Airglow caused by meteor induced electric fields

Ante Šiljić

Supervisor: izv.prof.dr. Dejan Vinković

Split, July 2014

Master Thesis in Physics

Department of Physics
Faculty of Science
University of Split

Abstract

We are exploring a possible physical mechanism responsible for formation of a shock-like airglow structure around a Leonid meteor that had been observed on November 18, 2001. There has been no satisfying explanation for this phenomenon to date. We test a physical mechanism of a meteor induced electric field that accelerates electrons in the ambient ionosphere at altitudes between 100km and 110km. Energized electron collide with atmospheric molecules and atoms and excite them. This creates an airglow around the meteor. We are using a Monte Carlo simulation of the process. The reconstructed airglow size and shape show a large resemblance with the shape and size of the observed Leonid meteor.

Contents

Chapter 1

Introduction

Motivation for this thesis comes from an observation of a Leonid meteor on 18 November 2001 (2). Images of the meteor were recorded at 1000 frames per second. Meteor was first observed at an altitude of about 123km with velocity of $\sim$72km/s. In addition to brightening of the atmosphere in its tail, a bright structure developed in front of the meteor (see figure 1.1). A bright structure resembles a shock-like structure with a large spatial size of several hundreds of meteors. It reached a maximum brightness at altitude of 104km. There has been no satisfying explanation of this large shock-like structure to date.

The Leonid observations were made at the University of Alaska's Poker Flat Research Range 30 miles northeast of Fairbanks, Alaska (65.12°N, 147.46°W) on the night of 18 November 2001. The authors of this detection note that at 10:42:59 UT a bright meteor passed through the field of view of their high-speed imager recording at 1000 frames per second. The high-speed imager is an intensified CCD with a 105mm f/0.95 lens giving a 6.4° x 6.4° field of view. The intensifier is sensitive to light in the wavelength range $\sim$500-900nm with peak sensitivity at 700 nm (2). The authors propose that far UV emissions from the hot meteor body may ionize ambient molecular oxygen. This would create a bright ionization boundary. However, this

process requires unrealistically large photon fluxes and authors note that this is an obstacle to this explanation.

In this thesis we are considering an alternative cause of airglow. We suggest that it is caused by a meteor induced electric field. According to the theoretical work (1) by Hans C. Stenbaek-Nielsen and Peter Jenniskens (2008.), interaction of an external ionospheric DC electric field with a dense meteor plasma trail will result in an amplification of the field near the trail edges. The idea of our simulation is to test this hypothesis by simulating electrons in ionosphere accelerated by meteor induced electric field. Such electrons could produce excitations

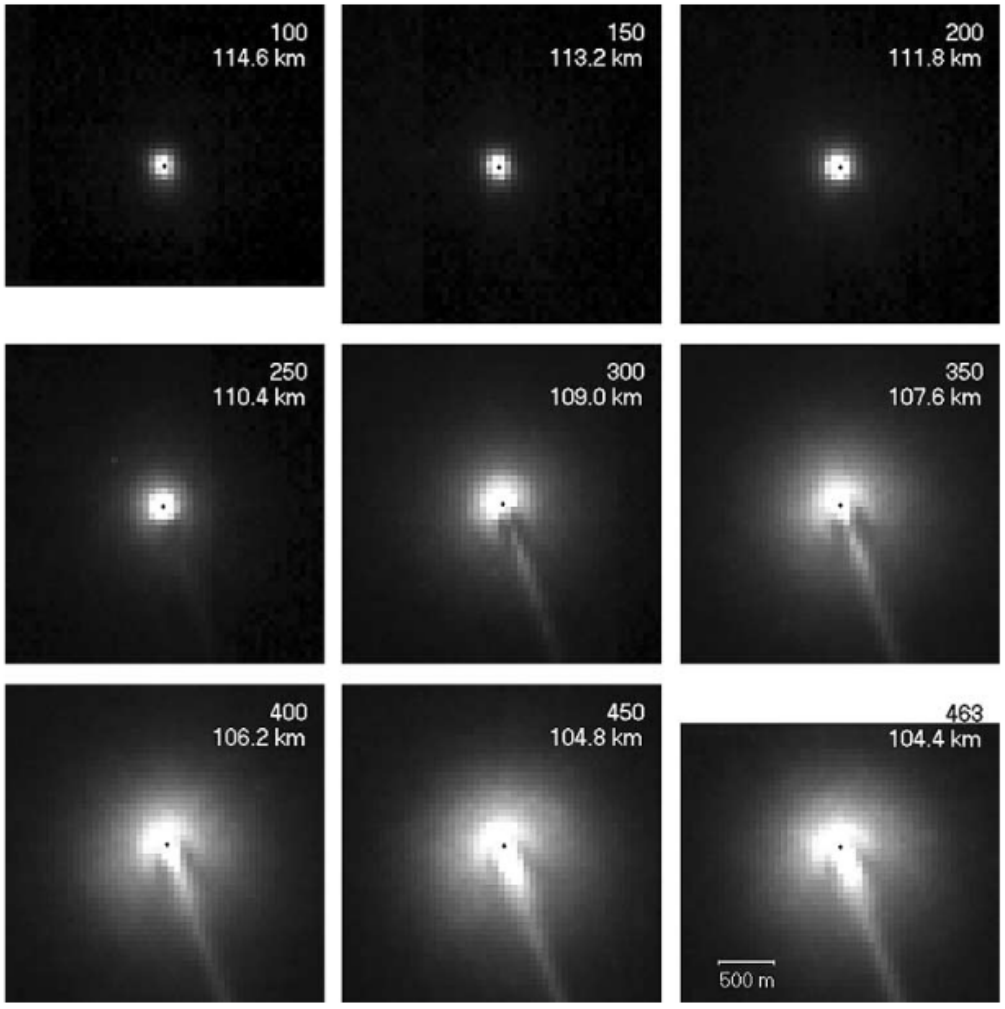

Figure 1.1: Image of a Leonid meteor recorded at 1000 frames per second on 18 November 2001 by (2)

in collisions with air molecules, which would explain the observed airglow.

Chapter 2

Theory

2.1 Model description

In this simulation a meteor is represented with a point like source of electric field moving on the path of the meteor. Starting altitude of the source of electric field is $110km$. At that point, before the start of simulation, we randomly arrange 1.4 million electrons around the source of electrical field. Each electron has a given vector of position and vector of velocity. The direction of velocity vector is random. Magnitude of velocity vector is obtained from the Maxwell's velocity distribution for the given temperature of atmosphere at this altitude. In this simulation the temperature is defined for every $1000m$ of altitude (7). Temperature profile used in simulation is given in figure 2.1. The simulation also includes the Earth magnetic field. The vector of magnetic field is pointing in the negative $\hat{z}$ direction, towards Earth, which is very close to the magnetic field vector direction at this geographic location of the recorded Leonid meteor.

Each electron has also a given value of P, which corresponds to the interaction probability

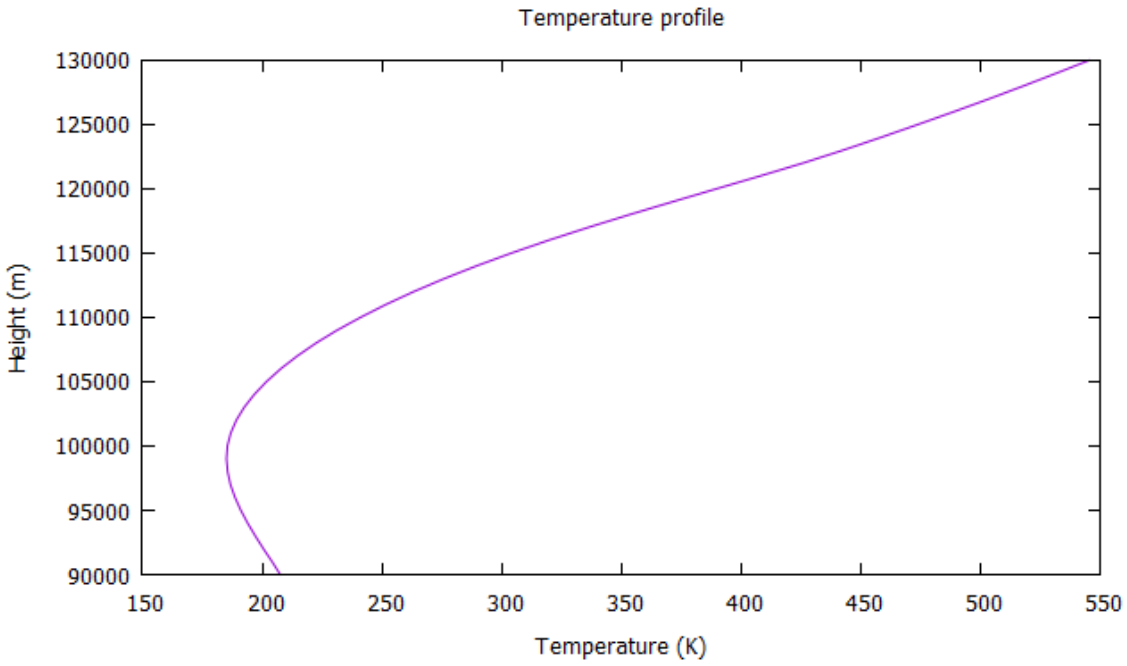

Figure 2.1: Temperature profile used in simulation - taken from (7)

in time T_{coll} and it is defined as

$$\int_0^{T_{coll}} dt \sum_{s=1}^{n} \int \sigma_s\left(\bar{v}\right) \bar{v} n_s(\vec{r}) f_b\left(\vec{r}, v_a(\vec{r})\right) d^3 v_a = -ln\left(1-\Re\right) = P, \tag{2.1}$$

where $\sigma_s\left(\bar{v}\right)$ is an interaction cross section at the relative speed $\bar{v} = |\vec{v_e} - \vec{v_a}|$ between the velocity of a projectile electron $\vec{v_e}$ and an atmospheric particle $\vec{v_a}$ of species s, n_s is the atmospheric number density of species s, f_b is distribution of velocities of atmospheric particles and $\Re$ is a random number.

At the start of the simulation the source of electrical field is at the altitude of 110*km* and it is moving in the negative $\hat{z}$ direction and negative $\hat{x}$ direction (velocity components: $v_{mx} < 0, v_{my} = 0, v_{mz} < 0$). The simulation ends when the source of electric field (i.e. meteor) reaches the altitude of 105*km*.

Every step of simulation lasts $\Delta t = 0.000001s$. In every step we move the source of electric field and each electron. If an electron is found inside the sphere of radius 20*m* around

the source of electric field then the electron is removed from the simulation. Electrons are not interacting with each other. They are moving under the influence of the electric and magnetic field. Electrons can also interact with the atmospheric atoms and molecules. They can either collide with, excite or ionize the air particles. By-product of every ionization is a new electron.

The most interesting interactions are collisions in which electron kinetic energy is between 6 and 50eV. Those collisions have the highest probability of producing photons with wavelengths corresponding to the airglow captured by camera in figure 1.1.

2.2 Arrangement of electrons

Electrons are initially randomly arranged inside a cylinder container positioned along the meteor trajectory. The cylinder is inclined so that vector perpendicular to cylinder base closes 68°

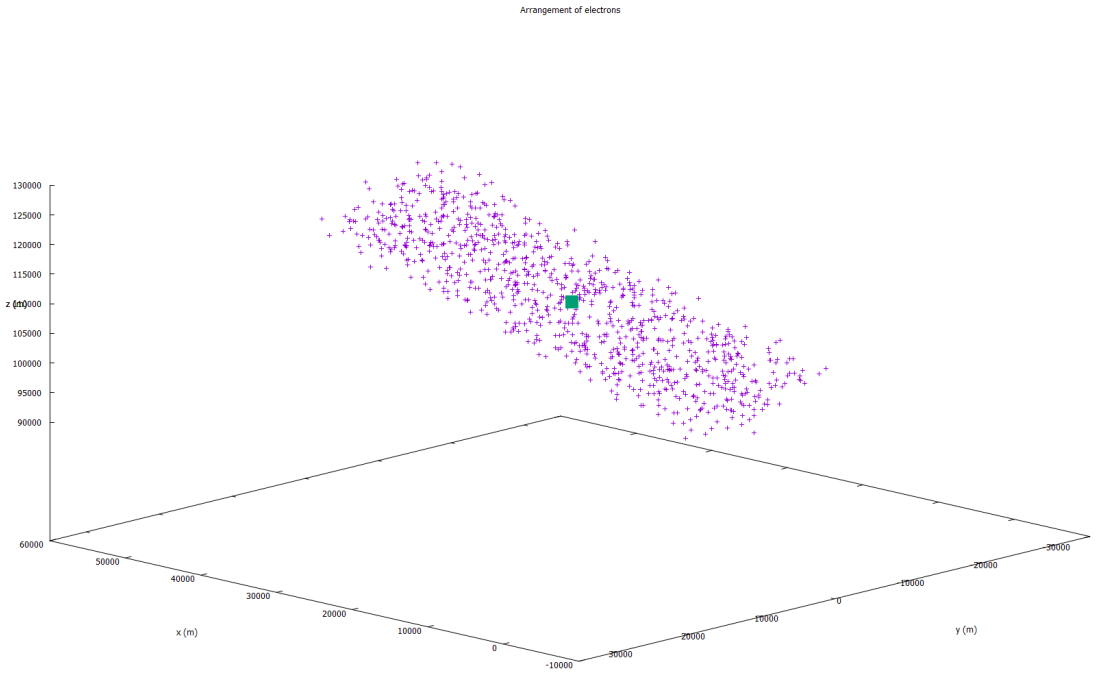

Figure 2.2: An example of the initial arrangement of electrons inside the cylinder container. The green square represents the meteor and the pink crosses are electrons

with the z axis (see figure 2.2). The angle of 68°is chosen so that it fits the angle of meteor path. Radius of the cylinder base is 10*km* and its length is 58389*m*. Height is chosen so that the center of one cylinder base is at altitude 100*km* and the other at 120*km*. This electron positioning is done in two steps. First we randomly arrange electrons inside a cylinder that is not inclined. Second we rotate each position vector of electron for 68° around $\hat{y}$ axes.

