kappa^2 \tau(1 + \tau^2)} \right\} S(x, Z)
\]

(3.40)

Here \( T(\tau) \) on the LHS in equation (3.37) is a valid rejection function because on the RHS the function in braces is positive and less than unity, when value of \( \tau \) varies within interval \([\tau_{min}, 1]\), and equal to 1 at the end points of this interval while the another part of this equation i.e. \( \frac{S(x, Z)}{Z} \) is less then unity for any value of dimensionless variable \( x \). In the expression (3.37) the functions \( \pi_i \) (1, 2) are normalized within interval \([\tau_{min}, 1]\) and can be sampled very easily by applying inverse transform method and the values of \( \tau \) can be generated randomly by applying together combination of composition and rejection methods [6].

3.2.2.2 Steps for sampling random values of “Fractional Energy” \( \tau \)

- Step 1 - Calculate all the parameters \( \kappa, \tau_{min}, t_1, t_2 \) depends on initial photon energy \( E \), but do not calculate “fractional energy” \( \tau \).
- Step 2 - Generate three random numbers \( r_1, r_2, r_3 \).
- Step 3 - According to point probability value of integer \( i \) (1, 2) is selected as
\[ p(1) = \frac{t_1}{t_1 + t_2} \quad p(2) = \frac{t_1}{t_1 + t_2} \] (3.41)

If \( r_1 < p(1) \) then \( i = 1 \) and

\[ \tau = \tau_{\text{min}}^{i=2} \] (3.42)

otherwise \( i = 2 \) and

\[ \tau = \left[ \tau_{\text{min}}^{i=2} + r_2 \left( 1 - \tau_{\text{min}}^{i=2} \right) \right]^\frac{1}{2} \] (3.43)

Here in expression (3.41) and (3.42) values of \( \tau \) determined using inverse transform method and \( \tau_{\text{min}} \) is defined by equation (3.7).

- Step 4: If \( r_3 > T(\tau) \), reject the value of \( \tau \) and go to Step 1.
- Step 5: The value of \( \tau \) is accepted.

After sampling the value of “fractional energy” \( \tau \), the energy of secondary photon, energy of recoiled electron, scattering angle of secondary photon and electron can be determined using expression (3.4), (3.35), (3.3) and (3.6).

Knowing the fact that the direction of produced electron and secondary photon are not necessarily coplanar, therefore the azimuthal angles \( \phi \) are randomly sampled in the interval \((0, 2\pi)\).

### 3.3 Photoelectric Effect

In the Photoelectric effect, a photon of energy \( E_p \) interacts with an atom and if the energy of photon is greater then ionization energy of interacting atomic shell \( i \) then it will get absorbed and an electron is emitted with energy \( E_{pe}(= E_p - U_i) \), where \( U_i \) is ionization energy of shell \( i \) (= K, L, M...etc) but if photon interacts with outer shell then emitted electron energy \( E_{pe}(= E_p) \) and then ionized atom emits x rays and Auger electron in order to reach to its ground state. Photo-ionization occurs only if photon energy \( E_p \) is greater then ionization energy \( U_i \) of interacting shell, due to this fact absorption edges are observed in photoelectric cross section \( \sigma_{\text{ph}} \) [4].

The photoelectric effect is described by the photo-absorption cross sections obtained from Evaluated Photon Data Library - EPDL (LLNL,1997), available at Lawrence Livermore National Laboratory. The total photo-absorption cross sections \( (\sigma_{\text{ph}}) \) numerical data tables are obtained from EPDL [16] and shell wise cross sections \( (\sigma_{\text{ph},i}) \) \( (i = K, L1, L2, L3 \text{ and } M1 \text{ to } M5) \) numerical data tables for the elements \( (Z = 1 - 99) \) are based on Scoefilds theoretical calculation[18].

#### 3.3.1 Algorithm for Photoelectric Simulation

- Let’s assume that a photon of energy \( E_p \) interacts with an element \( Z \) and as a result of this interaction it gets absorbed in its atomic shell \( i \).

- This photoelectric absorption events is simulated in two steps.

- Step 1: First step is to identify, which shell \( i \) \( (K, L, M) \) got ionized, using probability distribution function. The discrete probability distribution function of the shell \( i \), where photon gets absorbed is expressed as:

\[ p(1) = \frac{t_1}{t_1 + t_2} \quad p(2) = \frac{t_1}{t_1 + t_2} \]
Figure 3.8: Total photoelectric cross section using data obtained from EPDL.

\[ p_i = \sigma_{ph,i}(Z, E)/\sigma_{ph}(Z, E) \]  

(3.44)

Where \( \sigma_{ph,i} \) is the ionization cross section of the shell \( i \) and \( \sigma_{ph} \) is total cross section of the atom.

- Step 2 - Second step is to identify from which outer shell’s electron will fall to fill the ionized shell \( i \) resulted in first step and determine the kind of emission: X-ray or Auger electron and its energy. Probability of ionization determined as:

\[ p_{outer} = 1 - p_K - p_{L1} - \ldots - p_{M5} \]  

(3.45)

In step one, if the ionization occurs in K, L or M- shell then energy of photo-electron is \( E_{pe} (= E_p - U_i) \) or else if ionization occurs in outer shell then energy of photo electron is same as incident photon energy \( E_{pe} (= E_p - U_i) \).

### 3.3.2 Steps for simulation of photo-electron

In photoelectric effect the direction of emitted electron relative to absorbed photon, is determined by polar and azimuthal angles \( \theta_e \) and \( \phi_e \) (Figure 3.1). The azimuthal angle is distributed in the interval \((0, 2\pi)\), but photon under consideration is assumed to be non polarized, hence, the photo-electron angular distribution is independent of azimuthal angle \( \phi_e \). The sampling of polar angle \( \theta_e \) of photo-electron is done, using K-shell cross section given by Sauter (1931) with hydrogenic electron wave functions.

Differential cross section (DCS) given by Sauter can be expressed as:

\[ \frac{d\sigma_{ph}}{d\Omega_e} = \alpha^4 r_e^2 \left( \frac{Z}{\kappa} \right)^5 \frac{1}{\gamma} \frac{\sin^2 \theta_e}{(1 - \beta \cos \theta_e)^4} \left[ 1 + \frac{1}{2} \gamma (\gamma - 1) (\gamma - 2)(1 - \beta \cos \theta_e) \right] \]  

(3.46)

where \( \alpha = e^2/\hbar c \) is the fine-structure constant, \( r_e \) is classical electron radius, and
Figure 3.9: Angular scattering probability for photo-electron versus initial photon energy from 50 KeV to 18 MeV obtained by using expression (3.48). Resulting curve shows that photons having high energy has very high probability to get scattered in forward direction and photons having low energy can almost equal probability of getting scattered in forward or backward direction.

\[
\gamma = 1 + \frac{E_{pe}}{m_e c^2} \quad \beta = \frac{\sqrt{E_e(E_e + 2m_e c^2)}}{E_e + m_e c^2}
\]  \hspace{1cm} (3.47)

Let us introduce a variable \( \nu = 1 - \cos \theta_e \), except for a normalization constant, the angular distribution can be rewritten as:

\[
p(\nu) = (2 - \nu) \left( \frac{1}{A + \nu} + \frac{1}{2} \beta \gamma (\gamma - 1)(\gamma - 2) \right) \frac{\nu}{(A + \nu)^3}
\]  \hspace{1cm} (3.48)

where \( A = \frac{1}{\beta} - 1 \)

Analytically the variable \( \nu \) can be sampled randomly.

The above expression (3.48) of \( p(\nu) \) can be factorized and rewritten as:

\[
p(\nu) = g(\nu) \pi(\nu)
\]  \hspace{1cm} (3.49)

with

\[
g(\nu) = (2 - \nu) \left( \frac{1}{A + \nu} + \frac{1}{2} \beta \gamma (\gamma - 1)(\gamma - 2) \right)
\]  \hspace{1cm} (3.50)

and

\[
\pi(\nu) = \frac{A(A + 2)^2}{2} \frac{\nu}{(A + \nu)^3}
\]  \hspace{1cm} (3.51)
The variable $\nu$ varies in the interval $(0, 2)$ and the function $g(\nu)$ is positive and reaches its maximum when $\nu = 0$, while the function $\pi(\nu)$ is positive and normalized to unity.

Using the inverse transform method, random values from probability distribution $\pi(\nu)$ can be obtained by using sampling formula:

$$\int_0^\nu \pi(\nu')d\nu' = \xi, \quad \text{(3.52)}$$

Analytical result of expression $(3.52)$ will give

$$\nu = \frac{2A}{(A + 2)^2 - 4\xi} \left[2\xi + (A + 2)\xi^2\right] \quad \text{(3.53)}$$

Using the rejection method random sampling from Sauter’s distribution can be done as follows:

1. Obtain the value of $\nu$ from $\pi(\nu)$ using expression $(3.53)$.
2. Generate random number $\xi$.
3. If $\xi g(0) > g(\nu)$, go to step (1).
4. The value of $\cos \theta_e = 1 - \nu$ is accepted.

Knowing the fact that the direction of produced photoelectron and incident photon are not necessarily coplanar, therefore the azimuthal angles $\phi$ of the photoelectron is randomly sampled in the interval $(0, 2\pi)$.

Hence, the energy of photoelectron and its scattering angle is determined and will be saved in history of simulation.

### 3.4 Electron-positron pair production

Pair production is the type of process in which photon of energy $E_{pp}$ interacts with a nucleus or an electron, which absorbs its energy and momentum and as a result of this interaction electron-positron pairs get created in order to conserve energy and momentum. If the interaction occurs in the field of nucleus, the threshold energy for pair production is $2m_e c^2$ else if interaction occurs in field of an electron (at rest) then threshold energy for pair production is $4m_e c^2$ and as a result of this interaction electron itself get recoiled with some kinetic energy; this process is also called “triplet production” because it shows three visible tracks (as in case of cloud chamber).

The Pair production event is described by their total pair production cross sections obtained from XCOM program. The total pair production cross sections ($\sigma_{pp}$) numerical data tables are obtained from XCOM program.

In order to simulate events of pair production, we use semi empirical formula given by Boro et al 1994a [12]. The differential cross section for an energy $E$ to create a electron positron pair including low-energy correction and high-energy radiative correction, which is based on Bohrs approximation is given by Bethe and Heitler and can be expressed as

$$\frac{d\sigma_{pp}}{d\epsilon} = r_e^2 \alpha Z (Z + \eta) C_r \frac{2}{3} \left[2 \left(\frac{1}{2} - \epsilon\right)^2 \phi_1(\epsilon) + \phi_2(\epsilon)\right] \quad \text{(3.54)}$$

where

$$\phi_1(\epsilon) = g_1(b) + g_0(\kappa) \quad \phi_2(\epsilon) = g_1(b) + g_0(\kappa) \quad \text{(3.55)}$$

with
Figure 3.10: Total pair production cross section obtained using data from XCOM program.

\[ g_1(b) = \frac{7}{3} - 2 \ln (1 + b^2) - 6b \arctan (b^{-1}) - b^2 \left[ 4 - 4b \arctan (b^{-1}) - 3 \ln (1 + b^{-2}) \right] \quad (3.56) \]

\[ g_2(b) = \frac{11}{6} - 2 \ln (1 + b^2) - 3b \arctan (b^{-1}) + \frac{1}{2}b^2 \left[ 4 - 4b \arctan (b^{-1}) - 3 \ln (1 + b^{-2}) \right] \quad (3.57) \]

\[ g_0(\kappa) = 4 \ln (Rm_c/\hbar) - 4f_c(Z) + F_0(\kappa, Z) \quad (3.58) \]

where

\[ b = (Rm_c/\hbar) \frac{1}{2} \kappa \frac{1}{\epsilon (1 - \epsilon)} \quad (3.59) \]

In Eq. (3.58) values of \((Rm_c/\hbar)\) are given in Table given by Hubbell [13].

\(C_r = 1.009\) is the high-energy limit given by Mork and Olsen’s radiative correction [13].

The correction factor \(F_0(\kappa, Z)\) has been determined in order to have total cross section obtained from expression given in Eq. (3.58) (with \(\eta = 0\)) coincides with Hubbell total cross-section for pair production in the field of nucleus [7]. Analytical expression obtained by inspection and numerical fitting can be expressed as

\[ F_0(\kappa, Z) = \left( -0.1774 - 12.10a + 11.18a^2 \right) (2/\kappa)^{\frac{1}{2}} + \left( 8.523 + 73.26 - 44.41a^2 \right) - \left( 13.52 + 121.1a - 96.41a^2 \right) (2/\kappa)^{\frac{3}{2}} + \left( 8.946 + 62.05a - 63.41a^2 \right) (2/\kappa)^2 \quad (3.60) \]
The “reduced energy” $\epsilon = (E_\gamma + E_{RE})/E$ is the fraction of photon energy taken away by electron and its minimum and maximum values are determined as

$$\epsilon_{\text{min}} = \frac{E_{RE}}{E} = \kappa^{-1} \quad \text{and} \quad \epsilon_{\text{min}} = 1 - \frac{E_{RE}}{E} = 1 - \kappa^{-1} \quad (3.61)$$

The functions $\phi_1$ and $\phi_2$ are positive for high atomic elements except at endpoints of $\epsilon$ values given by Eq (3.61). In order make functions consistent they are set equal to zero when they take negative values.

The triplet contribution $\eta$ depends on photon energy is determined by the expression

$$\eta = [1 - \exp(-\nu)] \eta_\infty \quad (3.62)$$

where

$$\nu = \frac{0.2840 - 1.909a \ln (4/\kappa) + (0.1095 - 0.2206a) \ln^2 (4/\kappa) + (0.02888 - 0.04269a) \ln^3 (4/\kappa) + (0.002527 - 0.002623a) \ln^4 (4/\kappa)}{(0.02888 - 0.04269a) \ln^3 (4/\kappa) + (0.002527 - 0.002623a) \ln^4 (4/\kappa)} \quad (3.63)$$

In Eq. (3.62) values of $\eta_\infty$ for elements $Z = 1-99$ are from Table given by Hubbell [13].

### 3.4.1 Algorithm for sampling of “electron reduced energy” $\epsilon$.

The differential cross section (DCS) given by Bethe-Heitler Eq (3.52) depends on incident photon energy and the kinetic energy of produced electron $E_\gamma = \epsilon E - E_{RE}$, so by taking cumulative distribution function of DCS the energy of produced electron can be directly sampled. After sampling the kinetic energy of electron, kinetic energy of positron can be sampled from the relation $E_+ = E - E_\gamma - 2E_{RE}$. One should notice that the Bethe-Heitler Total DCS takes into account pair production and triplet production as well but all these events are simulated as if they were pairs.
This approximation is justified because in triplet production, mean free path of incident photon is much larger than recoiling electron.

