Documentation on simulating RREA and RFD

Brian Hare

April 13, 2016

1 table of constants

Name	symbol	value
speed of light	C	
charge of electron	-e	
molecular density of air	N_m	$2.688 \times 10^{25} m^{-3}$
average nuclear charge of air	Z_m	14.5
classical electron radius	r_e	$2.8179 \times 10^{-15} m$
mass of electron	m_e	
ionization potential of air	I	85.7 ev

2 dimensionless variables

Dimensionless variables are used in this simulation. This can complicate taking a physical equation, and relating it to a formula that can be used in the simulation, but once in the simulation the formulas tend to be simpler, thus easier to use. In the rest of this documentation, the normal symbols, e.g. ε for kinetic energy and $\vec{P}$ for momentum, will represent the values of those quantities in MKS units, alternate symbols will be used to represent the values of each quantity in dimensionless units, e.g. E for kinetic energy and $\vec{\rho}$ for momentum. The table below gives the units of all the dimensionless variables used in the simulation. Brackets around a symbol, e.g. $[E]$ for energy and $[\vec{\rho}]$ for momentum, represents the units of the dimensionless values, e.g. $\varepsilon = E\,[E]$.

Name	units	alternate symbol
time	$(2\pi N_m Z_m r_e^2 C)^{-1}$	τ
velocity	C	$\vec{\beta}$
position	$C \times [\tau]$	$\vec{\chi}$
momentum	$m_e C$	$\vec{\rho}$
energy	$m_e C^2$	E (kinetic only)
force	$\frac{m_e C}{[\tau]}$	$\frac{d\vec{\rho}}{d\tau}$
electric field	$\frac{m_e C}{e[\tau]}$	$\vec{\xi}$
magnetic field	$\frac{m_e}{e[\tau]}$	$\vec{\Upsilon}$

3 relativistic equations

This simulation deals with very relativistic particles, and so must use relativistically correct formula. Since, in the simulation, each particle stores position and momentum, of particular interest is simple formula relating momentum to other quantities.

First, is the definition of gamma:

$$\gamma = \frac{1}{\sqrt{1-\beta^2}} \tag{1}$$

Allowing us to define total energy:

$$\varepsilon_t = \gamma m_e C^2 \tag{2}$$

or in dimensionless units:

$$E_t = \gamma \tag{3}$$

The definition of kinetic energy:

$$\varepsilon_t = m_e C^2 + \varepsilon \tag{4}$$

and in dimensionless units:

$$E_t = 1 + E \tag{5}$$

Substitution gives us:

$$E = \gamma - 1 \tag{6}$$

Next is the definition of momentum:

$$\vec{P} = \gamma m_e \vec{V} \tag{7}$$

In dimensionless units:

$$\vec{\rho} = \gamma \vec{\beta} \tag{8}$$

We can relate momentum to energy:

$$\varepsilon_t^2 = (M_e C^2)^2 + (PC)^2 \tag{9}$$

dimensionless:

$$E_t^2 = 1 + \rho^2 \tag{10}$$

Or:

$$\gamma^2 = 1 + \rho^2 \tag{11}$$

thus:

$$E = \sqrt{1+\rho^2} - 1 \tag{12}$$

Substitution gives:

$$\beta^2 = \frac{\rho^2}{1+\rho^2} \tag{13}$$

4 forces and equations of motion

The simplest task that needs to be done in the simulation, is to simulate the motion of a particle given a electric field, magnetic field, and friction.

In dimensionless units, the equations of motion for an electron are:

$$\frac{d\vec{\rho}}{d\tau} = -\vec{\xi}(\vec{\chi}) - \frac{\vec{\rho}\times\vec{\Upsilon}(\vec{\chi})}{\sqrt{1+\rho^2}} + \hat{\rho}\frac{d\rho}{d\tau}_{friction}(\rho) \tag{14}$$

and

$$\frac{d\vec{\chi}}{d\tau} = \frac{\vec{\rho}}{\sqrt{1+\rho^2}} \tag{15}$$

This are simulated using a fourth-order runge-kutta technique. By first combining the two equations of motion into one:

$$\frac{d}{d\tau}\begin{bmatrix}\vec{\chi}\\ \vec{\rho}\end{bmatrix} = \begin{bmatrix} g(\tau,S)\\ f(\tau,S)\end{bmatrix} \tag{16}$$

We define the vector S so that:

$$\frac{d}{d\tau}S = S' \tag{17}$$

Then we can find S at the next time step, given a stepping time of $\Delta\tau$:

$$S_{n+1} = S_n + \frac{\Delta\tau}{6}(K_1 + 2K_2 + 2K_3 + K_4) \tag{18}$$

where:

$$K_1 = S'(\tau_n, S_n) \tag{19}$$

$$K_2 = S'(\tau_n + \frac{\Delta\tau}{2}, S_n + \frac{\Delta\tau}{2}K_1) \tag{20}$$

$$K_3 = S'(\tau_n + \frac{\Delta\tau}{2}, S_n + \frac{\Delta\tau}{2}K_2) \tag{21}$$

$$K_4 = S'(\tau_n + \Delta\tau, S_n + \Delta\tau K_3) \tag{22}$$

Given that high-precision is not really needed in solving the equations of motion, 4th order runge-kutta may be overkill. Work will need to be done to see if the time required to use 4th order runga-kutta is significant compared to the time required to calculate scattering processes. If so, maybe I should consider using a lower-order scheme.