Once we are done with arranging electrons we are obtaining their velocity in a cylindrical coordinate system. We are doing that from f_b - the Maxwell's distribution of velocities

$$f_b\left(v_e(\vec{r})\right) d^3v_e = \pi^{-3/2}\frac{1}{\bar{v}^3}e^{-\frac{{v_e}^2}{\bar{v}^2}}d^3v_e = \pi^{-3/2}\frac{1}{\bar{v}^3}e^{-\frac{{v_\varrho}^2}{\bar{v}^2}}e^{-\frac{{v_z}^2}{\bar{v}^2}}v_\varrho dv_\varrho d\varrho dv_z \tag{2.2}$$

$$1 = \int_{-\infty}^{+\infty} f_b\left(v_e(\vec{r})\right) d^3v_e \tag{2.3}$$

To determine φ_e we do:

$$\Re = \int_0^{\varphi_e} d\varphi \int_0^{\infty} dv_\varrho \int_{-\infty}^{\infty} dv_z \frac{v_\varrho}{\pi^{3/2}{\bar{v}_e}^3} e^{-\frac{{v_\varrho}^2}{\bar{v}^2}} e^{-\frac{{v_z}^2}{\bar{v}^2}} = \varphi_a \frac{1}{2\pi} \tag{2.4}$$

where $\bar{v} = \sqrt{\frac{2kT}{m_e}}$. $v_{\varrho,e}$ is determined from

$$\Re = \frac{2\pi \int_0^{v_{\varrho,e}} v_\varrho dv_\varrho \int_{-\infty}^{+\infty} f_b(\varrho_e, v_\varrho, v_z)}{2\pi \int_0^{+\infty} v_\varrho dv_\varrho \int_{-\infty}^{+\infty} f_b(\varrho_e, v_\varrho, v_z)} = 2\pi \frac{1}{\pi^{3/2}{\bar{v}_e}^3} \int_0^{v_{\varrho,e}} \frac{1}{2} d{v_\varrho}^2 e^{-\frac{{v_\varrho}^2}{\bar{v}^2}} \bar{v}\sqrt{\pi} \tag{2.5}$$

$$= \frac{1}{\bar{v}^2} | - \bar{v}^2 e^{-\frac{{v_{\varrho,e}}^2}{\bar{v}^2}} |_0^{v_{\varrho,e}} = 1 - e^{-\frac{{v_{\varrho,e}}^2}{\bar{v}^2}}$$

$$v_{\varrho,e} = \bar{v}\sqrt{\ln(\Re)} \tag{2.6}$$

and finally v_z is:

$$\Re = \frac{\int_{-\infty}^{v_{z,e}} f_b(\varrho_e, v_{\varrho,e}, v_z)dv_z}{\int_{-\infty}^{+\infty} f_b(\varrho_e, v_{\varrho,e}, v_z)dv_z} = \frac{1}{\bar{v}\sqrt{\pi}} \int_{-\infty}^{v_{z,e}} e^{-\frac{{v_z}^2}{\bar{v}^2}} dv_z = \frac{1}{\sqrt{\pi}} \int_{-\infty}^{\frac{v_{z,e}}{\bar{v}}} e^{-x^2} dx \tag{2.7}$$

We calculate equation 2.7 using two different integrals depending on $\Re$:

$$\Re < 0.5 => \frac{1}{\pi} \int_{\frac{|v_{z,e}|}{\bar{v}}}^{+\infty} e^{-x^2} dx = \frac{1}{\pi}(\int_0^{+\infty} e^{-x^2} dx - \int_0^{\frac{|v_{z,e}|}{\bar{v}}} e^{-x^2} dx) = \frac{1}{2} - \frac{1}{2} Erf(\frac{|v_{z,e}|}{\bar{v}}) \tag{2.8}$$

$$\Re >= 0.5 => \frac{1}{\pi}(\int_{-\infty}^{0} e^{-x^2} dx + \int_0^{\frac{|v_{z,e}|}{\bar{v}}} e^{-x^2} dx) = \frac{1}{2} + \frac{1}{2} Erf(\frac{|v_{z,e}|}{\bar{v}}) \tag{2.9}$$

Now we have electron velocity $\vec{v_e} = v_{\varrho,e} cos(\varphi_e)\hat{i} + v_{\varrho,e} sin(\varphi_e)\hat{j} + v_{z,e}\hat{k}$.

2.3 Electrons in electric and magnetic field

The meteor as the source of electric field is moving with velocity of 71km/s, starting at altitude of 120*km* and ending at 100*km*, with the inclination of 68°relative to $\hat{z}$.

$$\vec{v}_{ef} = v_{el} sin(68°)\hat{i} + v_{el} cos(68°)\hat{j} \tag{2.10}$$

The electric field is modeled as a point-like positive charge, with the magnitude of 4000 *V/m* at the distance of 1 *m* from the center of the meteor.

We use the leapfrog integration algorithm to calculate new positions and velocities. The name leapfrog comes from one of the ways to write this algorithm, where positions and velocities "leap over" each other. Positions are defined at times $t_i, t_{i+1}, t_{i+2}, \ldots$, spaced at constant intervals dt, while the velocities are defined at times halfway in between, indicated by

$t_{\frac{i-1}{2}}, t_{\frac{i+1}{2}}, t_{\frac{i+3}{2}}, \ldots$, where $t_{i+1} - t_{\frac{i+1}{2}} = t_{\frac{i+1}{2}} - t_i = dt/2$. The leapfrog integration scheme then reads:

$$r_i = r_{i-1} + v_{i-1/2}dt \tag{2.11}$$

$$v_{i+1/2} = v_{i-1/2} + a_i dt \tag{2.12}$$

Note that the accelerations $\mathbf{a}$ are defined only on integer times, just like the positions, while the velocities are defined only on half-integer times. We have to set up the velocity at its first half-integer time step. Starting with initial conditions r_0 and v_0, we take the first term in the Taylor series expansion to compute the first leap value for v:

$$v_{1/2} = v_0 + a_0 dt/2 \tag{2.13}$$

We are then ready to apply 2.11 to compute the new position r_1, using the first leap value for $v_{1/2}$. Next we compute the acceleration a_1, which enables us to compute the second leap value, $v_{3/2}$, using Eq. 2.12 and so on.

A second way to write the leapfrog looks quite different at first sight. Defining all quantities only at integer times, we can write:

$$r_{i+1} = r_i + v_i dt + a_i (dt)^2/2 \tag{2.14}$$

$$v_{i+1} = v_i + (a_i + a_{i+1})dt/2 \tag{2.15}$$

This is still the same leapfrog scheme, although represented in a different way. Notice that the increment in r is given by the time step multiplied by $v_i dt + a_i/2$, effectively equal to $v_{\frac{i+1}{2}}$. Similarly, the increment in v is given by the time step multiplied by $(a_i + a_{i+1})/2$, effectively

equal to the intermediate value $a_{\frac{i+1}{2}}$. In conclusion, although both positions and velocities are defined at integer times, their increments are governed by quantities approximately defined at half-integer values of time.

Magnetic field of Earth is not directly used in calculation of electron movement. The only consequence of magnetic field in our simulation is that conductivity in the direction parallel to the magnetic field is much greater than conductivity in perpendicular directions. The currents flow according to the Ohm's law, but the electric conductivity is anisotropic because of the effect of the magnetic field, and three conductivities are defined. Those are parallel, Pedersen and Hall conductivities. Parallel conductivity is for the direction parallel to the magnetic field line and denoted as σ_0 (5). This is same as that when there is no magnetic field, and much larger than Pedersen and Hall conductivities in the ionosphere. Pedersen conductivity σ_p is in direction vertical to the magnetic field and parallel to the electric field. Hall conductivity is for the direction vertical to both the magnetic and electric fields, it is denoted as σ_h. Because Pedersen and Hall conductivity are much smaller than parallel conductivity figure 2.3 in our simulation we approximate electric field is accelerating electrons only in the direction parallel to magnetic field. Magnetic field in our model is pointing in negative $\hat{z}$ direction. So in this simulation electrons are accelerated only in $\hat{z}$ direction. The direction of magnetic field is chosen this way to correspond to the magnetic field similar the value at the location where the meteor was observed.

2.4 Particle collision

We are interested in altitudes between approximately 90 *km* to 130 *km*. Three gas species are the most abundant at these altitudes: atomic oxygen (O), molecular oxygen (O_2) and molecular

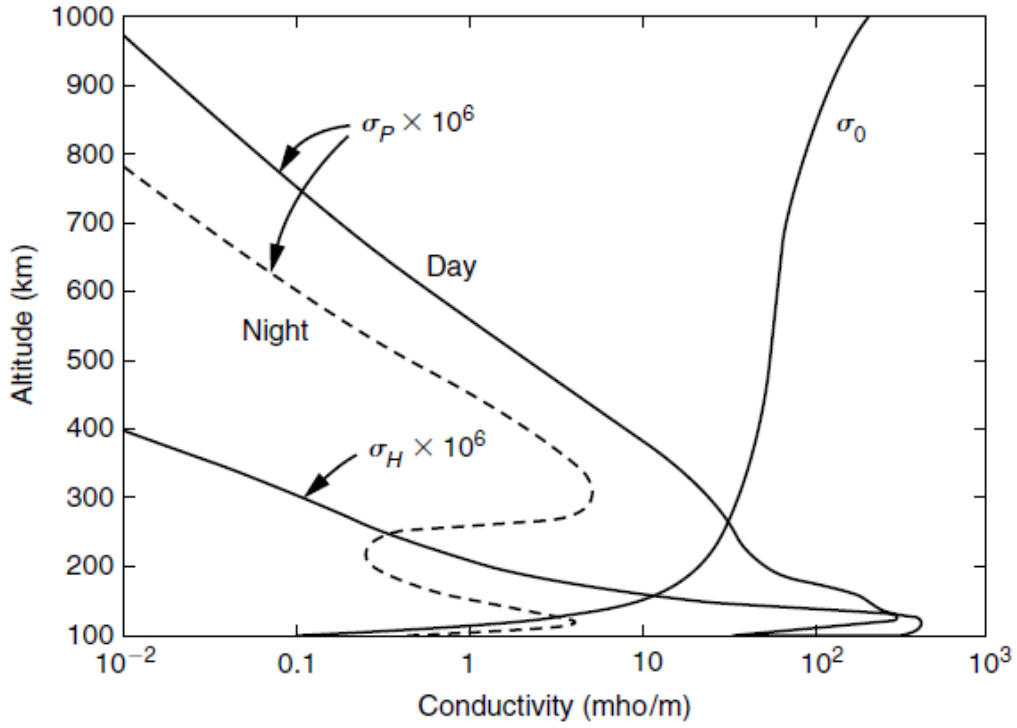

Figure 2.3: Typical conductivity values for the midlatitude daytime ionosphere (5)

nitrogen (N_2). We mark gas species as s (where $s \in \{1, 2, 3\}$).

Definition of collision probability in time T_{coll} is:

$$\int_0^{T_{coll}} dt \sum_{s=1}^{3} \int \sigma_s\left(\bar{v}\right) \bar{v} n_s(\vec{r}) f_b\left(\vec{r}, v_a(\vec{r})\right) d^3 v_a = -ln\left(1 - \Re\right) = P \tag{2.16}$$

where $\sigma_s\left(\bar{v}\right)$ is a collision cross section at relative speed $\bar{v} = |\vec{v_e} - \vec{v_a}|$ between the velocity of a projectile electron $\vec{v_e}$ and an atmospheric particle $\vec{v_a}$ of species s, n_s is the atmospheric number density of species s, f_b is distribution of velocities of atmospheric particles, $\Re$ is a random number.

We assume $\vec{v_e} \gg \vec{v_a}$ and simplify the integral, which is a good approximation because electrons have a small mass and a much higher speed than surrounding atmospheric atoms and molecules. The most probable thermal speed of electrons at T =200K (about 100km altitude) is $v_e = (2kT/m_e)^{1/2} = 5504T^{1/2} = 77.8$km/s. Thus, they react almost instantly to external

electric fields.