According to Eq. (3.61), electron reduced energy $\epsilon$ is distributed between $\epsilon_{\text{min}}$ and $\epsilon_{\text{max}}$, and its PDF (3.54) can be expressed as:

$$p_{pp} = 2 \left( \frac{1}{2} - \epsilon \right)^2 \phi_1 (\epsilon) + \phi_2 (\epsilon), \quad (3.64)$$

This distribution function is symmetrical about the point $\epsilon = \frac{1}{2}$ and the functions $\phi_1 (\epsilon)$ and $\phi_2 (\epsilon)$ are non-negative and will attain their maximum values at $\epsilon = \frac{1}{2}$.

The PDF in expression (3.64) can be rewritten as:

$$p_{pp} = u_1 U_1 (\epsilon) \pi_1 (\epsilon) + u_2 U_2 (\epsilon) \pi_2 (\epsilon) \quad (3.65)$$

with

$$u_1 = \frac{2}{3} \left( \frac{1}{2} - \frac{1}{\kappa} \right)^2 \phi_1 \left( \frac{1}{2} \right), \quad u_2 = \phi_1 \left( \frac{1}{2} \right), \quad (3.66)$$

$$\pi_1 (\epsilon) = \frac{1}{2} \left( \frac{1}{2} - \frac{1}{\kappa} \right)^{-3} \left( \frac{1}{2} - \epsilon \right)^2, \quad \pi_2 (\epsilon) = \frac{1}{2} \left( \frac{1}{2} - \frac{1}{\kappa} \right)^{-1} \quad (3.67)$$

and

$$U_1 (\epsilon) = \phi_1 (\epsilon) / \phi_1 \left( \frac{1}{2} \right), \quad U_2 (\epsilon) = \phi_2 (\epsilon) / \phi_2 \left( \frac{1}{2} \right). \quad (3.68)$$

Using the inverse-transform method, random values of $\epsilon$ can be sampled easily from the functions $\pi_i (\epsilon)$ normalized in the interval $[\epsilon_{\text{min}}, \epsilon_{\text{max}}]$. The functions $U_i (\epsilon)$ are positive and less than unity in between $\epsilon_{\text{min}}$ and $\epsilon_{\text{max}}$, i.e. they are valid rejection functions. Combination of composition and rejection methods is applied to sample random values of $\epsilon$ from the PDF given by expression (3.65).

### 3.4.2 Steps for random sampling of “electron reduced energy” $\epsilon$

- **Step 1** - Using discrete probabilities, value of integer $i (= 1, 2)$ is sampled

  $$p (1) = \frac{u_1}{u_1 + u_2} \quad \text{and} \quad p (2) = \frac{u_2}{u_1 + u_2} \quad (3.69)$$

- **Step 2** - Inverse-transform method is used to sample $\epsilon$ from $\pi_i (\epsilon)$

  If $i = 1$ then

  $$\epsilon = \frac{1}{2} + \left( \frac{1}{2} - \frac{1}{\kappa} \right) (2\xi - 1)^{\frac{3}{2}} \quad (3.70)$$

  else if $i = 2$ then

  $$\epsilon = \frac{1}{\kappa} + \left( \frac{1}{2} - \frac{1}{\kappa} \right) 2\xi \quad (3.71)$$

- **Step 3** - Generate a new random number $\xi$.

- **Step 4** - If $\xi > U_i (\epsilon)$, go to step 1.

- **Step 5** - Deliver $\epsilon$. 

27
3.4.3 Angular distribution of produced particles

The angular distribution of produced pairs sampled from the expression obtained from high-energy theory (Heitler, 1954; Motz, 1969)14.

\[ p(\cos \pm) = a (1 - \beta \pm \cos \theta \pm)^{-2} \quad (3.72) \]

where \( a \) is normalization constant and

\[ \beta \pm = \frac{\sqrt{E \pm (E \pm + 2ERE)}}{E \pm + 2ERE} \quad (3.73) \]

is the particle velocity in speed of light. Using the inverse-transfer method, random values of \( \cos \pm \) can be sampled using sampling formula

\[ \cos \theta \pm = \frac{2\xi - 1 + \beta \pm}{(2\xi - 1)\beta \pm + 1} \quad (3.74) \]

\[ \cos \theta \pm = \frac{2\xi - 1 + \beta \mp}{(2\xi - 1)\beta \mp + 1} \quad (3.75) \]

In expression (3.73) the energy of produced electron-positron pairs can be determined by using Eqs. (3.76), Eq. (3.77) and sampled electron reduced energy \( \epsilon \).

\[ E_- = \epsilon E - ERE \quad (3.76) \]

\[ E_+ = E - E_- - 2ERE \quad (3.77) \]

Knowing the fact that the direction of produced electron-positron pairs and incident photon are not necessarily coplanar, therefore the azimuthal angles \( \phi \pm \) of the electron and the positron are randomly sampled in the interval \((0, 2\pi)\).
Chapter 4

Conclusion and future work

Figure 4.1: Comparison of total cross sections of Compton scattering, photoelectric effect and pair production.

The outputs of simulation code are observed step by step as per schematic shown in Figure 5.1. In first step, simulation code determines total cross sections shown in Figure 4.1. Obtained total cross sections were compared to XCOM: Photon Cross Sections Database maintained by NIST (National Institute of Standards and Technology) and results are found similar to XCOM database.

Comparison of total cross sections shows that probability of occurrence of photoelectric scattering is almost twice of probability of compton scattering for the photons of energy below 10 KeV and then probability reduces with increasing energy. After 100 KeV chances of occurrence of photoelectric events decreases sharply in comparison of dominant compton scattering events, which remains stable through whole energy range. As energy range increases further in MeV, we can also observe
pair-production events but still Compton scattering remains dominant while photoelectric events chances decrease further and become the least dominant event in energy range of MeV.

In second step of simulation, path length for next interaction is determined by expression (3.1) and photon is moved to the site of next interaction. Path length for most of the photon found within limit of $10^{-100}$ km, satisfactory enough as per mission requirement because normally TLEs occurs in this range of altitude. After knowing the path length and kind of interaction photon will undergo, its angular scattering probability is determined and compared with Ostgaard[15] and PENELOPE, 2008[18] to make sure that code is calculating correct scattering angles.

Results of Energy Differential cross section shown in Figure 3.6 shows that energy taken away by secondary photon in Compton scattering is same for bound and unbound model for energy range above 100 KeV. This fact allows us to use unbound electron model in high energy domain and get same results that we got from bound model, with considerably less time for simulation.

![Graph showing distribution of escaped photons and initial number of photons](image)

Figure 4.2: Plot showing distribution of escaped photons (blue curve) and initial number of photons (brown curve) in energy range from 10 - 100 KeV at 30 Km altitude.

\[\text{Figure 4.2: Plot showing distribution of escaped photons (blue curve) and initial number of photons (brown curve) in energy range from 10 - 100 KeV at 30 Km altitude.} \]

\[\text{1Shown in Figure 3.4, 3.5, 3.9 and 3.11} \]
Figure 4.3: Plot showing distribution of escaped photons (blue, red and cyan curve) and initial number of photons (brown curve) in energy range from 100 - 1000 KeV at 30 Km altitude.

In last step, simulation codes checks whether photons were absorbed or escaped into the atmosphere. Plot in Figure 4.2 shows that distribution (blue curve) of number of photons escaped into the atmosphere is very less in comparison of distribution (brown curve) of initial number of photons in energy range from 10 -100 KeV at 30 Km altitude. This facts allows us to conclude that simulation codes gives correct results, as at lower energy most of the photons were absorbed by photoelectric absorption and very few escaped into the atmosphere (domination of photoelectric absorption in low energy range 10 - 100 KeV can be cross checked by Figure 4.1). In plot shown in Figure 4.3 shows that distribution of number of escaped secondary photons at higher energy (100-1000 KeV) is more than distribution (blue curve in Figure 4.2) of number of escaped photons at lower energy (10 - 100 KeV) at 30 Km altitude, this fact allows us to conclude that compton effect if dominated in higher energy range and as a result we see more secondary photons escaping into the atmosphere.

Plot in Figure 4.4 shows that distribution (blue, red and cyan curves) of number of photons escaped into the atmosphere from energy range 10 - 100 Km at 60 Km altitude is considerable in comparison of distribution (blue curve shown in figure 4.2) of escaped number of photons from energy range 10 - 100 Km at 30 Km altitude. This facts allows us to conclude that only photons without interactions (because initial photons energy and escaping photons energy is same) were escaped into to atmosphere and other photons were absorbed by photoelectric absorption. In plot shown in Figure 4.5 shows that distribution of number of escaped photons from energy range 100-1000 KeV at 60 Km altitude is higher than distribution (blue, red and cyan curves) of number of escaped photons (blue,red and cyan curves in Figure 4.4) at lower energy (10 - 100 KeV) at 60 Km altitude, this fact allows us to conclude that compton effect is dominated in higher energy range and this domination increases with increasing altitude.

In future simulation code will be modified further to combine photon transport model and electron transport model to see more complex simulation results and from next year a PhD candidate will continue this work.
Figure 4.4: Plot showing distribution of escaped photons (blue, red and cyan curve) and initial number of photons (brown curve) in energy range from 10 - 100 KeV at 60 Km altitude.

Figure 4.5: Plot showing distribution of escaped photons (blue, red and cyan curve) and initial number of photons (brown curve) in energy range from 100 - 1000 KeV at 60 Km altitude.
Chapter 5

Appendix

5.1 Tracing A Path

Gamma rays travel along straight line until they have an interaction and this path can be determined in the following way:

1. Change in position of coordinates and time can be written as,

\[ x = x_0 + \Omega_x S, \]
\[ y = y_0 + \Omega_y S, \]
\[ z = z_0 + \Omega_z S, \]
\[ t = t_0 + \frac{S}{\nu}, \]

where \( \nu \) is velocity of radiation and \( S \) is the distance to the next interaction with atomic electron.

Let's introduce one new term \( \lambda \), its the probability per unit path length of having an interaction.

As we are tracing a path in homogeneous medium, the probability that an interaction takes place after a flight through a distance greater than \( S \) is given by (Whitlock, 1986)[18]

\[ U(S) = \exp \left[ -\int_0^S \lambda(S')dS' \right] \quad (5.1) \]

\[ \lambda = \sigma \rho_{z1} \quad (5.2) \]

where \( z_1 = S\cos(\alpha) \) is starting altitude for density \( \rho_{z1} = \rho_0 e^{-a_z1} [m^{-3}] \), \( \sigma \) is total cross-section area in \([m^2]\). It gives dimension of \( \lambda \) in \([m^{-1}]\).

For simulation purpose we need to decide how to sample the next event. This can be done by equating a (cumulative) distribution function to uniform random variable and the expression [1] can be re-written as:

\[ U(S) = (1 - \xi) = \exp \left[ -\int_0^S \lambda(S')dS' \right] \quad (5.3) \]

Substituting value of \( \lambda \) from Eq. (83) into Eq. (84), we will get
\[ \log(1 - \xi) = \left[ - \int_0^S \sigma \rho_z dS' \right] \] (5.4)

Further substituting value of \( \rho_z = \rho_0 e^{-a \cdot z_1} \) into Eq. (85), we will get

\[ \log(1 - \xi) = \left[ - \int_0^S \sigma \rho_0 e^{-a \cdot z_1} dS' \right] \] (5.5)

\[ -\log(1 - \xi) = \sigma \rho_0 \left[ \int_0^S e^{-a \cdot S' \cos(\alpha)} dS' \right] \] (5.6)

\[ \frac{-\log(1 - \xi)}{\sigma \rho_0} = \frac{-1}{a \cdot \cos(\alpha)} \left[ e^{-a \cdot S' \cos(\alpha)} \right]_0^S \] (5.7)

\[ \frac{-\log(1 - \xi) \cdot a \cdot \cos(\alpha)}{\sigma \rho_0} = - \left[ e^{-a \cdot S \cos(\alpha)} \right]_0^S \] (5.8)

\[ \frac{\log(1 - \xi) \cdot a \cdot \cos(\alpha)}{\sigma \rho_0} = \left[ e^{-a \cdot S \cos(\alpha)} - 1 \right] \] (5.9)

\[ s \pm = \frac{-1}{a \cdot \cos(\alpha)} \log \left( 1 + \frac{\log(1 - \xi) \cdot a \cdot \cos(\alpha)}{\sigma \rho_0} \right) \] (5.10)

Where path length \( s \) in upward direction is symbolize by plus symbol and downward direction is symbolize by minus.