The frictional force due to ionization could be approximated using the Bethe formula, this however, produces some problems as the Bethe formula is at best an approximation, which is problematic because we care about finding the minimum electric field necessary for RREA and for RFD, which is sensitive to the energy loss due to

ionization, and more worrisome is that the Bethe formula is very imprecise and unstable at low energies, and there are a large number of low energy electrons in RREA. The solution is to use tabulated values instead. Particularly, we use tabulated values from ICRU report 37, which gives the energy loss due to ionization for electrons and positrons down to a very low energy. These tables are hard-coded into a program that is separate from the main simulation. When run, this program converts the values into dimensionless units, and it converts the energy values into dimensionless momentum squared (because momentum squared is easier to calculate at runtime than energy), and stores the two tables into a binary file. At startup, the simulation loads the energy losses due to ionization tables from the binary file, then during runtime these tables can be searched and energy loss rates returned using linear interpolation. At the time of writing this, the time required to search the lookup table is, at absolute worse, twice as slow as calculating the values using the Bethe formula. It is doubtful that the algorithm can be improved, but I believe that the increase in stability at low energy warrants the slowdown, and that the slowdown will be minor compared to the time to calculate other scattering processes (this has yet to be shown).

The tables in ICRU 37 extend to very high energy, and so probably will not need to generate new values using the Beth formula. However, as we include Moller scattering in the simulation, the ionization table generation program will need to take Moller scattering into account by subtracting off the energy loss due to Moller scattering at the appropriate energies. This has not been implemented.

5 shielded coulomb scattering

As the electrons and positrons travel through the simulation, they will collide off of atomic nuclei. We simulate this by including the effects of elastic scattering off of atomic nuclei via the shielded coulomb cross section. The differential cross section for the shielded coulomb cross section is:

$$\frac{d\sigma_{Coul}}{d\Omega} = \frac{1}{4}\left(\frac{Z_m r_e}{\beta^2\gamma}\right)^2 \frac{1-\beta^2\sin^2(\theta/2)}{\left(\sin^2(\theta/2)+\frac{\hbar}{4P^2a^2}\right)^2} \tag{23}$$

where

$$a = 183.8\lambda z_m^{-1/3} \tag{24}$$

Using the definition of cross-sections, we can relate the differential cross section to expected number of times a particle will be deflected into an angle:

$$\frac{dn}{d\Omega} = N_m Z_m V \Delta T \frac{d\sigma}{d\Omega} \tag{25}$$

where $\frac{dn}{d\Omega}$ is the expected number of times that a particle will be deflected by angle Ω during time ΔT. If $\frac{dn}{d\Omega} << 1$, it can be interpreted as a probability.

In dimensionless units, this turns into:

$$\frac{dn}{d\Omega} = \frac{\beta\Delta\tau}{2\pi r_e^2}\frac{d\sigma}{d\Omega} \tag{26}$$

Plugging in the differential cross section for elastic shielded coulomb scattering gives:

$$\frac{dn}{d\Omega} = \frac{\Delta\tau}{8\pi\beta}\left(\frac{Z_m}{\rho}\right)^2 \frac{1-\beta^2\sin^2(\theta/2)}{\left(\sin^2(\theta/2)+\frac{Z_m^{2/3}}{4\times183.8^2}\frac{1}{\rho^2}\right)^2} \tag{27}$$

Two decisions need to be made using this formula. 1) How many times does a particle scatter in one time step, and 2) what angles does it scatter into? The first question can be answered by integrating equation 27 over the unit sphere for a given particle energy and time step, then by sampling the Poisson distribution. For shielded coulomb scattering, this turns out to be many many times for any reasonable time step. The second question is answered using inverse transform sampling.

Inverse transform sampling is performed by first integrating the distribution over the relevant variable that needs to be sampled:

$$N(\theta) = \int_0^{\theta} sin(\theta)d\theta' \int_0^{2\pi} d\phi' \frac{dn}{d\Omega} \tag{28}$$

Normalizing and inverting the function such that:

$$N'\left(\frac{N(\theta)}{N(\pi)}\right) = \theta \tag{29}$$

Then we can then sample a number between 0 to 1, from the uniform distribution, U, and plug it into N' to get a sample for our variable:

$$N'(U) = \theta \tag{30}$$

Except for a few specific cases, inverse transform sampling cannot be done analytically. I will describe a method I have developed to do inverse transform sampling reliably. After sampling the inclination, the azimuth, ϕ can be sampled from the uniform distribution.