If we introduce a simulation time step Δt then we need to integrate the equation step by step, until the integral becomes equal to *P*:

$$\sum \Delta t \;\; v_e(\vec{r}) \sum_{s=1}^{3} \sigma_s \left(v_e(\vec{r})\right) n_s(\vec{r}) = P \tag{2.17}$$

Atmospheric number density of species n_s is dependent on altitude and we define it for every 1000 *m*. We do the same with temperature. n_s of all three gas species are given in figures 2.4, 2.5 and 2.6 (7).

Collision cross section of gas species is dependent on the kinetic energy of the colliding electron. They are given in figures 2.7 for O (from (3)), 2.8 for N_2 and 2.9 for O_2 (from (6)).

When we reach the end of integration, we have to select an atmospheric species for inter-

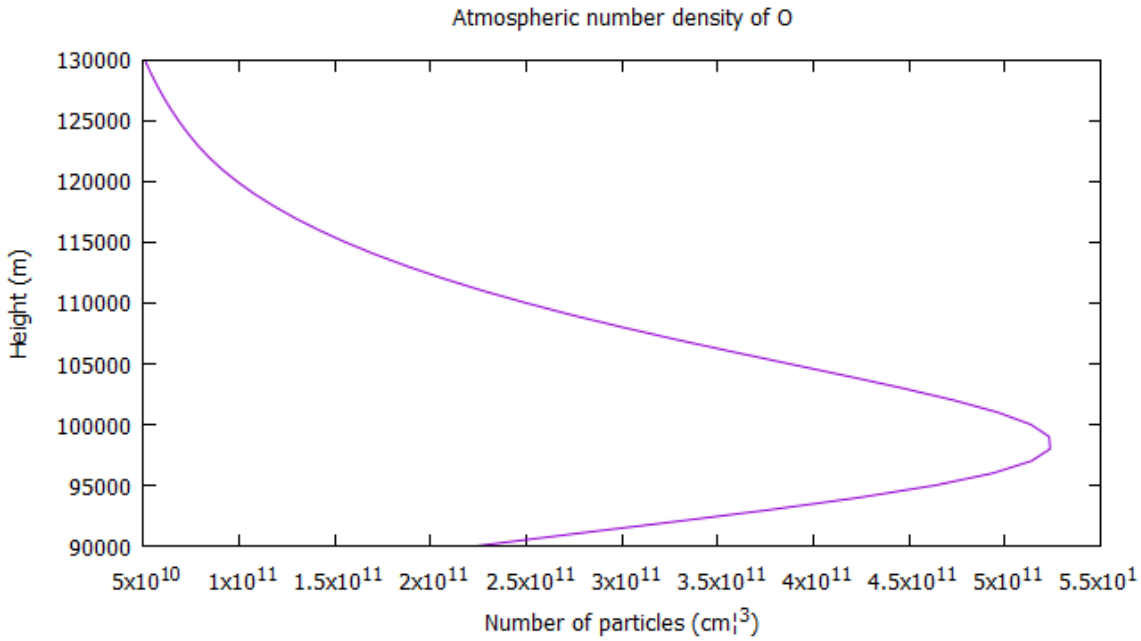

Figure 2.4: Atmospheric number density of *O*. Taken from (7)

action with the electron. For that we get a random number $\Re$ and:

$$\text{select } s = 1 \text{ if: } 0 \leq \Re < \frac{\sigma_1 n_1}{\sigma_1 n_1 + \sigma_2 n_2 + \sigma_3 n_3} \tag{2.18}$$

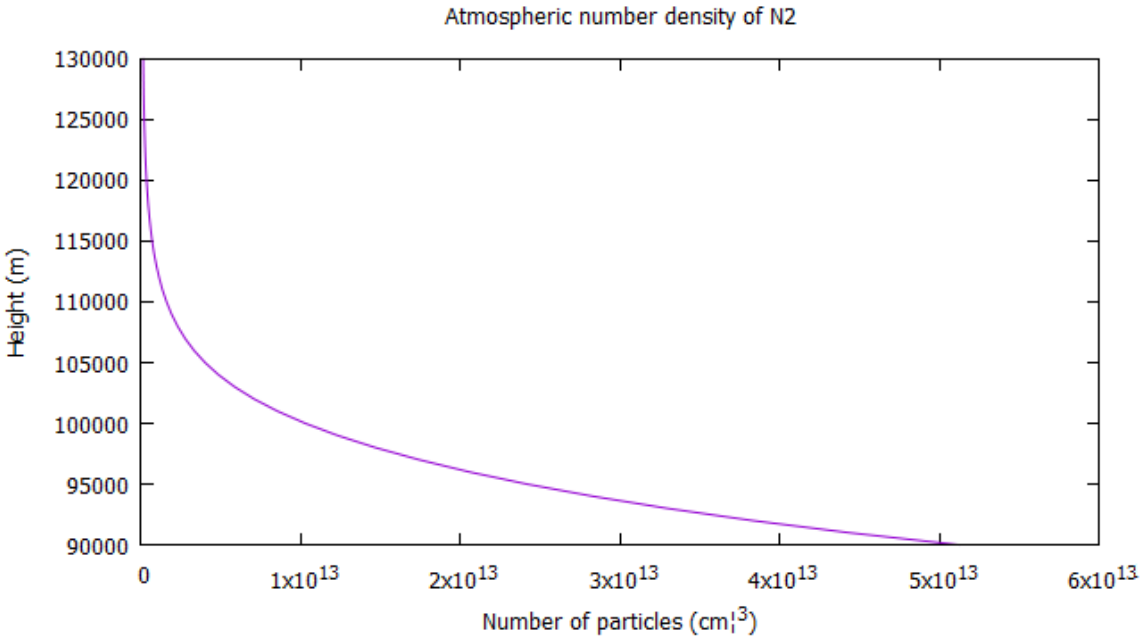

Figure 2.5: Atmospheric number density of N_2. Taken from (7)

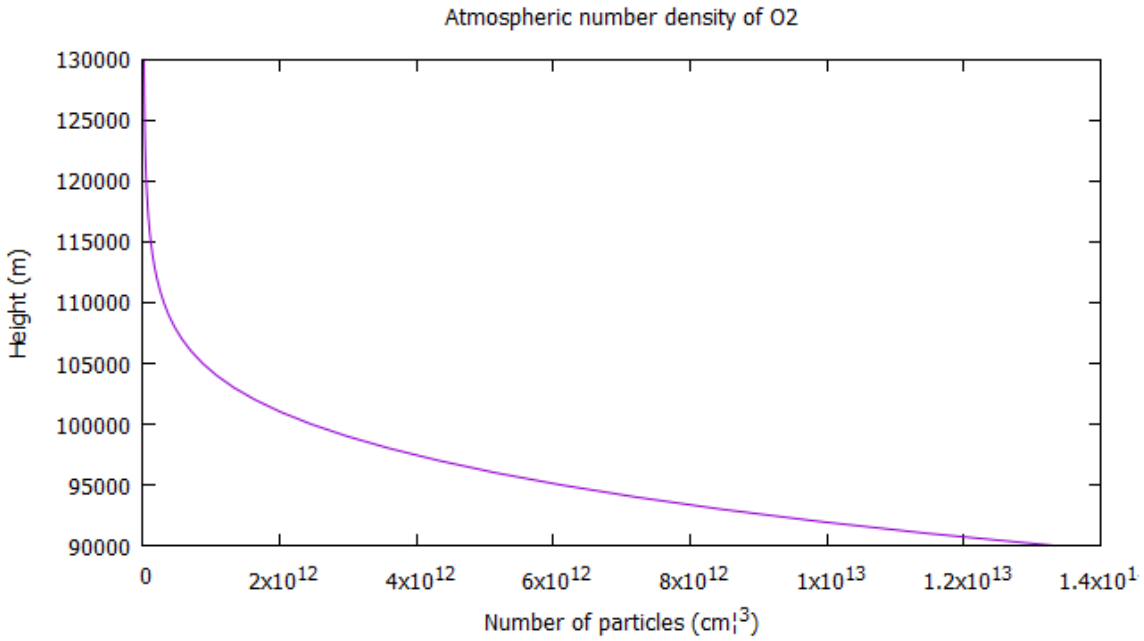

Figure 2.6: Atmospheric number density of O_2. Taken from (7)

$$\text{select } s = 2 \text{ if: } \frac{\sigma_1 n_1}{n_1 + n_2 + n_3} \leq \Re < \frac{\sigma_1 n_1 + \sigma_2 n_2}{\sigma_1 n_1 + \sigma_2 n_2 + \sigma_3 n_3}$$

$$\text{select } s = 3 \text{ if: } \frac{\sigma_1 n_1 + \sigma_2 n_2}{\sigma_1 n_1 + \sigma_2 n_2 + \sigma_3 n_3} \leq \Re < 1$$

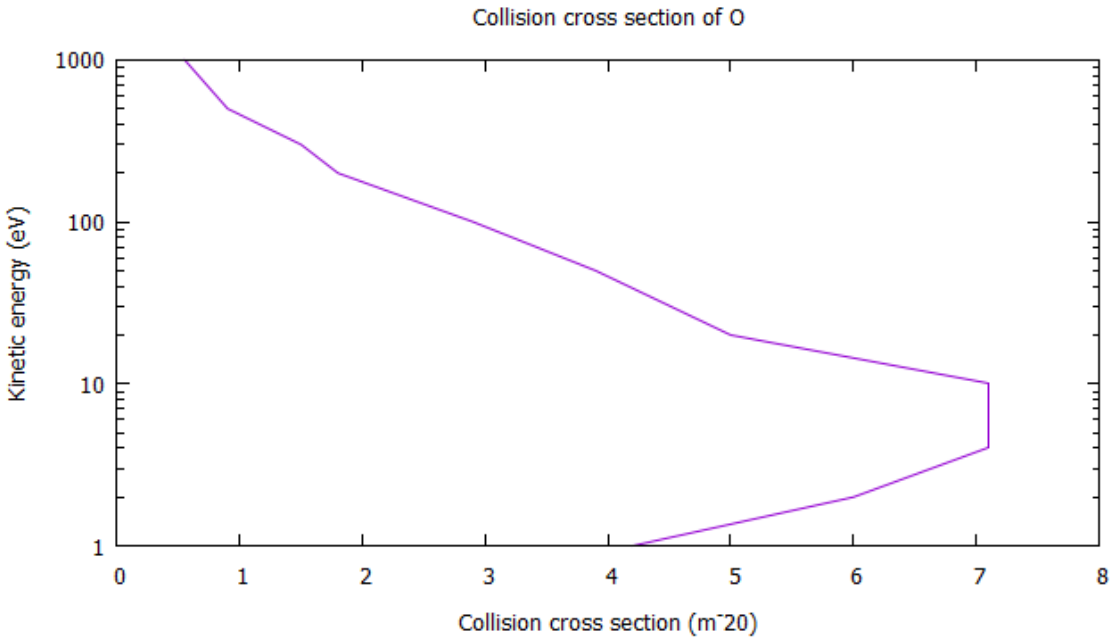

Figure 2.7: Collision cross section of O. Taken from (3)

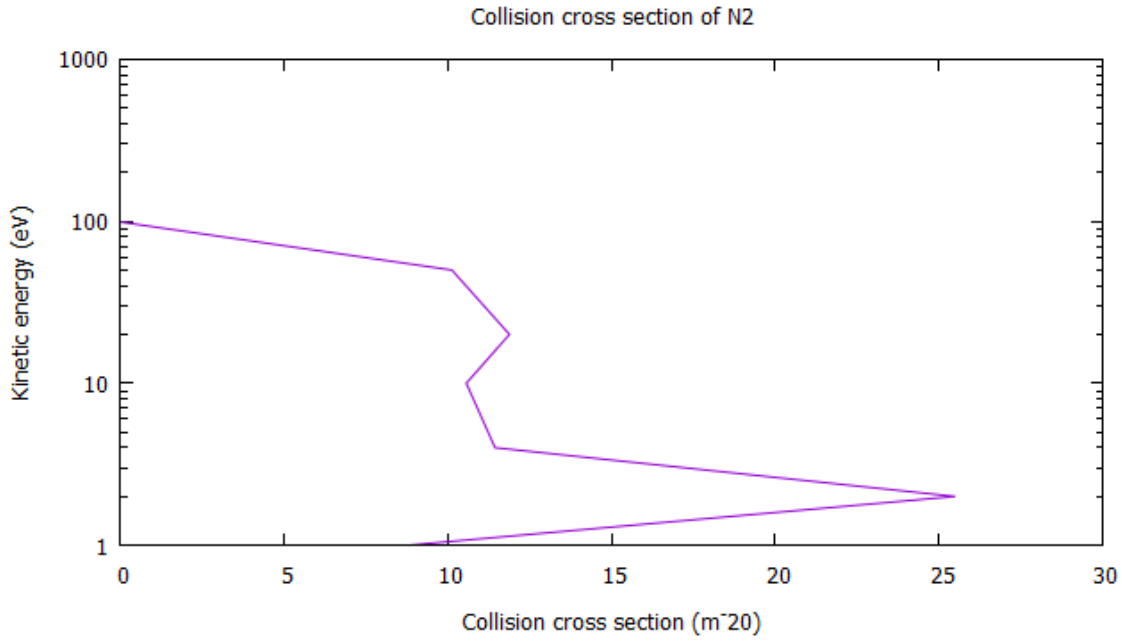

Figure 2.8: Collision cross section of N_2. Taken from (6)

where σ_1, σ_2 and σ_3 are interaction cross sections at the end of integration.