Path length \( s \) determined in Eq. 91 gives the distance between two successive interactions in a composite medium that contains mixture of gases (mainly oxygen and nitrogen) whose densities varies with changing altitude.
5.2 Schematic for Monte Carlo Simulation for Photon Transport

Figure 5.1: Scheme showing simulation of photon transport
Bibliography


Simulation Code in FORTRAN LANGUAGE For Transport Of Gamma -Rays Through Atmosphere Using Monte Carlo Method For The Taranis Mission Of CNES
Contents of Simulation Code

- **Main program calling all modules and subroutine**
- Module initialization
- Module collision
- Module constant geophysics
- Module constant physics
- Module constant simulation
- Module structure concentration
- Module structure cross section
- Module structure particle
- Module structure random number
- Module toolbox
program Tar Ga
  use initialisation
  ! use cross_section
  use const_geophys
  ! use const_phys
  use const_simul
  use concentration
  use collisions
  use particle
  use toolbox
  use random_nr
  implicit none

!!!!!!!!!!!!!!!!!!!!!! Here is our main program calling all subroutine!!!!!!!!!!
!!!!!!!!!!!!!! Parameters required to call suroutines init1 and init2 from module intialisation ..
real(double) :: E_insert, theta, ht
real(double) :: sigma_total1, s, aa, n_dens, pzi1, aa1
real(double), dimension(nb_ener) :: Sigma_total
!!!!!!!!&&&&&&&&&&&&&&&&&&&&& ___Parameters required to call suroutine call compton from moduke collisions
_________!!!!!!!!!!!!
real(double) :: theta_p, theta_e, Ee, Es, ui, E_posi, theta_posi
integer(kind=4) :: isp, isp_s
real(double), dimension(nb_ener) :: E_photon
!!!!!!!!&&&&&&&&&&&&&&&&&&&&& ___call compton a _________!!!!!!!!!!!!
real :: r(array)
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&Parameters required for energy flux
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
integer :: flu
real(double), dimension(nb_particles):: E_flux
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&Parameters required for counting flux
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
real(double), dimension(nb_ener) :: distri_E
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&& go to loop for discrete energy !!this is not required now
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&& counting flux &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
real(double) :: mat1, mat2, mat3
!real(double), dimension(nb_ener,nb_interect) :: matrix1, matrix2, matrix3
real(double), dimension(num_alttds,nb_ener) :: matrix1, matrix2, matrix3
integer :: m_i, m_j, m_k, last_j
real(double), dimension(nb_particles)::: last_theta
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&& photoelectic subshell &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
integer(kind=4) :: shellss, process, count_particle_es, count_particle_pp, count_particle_pe, count_particle_ab
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!! Open and save vertical and horizontal distance for next interaction!!!!!!
open (unit= 192, file = "z_r.txt")
open (unit= 194, file = "matrix1_mon100k.txt")
open (unit= 195, file = "matrix2_mon100k.txt")
open (unit= 196, file = "matrix3_mon100k.txt")
open (unit= 197, file = "matrix.txt")
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!! Sharing random number every where!!!!!!!!!!!!!!!!!!!!!!!!!!!!
call random_numbr()

!!! it will count number of particles
count_particle=0
count_particle_es=0
count_particle_pp=0
count_particle_pe=0
count_particle_ab=0
this subroutine initiates energy grid called E_photon and corresponding cross-section

```
call init(E_photon)
```

```
flu = 0
outer: do i_part=1,nb_particles

print*,i_part
    call altitude_s(i_part,flu)

inner: do i_inter=1,nb_interact

part(i_part)%nb_interaction=i_inter
    flu=flu+1
    if ((flu+n+50) .ge. array )then
        flu=0
        flu=flu+1
    call random_numbr()
end if

```

```

!!!!!! ----------------------------------!!!!!!!!!!
!!!!!! calling subroutine init2!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!! this subroutine initiates require initial energy (E_insert),
!!!!!! E_photon, ht-altitude and gives aa- slope, 
!!!!!!!n_dens-total number density  
!!!!!!ctt-n total cross section, isp_s - interating specie(oxygen or nitrogen)!!!!!!! 
!!!!!! It call MSIS,interpolation,calculate total cross section 
!!!!!!!and select interating specie based on probability!!
!!! print*,’ht’,ht
    !call collision2(aa,n_dens,sigma_total1,E_photon,isp,process,shellss,flu)

```

```

!!!!!! Choosing process, 1- based on algorihm and 2 is based on user

!!!!!!! ----------------------------------!!!!!!!!!!
! angle accepted between 0-85 deg and 95 to 180 deg!!!!!
    call angle_accept(i_part,i_inter)
!!!!!!! Calls subroutine path and gives S - distance to next interaction!!!!!!!
    call path(aa,n_dens,sigma_total1,part(i_part)%theta_es(i_inter),s,flu)

```

```

!!!!!!! ----------------------------------!!!!!!!!!!
!!!!!!! j number of interation and j+1 is next interation !!!!
!!! chk last_j=part(i_part)%nb_interaction
    last_j=part(i_part)%nb_interaction
    j=part(i_part)%nb_interaction+1

```

```

```

!!!!!!! ----------------------------------!!!!!!!!!!
!!!!!!! z_new and r_new are vertical and horizontal distance for new interaction !!!!
    call path_accept(i_part,i_inter,j,s)
```

```

```

```
part(i_part)%nb_interaction=i_inter 
part(i_part)%es1_last=part(i_part)%es1(part(i_part)%nb_interaction)

! print*, 'Particle-',i_part,'interaction-',i_inter 
! print*, 'initial altitude -',part(i_part)%z_path(1), 'verticle altitude-',part(i_part)%z_path(j) 
! print*, 'energy of particle-',part(i_part)%es1(i_inter), 'tot cross section-',sigma_total1 
! print*, 'path length-',s, 'initial angle-',part(i_part)%theta_es(i_inter) 
! print*, 'slopec-',aa, 'density-',n_dens 
! print*, 'specie-',name_species(isp), 'process-',name_proc(process)
! pause
end if
!!!!!!!!!!!!!!! ----------------------------------!!!!!!!!!!
if (process.eq.1) then
!!!!!! calling subroutine compton gives us Es-scattered energy (eV)of secondary photon and Ee-electron, 
!!!!!! theta_p,theta_e scattered theta of photon and electron in radians!!!
if (name_reac(model).eq. 'bound') then
    call compton(i_part,j,isp,ui,Es,flu)
    ui=E_p1
!!!!!!!!!!!!!!! ----------------------------------!!!!!!!!!!
!!!!!!!&&&&&&&&&&& for free electron...........
else if (name_reac(model).eq. 'unbound') then
    call compton_free(i_part,j,Es,flu)
    ui=E_p1
end if
else if (process.eq.2) then
    count_particle_pe=count_particle_pe+1
    call pe(i_part,j,shellss,isp,flu)
!last_theta(i_part)=acos((((part(i_part)%z_path(j-1))**2.)**(0.5))/(((((part(i_part)%r_path(j-1))**2.)+((part(i_part)%z_path(j-1))**2.)))**(0.5))))
! print*, 'Particle-',i_part,'interaction-',i_inter 
! print*, 'initial altitude -',part(i_part)%z_path(1), 'verticle altitude-',part(i_part)%z_path(j) 
! print*, 'energy of particle-',part(i_part)%es1(i_inter), 'tot cross section-',sigma_total1 
! print*, 'path length-',s, 'initial angle-',part(i_part)%theta_es(i_inter) 
! print*, 'slopec-',aa, 'density-',n_dens 
! print*, 'specie-',name_species(isp), 'process-',name_proc(process), 'Shell-',name_reac2(shellss)
! print*, 'scattered electron energy-',part(i_part)%ee1(j), 'scattered electron angle-',part(i_part)%theta_ee(j)
!
! pause
cycle outer
else if (process.eq.3) then
    count_particle_pp=count_particle_pp+1
    call pp(i_part,j,isp,flu)
!last_theta(i_part)=acos((((part(i_part)%z_path(j-1))**2.)**(0.5))/(((((part(i_part)%r_path(j-1))**2.)+((part(i_part)%z_path(j-1))**2.)))**(0.5))))
! print*, 'Particle-',i_part,'interaction-',i_inter 
! print*, 'initial altitude -',part(i_part)%z_path(1), 'verticle altitude-',part(i_part)%z_path(j) 
! print*, 'energy of particle-',part(i_part)%es1(i_inter), 'tot cross section-',sigma_total1 
! print*, 'path length-',s, 'initial angle-',part(i_part)%theta_es(i_inter) 
! print*, 'slopec-',aa, 'density-',n_dens 
! print*, 'specie-',name_species(isp), 'process-',name_proc(process)
! print*, 'scattered electron energy-',part(i_part)%ee1(j), 'scattered electron angle-',part(i_part)%theta_ee(j)
! print*, 'scattered positron energy-',part(i_part)%ep1(j), 'scattered positron angle-',part(i_part)%theta_ep(j)
!
! pause
cycle outer
end if

!!!!!!!!!!!!!!!If new scattered energy more then binding energy then further interation!!!!!!!!!!
if ((part(i_part)%es1(j-1)-Es) .ge. ui) then

! print*, 'Particle-',i_part,'interaction-',i_inter
! print*, 'initial altitude-',part(i_part)%z_path(1), 'verticle altitude-',part(i_part)%z_path(j)
! print*, 'energy of particle-',part(i_part)%es1(i_inter), 'tot cross section-',sigma_totall
! print*, 'path length-',s, 'initial angle-',part(i_part)%theta_es(i_inter)
! print*, 'slope-',aa,'density-',n_dens
! print*, 'specie-',name_species(isp), 'prosess-', name_proc(process)
! print*, 'scattered electron energy-',part(i_part)%ee1(j), 'scattered electron angle-',part(i_part)%theta_ee(j)
! print*, 'scattered photon energy-',part(i_part)%es1(j), 'scattered photon angle-',part(i_part)%theta_es(j)
! pause

cycle inner

!!!!!!!!!!!!!!!If new scattered energy less then binding energy then new particle initialisation because last particle is absorbed!!!!!!!!!!
else if ((part(i_part)%es1(j-1)-Es) .lt. ui) then

!part(i_part)%es1(j-1)=exp(((log(E_max)-log(E_p1))*(rando_1%rando(flu+10)))+log(E_p1))
!part(i_part)%es1(j-1)=exp(((log(max_beam_energy)-log(min_beam_energy))*(rando_1%rando(flu+10)))+log(E_p1))
call altitude_s(i_part,flu)

count_particle_ab=count_particle_ab+1

end if

end do inner
end do outer

print*,part%es1_last

count_particle=count_particle_ab+count_particle_pe+count_particle_pp+count_particle_es

print*,count_particle,count_particle_ab,count_particle_pe,count_particle_pp,count_particle_es

!! !!!Termination of process if number of particle is more then limited particle !!!!!!
!! !!!Loop for scanning number of particles having escaping angles 20-29 deg, 40-49deg &
!! !!!and 60-69deg for diff altitude (m_i),energy,(m_j) and number of particle (m_k) in diff ranges
!!
m_i=1.
m_j=1.
matrix1(m_i,m_j)=0.0
matrix2(m_i,m_j)=0.0
matrix3(m_i,m_j)=0.0

WRITE(*,*),'Started writing files4'
do m_i=1,(num_alttds-1)
do m_j=1,(nb_ener-1)

mat1=COUNT(part%es1_last .ge. E_photon(m_j).and. part%es1_last .lt.E_photon(m_j+1))

.mat2=COUNT(part%es1_last .ge. E_photon(m_j).and. part%es1_last .lt.E_photon(m_j+1))

.mat3=COUNT(part%es1_last .ge. E_photon(m_j).and. part%es1_last .lt.E_photon(m_j+1))

.matrix1(m_i,m_j)=matrix1(m_i,m_j)+mat1
!

.matrix2(m_i,m_j)=matrix2(m_i,m_j)+mat2
!

.matrix3(m_i,m_j)=matrix3(m_i,m_j)+mat3
!

((pi/180.)*theta_2)/(E_photon(m_j+1)-E_photon(m_j))

''
matrix2(m_i,m_j)=matrix2(m_i,m_j)+mat2

mat3=count(part%es1_last .ge. E_photon(m_j).and. part%es1_last .lt.E_photon(m_j+1)&
. and. part%z_path(1) .gt. Altitudes(m_i) .and. part%z_path(1) .le. Altitudes(m_i+1)&
. and. last_theta .ge.((pi/180.)*theta_5).and. last_theta .le.((pi/180.)*theta_6))/(E_photon(m_j+1)-E_photon(m_j))

matrix3(m_i,m_j)=matrix3(m_i,m_j)+mat3

!!!Writing number of particle (matrix1(m_i,m_j) anr their escaping altitude range
and angle
WRITE(194,*)Altitudes(m_i),E_photon(m_j+1),matrix1(m_i,m_j)
WRITE(195,*)Altitudes(m_i),E_photon(m_j+1),matrix2(m_i,m_j)
WRITE(196,*)Altitudes(m_i),E_photon(m_j+1),matrix3(m_i,m_j)

end do
end do

WRITE(*,*)'Finished'

100 format(e11.4,1x,e11.4)
200 format(a,1x,e11.4,1x,a,1x,e11.4)

end program Tar Ga
Contents of Simulation Code

- Main program calling all modules and subroutine
- **Module initialization**
  - Module collision
  - Module constant geophysics
  - Module constant physics
  - Module constant simulation
  - Module structure concentration
  - Module structure cross section
  - Module structure particle
  - Module structure random number
  - Module toolbox
module initialisation
use const_geophys
use const_phys
use const_simul
use collisions
use cross_section
use concentration
use particle
use random_nr
implicit none
contains

! this subroutine initiates energy grid called E_photon and corresponding cross-section!!!!!
subroutine init1(E_photon)
integer(kind=4) :: isp,ipro,ireac,iener,atomic_nr,iii
real(double), dimension(nb_ener) :: E_photon,Sigma_total
real(double), dimension(nb_ener) :: Sigma_total_uo,Sigma_total_un,Sigma_total_o,Sigma_total_n
open (unit=92, file = "energy_&_ocs.txt")
open (unit=93, file = "energy_&_ncs.txt")
open (unit=94, file = "energy_&_uocs.txt")
open (unit=95, file = "energy_&_uncs.txt")
open (unit=96, file = "new.txt")
!!! simulating energy grid fron 1kev to 20 mev, i and iii is just number count
!!!E_p1 is from const_simul and E_photon is energy grid
i=1
iii=1
E_photon(i)=E_p1
do while(E_photon(i) .le. E_max)
    i=i+1
    iii=iii+1
    if (E_photon(i-1) .lt. E_p2) then
        E_photon(i)=E_photon(i-1)+E1!E1
    else if (E_photon(i-1) .ge. E_p2 .and. E_photon(i-1) .lt. E_p22) then
        E_photon(i)=E_photon(i-1)+E1!E2
    else if (E_photon(i-1) .ge. E_p22 .and. E_photon(i-1) .lt. E_p222) then
        E_photon(i)=E_photon(i-1)+E2!E3
    else if (E_photon(i-1) .ge. E_p222 .and. E_photon(i-1) .lt. E_p2222) then
        E_photon(i)=E_photon(i-1)+E2!E6
    else if (E_photon(i-1) .ge. E_p2222 .and. E_photon(i-1) .le. E_max) then
        E_photon(i)=E_photon(i-1)+E2!E7
    end if
end do
print*,iii
pause

! giving name to species and process !!!!

sp%name=name_species
sp(1)%pro(1)%reac(1)%name=name_reac(1)
sp(1)%pro(1)%reac(2)%name=name_reac(2)
sp(2)%pro(1)%reac(1)%name=name_reac(1)
sp(2)%pro(1)%reac(2)%name=name_reac(2)
sp(1)%pro(2)%reac%=name_reac2
sp(2)%pro(2)%reac%=name_reac2

!!!this loop will calculate cross-section for all species and for all process.
do isp=1,nb_spc
do ipro=1,nb_proc
   do ireac=1,nb_reac
      sp(isp)%pro(ipro)%reac(ireac)%e=E_photon
      do iener=1,nb_ener
         if(ipro .eq. 1 .and. ireac .le. 2 )then
            call cs(iener,isp,ipro,ireac)
         end if
      end do
      if(ipro .eq. 2 )then
         call pe_cs(isp,ipro,ireac)
      end if
      if(ipro .eq. 3 .and. ireac .eq. 1 )then
         call pp_cs(isp,ipro,ireac)
      end if
   end do
end do
end do