After the inclination and azimuth have been sampled, we can rotate the momentum vector by finding an orthogonal, but not normal, vector basis in which the momentum is one axis. First, we use cross products to find two vectors, $\vec{Bv}$ and $\vec{Cv}$ orthogonal to each other, and orthogonal to the momentum vector, $\vec{\rho}$:

$$\vec{Bv} = \frac{\vec{\varsigma}\times\vec{\rho}}{\left|\vec{\varsigma}\times\vec{\rho}\right|} \tag{31}$$

Where $\vec{\varsigma}$ is any unit vector chosen so that equation 31 is not singular.

$$\vec{Cv} = \vec{Bv}\times\vec{\rho} \tag{32}$$

$\vec{Cv}$ is guaranteed to have the same magnitude as the momentum, since $\vec{Bv}$ is guaranteed to be a unit vector orthogonal to the momentum.

Using these new basis vectors, we can calculate the new momentum:

$$\vec{\rho}_{new} = cos(\theta)\vec{\rho} + sin(\theta)cos(\phi)\vec{Bv}\left|\vec{\rho}\right| - sin(\theta)sin(\phi)\vec{Cv} \tag{33}$$

This process needs to be ran however many times scattering happened to occur for that time step. This process is very slow. So what is done, is that this process is ran many many times before the simulation, and a distribution for θ is built up for different energies and time steps. This distribution is then saved in a table and can be opened and sampled relatively quickly at simulation time.

6 numerical inverse transform sampling

My method for inverse transform sampling relies upon quadratic interpolation. We need a quadratic interpolant:

$$\bar{Y}(x) = W_1 + W_2 x + W_3 x^2 \tag{34}$$

Such that $\bar{Y}$ is precise at three points:

$$\bar{Y}(X_L) = Y_L$$

$$\bar{Y}(X_M) = Y_M$$

$$\bar{Y}(X_R) = Y_R$$

Solving the 3x3 matrix reveals:

$$W_3 = \frac{(X_M - X_L)(Y_R - Y_L) - (X_R - X_L)(Y_M - Y_L)}{(X_M - X_L)(X_R^2 - X_L^2) - (X_R - X_L)(X_M^2 - X_L^2)} \tag{35}$$

$$W_2 = \frac{Y_M - Y_L}{X_M - X_L} - W_3 \frac{X_M^2 - X_L^2}{X_M - X_L} \tag{36}$$

$$W_1 = Y_L - W_2 X_L - W_3 X_L^2 \tag{37}$$

Integrating this interpolant gives us an approximation for the cumulative integral of our function;

$$\int_{X_L}^{x} Y(x')dx' \approx \int_{X_L}^{x} \bar{Y}(x')dx' = W_1 x + \frac{W_2}{2}x^2 + \frac{W_3}{3}x^3 - (W_1 X_L + \frac{W_2}{2}X_L^2 + \frac{W_3}{3}X_L^3) \tag{38}$$

Using this integral over multiple regions, we can have a method adaptive cumulative quadrature. This is done by first splitting the region we want to integrate over into multiple sections (say, 5). We then sample the function we want to integrate at the endpoints and at the middle of each section and estimate the area of each section using equation 38. We then split each section into a left and right subsections, and sample the function at the center of each subsection to get a slightly better estimate of the area. Finally we test our precision using equation 39 below. Where $float(x)$ is the floating point representation of x. A_s, A_l, and A_r are the areas of some section and its left and right subsections, and F is a factor that controls our precision.

$$float(F * A_s + (A_s - A_l - A_r)) == float(F * A_s) \tag{39}$$

For any subsection, if this condition is true, then we are done and move onto the next section. If this condition is not true then the two subsections are each split into two subsections and this procedure is repeated until condition 39 is true. In this way we can very precisely sample the integral of any function at a number of un-equally spaced points. I am not sure how this method will behave if the value of the integral is at or close to zero. I am using low-order polynomial quadrature instead of a more advance method, say a high-order Gaussian quadrature, because this method is relatively simple and easily produced a cumulative integral.

Finally, this cumulative integral is normalized and is inverted by plugging the value of the integral at each point into our interpolant (equation 34) for the x values and the points that the integral was sampled at for the y-values. We can then sample our distribution by sampling a variable from the uniform distribution and plugging that number into this inverse interpolant. This method can only sample one-dimensional distributions, but multi-dimensional distributions can always be turned into multiple one-dimensional distributions via integration.