Once we have chosen one of gas species we are obtaining particle velocity in the same way we did for electrons. For that we use f_b - distribution of velocities of atmospheric particles:

$$f_b\left(v_a(\vec{r})\right) d^3v_a = \pi^{-3/2}\frac{1}{\bar{v}^3}e^{-\frac{{v_a}^2}{\bar{v}^2}} d^3v_a = \pi^{-3/2}\frac{1}{\bar{v}^3}e^{-\frac{{v_\varrho}^2}{\bar{v}^2}} e^{-\frac{{v_z}^2}{\bar{v}^2}} v_\varrho dv_\varrho d\varrho dv_z \tag{2.19}$$

$$1 = \int_{-\infty}^{+\infty} f_b\left(v_a(\vec{r})\right) d^3v_a \tag{2.20}$$

To determine φ_a we do:

$$\Re = \int_0^{\varphi_a} d\varphi \int_0^{\infty} dv_\varrho \int_{-\infty}^{\infty} dv_z \frac{v_\varrho}{\pi^{3/2}{\bar{v}_a}^3} e^{-\frac{{v_\varrho}^2}{\bar{v}^2}} e^{-\frac{{v_z}^2}{\bar{v}^2}} = \varphi_a \frac{1}{2\pi} \tag{2.21}$$

where $\bar{v} = \sqrt{\frac{2kT}{m_a}}$. $v_{\varrho,a}$ is determine:

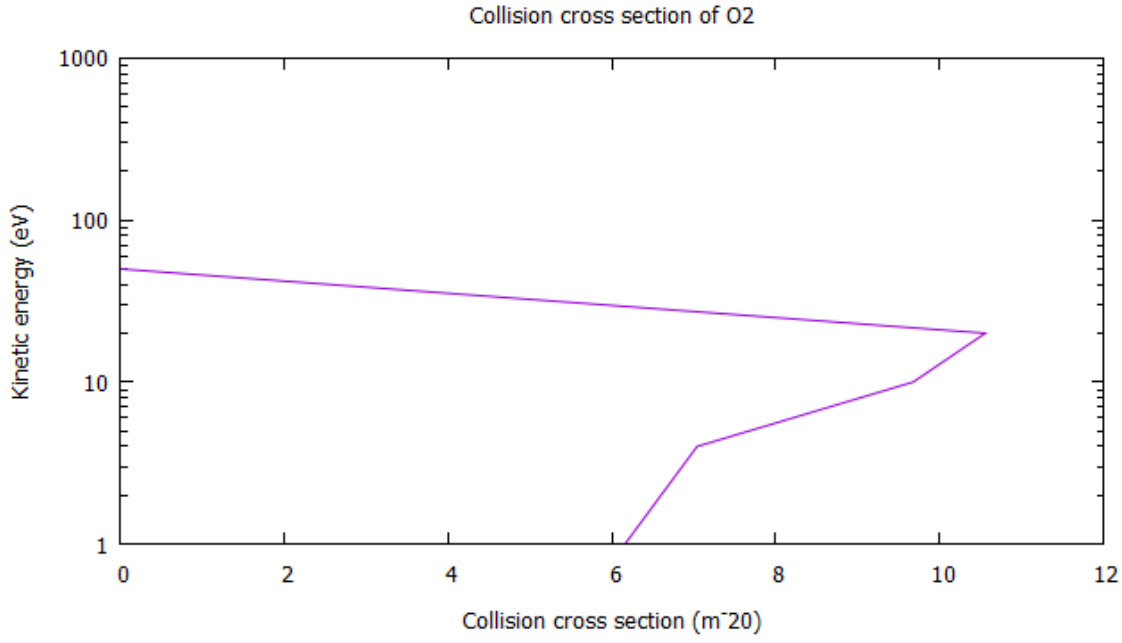

Figure 2.9: Collision cross section of O_2. Taken from (6)

$$\Re = \frac{2\pi \int_0^{v_{\varrho,a}} v_\varrho dv_\varrho \int_{-\infty}^{+\infty} f_b(\varrho_a, v_\varrho, v_z)}{2\pi \int_0^{+\infty} v_\varrho dv_\varrho \int_{-\infty}^{+\infty} f_b(\varrho_a, v_\varrho, v_z)} = 2\pi \frac{1}{\pi^{3/2} \bar{v_a}^3} \int_0^{v_{\varrho,a}} \frac{1}{2} dv_\varrho{}^2 e^{-\frac{v_\varrho{}^2}{\bar{v}^2}} \bar{v}\sqrt{\pi} \tag{2.22}$$

$$= \frac{1}{\bar{v}^2} | - \bar{v}^2 e^{-\frac{v_{\varrho,a}{}^2}{\bar{v}^2}} |_0^{v_{\varrho,a}} = 1 - e^{-\frac{v_{\varrho,a}{}^2}{\bar{v}^2}}$$

$$v_{\varrho,a} = \bar{v}\sqrt{\ln(\Re)} \tag{2.23}$$

and finally v_z is:

$$\Re = \frac{\int_{-\infty}^{v_{z,a}} f_b(\varrho_a, v_{\varrho,a}, v_z) dv_z}{\int_{-\infty}^{+\infty} f_b(\varrho_a, v_{\varrho,a}, v_z) dv_z} = \frac{1}{\bar{v}\sqrt{\pi}} \int_{-\infty}^{v_{z,a}} e^{-\frac{v_z{}^2}{\bar{v}^2}} dv_z = \frac{1}{\sqrt{\pi}} \int_{-\infty}^{\frac{v_{z,a}}{\bar{v}}} e^{-x^2} dx \tag{2.24}$$

We calculate 2.24 using two different integrals depending on $\Re$:

$$\Re < 0.5 => \frac{1}{\pi} \int_{\frac{|v_{z,a}|}{\bar{v}}}^{+\infty} e^{-x^2} dx = \frac{1}{\pi} (\int_0^{+\infty} e^{-x^2} dx - \int_0^{\frac{|v_{z,a}|}{\bar{v}}} e^{-x^2} dx) = \frac{1}{2} - \frac{1}{2} Erf(\frac{|v_{z,a}|}{\bar{v}}) \tag{2.25}$$

$$\Re >= 0.5 => \frac{1}{\pi} (\int_{-\infty}^{0} e^{-x^2} dx + \int_0^{\frac{|v_{z,a}|}{\bar{v}}} e^{-x^2} dx) = \frac{1}{2} + \frac{1}{2} Erf(\frac{|v_{z,a}|}{\bar{v}}) \tag{2.26}$$

Now we know velocity of target particle $\vec{v_a} = v_{\varrho,a} cos(\varphi_a)\hat{i} + v_{\varrho,a} sin(\varphi_a)\hat{j} + v_{z,a}\hat{k}$. We need the velocity after scatering. To determine velocities after scattering first we randomly choose unitary vector $\hat{n}$:

$$\varphi_n = 2\pi\Re \tag{2.27}$$

$$\theta_n = \pi\Re$$

$$\hat{n} = cos(\varphi_n)sin(\theta_n)\hat{i} + sin(\varphi_n)cos(\theta_n)\hat{j} + cos(\theta_n)\hat{k}$$

Now we can determine electron velocity after scattering:

$$\vec{v_e}' = \vec{v_c} + v'\hat{n} \tag{2.28}$$

where $\vec{v_c}$ is center of mass velocity and v' is determine from conservation of momentum:

$$\vec{v_c} = \frac{m_e\vec{v_e} + m_a\vec{v_a}}{m_e + m_a} \tag{2.29}$$

$$v' = \frac{m_a}{m_a + m_e}|\vec{v_e} - \vec{v_a}| \tag{2.30}$$

2.5 Ionization

For electrons with higher energies, in addition to elastic collisions, we are introducing possibility of ionization. For that we add another three members in summation 2.17, one for each gas species:

$$\sum \Delta t \;\; v_e(\vec{r}) \sum_{s=1}^{6} \sigma_s \left(v_e(\vec{r})\right) n_s(\vec{r}) = \boldsymbol{P} \tag{2.31}$$

where $\sigma_1\left(\bar{v}\right)$,$\sigma_2\left(\bar{v}\right)$ and $\sigma_3\left(\bar{v}\right)$ are cross sections for elastic collision and $\sigma_4\left(\bar{v}\right)$, $\sigma_5\left(\bar{v}\right)$ and $\sigma_6\left(\bar{v}\right)$ are cross sections for ionisation of each gas species. Cross sections $\sigma_1\left(\bar{v}\right)$ and $\sigma_4\left(\bar{v}\right)$ correspond to the same gas species and different interactions among particles. The same goes for other two pairs $\sigma_2\left(\bar{v}\right)$ and $\sigma_5\left(\bar{v}\right)$, as well as $\sigma_3\left(\bar{v}\right)$ and $\sigma_6\left(\bar{v}\right)$. Cross sections for ionisation used in this simulation are shown in figures 2.10 (taken from (3)), 2.11 and 2.12 (taken from (6)). Atmospheric number density of species n_1 is obviously equal to n_4 because both are referring to the same gas species. The same goes for n_2 and n_5 and of course n_3 and n_6. At

the end of integration we are choosing between ionization and elastic collision much the same way we do between gas species in 2.18:

$$\text{select elastic collision if: } 0 \leq \Re < \frac{\sigma_1 n_1 + \sigma_2 n_2 + \sigma_3 n_3}{\sigma_1 n_1 + \sigma_2 n_2 + \sigma_3 n_3 + \sigma_4 n_4 + \sigma_5 n_5 + \sigma_6 n_6} \tag{2.32}$$

$$\text{select ionisation if: } \frac{\sigma_1 n_1 + \sigma_2 n_2 + \sigma_3 n_3}{\sigma_1 n_1 + \sigma_2 n_2 + \sigma_3 n_3 + \sigma_4 n_4 + \sigma_5 n_5 + \sigma_6 n_6} \leq \Re$$

If elastic collision occurs then we select a gas species in the way described in equation 2.18. In a case of ionization we use the same equation, just replacing $\sigma_1(\bar{v})$ with $\sigma_4(\bar{v})$, $\sigma_2(\bar{v})$ with $\sigma_5(\bar{v})$ and $\sigma_3(\bar{v})$ with $\sigma_6(\bar{v})$. Once we have chosen one of the gas species we do the same calculation as for elastic collision except we are colliding electron with another electron. Target electron in calculation is considered as stationary before collision. After collision a new electron is treated as a free electron as all others in the simulation.

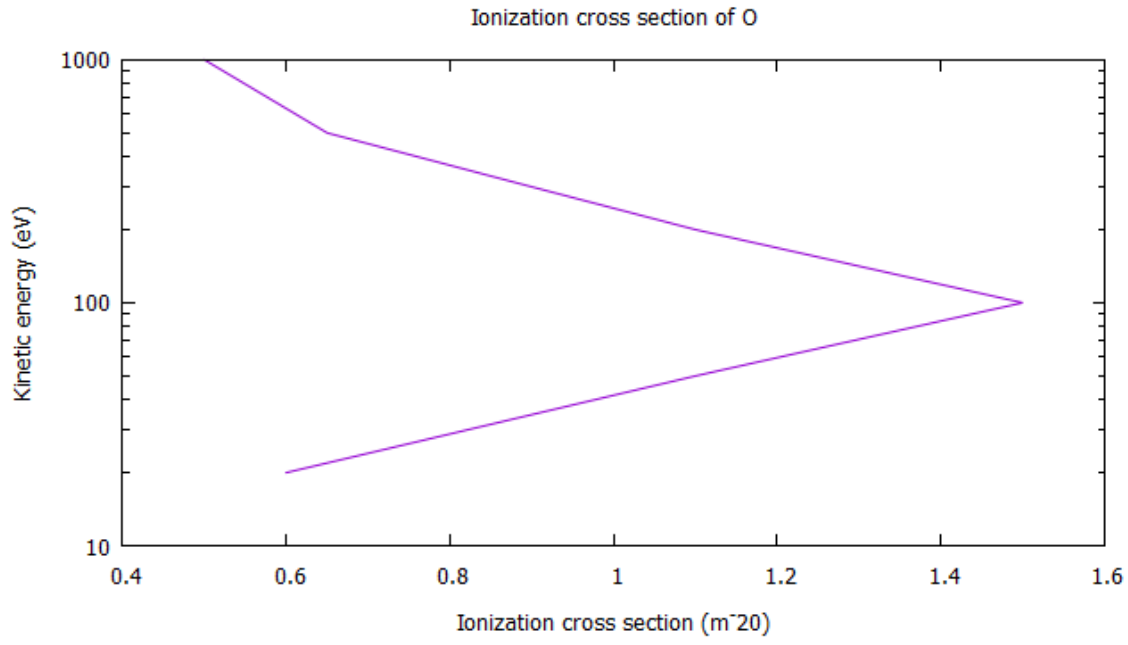

Figure 2.10: Ionization cross section of *O*. Taken from (3)