!!!Writing cross section for their corresponding energy
WRITE(92,100)(sp(1)%pro(1)%reac(1)%cs(i),sp(1)%pro(1)%reac(1)%e(i),i=1,nb_ener)
WRITE(93,100)(sp(2)%pro(1)%reac(1)%cs(i),sp(2)%pro(1)%reac(1)%e(i),i=1,nb_ener)
WRITE(94,100)(sp(1)%pro(1)%reac(2)%cs(i),sp(1)%pro(1)%reac(2)%e(i),i=1,nb_ener)
WRITE(95,100)(sp(2)%pro(1)%reac(2)%cs(i),sp(2)%pro(1)%reac(2)%e(i),i=1,nb_ener)

!!! cross section for compton effect, sp1 is o2 and sp2 is n2
!!! total cross section cm^2 into m^2 and also multiplying by two to calculate molecular cross section
sp(1)%pro(1)%reac(1)%cs=(sp(1)%pro(1)%reac(1)%cs)*cm_mtr*2.0
sp(2)%pro(1)%reac(1)%cs=(sp(2)%pro(1)%reac(1)%cs)*cm_mtr*2.0
sp(1)%pro(1)%reac(2)%cs=(sp(1)%pro(1)%reac(2)%cs)*cm_mtr*2.0
sp(2)%pro(1)%reac(2)%cs=(sp(2)%pro(1)%reac(2)%cs)*cm_mtr*2.0
return
100 format(e11.4,1x,e11.4)
end subroutine init1

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
This subroutine generates energy flux
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
subroutine init(E_photon)
real(double), dimension(nb_ener) :: E_photon
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!Parameters required for energy flux
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
integer :: flu,x_factor
real(double), dimension(nb_ener) :: E_flux

real(double), dimension(nb_ener) :: distri_E
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
open (unit= 193, file = "flux_ener.txt")
call init1(E_photon)
E_photon=E_photon

!!!!! Generating energy flux!!!!!!!!!!!!
flu=0
do flu=1,nb_particles
   if (power .eq. -1)then
      part(flu)%es1(1)=exp(((log(E_max)-log(E_p1))*(rando_1%rando(flu+10+n)))+log(E_p1))
      E_flux(flu)=part(flu)%es1(1)
   else if (power .lt. -1 .or. power .gt. -1 .and. power .ne. 0)then
      part(flu)%es1(1)=(((E_max)**(power+1.))-(E_p1)**(power+1.))*(rando_1%rando(flu+10+n))+((E_p1)**(power+1.))**(1./(power+1.))
      E_flux(flu)=part(flu)%es1(1)
   else if (power .eq. 0)then
      x_factor=1
      part(flu)%es1(1)=max Beam energy*(rando_1%rando(flu+n+x_factor))
E_flux(flu)=part(flu)%es1(1)
if (part(flu)%es1(1) .lt. min_beam_energy) then
  x_factor=x_factor+1
  go to 9
end if
end do

!!!!! Generating energy flux   end!!!!!!!!!!!!
!!!!! es1- scattered energy, ee1- electron energy, theta_es - angle of scattered photon in radians,
!!!!! theta ee- angle of scattered electron in radians, s_path- distance of next iteration in meters
!!!!! z_path- vertical distance of next iteration in meters !!!!!!!!
!!!!! r_path-horizontal distance of next iteration in meters !!!!!!!!

part(1:nb_particles)%ee1(1)=0.0
if (angle_distri .eq. 1) then
  part(1:nb_particles)%theta_es(1)=(pi/180.)*initial_angle
else if (angle_distri .eq. 2) then
  flu=0
do flu=1, nb_particles
    part(flu)%theta_es(1)=(pi/180.)*(rando_1%rando(flu))
  end do
end if

part(1:nb_particles)%theta_ee(1)=0.0
part(1:nb_particles)%s_path(1)=0.0
part(1:nb_particles)%z_path(1)=Altitudes(disc)
part(1:nb_particles)%r_path(1)=0.0
part(1:nb_particles)%theta_ep(1)=0.0
part(1:nb_particles)%ep1(1)=0.0
do i=1, (nb_ener-1)
  distri_E(i)=(count(E_flux.ge. E_photon(i) .and. E_flux.lt. E_photon(i+1)))/(E_photon(i+1)-E_photon(i))
  WRITE(193,100)(distri_E(i),(E_photon(i))
WRITE(193,100)distri_E(i), (E_photon(i)+E_photon(i+1)-E_photon(i))/(E_photon(i+1)-E_photon(i))
end do
print*, 'energy flux file written'
pause
100 format(e11.4,1x,e11.4)
return
end subroutine init
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!! This subroutine will give us altitude distribution for initial photons

subroutine altitude_s(i_part,flu)
implicit none
integer :: flu,i_part

5 flu=flu+1
if (alt_distri .eq. 2) then
  part(i_part)%z_path(1)=ALT*(rando_1%rando(flu+n))
  if ( part(i_part)%z_path(1) .lt. MIN_ALT) go to 5
else if (alt_distri .eq. 1) then
  part(i_part)%z_path(1)=ALT*(rando_1%rando(flu+n))
  if ( part(i_part)%z_path(1) .lt. 10000.) go to 5
  if ( part(i_part)%z_path(1) .ge. 10000. .and. part(i_part)%z_path(1) .lt. 15000.) then
    part(i_part)%z_path(1)=10000.
  else if( part(i_part)%z_path(1) .ge. 15000. .and. part(i_part)%z_path(1) .lt. 20000.) then
    part(i_part)%z_path(1)=20000.
end subroutine
else if (part(i_part)%z_path(1) .ge. 20000. .and. part(i_part)%z_path(1) .lt. 30000.) then
    part(i_part)%z_path(1) = 30000.
else if (part(i_part)%z_path(1) .ge. 30000. .and. part(i_part)%z_path(1) .lt. 40000.) then
    part(i_part)%z_path(1) = 40000.
else if (part(i_part)%z_path(1) .ge. 40000. .and. part(i_part)%z_path(1) .lt. 50000.) then
    part(i_part)%z_path(1) = 50000.
else if (part(i_part)%z_path(1) .ge. 50000. .and. part(i_part)%z_path(1) .lt. 60000.) then
    part(i_part)%z_path(1) = 60000.
end if
else if (alt_distri.eq. 3) then
    part(i_part)%z_path(1) = ALT_mono
    part(i_part)%z_path(1) = max ALT beam*(rando_1%rando(flu+n))
    if (part(i_part)%z_path(1) .lt. min ALT beam) go to 5
end if
return
end subroutine altitude_s
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Contents of Simulation Code

- Main program calling all modules and subroutine
- Module initialization
- **Module collision**
  - Module constant geophysics
  - Module constant physics
  - Module constant simulation
  - Module structure concentration
  - Module structure cross section
  - Module structure particle
  - Module structure random number
  - Module toolbox
This is included in order to generate random number

```fortran
include 'mkl_vsl.f90'

module collisions
    ! use initialisation
    use toolbox
    use const_geophys
    use const_phys
    use cross_section
    use const_simul
    use particle
    use random_nr
    USE MKL_VSL_TYPE
    USE MKL_VSL
    USE IFPORT

implicit none

contains

!! subroutine cs(iener,isp,ipro,ireac) is built to get total cross_section
!! for bound and unbound electron. It is based on J. Baro, (1993), ANALYTICAL CROSS SECTIONS FOR MONTE CARLO
!! SIMULATION OF PHOTON TRANSPORT, page 535, expression (23) for unbound case and expression (27) for bound
!! case but S(q,z) in expression (23) is replaced by S(x,Z) in expression (29).
!!
!! subroutine cs(iener,isp,ipro,ireac)
!! iener - number of energies
!! isp - number of species
!! ipro - number of processes
!! ireac - number of reaction, to understand see - module initialisation and subroutine init.

real(double) :: Sigma_total,IPEM,H,A11,B11,N11,F,z1,b1,b2,b3,b4,b5
integer(kind=4) :: isp,ipro,ireac,iener
integer(kind=4) :: i,ii
character*20 :: text

! Selection of atom (oxygen or nitrogen, etc.) for which total cross_section is going to be calculate!
! When text is equal to O2.txt then it will calculate oxygen cross section
! When text is equal to N2.txt then it will calculate nitrogen cross section
! (sp(isp)%name) - to understand this see - (type * specie * in) module cross_section,
! text=trim(sp(isp)%name)//'.txt'
! print*,text
! pause

! Below file opens the atomic parameters
! z1 - atomic number and b1,b2,b3,b4,b5 are from J. Baro, (1993), ANALYTICAL CROSS SECTIONS FOR MONTE CARLO
! SIMULATION OF PHOTON TRANSPORT, page 542
open(unit=96,file=text)
!z1
read (96,*)z1,b1,b2,b3,b4,b5
close(96)

! to understand see - module cross_section,
! this part is to initiate calculation of total cross section for bound case
if (sp(isp)%pro(ipro)%reac(ireac)%name .eq. 'bound') then
    ! IPEM is the ratio of initial photon energy and electron rest energy
    ! A11 - is the lower limit of integration expression (27) on page 535 [1] that
    ! correspond to tau minimum, from expression (20) on page 534 [1]
    ! B11 - is the upper of integration expression (27) on page 535 [1] that
    ! correspond to tau max, from expression (20) on page 534 [1]
    ! A11=(1./(1.+(IPEM*2.)))
    ! B11=1.
! N11 - It's the number to trapezoid that summed up together to give integration.
! When it increase it increases the accuracy of integration.
! To understand this - http://www.assakkaf.com/courses/ence202/lectures/fortran/lecture4.pdf page 18
N11 = 200.
H = (B11 - A11) / N11

! func_bound is the function written next to this subroutine
Sigma_total = (func_bound(A11, IPEM, z1, b1, b2, b3, b4, b5) + func_bound(B11, IPEM, z1, b1, b2, b3, b4, b5)) / 2.0

IF (N11 .GT. 1) THEN
F = A11
DO 1 I = 1, N11 - 1
F = F + H
Sigma_total = Sigma_total + func_bound(F, IPEM, z1, b1, b2, b3, b4, b5)
1 CONTINUE
END IF
Sigma_total = Sigma_total * H

! This is total cross section for bound case
sp(isp)%pro(ipro)%reac(ireac)%cs(iener) = Sigma_total
!
WRITE(*,*) sp(isp)%pro(ipro)%reac(ireac)%e(iener), sp(isp)%pro(ipro)%reac(ireac)%cs(iener)
! PAUSE
! WRITE(92,*),sp(1)%pro(1)%reac(1)%cs(iener),sp(1)%pro(1)%reac(1)%e(iener)
! WRITE(93,*),sp(2)%pro(1)%reac(1)%cs(iener),sp(2)%pro(1)%reac(1)%e(iener)

! To understand this see - module cross_section,
! This part is to initiate calculation of total cross section for unbound case
else if (sp(isp)%pro(ipro)%reac(ireac)%name .eq. 'unbound') then

IPEM = sp(isp)%pro(ipro)%reac(ireac)%e(iener) / ERE
A11 = (1. / 1. + (IPEM * 2.))
B11 = 1.
N11 = 200.
H = (B11 - A11) / N11

! func_unbound is the function written next to function bound
Sigma_total = (func_unbound(A11, IPEM, z1, b1, b2, b3, b4, b5) + func_unbound(B11, IPEM, z1, b1, b2, b3, b4, b5)) / 2.0

IF (N11 .GT. 1) THEN
F = A11
DO 2 ii = 1, N11 - 1
F = F + H
Sigma_total = Sigma_total + func_unbound(F, IPEM, z1, b1, b2, b3, b4, b5)
2 CONTINUE
END IF
Sigma_total = Sigma_total * H

! This is total cross section for unbound case
sp(isp)%pro(ipro)%reac(ireac)%cs(iener) = Sigma_total
!
WRITE(*,*) sp(isp)%pro(ipro)%reac(ireac)%e(iener), sp(isp)%pro(ipro)%reac(ireac)%cs(iener)
! PAUSE
! WRITE(94,*),sp(1)%pro(1)%reac(2)%cs(iener),sp(1)%pro(1)%reac(2)%e(iener)
! WRITE(95,*),sp(2)%pro(1)%reac(2)%cs(iener),sp(2)%pro(1)%reac(2)%e(iener)
end if
!
print*, sp(1)%pro(1)%reac(1)%cs(iener)
! close(92)
! close(93)
! close(94)
! close(95)
return
end subroutine cs

! Function for bound case

FUNCTION func_bound(F,IPEM1,z1,b1,b2,b3,b4,b5)

double precision :: z1,b1,b2,b3,b4,b5,yt,zt,ct,xt,syt,szt,stt,IPEM1,F,func_bound

yt=((3.14159*((2.8179E-13)**2))/(IPEM1**3)) !print*, "yt",yt
zt=((1/(F**2))+(((IPEM1**2)-(2*IPEM1-2))/F)+((2*IPEM1+1)+(IPEM1**2)*F)) !print*, "zt",zt
ct=(1-((1-F)/(IPEM1*F))) ! print*, "ct",ct
xt=(20.6074*IPEM1)*((2.*(1.-ct)**0.5)) ! print*, "xt",xt

syt=(1+(b1*(xt**2))+(b2*(xt**3))+b3*(xt**4)) !print*, "syt",sxt
szt=((1+b4*(xt**2)+b5*(xt**4))**2) !print*, "szt",sxt
stt=(syt/szt) !print*, "syt",sxt

S(x,Z) from expression (29) [J. Baro, (1993), ANALYTICAL CROSS SECTIONS FOR MONTE CARLO SIMULATION OF PHOTON TRANSPORT ,page 535]
sxt=(z1*(1-stt)) !print*, "sxt",sxt

!func_bound from expression (27) [J. Baro, (1993), ANALYTICAL CROSS SECTIONS FOR MONTE CARLO SIMULATION OF PHOTON TRANSPORT ,page 535]
func_bound= (yt*zt*sxt) !print*, "z",Z

RETURN
END function func_bound

FUNCTION func_unbound(F,IPEM1,z1,b1,b2,b3,b4,b5)

double precision :: z1,b1,b2,b3,b4,b5,yt,zt,ct,xt,syt,szt,stt,IPEM1,F,func_unbound

yt=((3.14159*((2.8179E-13)**2))/(IPEM1**3)) !print*, "yt",yt
zt=((1/(F**2))+(((IPEM1**2)-(2*IPEM1-2))/F)+((2*IPEM1+1)+(IPEM1**2)*F)) !print*, "zt",zt

!func_unbound from expression (27) [J. Baro, (1993), ANALYTICAL CROSS SECTIONS FOR MONTE CARLO SIMULATION OF PHOTON TRANSPORT ,page 535]
func_unbound= (yt*zt*z1) !print*, "z",Z

RETURN
END function func_unbound

subroutine pe_cs(isp,ipro,ireac) 

!!!! Subroutine to extract photoelectroic cross section
!!!! Subroutine to read cross section and sub shell cross section for photoionisation

!
integer(kind=4) :: isp,ipro,ml1,ml2,ireac
character*66 :: text
integer(kind=4) :: mat,mf,mt,ns,i1,i2,nr,np,n_n,nz,n1,n2
real(double) :: c1,c2
integer(kind=4),dimension(:),allocatable :: nbt,int
real(double),dimension(:),allocatable :: etemp,pekcs,temp,pel1cstemp,pel2cstemp,pel3cstemp
logical :: flg_log

!! opening the file containing cross section and subshell cross section

open(unit=10,file="petcs_check.dat")
open(unit=11,file="pekcs_check.dat")
open(unit=12,file="petcs.txt")
open(unit=13,file="pel1cs_check.dat")
open(unit=14,file="pel2cs_check.dat")
open(unit=21,file="findpel3cs_check.dat")
open(unit=16,file="npetcs_check.dat")
open(unit=17,file="npekcs_check.dat")
open(unit=18,file="npel1cs_check.dat")
open(unit=19,file="npel2cs_check.dat")
open(unit=20,file="npel3cs_check.dat")
open(unit=101,file="petcs_inter.dat")
open(unit=102,file="pekcs_inter.dat")
open(unit=103,file="pel1cs_inter.dat")
open(unit=104,file="pel2cs_inter.dat")
open(unit=105,file="pel3cs_inter.dat")
open(unit=106,file="npetcs_inter.dat")
open(unit=107,file="npekcs_inter.dat")
open(unit=108,file="npel1cs_inter.dat")
open(unit=109,file="npel2cs_inter.dat")
open(unit=110,file="npel3cs_inter.dat")

rewind(12)
!! for each species

write(*, '(a54)'), &
     '******************************************************************************'
write(*, '(a50,1x,a2)'),'Reading Photoelectric tot_cs for specie:',sp(isp)%name
sp(isp)%pro(ipro)%name=name_proc(ipro)
read(12,10)text,mat,mf,mt,ns

if(ireac .eq. 1) then
  mf1=23
  mt1=522
else if(ireac .eq. 2) then
  mf1=23
  mt1=534
else if(ireac .eq. 3) then
  mf1=23
  mt1=535
else if(ireac .eq. 4) then
  mf1=23
  mt1=536
else if(ireac .eq. 5) then
  mf1=23
  mt1=537
end if
!!! we look for the tot-cs mf= 23
  do while (mat/=100*z_an(isp) .or. mf/=mf1 .or. mt/=mt1)
    read(12,10)text,mat,mf,mt,ns
  end do
print*,text,'mat',mat,mf,mt,ns

!!! we read cross section
allocate(etemp(np),cstemp(np))
read(12,14)(etemp(n_n),cstemp(n_n),n_n=1,np)
!if (ireac .eq. 1) then
!! cs of the molecule is the sum of cs of its atom
  cstemp=2.*cstemp
!end if
!! conversion of cross section form barn to cm2
  cstemp=cstemp/(1.e24)
!! conversion of cs from cm^2 to m^2
  cstemp=cstemp*cm_mtr
!! we makr linear interpolation of the cs
flg_log=.false.
call interpt_d(flg_log,etemp,cstemp,sp(isp)%pro(ipro)%reac(ireac)%e&
    ,sp(isp)%pro(ipro)%reac(ireac)%cs)
!! below the lowest eveluated energy in petcs file, the cs is set to 0.0
where (sp(isp)%pro(ipro)%reac(ireac)%e < etemp(1))
  sp(isp)%pro(ipro)%reac(ireac)%cs_t=0.0
end where

!!! we write the cs in petcs_check.dat
if (isp .eq. 1)then
  if(ireac .eq. 1 )then
    write(101,100)(sp(1)%pro(ipro)%reac(1)%cs(i),sp(1)%pro(ipro)%reac(1)%e(i),i=1,nb_ener)
    write(10,')',sp(1)%pro(ipro)%name,'for specie',sp(isp)%name
    do i=1,np
      write(10,15),'cross-section=',cstemp(i),'energy=',etemp(i)
    end do
  else if(ireac .eq. 2 )then
    write(102,100)(sp(1)%pro(ipro)%reac(2)%cs(i),sp(1)%pro(ipro)%reac(2)%e(i),i=1,nb_ener)
    write(11,')',sp(1)%pro(ipro)%name,'for specie',sp(isp)%name
    do i=1,np
      write(11,15),'cross-section=',cstemp(i),'energy=',etemp(i)
    end do
  else if(ireac .eq. 3 )then
    write(103,100)(sp(1)%pro(ipro)%reac(3)%cs(i),sp(1)%pro(ipro)%reac(3)%e(i),i=1,nb_ener)
    write(13,')',sp(1)%pro(ipro)%name,'for specie',sp(isp)%name
    do i=1,np
      write(13,15),'cross-section=',cstemp(i),'energy=',etemp(i)
end do
else if (ireac .eq. 4) then
    write(104,100) (sp(1)%pro(ipro)%reac(4)%cs(i), sp(1)%pro(ipro)%reac(4)%e(i), i=1,nb_ener)
    write(14,*), ''
    write(14,16), sp(isp)%pro(ipro)%name, 'for specie', sp(isp)%name
    do i=1,np
        write(14,15), 'cross-section=', cstemp(i), 'energy=', etemp(i)
    end do
else if (ireac .eq. 5) then
    write(105,100) (sp(1)%pro(ipro)%reac(5)%cs(i), sp(1)%pro(ipro)%reac(5)%e(i), i=1,nb_ener)
    write(21,*), ''
    write(21,16), sp(isp)%pro(ipro)%name, 'for specie', sp(isp)%name
    do i=1,np
        write(21,15), 'cross-section=', cstemp(i), 'energy=', etemp(i)
    end do
end if
else if (isp .eq. 2) then
    if (ireac .eq. 1) then
        write(106,100) (sp(2)%pro(ipro)%reac(1)%cs(i), sp(2)%pro(ipro)%reac(1)%e(i), i=1,nb_ener)
        write(16,*), ''
        write(16,16), sp(isp)%pro(ipro)%name, 'for specie', sp(isp)%name
        do i=1,np
            write(16,15), 'cross-section=', cstemp(i), 'energy=', etemp(i)
        end do
    else if (ireac .eq. 2) then
        write(107,100) (sp(2)%pro(ipro)%reac(2)%cs(i), sp(2)%pro(ipro)%reac(2)%e(i), i=1,nb_ener)
        write(17,*), ''
        write(17,16), sp(isp)%pro(ipro)%name, 'for specie', sp(isp)%name
        do i=1,np
            write(17,15), 'cross-section=', cstemp(i), 'energy=', etemp(i)
        end do
    else if (ireac .eq. 3) then
        write(108,100) (sp(2)%pro(ipro)%reac(3)%cs(i), sp(2)%pro(ipro)%reac(3)%e(i), i=1,nb_ener)
        write(18,*), ''
        write(18,16), sp(isp)%pro(ipro)%name, 'for specie', sp(isp)%name
        do i=1,np
            write(18,15), 'cross-section=', cstemp(i), 'energy=', etemp(i)
        end do
    else if (ireac .eq. 4) then
        write(109,100) (sp(2)%pro(ipro)%reac(4)%cs(i), sp(2)%pro(ipro)%reac(4)%e(i), i=1,nb_ener)
        write(19,*), ''
        write(19,16), sp(isp)%pro(ipro)%name, 'for specie', sp(isp)%name
        do i=1,np
            write(19,15), 'cross-section=', cstemp(i), 'energy=', etemp(i)
        end do
    else if (ireac .eq. 5) then
        write(110,100) (sp(2)%pro(ipro)%reac(5)%cs(i), sp(2)%pro(ipro)%reac(5)%e(i), i=1,nb_ener)
        write(20,*), ''
        write(20,16), sp(isp)%pro(ipro)%name, 'for specie', sp(isp)%name
        do i=1,np
            write(20,15), 'cross-section=', cstemp(i), 'energy=', etemp(i)
        end do
    end if
end if

dallocate(nbt,int)
dallocate(etemp,cstemp)

!pause

10 format(a66,i4,i2,i3,i5)
12 format(2e11.0,4i11,i4,i2,i3,i5)
13 format(6i11)
14 format(6e11.0)
15 format(a7,e11.4,2x,a10,e11.4)
16 format(a4,a13,a2)
100 format(e11.4,1x,e11.4)

return

end subroutine pe_cs

!!!!!!!! Subroutine to extract pair production cross section
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

subroutine pp_cs(isp,ipro,ireac)
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!! subroutine to read pair production cross section and sub shell cross section for photoionisation

integer(kind=4) ::isp,ipro,mf1,mt1,ireac