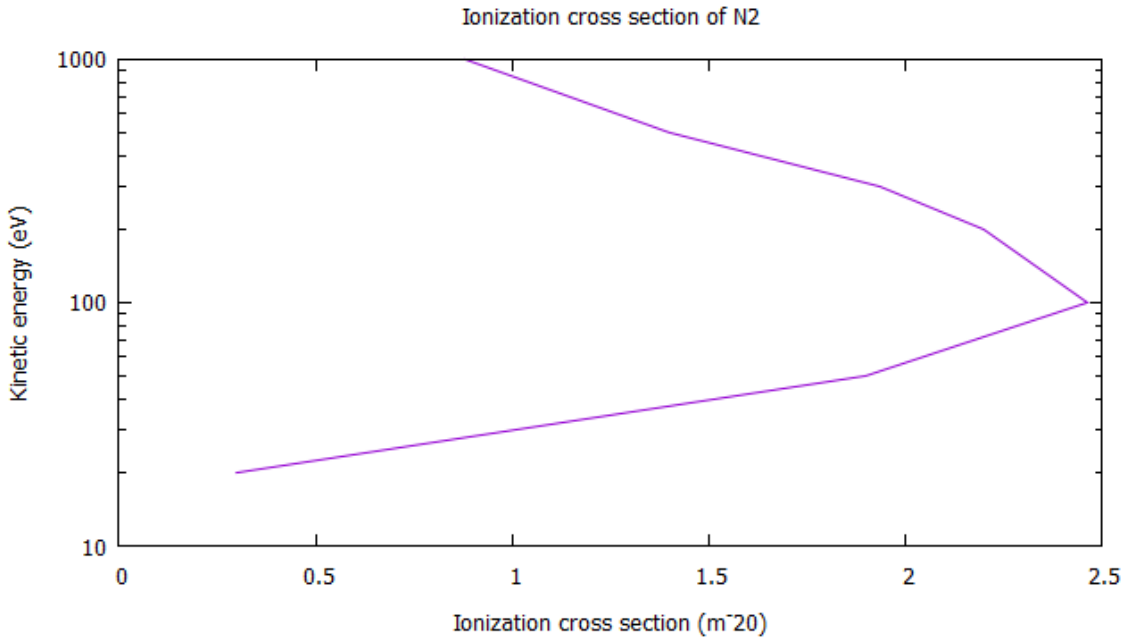

Figure 2.11: Ionization cross section of N_2. Taken from (6)

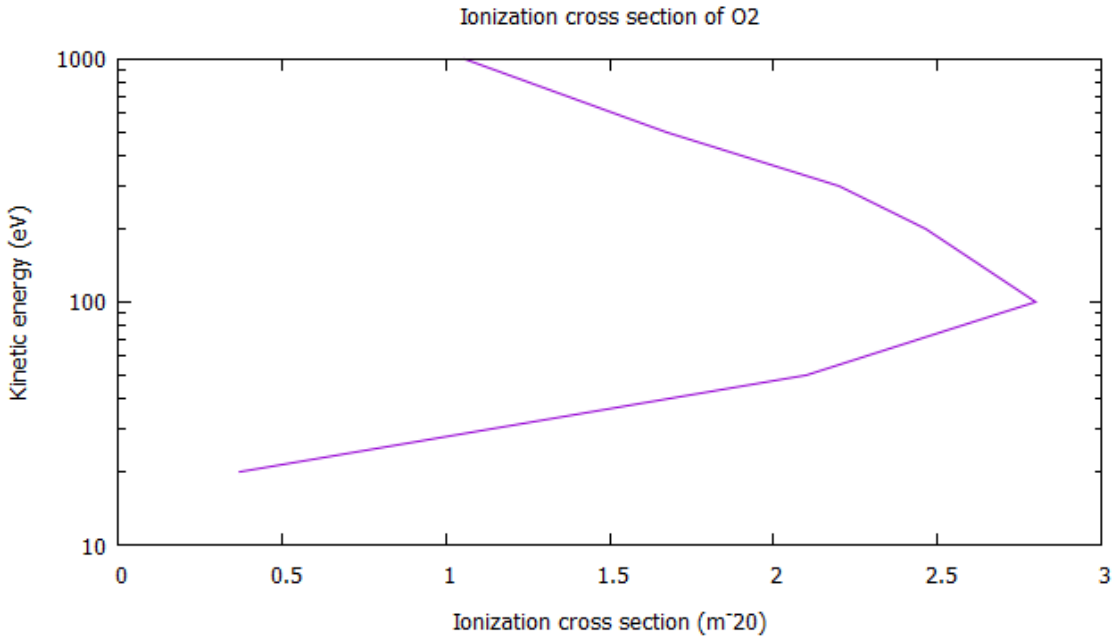

Figure 2.12: Ionization cross section of O_2. Taken from (6)

2.6 Airglow caused by accelerated electrons

As mentioned in introduction, camera that recorded images in figure 1.1 is sensitive to light in the wavelength range 500-900*nm* with peak sensitivity at 700 *nm*. To reconstruct airglow seen in figure 1.1 we consider excitation of two major electronic levels: $O(^1S)$ and $N_2(B^3\pi_g)$ by electron impacts. The excitation leads to emissions of oxygen green line 557.7*nm* and of N_2(*1P*) band in 478-2531 *nm* range (1). The oxygen red line emission in visual can be neglected below 130*km* because it is strongly quenched.

In this simulation we are recording positions of electron collisions with O and N_2, which have the largest probability for emissions. From figures 2.13 and 2.14 we can see that collisions in which electrons have kinetic energy in range 6-50 *eV* have the largest probability for such emissions. So we are printing positions of collisions with the largest probability for generating photons that would be caught on camera.

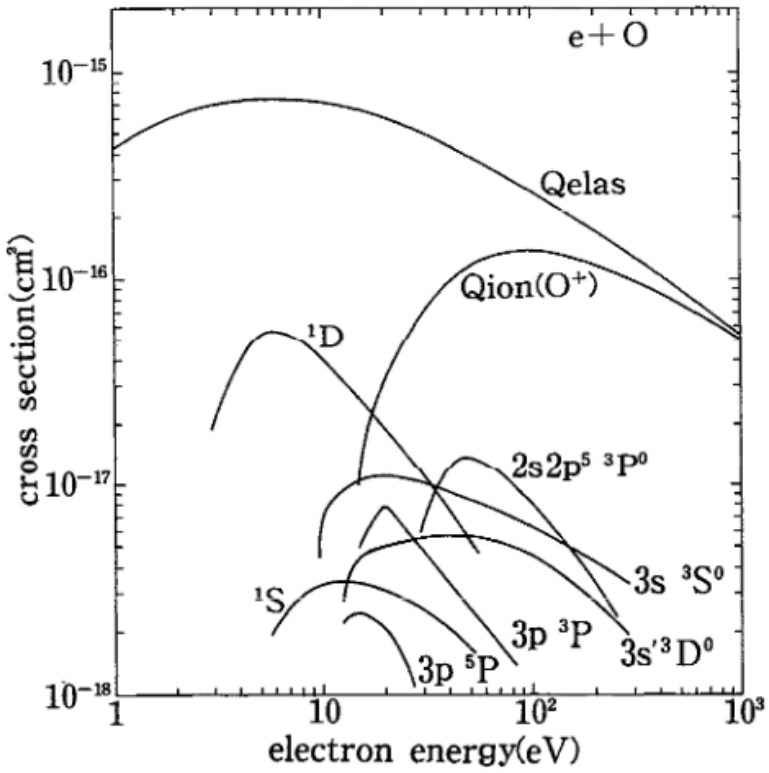

Figure 2.13: Cross sections for the electron collision with atomic oxygen. Taken from (3)

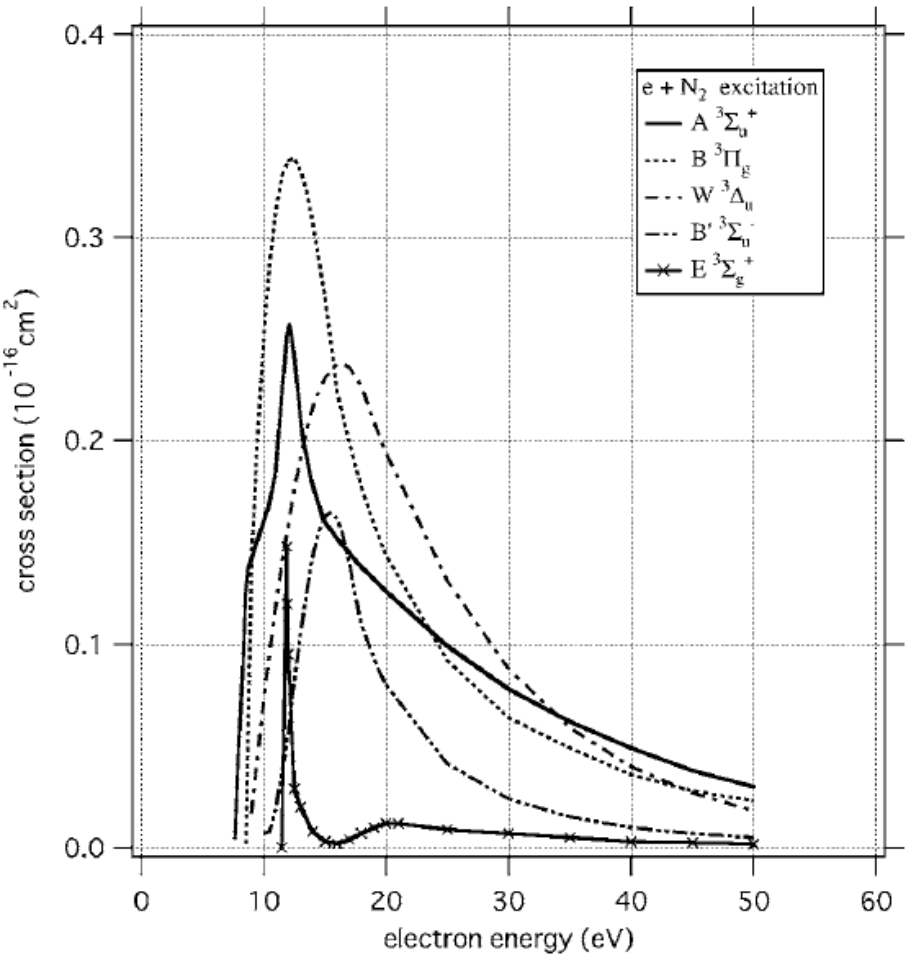

Figure 2.14: Recommended values of the cross sections for the excitation of electronic states of lower states. Taken from (4)

Chapter 3

Results

Histograms of collected photons at different radial distances from the meteor for altitudes of 105 and 107 *km* are shown in figures 3.1 and 3.1. We can clearly see that the highest number of photons are generated at radial distances from $\sim$100 to 300 *m*. Generation of photons is decreasing with an exponential rate as we move further away from the meteor.

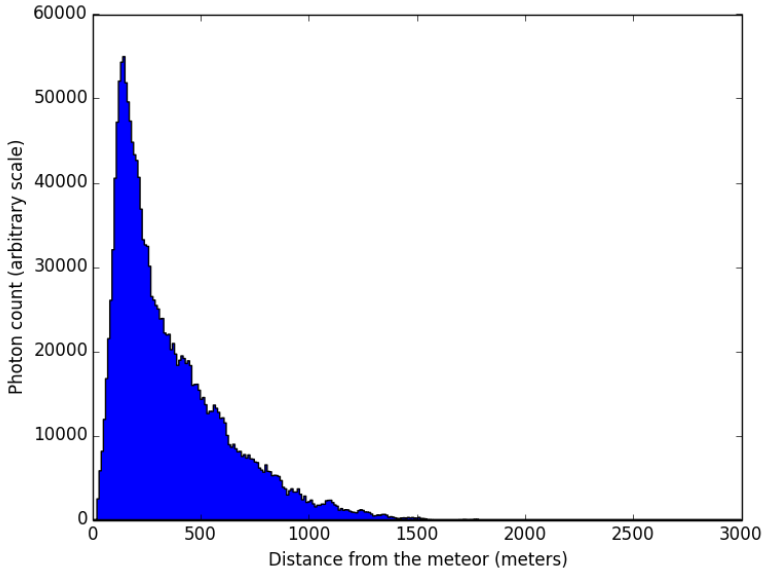

Figure 3.1: Histogram showing the number of generated photons depending on radial distance from the source of electric field at altitude of 105 *km* during 0.001 *s*. Electric field is 4000 *V/m* at 1 *m* distance from source.

Figures 3.3 and 3.4 are showing the spatial arrangement of photons generated at altitudes of 105 and 107 *km*. Figures are divided in pixels of 15 x 15 *m* in size. In every pixel we add photons and represent a bigger number as a brighter pixel.

Figures 3.6 and 3.5 are showing the histogram and spatial arrangement of photons at altitude of 107 *km* for a stronger electric field than previous figures. The electric field in previous figures is 4000 *V/m* at 1*m* from the center of meteor, while here it is 10000 *V/m*. As expected

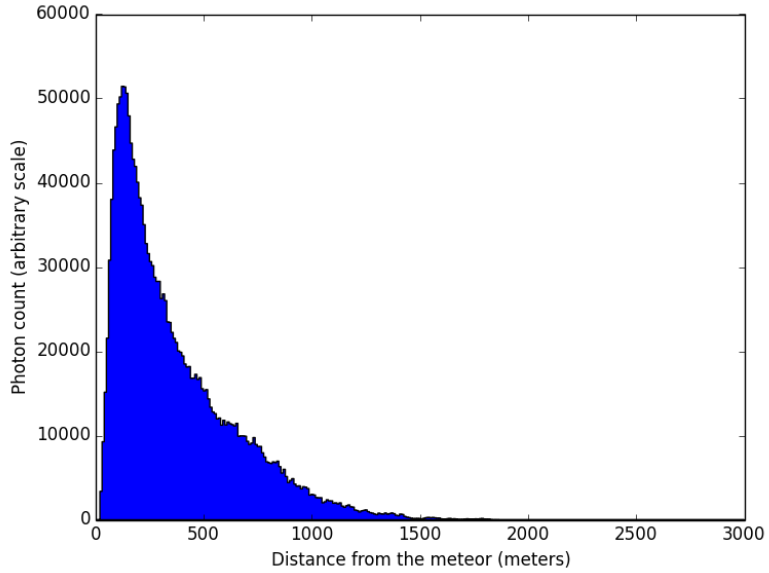

Figure 3.2: Histogram showing the number of generated photons depending on radial distance from the source of electric field at altitude of 107 *km* during 0.001 *s*. Electric field is 4000 *V/m* at 1 *m* distance from source.

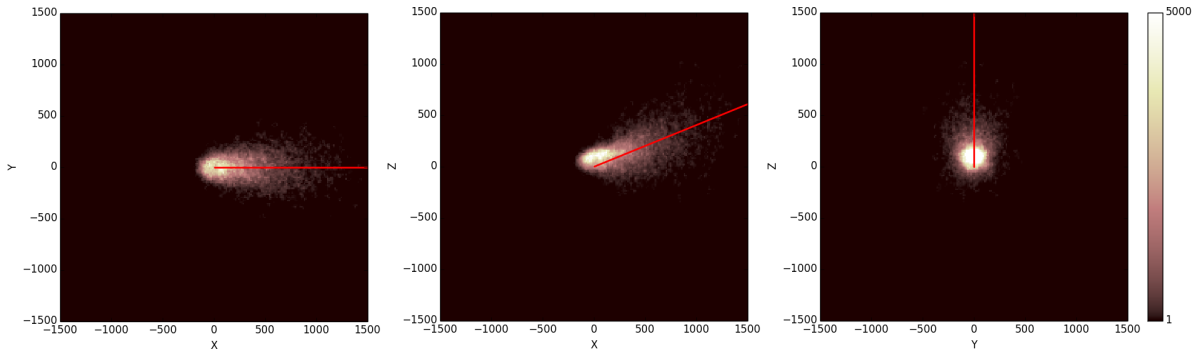

Figure 3.3: Spatial arrangement of photons around the meteor at altitude of 105 *km* during 0.001 *s*. Electric field is 4000 *V/m* at 1 *m* distance from source. Images have pixels of 15 x 15 *m* in size.

the number of photons generated is larger for a stronger field, but the greatest number of photons is still generated at the radial distances of $\sim$ 100 to 300 *m*.

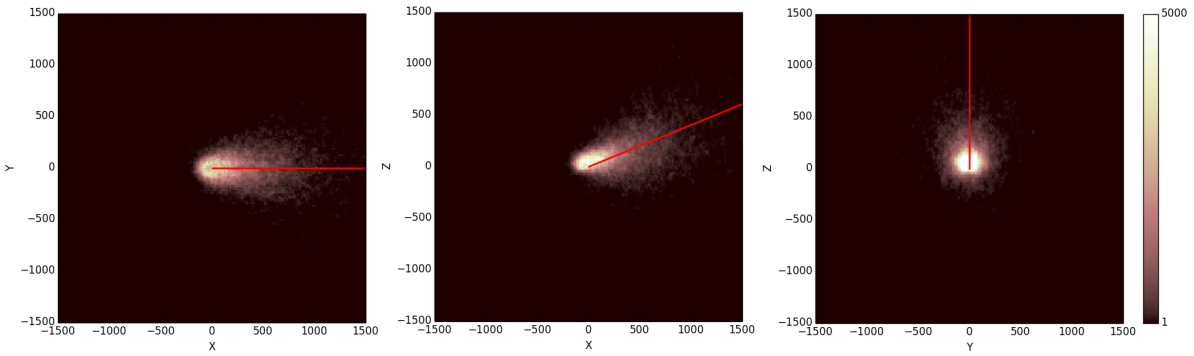

Figure 3.4: Spatial arrangement of photons around the meteor at altitude of 107 *km* during 0.001 *s*. Electric field is 4000 *V/m* at 1 *m* distance from source. Images have pixels of 15 x 15 *m* in size.

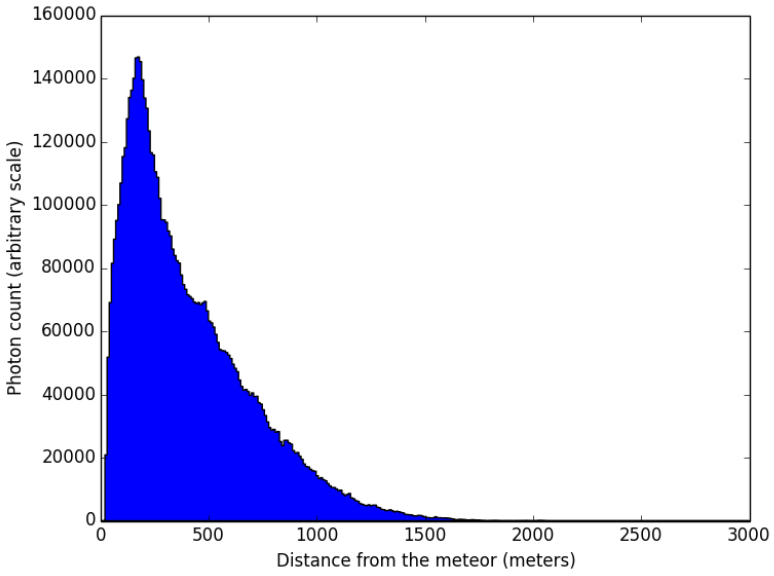

Figure 3.5: Histogram showing the number of generated photons depending on radial distance from the source of electric field at altitude of 107 *km* during 0.001 *s*. Electric field is 10000 *V/m* at 1 *m* distance from source.

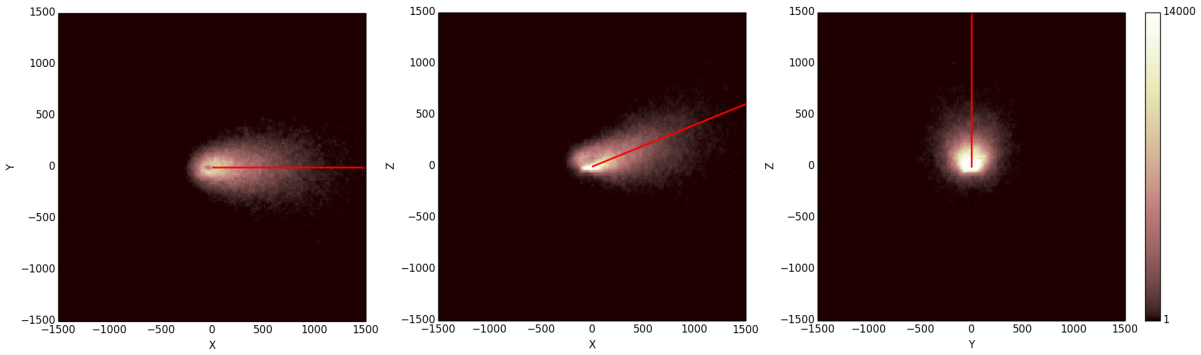

Figure 3.6: Spatial arrangement of photons around the meteor at altitude of 107 *km* during 0.001 *s*. Electric field is 10000 *V/m* at 1 *m* distance from source. Images have pixels of 15 x 15 *m* in size.

Chapter 4

Discussion and conclusions

With this simulation we tested a possibility of reproducing the shock-like airglow structure with accelerated electrons in meteor induced field. Electrons reach kinetic energies high enough to excite oxygen and nitrogen through collisions. Even though we were not able to reproduce exact shape of the shock-like structure, the size and overall resemblance is promising. Especially if we consider that our electric field is just a simple point source approximation. As shown in figures 3.1 and 3.2 the largest number of photons is generated at radial distance of $\sim$ 100 to 300 m from the meteor, which agrees with the observed airglow size. As mentioned before, the shape of structure given by simulation figures 3.3 and 3.4 closely resembles the observed one. Furthermore, it must be mentioned that no other source of radiation has been accounted for in simulation, so the missing tail is expected.

Much more work is needed to show that the meteor induced electrical field is responsible for the formation of such a structure. In this simulation the meteor induced electrical field is represented by electric field of a point-like charge, which could be replaced by more exact representation of the solution given in (1). Also, the magnetic field constrains the movement of electrons to their acceleration only in direction parallel to the magnetic field, but a full 3D

movement could be modeled in the future. Another improvement is possible in calculation of photon emission. We consider that every collision of electrons with air molecules results in generation of photons if electrons have kinetic energy in range 6 to 50 eV, but this could be replaced with the exact cross-section for each type of excitation.