character*66 ::text
integer(kind=4) ::mat,mf,mt,ns,i1,i2,nr,np,n_n,nz,n1,n2
real(double) ::c1,c2
integer(kind=4), dimension(:), allocatable ::nbt, int
real(double), dimension(:), allocatable ::etemp, pekcstemp, pekc, cstem, pel1cstemp, pel2cstemp, pel3cstemp
logical ::flg_log

!! opeing the file to write total pair production cross section

open(unit=22, file="pptcs_check.dat")
open(unit=122, file="pptcs_inter.dat")

open(unit=66, file="nppptcs_check.dat")
open(unit=160, file="nppptcs_inter.dat")

!! opeing the file containing cross section and subshell cross section
! open(unit=12, file="petcs.txt")
open(unit=12, file="pp_electron_cs.txt")
open(unit=13, file="npp_electron_cs.txt")

rewind (12)
np=80
! mf1=23
! mt1=515
! write(*,'(a54)'),&
! '&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&'
! write(*,'(a50,1x,a2)'),'Reading Pair production tot_cs for specie:', sp(isp)%name
! do while (mat/=100*z_an(isp) .or. mf/=mf1 .or. mt/=mt1)
! read(12,10)text,mat,mf,mt,ns
! end do
!
! write(*,'(a54)'),'Found total cross section'
! read(12,12)c1,c2,i1,i2,nr,np,mat,mf,mt,ns
! print*,c1,c2,i1,i2,'nr',nr,'np',np,mat,mf,mt,ns
! allocate(nbt(nr), int(nr))
! read(12,13),(nbt(n_n), int(n_n), n_n=1,nr)
! print*,(nbt(n_n), int(n_n), n_n=1,nr)
!!! we read cross section
 allocate(etemp(np), cstem(np))

! read(12,14)(etemp(n_n), cstem(n_n), n_n=1,np)
do i=1,np
 n_n=i
if (isp .eq. 1) then
 read(12,*),etemp(n_n), cstem(n_n)
else if (isp .eq. 2) then
 read(13,*),etemp(n_n), cstem(n_n)
end if
end do

!! converting energy into eV from Mev
 etemp=etemp*(10**6.)
!! conversion of cross section from barn to cm2
cstem=cstem/(1.242)
!! cs of the molecule is the sum of cs of its atom
cstemp=2.*cstemp

!! conversion of cs from barn to m^2
cstemp=cstemp*cm_mtr
!! we make linear interpolation of the cs
flg_log=.false.
call interpt_d(flg_log,etemp,cstemp,sp(isp)%pro(ipro)%reac(ireac)%e &
,sp(isp)%pro(ipro)%reac(ireac)%cs)

!! below the lowest evaluated energy in petcs file, the cs is set to 0.0
where (sp(isp)%pro(ipro)%reac(ireac)%e < etemp(1))
      sp(isp)%pro(ipro)%reac(ireac)%cs_t=0.0
end where
!! we write the cs in petcs_check.dat
if (isp.eq.1)then
  if (ireac.eq.1) then
    write(122,100)(sp(1)%pro(ipro)%reac(1)%cs(i),sp(1)%pro(ipro)%reac(1)%e(i),i=1,nb_ener)
    write(22,*),'','for specie',sp(isp)%name
do i=1,np
    write(10,15),'cross-section=',cstemp(i),'energy=',etemp(i)
  end do
end if
else if (isp.eq.2)then
  if (ireac.eq.1) then
    write(160,100)(sp(2)%pro(ipro)%reac(1)%cs(i),sp(2)%pro(ipro)%reac(1)%e(i),i=1,nb_ener)
    write(66,*),'','for specie',sp(isp)%name
do i=1,np
    write(16,15),'cross-section=',cstemp(i),'energy=',etemp(i)
  end do
end if
end if

10 format(a66,i4,i2,i3,i5)
12 format(2e11.0,i4,i2,i3,i5)
13 format(a61)
14 format(6i11)
15 format(a7,e11.4,2x,a10,e11.4)
16 format(a4,a13,a2)
100 format(e11.4,a10,e11.4)
return
end subroutine pp_cs

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!! This subroutine will give density(n_tot1),slope(aa) for path length and it will also
!! tell us kind of interaction(process), cross section(sigma_total1), interacting species(isp) and its
!! subshell(shellss)
subroutine collision2(aa,n_tot1,sigma_total1,E_photon,isp,process,shellss,flu)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!C Least square fitting of a straight line to data points
real ::loop,ddd(nb_conc),ddd11(nb_conc)
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!C Least square fitting of a straight line to data points
double precision ::sumx,sumy,sumsqx,sumxy,pzi,y,deno,slope,b,x
!conversions
double precision ::ALT_mtr,ddd6_mtr,ddd6_pmtr,ddd3_mtr,ddd4_mtr,loop_mtr ,i_mtr ,ht
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
double precision ::E_insert

double precision ::sigma_total1, aa,n_tot1,ratio_cs,ctt1
INTEGER, DIMENSION(1) :: AD1
integer(kind=4) :: ipro, n_n, n_nn, n_n1, n_nnl, model1
real(double), dimension(:,), allocatable :: cs_temp, ratio_cs1
!!! for energy grid
real(double), dimension(nb_ener) :: E_photon, Sigma_total

!!! selecting oxygen or nitrogen
integer(kind=4) :: isp, process, flu
!!! parameter for calculating probability for photoelectric subshells
double precision :: sigma_total1ss, p_outer
real(double), dimension(nb_ener) :: Sigma_totalss
integer(kind=4) :: n_nss, n_nns, nreacss, n_n1ss, shellss
real(double), dimension(:,), allocatable :: cs_tempss, ratio_cs1ss

!!!!!!!!!!!!!! Generating Random Numbers!!!!!!!!!!!!!!!!!!!!

!!! for random number

!!! ht-initial height is divided by 1000 to convert it into kilo-meters.
ht=part(i_part)%z_path(i_inter)/1000.

!!!! E_insert is the energy of interacting(i_inter) photon of that particle (i_part)
E_insert=part(i_part)%es1(i_inter)

!!!!!!!!!!!!! calling msis to determine air and its component densities for different altitudes
!open (unit= 99, file = "densities.txt")
sumx=0.0
sumy=0.0
sumsqx=0.0
sumxy=0.0
do loop=1.,ht
    call msis90(IYD,SEC,loop,GLAT,GLONG,STL,F107A,F107,AP,MASS,ddd,Tem)
    !To understand meaning of number dddxxx(1-8) see module const_simul, section msis
    !Here output of ddd(6) is in- g/cm^3 and ddd(1),ddd(2) are in- cm^-3, loop is in- KM
    !write(99,*)(ddd6,loop,ddd3,ddd4)
    !ddd6_mtr g/cm^3  into Kg/m^3  and for ddd3_mtr,ddd4_mtr 1/cm^3  into 1/m^3
    ddd6_mtr=ddd6*1e3
    ddd3_mtr=ddd3*1e6
    ddd4_mtr=ddd4*1e6
    ddd11(2)=ddd3_mtr
    ddd11(1)=ddd4_mtr
    ddd11(3)=ddd5*1e6
    ddd11(4)=ddd7*1e6
    ddd11(5)=ddd8*1e6
    ddd11(6)=ddd1*1e6
    ddd11(7)=ddd2*1e6
    ddd11(8)=ddd6_mtr

!!!!!!!!!!!!!!!!!!!! Km to meter
loop_mtr=loop*1e3

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! converting air density in Kg/m3 to particle per m3
! ddd6 pmtr=((ddd6_mtr*1e3)/28.84)*Na
n_totl=0.0
do i=1,nb_conc_use
    n_totl=n_totl+ddd11(i)
end do

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!C Least square fitting of a straight line to data points
!C Least square fitting of a straight line to data points
deno=log(sumqxy-sumx*sumx)/deno
b=(sumsqx*sumy-sumx*sumxy)/deno
!open (unit= 99, file = "fit_use.txt")
!open (unit= 100, file = "cd.txt")
do  i=1,ALT
   i_mtr=i
   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!
   y=slope*i_mtr+b
   x=exp(y)
!!! pzi is column density in Kg/m^2 and i_mtr is altitude in meter
end do

aa=-slope
pzi=((-((exp(b))/(slope)))*((slopes)(i_mtr)))
!!!!!!!!!!!!!!!! sigma total!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

AD1 = MINLOC(E_photon,E_photon .GT. E_insert )
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!! here we calculate (3 (process) * 2 (species)) cross sections, model - given in const_simul tell us
!!! weather its bound or unbound compton scattering
Sigma_total(AD1(1))=0.0
n_n=nb_sp*n*nb_proc
n_nn=0
allocate (cs_temp(n_n))
do  ipro=1,n*nb_proc
   do  i=1,nb_conc_use
      n_n=nn+n_n+1
      if(ipro .eq. 1)then
         model1=model
      else if(ipro .gt. 1)then
         model1=1
      end if
      Sigma_total(AD1(1))=((ddd11(i)/n_tot1)*
      ((sp(i)%qro(ipro)%reac(model1)%cs(AD1(1)-1))+(E_insert-E_photon(AD1(1)-1))&
      (((sp(i)%qro(ipro)%reac(model1)%cs(AD1(1)-1))&&
      -(sp(i)%qro(ipro)%reac(model1)%cs(AD1(1)-1)))/(E_photon(AD1(1))-E_photon(AD1(1)-1))))+Sigma_total(AD1(1))
      cs_temp(n_nn)=Sigma_total(AD1(1))
   end do
end do
!!!!!!!!!!!!!!!! Interpolation end!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!! Calculating point probablility
!!!! ratio cs1 id the ratio of interacting cross section to total cross section
sigma_total1=Sigma_total(AD1(1))
n_n=nb_sp*n*nb_proc
allocate (ratio_cs1(n_n1))
do  i=1,n_n1
   ratio_cs1(i)=cs_temp(i)/Sigma_total(AD1(1))
end do

!print*,Sigma_total(AD1(1))
!pause
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!! deciding which process and species ( oxygen or nitrogen) both in bound and
unbound!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!! process 1 is compton ,2 is photoelectric ,3 is pair production
!!!! isp .1 is O2 and  isp .2 is N2

if (rando_1%rando(flu) .le. ratio_cs1(1)) then
   isp=1
   process=1
ratio_cs=ratio_cs1(1)
else if ( rando_1%rando(flu) .gt. ratio_cs1(2) .and. rando_1%rando(flu) .le. ratio_cs1(3)) then
  isp=1
  process=2
  ratio_cs=ratio_cs1(3)
else if ( rando_1%rando(flu) .gt. ratio_cs1(4) .and. rando_1%rando(flu) .le. ratio_cs1(5)) then
  isp=1
  process=3
  ratio_cs=ratio_cs1(5)
else if ( (rando_1%rando(flu) .gt. ratio_cs1(1)) .and. rando_1%rando(flu) .le. ratio_cs1(2)) then
  isp=2
  process=1
  ratio_cs=ratio_cs1(2)
else if ( rando_1%rando(flu) .gt. ratio_cs1(3) .and. rando_1%rando(flu) .le. ratio_cs1(4)) then
  isp=2
  process=2
  ratio_cs=ratio_cs1(4)
else if ( rando_1%rando(flu) .gt. ratio_cs1(5) .and. rando_1%rando(flu) .le. ratio_cs1(6)) then
  isp=2
  process=3
  ratio_cs=ratio_cs1(6)
end if

flu=flu+1

if (process .eq. 2) then
  Sigma_totalss(AD1(1))=0.0
  n_nss=nb_pe_subshell
  n_nbss=0
  allocate (cs_tempss(n_nss))
  do ireacss=2,nb_reac
    do i=isp,isp
      Sigma_totalss(AD1(1))=((ddd11(i)/n_tot1)*&
      ((sp(i)%pro(2)%reac(ireacss)%cs(AD1(1)-1))+(E_insert-E_photon(AD1(1)-1)))&
      (((sp(i)%pro(2)%reac(ireacss)%cs(AD1(1)))&
      -(sp(i)%pro(2)%reac(ireacss)%cs(AD1(1)-1)))/(E_photon(AD1(1))-E_photon(AD1(1)-1))))
      cs_tempss(n_nbss)=Sigma_totalss(AD1(1))
    end do
  end do
end if

!!!!!!! Calculating point probability
sigma_totalss=Sigma_totalss(AD1(1))

allocate (ratio_cs1ss(n_n1ss))
do i=1,n_n1ss
  if (isp .eq. 1) then
    ratio_cs1ss(i)=cs_tempss(i)/(cs_temp(3)-cs_temp(2))
  else if (isp .eq. 2) then
    ratio_cs1ss(i)=cs_tempss(i)/(cs_temp(4)-cs_temp(3))
  end if
end do

p_outer=1-ratio_cs1ss(1)-ratio_cs1ss(2)-ratio_cs1ss(3)-ratio_cs1ss(4)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!deciding which shell and species ( oxygen or nitrogen) both in bound and unbound
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!! process 1 is compton ,2 is photoelectric ,3 is pair production

if (rando_1%rando(flu) .le.ratio_cs1ss(4) ) then
  shellss=5
else if (rando_1%rando(flu) .le. ratio_cs1ss(3) ) then
  shellss=4
else if (rando_1%rando(flu) .le. ratio_cs1ss(2) ) then
  shellss=3
else if (rando_1%rando(flu) .le. p_outer) then
  shellss=0
else if (rando_1%rando(flu).gt. ratio_cs1ss(2) .and. rando_1%rando(flu) .le. 1) then
  shellss=2
end if
flu=flu+1
end if
!!!!!----------------------------------------------------------
deallocate(cs_temp)
deallocate(ratio_cs1)

return
end subroutine collision2

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

subroutine angle_accept(i_part,i_inter)
imPLICIT none
integer :: i_inter,i_part

!!!! angle accepted between 0-85 deg and 95 to 180 deg!!!!!
  if(part(i_part)%theta_es(i_inter) .ge. ((pi/180.)*85.) .and. part(i_part)%theta_es
(i_inter) .le. (((pi/180.)*90.)) then
    part(i_part)%theta_es(i_inter) =((pi/180.)*85.)
  end if
  else if (part(i_part)%theta_es(i_inter) .gt. ((pi/180.)*90.) .and. part(i_part)%theta_es
(i_inter) .le. (((pi/180.)*95.)) then
    part(i_part)%theta_es(i_inter) =((pi/180.)*95.)
  end if
else if (part(i_part)%theta_es(i_inter) .lt. ((pi/180.)*85.) .or. part(i_part)%theta_es
(i_inter) .gt. ((pi/180.)*95.)) then
    part(i_part)%theta_es(i_inter)=part(i_part)%theta_es(i_inter)
end if
!!!!!----------------------------------!!!!!!!!!
return
end subroutine angle_accept

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
subroutine path(aa,pzi,Sigma_total1,theta,s,flu)
imPLICIT none
! Deceleration by letter
!double precision ::Sigma_total1,aa,pzi,s,s1,s2,s3,s4,sd
double precision r1(1) ,z1,theta
integer(kind=4) ::ii,flu
!!!!!!!!!!!!!!!!!!!! Generating Random Numbers!!!!!!!!!!!!!!!!!!!!

! calculation Begin
! aa is negative as we have negative slope

s1=((1.)/(aa*cos(theta)))
s3=(((log((1.-rando_1%rando(flu))*aa*cos(theta)))/(Sigma_total1*pzi))
  if ( theta .le. (pi/2.)) then
    s2=log(1.-s3)
  else if ( theta .gt. (pi/2.)) then
    s2=log(1.+s3)
  end if
s4=((log((1.-rando_1%rando(flu))*aa*cos(theta)))
!!!!!!!!!!!!!!!!!! s is in meters ______
s=s1*s2
sd=(-1./((Sigma_total1*pzi))*log(rando_1%rando(flu))
!print*,'s',s
!pause
return
end subroutine path

subroutine path_accept(i_part,i_inter,j,s)
implicit none
integer :: i_inter,i_part,j
real(double) :: s,z_new,r_new
!!!! z_new and r_new are vertical and horizontal distance for new interaction !!!!!

!!!!!! Horizontal Distance Calculation !!!!!!

z_new=s*cos(part(i_part)%theta_es(i_inter))
r_new=s*sin(part(i_part)%theta_es(i_inter))
!!!!!!!! ----------------------------------!!!!!!!!!
!!!!!!! Vertical Distance Calculation !!!!!!!

if(part(i_part)%theta_es(j-1).le.(pi/2.)) then
  part(i_part)%z_path(j)=part(i_part)%z_path(j-1)+(((z_new/1.)**2.)**0.5)
else if (part(i_part)%theta_es(j-1).gt.(pi/2.)) then
  part(i_part)%z_path(j)=part(i_part)%z_path(j-1)-(((z_new/1.)**2.)**0.5)
end if
!print*,'z',part(i_part)%z_path(j)
!pause
return
end subroutine path_accept