Appendices

Appendix A

Code

```
#include <iostream>
#include <fstream>
#include <math.h>
#include <array>
#include <vector>
#include <iomanip>

#ifndef ELEKTRON_H
#define ELEKTRON_H
class elektron {
public:
elektron();

elektron(double, double, double, double, double, double, double);
~elektron();

int getInVis() const;
double getBrzinaVx() const;
double getBrzinaVy() const;
double getBrzinaVz() const;
double getRadiVx() const;
double getRadiVy() const;
double getRadiVz() const;
double getTr() const;

void setInVis(int);
void setRadiVx(double);
void setRadiVy(double);
void setRadiVz(double);
void setBrzinaVx(double);
void setBrzinaVy(double);
void setBrzinaVz(double);
void setTr(double);
```

```
private:
double nradiV[3];
double nbrzinaV[3];
int nInVis;
double nTr;
};
#endif

elektron::elektron(){
for (int i = 0; i <= 2; i++){
nbrzinaV[i] = 0.0;
nradiV[i] = 0.0;
}
}

elektron::elektron(double Tr, double brzinax, double brzinay,
 double brzinaz, double radix, double radiy, double radiz){
nTr = Tr;
nbrzinaV[0] = brzinax;
nradiV[0] = radix;
nbrzinaV[1] = brzinay;
nradiV[1] = radiy;
nbrzinaV[2] = brzinaz;
nradiV[2] = radiz;

if (radiz<130000) nInVis = (int)round(radiz / 1000.0 - 90.0);
if (radiz>130000) nInVis = 40;
if (radiz<90) nInVis = 0;
}

elektron::~elektron(){
}

int elektron::getInVis() const {
return nInVis;
}
double elektron::getTr() const {
return nTr;
}
double elektron::getBrzinaVx() const {
return nbrzinaV[0];
}
double elektron::getBrzinaVy() const {
return nbrzinaV[1];
}
double elektron::getBrzinaVz() const {
return nbrzinaV[2];
```

```
}
double elektron::getRadiVx() const {
return nradiV[0];
}
double elektron::getRadiVy() const {
return nradiV[1];
}
double elektron::getRadiVz() const {
return nradiV[2];
}

void elektron::setInVis(int inVis){
nInVis = inVis;
}
void elektron::setTr(double Tr){
nTr = Tr;
}
void elektron::setBrzinaVx(double brzina){
nbrzinaV[0] = brzina;
}
void elektron::setBrzinaVy(double brzina){
nbrzinaV[1] = brzina;
}
void elektron::setBrzinaVz(double brzina){
nbrzinaV[2] = brzina;
}
void elektron::setRadiVx(double koor){
nradiV[0] = koor;
}
void elektron::setRadiVy(double koor){
nradiV[1] = koor;
}
void elektron::setRadiVz(double koor){
nradiV[2] = koor;
}

#ifndef POLJE_H
#define POLJE_H
class polje {
public:
polje();

polje(double, double, double, double,double,double,double);
~polje();

double getQ() const;
double getBrzinaVx() const;
```

```
double getBrzinaVy() const;
double getBrzinaVz() const;
double getRadiVx() const;
double getRadiVy() const;
double getRadiVz() const;

void setQ(double);
void setBrzinaVx(double);
void setBrzinaVy(double);
void setBrzinaVz(double);
void setRadiVx(double);
void setRadiVy(double);
void setRadiVz(double);

private:
double nbrzinaV[3];
double nradiV[3];
double nQ;
};
#endif

polje::polje(){
nQ = 0;
for (int i = 0; i <= 2; i++){
nbrzinaV[i] = 0.0;
nradiV[i] = 0.0;
}
}

polje::polje(double C, double radix, double radiy, double radiz,
 double vx, double vy, double vz){
nradiV[0] = radix;
nradiV[1] = radiy;
nradiV[2] = radiz;
nbrzinaV[0] = vx;
nbrzinaV[1] = vy;
nbrzinaV[2] = vz;
nQ = C;
}

polje::~polje(){
}

double polje::getBrzinaVx() const {
return nbrzinaV[0];
}
double polje::getBrzinaVy() const {
```

```
return nbrzinaV[1];
}
double polje::getBrzinaVz() const {
return nbrzinaV[2];
}
double polje::getRadiVx() const {
return nradiV[0];
}
double polje::getRadiVy() const {
return nradiV[1];
}
double polje::getRadiVz() const {
return nradiV[2];
}
double polje::getQ() const {
return nQ;
}

void polje::setBrzinaVx(double brzina){
nbrzinaV[0] = brzina;
}
void polje::setBrzinaVy(double brzina){
nbrzinaV[1] = brzina;
}
void polje::setBrzinaVz(double brzina){
nbrzinaV[2] = brzina;
}
void polje::setRadiVx(double koor){
nradiV[0] = koor;
}
void polje::setRadiVy(double koor){
nradiV[1] = koor;
}
void polje::setRadiVz(double koor){
nradiV[2] = koor;
}
void polje::setQ(double konst){
nQ = konst;
}

#define IM1 2147483563
#define IM2 2147483399
#define AM (1.0/IM1)
#define IMM1 (IM1-1)
#define IA1 40014
#define IA2 40692
#define IQ1 53668
```

```
#define IQ2 52774
#define IR1 12211
#define IR2 3791
#define NTAB 32
#define NDIV (1+IMM1/NTAB)
#define EPS 1.e-14
#define RNMX (1.0-EPS)

double ran2(int *idum)
{
int j;
int k;
static int idum2 = 123456789;
static int iy = 0;
static int iv[NTAB];
double temp;

if (*idum <= 0) {                    // *idum < 0 ==> initialize
if (-(*idum) < 1)
*idum = 1;
else
*idum = -(*idum);
idum2 = (*idum);

for (j = NTAB + 7; j >= 0; j--) {
k = (*idum) / IQ1;
*idum = IA1*(*idum - k*IQ1) - k*IR1;
if (*idum < 0) *idum += IM1;
if (j < NTAB) iv[j] = *idum;
}
iy = iv[0];
}
k = (*idum) / IQ1;
*idum = IA1*(*idum - k*IQ1) - k*IR1;
if (*idum < 0) *idum += IM1;

k = idum2 / IQ2;
idum2 = IA2*(idum2 - k*IQ2) - k*IR2;
if (idum2 < 0) idum2 += IM2;

j = iy / NDIV;
iy = iv[j] - idum2;
iv[j] = *idum;
if (iy < 1) iy += IMM1;

if ((temp = AM*iy) > RNMX)
return RNMX;
```

```
else
return temp;
}

#undef IM1
#undef IM2
#undef AM
#undef IMM1
#undef IA1
#undef IA2
#undef IQ1
#undef IQ2
#undef IR1
#undef IR2
#undef NTAB
#undef NDIV
#undef EPS
#undef RNMX

#define PI 3.14159265
#define inVisina 41
#define vrstaCestice 3
#define brElektrona 10000

using namespace std;

double pv(double, double);
double vRoMete(double,double);
double vCM(double,double, double, double);
double konV(double,double,double,double,double[],double);
double InErf(double);
double Ek(double, double);
double PPc(int,double,double[][11]);
double PPi(int,double,double[][7]);
int GetInCesMet(double, double, double, double,double,double,double);
double Sudar(double, double, double, double, double, double,
 double, double, double );

int main() {
bool coll;
int inCesMet;
int idum = 3;
double delH = 500.0;
double delT = 0.000001;
double D,EK,v2,TT,N,PP, V, RAN1,RAN,  ROVMete, prosBrz;
double  VMete[3], Vn[3], Vc[3],rd[3],aiz,afz;
```

```
const double me = 9.109E-31;
const double qe = -1.602E-19;

const double mO = 15.999*1.6605E-27;
const double mN = 14.007*1.6605E-27);
const double mO2 = 2 * mO ;
const double mN2 = 2 * mN ;

double koncetracija[vrstaCestice][inVisina];
double UPc[vrstaCestice][11];
double UPi[vrstaCestice][7];
double maseCes[vrstaCestice] = { mO, mN2, mO2};
double tempe[inVisina];
ifstream uV,udarniC,udarniI;
ofstream f108,f107,f106,f105,b108,b107,b106,b105,kor105,kor106,kor107
,kor108;

uV.open("ulazVis.txt");
if (uV.fail()){       ///provjera
cerr << "EEE" << endl;
exit(1);
}

for (int i = 0; i < inVisina; i++){
for (int j = 0; j < vrstaCestice; j++){
uV >> koncetracija[j][i];
}
uV >> tempe[i];
}
uV.close();

udarniC.open("UPc.txt");
if (udarniC.fail()){
cerr << "CCC" << endl;
exit(1);
}

for (int i = 0; i < vrstaCestice; i++){
for (int j = 0; j < 11; j++){
udarniC >> UPc[i][j];
}
}
udarniC.close();

udarniI.open("UPi.txt");
if (udarniI.fail()){
cerr << "III" << endl;
```

```
exit(1);
}

for (int i = 0; i < vrstaCestice; i++){
for (int j = 0; j < 7; j++){
udarniI >> UPi[i][j];
}
}
udarniI.close();

vector<elektron> niz;

for (int j = 0; j < brElektrona; j++){
double z = (double)ran2(&idum) * 53389 ;
RAN = (double)ran2(&idum)*2*PI;
RAN1=(double)ran2(&idum);
double x = 10000 * RAN1 * cos(RAN);
double y = 10000 * RAN1 * sin(RAN);
double z1 = -x*sin(0.0174532925*(68)) + z*cos(0.0174532925*(68))+100000;
double x1 = x*cos(0.0174532925*(68)) + z*sin(0.0174532925*(68));
int InVis;
if (z<130000)  InVis = (int)round(z1 / 1000.0 - 90.0);
else InVis = 40;

prosBrz = pv(me, tempe[InVis]);

RAN = (double)ran2(&idum);
ROVMete = vRoMete(prosBrz, (double)ran2(&idum));
VMete[0] = ROVMete*cos(2 * PI*RAN);
VMete[1] = ROVMete*sin(2 * PI*RAN);
RAN = (double)ran2(&idum) - 0.5;
if (RAN < 0.0) VMete[2] = -prosBrz*InErf(2 * -RAN);
else VMete[2] = prosBrz*InErf(2 * RAN);

elektron AAA(-log((double)ran2(&idum)), VMete[0], VMete[1], VMete[2],
 x1, y,z1 );
niz.push_back(AAA);
}

double z = 53389/2 ;

polje EL(10000.0, z*sin(0.0174532925*(68)), 0.0, z*cos(0.0174532925*(68)
)+100000, -sin(0.0174532925*(68)) * 71000, 0.0, -cos(0.0174532925*(68))
 * 71000); //E polje

f105.open("f105.txt");
f106.open("f106.txt");
```

```
f107.open("f107.txt");
f108.open("f108.txt");
b105.open("b105.txt");
b106.open("b106.txt");
b107.open("b107.txt");
b108.open("b108.txt");
kor105.open("kor105.txt");
kor106.open("kor106.txt");
kor107.open("kor107.txt");
kor108.open("kor108.txt");

TT = -5100 / EL.getBrzinaVz();
for (double u = 0.0; u < TT; u = u + delT){

if (EL.getRadiVz()<108000 && EL.getRadiVz()>108000 + EL.getBrzinaVz()
*0.001) kor108 << fixed << EL.getRadiVx() << " " << EL.getRadiVy()
 << " " << EL.getRadiVz() << endl;
if (EL.getRadiVz()<107000 && EL.getRadiVz()>107000 + EL.getBrzinaVz()
*0.001) kor107 << fixed << EL.getRadiVx() << " " << EL.getRadiVy()
 << " " << EL.getRadiVz() << endl;
if (EL.getRadiVz()<106000 && EL.getRadiVz()>106000 + EL.getBrzinaVz()
*0.001) kor106 << fixed << EL.getRadiVx() << " " << EL.getRadiVy()
<< " " << EL.getRadiVz() << endl;
if (EL.getRadiVz()<105000 && EL.getRadiVz()>105000 + EL.getBrzinaVz()
*0.001) kor105 << fixed << EL.getRadiVx() << " " << EL.getRadiVy()
 << " " << EL.getRadiVz() << endl;

EL.setRadiVx(EL.getRadiVx() + EL.getBrzinaVx()*delT);
EL.setRadiVy(EL.getRadiVy() + EL.getBrzinaVy()*delT);
EL.setRadiVz(EL.getRadiVz() + EL.getBrzinaVz()*delT);

for (int j = 0; j < (int)niz.size(); j++)        {

rd[0] = niz[j].getRadiVx() - EL.getRadiVx();
rd[1] = niz[j].getRadiVy() - EL.getRadiVy();
rd[2] = niz[j].getRadiVz() - EL.getRadiVz();

D = sqrt(rd[0] * rd[0] + rd[1] * rd[1] + rd[2] * rd[2]);
if (D < 20 ){
if (j + 1 == (int)niz.size()){
niz.pop_back();
}
else {
niz.erase(niz.begin() + j);
}
}
else{
```

```
if (niz[j].getTr() > 0.0){
rd[0] = rd[0] / D;
rd[1] = rd[1] / D;
rd[2] = rd[2] / D;

aiz = rd[2] * qe*EL.getQ() / (me*D*D);

niz[j].setRadiVx(niz[j].getRadiVx() + niz[j].getBrzinaVx()*delT );
niz[j].setRadiVy(niz[j].getRadiVy() + niz[j].getBrzinaVy()*delT );
niz[j].setRadiVz(niz[j].getRadiVz() + niz[j].getBrzinaVz()*delT
+ 0.5*aiz * delT*delT);

rd[0] = niz[j].getRadiVx() - EL.getRadiVx();
rd[1] = niz[j].getRadiVy() - EL.getRadiVy();
rd[2] = niz[j].getRadiVz() - EL.getRadiVz();

D = sqrt(rd[0] * rd[0] + rd[1] * rd[1] + rd[2] * rd[2]);

rd[0] = rd[0] / D;
rd[1] = rd[1] / D;
rd[2] = rd[2] / D;

afz = rd[2] * qe*EL.getQ() / (me*D*D);

niz[j].setBrzinaVz(niz[j].getBrzinaVz() + 0.5 * delT*(aiz + afz));
if (niz[j].getRadiVz() > ((niz[j].getInVis() + 90) * 1000 + delH)) {
if (niz[j].getInVis() == 40) niz[j].setInVis(40);
else niz[j].setInVis(niz[j].getInVis() + 1);
}
if (niz[j].getRadiVz() < ((niz[j].getInVis() + 90) * 1000 - delH)) {
if (niz[j].getInVis() == 0) niz[j].setInVis(0);
else niz[j].setInVis(niz[j].getInVis() - 1);
}

v2 = niz[j].getBrzinaVx()*niz[j].getBrzinaVx() +
 niz[j].getBrzinaVy()*niz[j].getBrzinaVy() +
 niz[j].getBrzinaVz()*niz[j].getBrzinaVz();
EK = Ek(me, v2);

for (int k = 0; k < vrstaCestice; k++){
N = koncetracija[k][niz[j].getInVis()] * pow(10, 6);

PP = PPc(k, EK, UPc);

niz[j].setTr(niz[j].getTr() - delT*PP*N*sqrt(v2));

if (EK>15){
```

```
PP = PPi(k, EK, UPi);
niz[j].setTr(niz[j].getTr() - delT*PP*N*sqrt(v2));
}
}
}
else{
v2 = niz[j].getBrzinaVx()*niz[j].getBrzinaVx() +
 niz[j].getBrzinaVy()*niz[j].getBrzinaVy() +
 niz[j].getBrzinaVz()*niz[j].getBrzinaVz();
EK = Ek(me, v2);

if (EK < 15.0) coll = true;
else {
RAN = (double)ran2(&idum);
if (RAN < Sudar(koncetracija[0][niz[j].getInVis()],
koncetracija[1][niz[j].getInVis()], koncetracija[2][niz[j].getInVis()],
PPc(0, EK, UPc), PPc(1, EK, UPc), PPc(2, EK, UPc), PPi(0, EK, UPi),
PPi(1, EK, UPi), PPi(2, EK, UPi)))coll = true;
else coll = false;
}
if (coll == true) {

inCesMet = GetInCesMet((double)ran2(&idum), koncetracija[0]
[niz[j].getInVis()],koncetracija[1][niz[j].getInVis()],
koncetracija[2][niz[j].getInVis()], PPc(0, EK,
UPc), PPc(1, EK, UPc), PPc(2, EK, UPc));

if (EL.getRadiVz()<108000 && EL.getRadiVz()>108000 + EL.getBrzinaVz()
*0.001){if (EK>6 && EK < 50 && inCesMet!=1){b108 << round(D)
 << endl;
f108 << fixed << niz[j].getRadiVx() << " " << niz[j].getRadiVy()
 << " " << niz[j].getRadiVz() << endl;
}
}
if (EL.getRadiVz()<107000 && EL.getRadiVz()>107000 + EL.getBrzinaVz()
*0.001){if (EK>6 && EK < 50 && inCesMet!=1){b107 << round(D)
 << endl;
f107 << fixed << niz[j].getRadiVx() << " " << niz[j].getRadiVy()
 << " " << niz[j].getRadiVz() << endl;
}
}
if (EL.getRadiVz()<106000 && EL.getRadiVz()>106000 + EL.getBrzinaVz()
*0.001){if (EK>6 && EK < 50 && inCesMet!=1){b106 << round(D)
 << endl;
f106 << fixed << niz[j].getRadiVx() << " " << niz[j].getRadiVy()
 << " " << niz[j].getRadiVz() << endl;
}
```

```
}
if (EL.getRadiVz()<105000 && EL.getRadiVz()>105000 + EL.getBrzinaVz()
*0.001){if (EK>5 && EK < 50 && inCesMet!=1){b105 << round(D)
 << endl;
f105 <<  fixed << niz[j].getRadiVx() << " " << niz[j].getRadiVy()
 << " " << niz[j].getRadiVz() << endl;
}
}

prosBrz = pv(maseCes[inCesMet], tempe[niz[j].getInVis()]);
RAN = (double)ran2(&idum);
ROVMete = vRoMete(prosBrz, (double)ran2(&idum));
VMete[0] = ROVMete*cos(2 * PI*RAN);
VMete[1] = ROVMete*sin(2 * PI*RAN);
RAN = (double)ran2(&idum) – 0.5;
If (RAN < 0.0) VMete[2] = –prosBrz*InErf(2 * –RAN);
else VMete[2] = prosBrz*InErf(2 * RAN);

RAN = (double)ran2(&idum);
RAN1 = (double)ran2(&idum);

Vn[0] = cos(2 * PI*RAN)*sin(PI*RAN1);
Vn[1] = sin(2 * PI*RAN)*sin(PI*RAN1);
Vn[2] = cos(PI*RAN1);

Vc[0] = vCM(maseCes[inCesMet], niz[j].getBrzinaVx(), VMete[0], me);
Vc[1] = vCM(maseCes[inCesMet], niz[j].getBrzinaVy(), VMete[1], me);
Vc[2] = vCM(maseCes[inCesMet], niz[j].getBrzinaVz(), VMete[2], me);

V = konV(maseCes[inCesMet], niz[j].getBrzinaVx(), niz[j].getBrzinaVy(),
 niz[j].getBrzinaVz(), VMete, me);

niz[j].setBrzinaVx(Vc[0] + V*Vn[0]);
niz[j].setBrzinaVy(Vc[1] + V*Vn[1]);
niz[j].setBrzinaVz(Vc[2] + V*Vn[2]);
}
else{
RAN = (double)ran2(&idum);
RAN1 = (double)ran2(&idum);

Vn[0] = cos(2 * PI*RAN)*sin(PI*RAN1);
Vn[1] = sin(2 * PI*RAN)*sin(PI*RAN1);
Vn[2] = cos(PI*RAN1);

for (int l = 0; l < 3; l++) VMete[l] = 0.0;

Vc[0] = vCM(me, niz[j].getBrzinaVx(), VMete[0], me);
```

```
Vc[1] = vCM(me, niz[j].getBrzinaVy(), VMete[1], me);
Vc[2] = vCM(me, niz[j].getBrzinaVz(), VMete[2], me);

V = konV(me, niz[j].getBrzinaVx(), niz[j].getBrzinaVy(),
 niz[j].getBrzinaVz(), VMete, me);

niz[j].setBrzinaVx(Vc[0] + V*Vn[0]);
niz[j].setBrzinaVy(Vc[1] + V*Vn[1]);
niz[j].setBrzinaVz(Vc[2] + V*Vn[2]);

elektron novi(-log((double)ran2(&idum)), Vc[0] - V*Vn[0], Vc[0] - V*Vn[0],
 Vc[0] - V*Vn[0], niz[j].getRadiVx(), niz[j].getRadiVy(),
 niz[j].getRadiVz());
niz.push_back(novi);
}
niz[j].setTr(-log((double)ran2(&idum)));
}
}
}
}

f105.close();
f106.close();
f107.close();
f108.close();
b105.close();
b106.close();
b107.close();
b108.close();
kor105.close();
kor106.close();
kor107.close();
kor108.close();
cin.clear();
system("pause");
return 0;
}

double Sudar(double n0, double n1, double n2, double pp0, double pp1,
 double pp2, double ppi0, double ppi1, double ppi2){
return (n0*pp0 + n1*pp1 + n2*pp2) / (n0*(pp0 + ppi0) + n1*(pp1*ppi1) + n2
*(pp2 + ppi2));
}

double PPc(int Ind,double EK,double PP[][11]){
if (EK < 1.5) return PP[Ind][0]*pow(10,-20);
if (EK < 3) return PP[Ind][1] * pow(10, -20);
```

```
if (EK < 6.5) return PP[Ind][2] * pow(10, -20);
if (EK < 20) return PP[Ind][3] * pow(10, -20);
if (EK < 40) return PP[Ind][4] * pow(10, -20);
if (EK < 75) return PP[Ind][5] * pow(10, -20);
if (EK < 150) return PP[Ind][6] * pow(10, -20);
if (EK < 250) return PP[Ind][7] * pow(10, -20);
if (EK < 400) return PP[Ind][8] * pow(10, -20);
if (EK < 750) return PP[Ind][9] * pow(10, -20);
else return PP[Ind][10];
}

double PPi(int Ind, double EK, double PP[][7]){
if (EK < 40) return PP[Ind][0] * pow(10, -20);
if (EK < 75.0) return PP[Ind][1] * pow(10, -20);
if (EK < 150.0) return PP[Ind][2] * pow(10, -20);
if (EK < 250.0) return PP[Ind][3] * pow(10, -20);
if (EK < 400.0) return PP[Ind][4] * pow(10, -20);
if (EK < 750.0) return PP[Ind][5] * pow(10, -20);
else return PP[Ind][6];
}

int GetInCesMet(double ran,double n0,double n1, double n2,
double pp0,double pp1,double pp2){
if (ran < ((n0*pp0) / (n0*pp0 + n1*pp1 + n2*pp2))) return 0;

if (ran < ((n0*pp0+n1*pp1) / (n0*pp0 + n1*pp1 + n2*pp2))) return 1;
else return 2;
}

double InErf(double ran){
double ERF = 0.0;
for (double i = 0.01; i <= 2; i = i + 0.01){
if (abs(erf(ran) - ERF) > (abs(ERF - erf(i))))ERF = erf(i);
else return ERF;
}
return ERF;
}

double pv(double masa,double temp){
double K = 1.3806505E-23;
return sqrt(2 * K * temp /(masa));
}

double vRoMete(double pv, double ran){
return pv*sqrt(-log(ran));
}
```

```
double vCM(double masa,double vCes,double vMete, double me){
return (me * vCes + masa * vMete) / (me + masa);
}

double konV(double mase,double Vx,double Vy,double Vz,double V[]
,double me)
{
return(mase / (me + mase))*abs(sqrt(Vx * Vx + Vy * Vy + Vz * Vz) - sqrt
(V[0] * V[0] + V[1] * V[1] + V[2] * V[2]));
}

double Ek(double masa,double v2){
return 0.5*masa*v2*6.241509E18;
}
```

Appendix B

Input file with the temperatures and atmospheric number densities of species

2.235E+11	5.125E+13	1.333E+13	207.6
2.747E+11	4.435E+13	1.148E+13	204.2
3.272E+11	3.830E+13	9.869E+12	200.6
3.780E+11	3.298E+13	8.450E+12	197.2
4.243E+11	2.830E+13	7.203E+12	194.0
4.636E+11	2.418E+13	6.109E+12	191.1
4.938E+11	2.057E+13	5.153E+12	188.7
5.140E+11	1.742E+13	4.321E+12	186.8
5.237E+11	1.468E+13	3.602E+12	185.6
5.232E+11	1.231E+13	2.984E+12	185.1
5.137E+11	1.028E+13	2.457E+12	185.5
4.964E+11	8.545E+12	2.012E+12	186.9
4.734E+11	7.082E+12	1.640E+12	189.2
4.465E+11	5.859E+12	1.331E+12	192.5
4.174E+11	4.843E+12	1.079E+12	196.7
3.874E+11	4.003E+12	8.726E+11	201.8
3.575E+11	3.312E+12	7.056E+11	207.8
3.283E+11	2.745E+12	5.707E+11	214.7
3.005E+11	2.280E+12	4.621E+11	222.5
2.743E+11	1.900E+12	3.748E+11	231.3
2.500E+11	1.588E+12	3.049E+11	241.1
2.275E+11	1.332E+12	2.487E+11	251.7
2.070E+11	1.123E+12	2.037E+11	263.3
1.882E+11	9.498E+11	1.675E+11	275.9
1.711E+11	8.071E+11	1.382E+11	289.5
1.556E+11	6.888E+11	1.147E+11	304.1
1.417E+11	5.906E+11	9.558E+10	319.7
1.291E+11	5.089E+11	8.009E+10	336.3
1.179E+11	4.407E+11	6.750E+10	353.6
1.078E+11	3.836E+11	5.725E+10	371.6
9.891E+10	3.359E+11	4.888E+10	389.9
9.106E+10	2.960E+11	4.204E+10	408.2

8.421E+10	2.626E+11	3.644E+10	426.0
7.826E+10	2.347E+11	3.185E+10	442.7
7.302E+10	2.110E+11	2.804E+10	458.6
6.833E+10	1.905E+11	2.482E+10	474.0
6.409E+10	1.728E+11	2.208E+10	489.1
6.026E+10	1.573E+11	1.974E+10	503.8
5.678E+10	1.436E+11	1.773E+10	518.1
5.361E+10	1.316E+11	1.599E+10	532.0
5.071E+10	1.209E+11	1.447E+10	545.6

In every row are given values for one altitude. Starting with first at 90 *km* and ending with last at 130 *km*. Values are given for every 1000 *m*. First three columns are values of atmospheric number densities of species *O*, N_2 and O_2 in cm^3. Fourth column is value of temperature in *K*.

Appendix C

Input file with cross sections for elastic collision

```
4.2   6   7.1   7.1   5   3.9   2.9   1.8   1.5   0.9   0.55
8.8   25.5   11.44   10.56   11.88   10.12   0.0   0.0   0.0   0.0   0.0
6.16   6.6   7.04   9.68   10.56   0.0   0.0   0.0   0.0   0.0   0.0
```

In every row are given values for one gas spaces in x$10^{-20}m^2$. Rows are representing different kinetic energies of electrons corresponding to:

```
1   2   4   10   20   50   100   200   300   500   1000  eV
```

Appendix D

Input file with cross sections for ionisation

```
0.6    1.1    1.5    1.1    0.9    0.65    0.5
0.299    1.9    2.464    2.2    1.936    1.4    0.88
0.3696    2.1    2.8    2.464    2.2    1.672    1.056
```

In every row are given values for one gas spaces in x$10^{-20}m^2$. Rows are representing different kinetic energies of electrons corresponding to:

```
20    50    100    200    300    500    1000  eV
```

Bibliography

[1] Hans C. Stenbaek-Nielsen, Peter Jenniskens, *Meteor plasma trails: effects of external electric field*, Annales Geophysicae, 2008.

[2] Y. S. Dimant, M. M. Oppenheim, and M. Milikh, *A 'shocking' Leonid meteor at 1000 fps*, Advances in Space Research, 2003.

[3] Y. Itikawa and A. Ichimura, *Cross Sections for Collisions of Electrons and Photons with Atomic Oxygen*, Journal of Physical and Chemical, 1990.

[4] Y. Itikawa, *Cross Sections for Electron Collisions with Nitrogen Molecules*, Journal of Physical and Chemical, 2006.

[5] Michael C. Kelley, *The Earth's Ionosphere: Plasma Physics and Electrodynamics*, Elsevier Inc., 2009.

[6] Kaye and Laby Online, *Electron collisions*, July, 2014. http://www.kayelaby.npl.co.uk/atomic_and_nuclear_physics/4_4/4_4_6.html

[7] The Community Coordinated Modeling Center, *MSIS-E-90 Atmosphere Model*, July, 2014. http://ccmc.gsfc.nasa.gov/modelweb/models/msis_vitmo.php