subroutine compton(i_part,j,isp,ui,Es,flu)
implicit none

integer :: flu
double precision a1, a2, ans, asp, asp1, b1, b2, b3, b4, b5, cos_theta, cos_e, cs_e, DST, Ee, Es, IPE, IPEM, pei, n1, m1, o1, p1, q1, pad
double precision R_E, s1, s2, sigma_total, sxz, sin_theta, thet, thel, theta_e_deg, theta_deg, hsf, tau, tmin, theta, theta_e, z1, ui

! Deceleration by alphabats
!n
double precision Ex, ii, k, s, T, x

double precision ER_E

! Deceleration for random numbers
!!!!!!! this is to call species !!!!!!!!!!!!!!!!!!!!!!!
integer(kind=4) :: isp, i_part, j
character*20 :: text
!!!!!!! this is to call species !!!! Generating Random Numbers!!!!!!!!!!!!!!!!!!!!!!!

!n
text=trim(name_species(isp))//'.txt'

!print*, text

pei = pi
ER_E = ERE
R_E = RE

open (unit=92,file=text)
read (92,*)z1, b1, b2, b3, b4, b5, ui
  ! print*,z1, b1, b2, b3, b4, b5, ui
close(92)
1 flu=flu+1

! calculation Begin
Ex=part(i_part)%es1(j-1)
IPE = Ex
!
IPEM = IPE/ER_E
k=IPE/ER_E
!
a1 = log(1.+(IPEM*2.))

a2 = ((1.+IPEM)*(IPEM*2.))/((1.+IPEM*2.)**2.)
!
s1 = a1/(a1+a2)
!
s2 = a2/(a1+a2)
!
tmin = 1./(1.+(IPEM*2.))
!
!! ! ! ! ! ! ! ! Conditions!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
flu=flu+1
!
if (rando_1%rando(flu) .lt. s1) then
flu=flu+1

tau = tmin**rando_1%rando(flu)

cos_theta=(1.-((1.-tau)/(IPEM*tau)))
theta =acos(cos_theta)
the1=(180./pei)*theta

sin_theta=(1.-(cos_theta)**2.)**0.5
!
flu=flu+1
else

tau = ((tmin**2.)+(rando_1%rando(flu))*(1.-tmin**2.))**0.5
!
cos_theta=1.-((1.-tau)/(IPEM*tau))
!theta = (180./pei)*acos(cos_theta)
theta =acos(cos_theta)
the1=(180./pei)*theta
sin_theta=(1.-(cos_theta)**2.)**0.5
!
flu=flu+1
end if
!!! x is dimensionless variable, used instead of momentum transfer q
x = (20.6074*IPEM)*(1. - cos_theta)**0.5
!
    print*, "x", x
    !write (143,*)"x", x
!!! analytical incoherent scattering function
sxz = (z1*(1-((1+(b1*(x**2))+(b2*(x**3))+(b3*(x**4)))/(1+(b4*(x**2))+(b5*(x**4)))*2)))/((1+(b1*(x**2))+(b2*(x**3))+(b3*(x**4)))/(1+(b4*(x**2))+(b5*(x**4)))*2))
!
    !sxz =1
    !print*, "sxz", sxz
    !write (143,*)"sxz", sxz
!!! valid rejection function
if ( IPE .ge. ui) then
hsf=1
else if( IPE .lt. ui) then
!
    print*, "Energy is too low for bound Model, go for unbound or enter energy more then",ui
hsf=0
end if
!
T = (1.-((1.-tau)*(((2.*IPEM)+1.)*tau)-1.))/((IPEM**2.)*tau*(1.+(tau**2.)))*(sxz/z1)*hsf
!
    print*, "T", T
    !write (143,*)"T", T
!!! Generating third random number!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
    errcode=vsrnguniform( method_uni, stream, n, r3, a, sigma )
flu=flu+1
!
if ( rando_1%rando(flu) .gt. T ) go to 1
!
Es=tau*Ex
if (((1.-tau)*Ex)-ui) .ge. 0
    !print*, "Es",Es
    Ee=((1.-tau)*Ex)
else
    Ee=(1.-tau)*Ex
end if
!
  print*, "Ee",Ee
  cos_e=(1.+IPEM)*(((1.-tau)/(IPEM*(2.+(IPEM*(1.-tau)))))*0.5)
theta_e =acos(cos_e)
theta_e_deg=(180./pei)*theta_e
theta_deg=(180./pei)*theta
!
    part(i_part)%es1(j)=Es
    part(i_part)%ee1(j)=Ee
    part(i_part)%theta_es(j)=theta
    part(i_part)%theta_ee(j)=theta_e
!! Deceleration by alphabats
return
end subroutine compton
!!!!!!!!!!!!!!!!!!!!!!!!!!!! Free __compton!!!!!!!!!!!!!!!!!!!!!!!!!!!!
subroutine compton_free(i_part,j,Es,flu)
implicit none
!
integer(kind=4) ::i_part,j
double precision a1, a2, cos_theta,cos_e,cs_e,cs_e1, DST ,d1,d2, Ee,EREx,Es,e_density,
IPE,IPEM, ml, mex,n11,o1,p1,q1,pad
double precision REx, s1,s2,sxz,sin_theta,set,set1, thet,theta,theta_e , 
tau,tau1,tau2,tmin,tau_d,pix
integer ::flu
!! Deceleration by letter
return
real aa, Sigma

double precision p_in, pi_1, pi_2, p_in1, c, Ex, Ek, ii, k, s, T, T1, xx, x, q, z1

open (unit=143, file = "19april.txt")
open (unit=144, file = "rejected.txt")
open (unit=145, file = "ostgard.txt")

!!!!!!!!!!!!!!!!! Generating Random Numbers!!!!!!!!!!!!!!!!!!!!!!!!!!!!

pix = pi
mex = me
ERE = ere
RE = re
ii = 0

!1 ii = ii + 5
!1 flu = flu + 1
Ex = part(i_part)%es1(j - 1)

IPE = Ex
IPEM = IPE/ERE
k = IPE/ERE

tau = (1./(1.+(IPEM*2.)))
a1 = log(1./tau)
a2 = ((1.-(tau**2))/2.)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! sampl Eug tau

if (a1 .ge. ((a1+a2)*rando_1%rando(flu))) then
    flu = flu + 1
    s1 = exp(a1*(rando_1%rando(flu)))
    tau = tmin*s1
else
    if (IPEM .ge. (IPEM+1.)*(rando_1%rando(flu))) then
        flu = flu + 1
        tau_d = amax1(rando_1%rando(flu), (rando_1%rando(flu+1))
        tau = tmin+(1.-tmin)*tau_d
    else
        flu = flu + 1
        tau_d = (rando_1%rando(flu))
        tau = tmin+(1.-tmin)*tau_d
    end if
end if

T = (ERE*tau)/(IPE*tau)

sin_theta = (T*(2.-T)**(0.5))
sxz = (1.-((tau*(sin_theta**2))/(1+(tau**2))))

flu = flu + 1
if (rando_1%rando(flu) .lt. sxz) go to 1

Es = tau*IPE

theta = acos(cos_theta)
Ee = (1.-tau)*IPE

cos_theta = (((IPE+ERE)/(IPE*Es))-(IPE/(IPE*ERE))/(IPE*Es))

theta_e = acos(cos_e)

part(i_part)%es1(j) = Es
part(i_part)%ee1(j) = Ee
part(i_part)%theta_es(j) = theta
part(i_part)%theta_e(j) = theta_e

!print*, part(i_part)%es1(j), part(i_part)%ee1(j), 'theta', part(i_part)%theta_es(j), part(i_part)%theta_e(j), Ex

return
end subroutine compton_free

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
subroutine pe(i_part,j,shellss,isp,flu)

implicit none
!!!&for random number&!!!
!!&parameters!!
double precision ::Epx
integer(kind=4) ::shellss,isp,i_part,j
integer ::flu
double precision :beta,Epe,ui,aeta,vai,
gamma,gvai,gvai0,cos_theta_pe,mecsq_ev
!!&parameter for distribution s!!
double precision ::pvai
integer(kind=4) ::i500
!!!!!!!!!!!!!!!!! Generating Random Numbers!!!!!!!!!!!!!!!!!!!!

!!mecsq_ev=ere
if (isp .eq. 1)then
  if (shellss .eq. 2)then
    ui=871.4101
  else if (shellss .eq. 3)then
    ui=138.1197
  else if (shellss .eq. 4)then
    ui=77.4153
  else if (shellss .eq. 5)then
    ui=35.11
  end if
else if (isp .eq. 2)then
  if (shellss .eq. 2)then
    ui=667.046
  else if (shellss .eq. 3)then
    ui=97.8902
  else if (shellss .eq. 4)then
    ui=47.4735
  else if (shellss .eq. 5)then
    ui=29.603
  end if
end if

Epx=part(i_part)%es1(j-1)
if (shellss .eq. 0)then
  Epe=Epx
else
  Epe=Epx-ui
end if
gamma=1.+(Epe/(mecsq_ev))
beta=((Epe*(Epe+2.*mecsq_ev))**0.5)/(Epe+mecsq_ev)
aeta=(1./beta)-1.

! 1 i=i+200
1 flu=flu+1
vai=((2.*aeta)/((aeta+2.)**2.)-4.*(rando_1%rando(flu)))*((2.*(rando_1%rando(flu))+(aeta+2.)*((rando_1%
  rando(flu))**0.5)))
gvai=(2.-vai)*((1./(aeta+vai))+(0.5*beta*gamma*(gamma-1.)*(gamma-2.)))
gvai0=(2.-0.0)*((1./(aeta+0.0))+(0.5*beta*gamma*(gamma-1.)*(gamma-2.)))
flu=flu+1
if (((rando_1%rando(flu))>gvai0) .gt. gvai) go to 1

  cos_theta_pe=1.-vai
  cos_theta_pe=acos(cos_theta_pe)
return
end subroutine pe
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!for random number!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!parameters!!!!!!!!!!!!!!!!!!!!!!!!!!!!
double precision ::Ex
integer(kind=4) ::isp,i_part,j
double precision ::cos_thetaﭷpe,cos_theta มีนาคมpp,Eppe,Eppp,kpp,rmech,bpp,frac_pp,glb,g2b,zpp,app,fcz,fokz,g0k,fhi1,fhi2,ppfrac_enener,betap,betan
double precision ::u1,u2,frac_pp,u11,u22,frac_pp1,bpp1,glb1,g2b1,fhi11,fhi22,ppfrac_ener,betap,betan
integer ::flu
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Ex=part(i_part)%es1(j-1)

kpp=Ex/ERE

if (isp .eq. 1) then
    rmech=54.033
    zpp=8.
else if(isp .eq. 2) then
    rmech=57.524
    zpp=7.
end if

!!! app is constant given by multiplication of fine structure constant and atomic number zpp

app=alpha*zpp

frac_pp=0.5

bpp=rmech*(1./2.)*kpp)*1.)/(frac_pp*(1.-frac_pp)))

glb=(7./3.)*2.*(log(1. +bpp**2.)) -6.*bpp*atan((bpp)**(-1.)) -(bpp**2.)*((4.-4.*bpp*atan((bpp)**(-1.)))-3.*log(1. +((bpp)**(-2.)))

g2b=(11./6.)*2.*log1. +bpp**2.)) -3.*bpp*atan((bpp)**(-1.)) +&
0.5*(bpp**2.)*((4.-4.*bpp*atan((bpp)**(-1.)))-3.*log(1. +((bpp)**(-2.)))

fcz=(app**2.)*(((1.+((app**2.))**(-1.)))(+0.20059-0.03693*(app**2.)+0.00835*(app**4.)-0.00201*app**6.)&
&
0.00049*(app**8.)-0.00012*(app**10.)+0.00003*(app**12.))

fokz=(0.1774*(12.10*app)+0.0174*(app**2.))*(2./kpp)*0.5)+&
0.523*(73.26*app) -44.41*(app**2.))*2./kpp-4&
(13.52*(121.1*app) -96.41*(app**2.))*2./kpp)+&
(8.524*(62.05*app) -63.41*(app**2.))*2./kpp)

g0k=4.*log(rmech)-4.*fcz+fokz

fhi1=glb+g0k

fhi2=g2b+g0k

u1=(2./3.)*(((1./2.)-(1./kpp))**2.)*fhi1

u2=fhi2

! 1  i=i+200
1 if ((rando_1%rando(0u)) .lt. (u1/u1+u2)) then
2 flu=flu+1
3 frac_pp=(((1.2.)-(1./kpp))**2.))**0.5)
4 frac_pp=frac_pp
5 bpp1=frac_pp
6 bpp2=1./2.)*pp1*(1.-frac_pp))
7 g1b=(7./3.)*2.*(log1. +bpp1**2.)) -6.*bpp1*atan((bpp1)**(-1.)) -(bpp1**2.)*((4.-4.*bpp1*atan((bpp1)**(-1.)))-3.*log1. +((bpp1)**(-2.)))
8 g2b=(11./6.)*2.*(log1. +bpp1**2.)) -3.*bpp1*atan((bpp1)**(-1.)) +&
9 0.5*(bpp1**2.)*((4.-4.*bpp1*atan((bpp1)**(-1.)))-3.*log1. +((bpp1)**(-2.)))
10 fhi1=g1b+g0k
11 u1=fhi1/fhi1
12 flu=flu+1
if ((rando_1%rando(flu)) .gt. u11) go to 1
else
  flu=flu+1
  fract_pp=((1./2.)+(1./2.)-(1./kpp))*(2.*(rando_1%rando(flu)))
  frac_pp1=frac_pp
  bpp1=rmech*(1./((1.-frac_pp1)*(1.-frac_pp1)))
  g1b1=(7./3.)*(-2.)*log((1.0+bpp1**2.0)*-6.0*bpp1*atan((bpp1)**(-1.0))-2.0*(4.0-4.0*bpp1*atan((bpp1)**(-1.0))-3.0*log(1.0+(bpp1)**(-2.0))))
  g2b1=(11./6.)*(-2.)*log((1.0+bpp1**2.0)*-3.0*bpp1*atan((bpp1)**(-1.0)))+&
  0.5*(bpp1**2.0)*(4.0-4.0*bpp1*atan((bpp1)**(-1.0))-3.0*log(1.0+(bpp1)**(-2.0)))
fi22=g2b1+g0k
u11=fi22/fhi2
flu=flu+1
if ((rando_1%rando(flu)) .gt. u22) go to 1
end if
ppfrac_en=frac_pp
Eppe=ppfrac_en*Epx-ERE
Eppp=Epx-Eppe-2.*ERE
betap=((Eppp*(Eppp+2.*ERE))**0.5)/(Eppp+ERE)
betan=((Eppe*(Eppe+2.*ERE))**0.5)/(Eppe+ERE)
cos_theta_ppe=((2.0*(rando_1%rando(i+19+n)))-1.0+betan)/(((2.0*(rando_1%rando(i+19+n)))-1.0)*betan)+1.0)
cos_theta_ppp=((2.0*(rando_1%rando(i+19+n)))-1.0+betap)/(((2.0*(rando_1%rando(i+19+n)))-1.0)*betap)+1.0)
cos_theta_ppe=acos(cos_theta_ppe)
cos_theta_ppp=acos(cos_theta_ppp)
part(i_part)%es1(j)=0.0
part(i_part)%ee1(j)=Eppe
part(i_part)%theta_es(j)=0.0
part(i_part)%theta_ee(j)=cos_theta_ppe
part(i_part)%ep1(j)=Eppp
part(i_part)%theta_ep(j)=cos_theta_ppp
! print*,part(i_part)%es1(j-1),part(i_part)%es1(j),part(i_part)%ee1(j),part(i_part)%theta_es(j),part
(i_part)%theta_ee(j),part(i_part)%ep1(j),part(i_part)%theta_ep(j)
! pause
return
end subroutine pp
! !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!__free__compton_end!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
end module collisions
! !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!Reference:
!! SIMULATION OF PHOTON TRANSPORT ,page 535
Contents of Simulation Code

- Main program calling all modules and subroutine
- Module initialization
- Module collision
- **Module constant geophysics**
  - Module constant physics
  - Module constant simulation
  - Module structure concentration
  - Module structure cross section
  - Module structure particle
  - Module structure random number
- Module toolbox
module const_geophys

implicit none

! Name and number of neutral species
integer(kind=4),parameter :: nb_spc=2,nb_pe_subshell=4
character(len=20),dimension(nb_spc),parameter :: name_species=/'O2','N2'/
integer(kind=4),dimension(nb_spc),parameter :: z_an=/(8,7/)
character(len=20),dimension(nb_spc),parameter :: name_reac=/'bound','unbound'/
character(len=20),dimension(5),parameter :: name_reac2=/'total','k','l1','l2','l3'/
character(len=20),dimension(3),parameter :: name_proc=/'compton','photoelectric','pair-production'/
!!!!!!!MSIS_open!!!

integer(kind=4),parameter :: nb_conc=8
integer(kind=4),parameter :: nb_conc_use=2
!!! 1- oxygen,2-nitrogen,3-argon,4-Hydrogen,5 -nitrogen atom,6-helium,7-oxygen,8-total mass all in meters^-3
character(len=20),dimension(nb_conc),parameter :: name_conc=/'HE','O','N2','O2','AR','TOT','H','N'/
!!!!!!!MSIS_close!!!

end module const_geophys
Contents of Simulation Code

- Main program calling all modules and subroutine
- Module initialization
- Module collision
- Module constant geophysics
- **Module constant physics**
- Module constant simulation
- Module structure concentration
- Module structure cross section
- Module structure particle
- Module structure random number
- Module toolbox
module const_phys

use const_simul

implicit none

! Charge electronique (c)
real(double), parameter :: ec=1.60217733d-19

! Boltzmann constant (J/K)
real(double), parameter :: k=1.380658d-23

! electronic mass (kg)
real(double), parameter :: me=9.1d-31

! proton mass (kg)
real(double), parameter :: mp=1.66054d-27

! permitivity (F/m)
real(double), parameter :: Epsilon0=8.854187817d-12

! Pi
real(double), parameter :: pi=3.14159265358979d0

!g
real(double), parameter :: g=9.81d0

!c
real(double), parameter :: c=2.99792458d8

!mc2
real(double), parameter :: mc2=me*c**2.d0

!ERE
real(double), parameter :: ERE=(mc2)/ec

!Radius of electron
real(double), parameter :: RE=2.8179e-13

!a0
real(double), parameter :: a0=0.529177249d-10

!alpha
real(double), parameter :: alpha=7.29735308d-3

! Avogadro's Number (1/mol)
real(double), parameter :: Na=6.022d23

end module const_phys
Contents of Simulation Code

- Main program calling all modules and subroutine
- Module initialization
- Module collision
- Module constant geophysics
- Module constant physics
- **Module constant simulation**
- Module structure concentration
- Module structure cross section
- Module structure particle
- Module structure random number
- Module toolbox
module const_simul

  implicit none

  integer (kind=4) :: i,j,i_part, i_inter

  !!! Select type of modal
  !!! 1 - Bound
  !!! 2 - Unbound
  integer(kind=4),parameter :: model=2

  !!! Select type of altitude distribution
  !!! 1 - discrete
  !!! 2 - distributed
  !!! 3 - monoaaltitude
  !!! 4- beam type ... after selectiong this please also set initial and final energy of beam
  integer(kind=4),parameter :: alt_distri=2

  !!! ALT mono is the initial altitude for all particles
  real ,parameter :: ALT_mono=30000.
  real ,parameter :: max_ALT_beam=60000.
  real ,parameter :: min_ALT_beam=10000.

  !!!Select total number of discrete altitude
  integer(kind=4),parameter :: num_alttds=7.

  !!! power of energy flux
  !!! 1 - discrete
  !!! 2 - distributed
  !!! Select parameter for power law distribution
  !!! power=-1. for power law distribution
  !!! power=0. for beam type energy distribution, please also select max abd min beam energy
  !!!
  real :: power=-1.
  real ,parameter :: max_beam_energy=10.e6
  real ,parameter :: min_beam_energy=10.e3

  !!!Select initial angle for particles
  !!! mono -1
  !!! distributed -2
  integer(kind=4),parameter :: angle_distri=1
  :: initial_angle=0.

  !!! Sampling angles!!!!!!!!!!!!!!!!!!!!!!
  real ,parameter :: theta_1=90.
  real ,parameter :: theta_2=30.
  real ,parameter :: theta_3=30.
  real ,parameter :: theta_4=60.
  real ,parameter :: theta_5=60.
  real ,parameter :: theta_6=90.

  !!! choosing process type
  !!! 1 - automatic process by algorithm
  !!! 2 - defined by user
  ! integer(kind=4),parameter :: user_defined=1
  !!! choose process
  !!! 1-compton
  !!! 2- photoelectric
  !!! 3- pair production
  integer(kind=4) :: selected_process=1

  ! precision
  integer (kind=4), parameter :: double=kind(1.d0)

  ! number of states (excitation, ionization etc...) !10
  integer(kind=4),parameter :: nb_proc=3
  ! number of states (excitation, ionization etc...) !20
integer(kind=4),parameter :: nb_reac=5
! number of energies
integer(kind=4),parameter :: nb_ener=1091120
!
! It defines array of allowable random number before it was 3e4
integer(kind=4),parameter :: n=1e5
!
! Number of random number sampled in loop before it was 1e4
integer(kind=4),parameter :: array=6*n
!
! Number of random number sampled in loop
integer(kind=4),parameter :: l=28

!! number of initial energy particles
!! number of energies
integer(kind=4),parameter :: nb_particles=100000
!! Number of interaction
integer(kind=4),parameter :: nb_interect=100

! conversion factor for cross section from cm2 to m2
real,double),parameter :: cm_mtr=1e-4
!!!!!!!!!!!!

!!!!!!!!!!!!

!!  ALT lower Boundary- ALTITUDE(M)
real,parameter :: MIN_ALT=1000.

!! IYD - YEAR AND DAY AS YYDDD or DDD (day of year from 1 to 365)
integer(kind=4),parameter :: IYD=172
!! MASS - MASS NUMBER (ONLY DENSITY FOR SELECTED GAS IS CALCULATED. MASS 0 IS TEMPERATURE. MASS 48 FOR ALL.
integer(kind=4),parameter :: MASS=48

!! SEC - UT(SEC)
real,parameter :: SEC=29000.
!! ALT - ALTITUDE(KM)
real,parameter :: ALT=70000.
!! GLAT - GEODETI C LATITUDE(DEG)
real,parameter :: GLAT= 55.
!! GLONG - GEODETI C LONGITUDE(DEG)
real,parameter :: GLONG=15.
!! STL - LOCAL APPARENT SOLAR TIME(HRS)
real,parameter :: STL=16.
!! F107A - 3 MONTH AVERAGE OF F10.7 FLUX
real,parameter :: F107A=150.
!! F107 - DAILY F10.7 FLUX FOR PREVIOUS DAY
real,parameter :: F107=150.
!! AP - MAGNETIC INDEX(DAILY) OR WHEN SW(9)=-1.
real,dimension(7),parameter :: AP=4.

!! OUTPUT:
!! T(1) - EXOSPHERIC TEMPERATURE
!! T(2) - TEMPERATURE AT ALT
real,dimension(2) :: Tem

!! D(1) - HE NUMBER DENSITY(CM-3)
!! D(2) - O NUMBER DENSITY(CM-3)
!! D(3) - N2 NUMBER DENSITY(CM-3)
!! D(4) - O2 NUMBER DENSITY(CM-3)
!! D(5) - AR NUMBER DENSITY(CM-3)
!! D(6) - TOTAL MASS DENSITY(GM/CM3)
!! D(7) - H NUMBER DENSITY(CM-3)
!! D(8) - N NUMBER DENSITY(CM-3)
!! Co efficient and coloumn density in unit kg/m4 and kg/m2 of density profile
!! path length
!! double precision :: aa,,pzi,s
!! ALT lower Boundary- ALTITUDE(M)
real,parameter :: MIN_ALT=1000.
!! ALT upper boundaey - ALTITUDE(M)
real ,parameter :: MAX_ALT=100000.

real(double), dimension(num_alttds),parameter :: Altitudes=(/1000., 10000., 20000., 30000.,
40000., 50000., 60000./)

!!!!!!!!!
!!!!!!!!!!!! MSIS_close !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Maximum Energie
   real(double),parameter :: E_max=1.e7!20.e6
!period!
   real(double),parameter :: E_p1=1.e3, E_p2=1.e4
! step1 ! Energy from zero to 100 kev increase by step E1                E1=1.e3
   real(double),parameter :: E1=1.e3
!period2!
   real(double),parameter :: E_p11=1.e4, E_p22=1.e5
! step2 ! Energy from 100 kev to 1000 kev increase by step E2             E2=1e4
   real(double),parameter :: E2=1.e4
!period3!
   real(double),parameter :: E_p111=1.e5, E_p222=1.e6
! step3 ! Energy from 100 kev to 1000 kev increase by step E3             E3=1e5
   real(double),parameter :: E3=1.e5
   real(double),parameter :: E3=1.e5
   real(double),parameter :: E3=1.e5
   real(double),parameter :: E3=1.e5
   real(double),parameter :: E3=1.e5
! step3 ! Energy from 100 kev to 1000 kev increase by step E3
   real(double),parameter :: E6=1.e6, E7=1.e7
! above E3 E increases as E**coef
   real(double),parameter :: coef=4.0
!
! epsilon
   real(double),parameter :: epsilon=1e-34
! mean of gaussian uniform random number
   real :: a = 0.0
! standard deviation of gaussian uniform random number
   real :: sigma = 1.0

end module const_simul
Contents of Simulation Code

- Main program calling all modules and subroutine
- Module initialization
- Module collision
- Module constant geophysics
- Module constant physics
- Module constant simulation
- **Module structure concentration**
- Module structure cross section
- Module structure particle
- Module structure random number
- Module toolbox
module concentration

use const_simul
use const_geophys

implicit none

type conc
    character (len=20) :: name
    real, dimension(nb_conc) :: Den
end type conc

type(conc), dimension(nb_conc) :: cn

end module concentration
Contents of Simulation Code

• Main program calling all modules and subroutine
• Module initialization
• Module collision
• Module constant geophysics
• Module constant physics
• Module constant simulation
• Module structure concentration
• Module structure cross section
• Module structure particle
• Module structure random number
• Module toolbox
module cross_section

use const_simul
use const_geophys

implicit none

! cross_sec structure
! Each collisional process is described by :
! a species : 02, N2 etc
! a process : vibrational, electronic, rotational, ionization, Bremsstrahlung etc...
! a reaction including a name an a list of parameters
! sp%pro%reac

type reactions
  character (len=40) :: name
  real(double), dimension(:), allocatable :: e_pe, cs_t, cs_k, cs_l1, cs_l2, cs_l3
  real(double), dimension(nb_ener) :: e, cs
  ! real(double) :: energy_loss
  ! real(double) :: E_min
  ! real(double) :: E_max
  ! real(double) :: weight
  ! real(double) :: ebr
  ! real(double) :: qscale
  ! integer(kind=4) :: n_ener
  real(double), dimension(nb_ener) :: e_int, cs_int
  real(double), dimension(array) :: rn
  integer(kind=4), dimension(nb_spc) :: z_nr
end type reactions

type processes
  character (len=40) :: name
  integer(kind=4) :: nb_reac
  type(reactions), dimension(nb_reac) :: reac
end type processes

type specie
  character (len=20) :: name
  integer(kind=4) :: nb_pro
  type(processes), dimension(nb_proc) :: pro
end type specie

type(specie), dimension(nb_spc) :: sp

end module cross_section
Contents of Simulation Code

- Main program calling all modules and subroutine
- Module initialization
- Module collision
- Module constant geophysics
- Module constant physics
- Module constant simulation
- Module structure concentration
- Module structure cross section
- Module structure particle
- Module structure random number
- Module toolbox
module particle

use const_simul
use const_geophys

implicit none

!! cross_sec structure
!! Each collisional process is described by :
!! a species : O2, N2 etc
!! a process : vibrational, electronic, rotational, ionization, Bremsstrahlung etc...
!! a reaction including a name an a list of parameters
!! sp%pro%reac

! type particles
character (len=40) :: name
real(double) :: energy_loss
real(double) :: E_min
real(double) :: E_max
real(double) :: weight
real(double) :: ebr
real(double) :: qscale
integer(kind=4) :: n_ener
real(double),dimension(nb_interect) :: es1
real(double),dimension(nb_interact) :: esl_last
real(double),dimension(nb_ener) :: cs
real(double),dimension(nb_ener) :: e_int
real(double),dimension(nb_ener) :: cs_int
real(double),dimension(array) :: rn
integer(kind=4),dimension(nb_spc) :: z_nr
real(double),dimension(nb_interect) :: ee1
real(double),dimension(nb_interect) :: theta_es
real(double),dimension(nb_interect) :: theta_ee
real(double),dimension(nb_interect) :: s_path
real(double),dimension(nb_interect) :: z_path
real(double),dimension(nb_interect) :: r_path
real(double),dimension(nb_interect) :: ep1
real(double),dimension(nb_interect) :: theta_ep

integer
nb_interaction

integer
nb_particle

! type processes
character (len=40) :: name
integer(kind=4) :: nb_reac
! type(reactions),dimension(nb_reac) :: reac

! end type processes

! type specie
character (len=20) :: name
integer(kind=4) :: nb_pro
! type(processes),dimension(nb_proc) :: pro

! end type specie

type(particles),dimension(nb_particles) :: part

end module particle
Contents of Simulation Code

- Main program calling all modules and subroutine
- Module initialization
- Module collision
- Module constant geophysics
- Module constant physics
- Module constant simulation
- Module structure concentration
- Module structure cross section
- Module structure particle
- **Module structure random number**
- Module toolbox
module random_nr

use const_simul
use const_geophys

implicit none

type ra
  real ,dimension(array) :: rando
end type ra

type(ra) :: rando_1

end module random_nr
Contents of Simulation Code

- Main program calling all modules and subroutine
- Module initialization
- Module collision
- Module constant geophysics
- Module constant physics
- Module constant simulation
- Module structure concentration
- Module structure cross section
- Module structure particle
- Module structure random number
- **Module toolbox**
module toolbox

use const_geophys
use const_phys
use cross_section
use const_simul
USE MKL_VSL_TYPE
USE MKL_VSL
USE IFPORT
use random_nr

! implicit none
contains

!!!&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
!-------------------------- INTERPT_D ---------------------------

subroutine interpt_d(flg_log,x0,y0,x,y)
implicit none
logical :: flg_log
real*8 :: x0(:),y0(:),x(:),y(:)
integer :: n_n,i

open(unit=15,file="e_int_cs_int.dat")
n_n=size(x0)
y0=max(y0,1.e-60)
if (flg_log) then
  do i=1,n_n-1
    where (x>=x0(i).and.x<=x0(i+1))
      y=dlog(y0(i))+dlog(y0(i+1))-dlog(y0(i))*(x-x0(i))/(x0(i+1)-x0(i))
    endwhere
  enddo
  where (x<x0(1))
    y=dlog(y0(1))+min(0.,(dlog(y0(2))-dlog(y0(1)))*(x-x0(1))/(x0(2)-x0(1)))
  endwhere
  y=dexp(y)
else
  do i=1,n_n-1
    where (x>=x0(i).and.x<=x0(i+1))
      y=y0(i)+(y0(i+1)-y0(i))*(x-x0(i))/(x0(i+1)-x0(i))
    endwhere
  enddo
  where (x<x0(1))
    y=y0(1)*min(1.,(y0(2)/y0(1))**((x-x0(1))/(x0(2)-x0(1))))
  endwhere
write(15,100)(x(i),y(i),i=1,nb_ener)
! if (x(n_n)>maxval(x)) then
!  where (x>x0(n_n))
!    y=y0(n_n)*min(1.,(y0(n_n)/y0(n_n-1))**((x-x0(n_n))/(x0(n_n)-x0(n_n-1))))
!  endif
! where(x>maxval(x))
!  y=0.0
! endwhere
100 format(e11.4,1x,e11.4)
endif
return
end subroutine interpt_d

module toolbox

!--------------------------------------------------
! random number subroutine
!

subroutine random_numbr()

implicit none

!!! Parameters required to call random number
integer(4) :: tmphour, tmpminute, tmpsecond, tmphund
integer :: brng, method_uni, seed1
TYPE (VSL_STREAM_STATE) :: stream
integer :: ercode
!real :: r(array)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!! calling random number
brng=VSL_BRNG_MT19937
method_uni=VSL_RNG_METHOD_UNIFORM_STD
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!! calling cpu time
CALL GETTIM(tmphour, tmpminute, tmpsecond, tmphund)
seed1 = (tmphund)
ercode=vslnewstream( stream, brng, seed1)
ercode=vsrnguniform( method_uni, stream, 6*n, rando_1%rando(1:array), a, sigma )

return
end subroutine random_numbr

!--------------------------------------------------
end module toolbox