Documentation for the radioactive package

João Caldeira

October 3, 2016

This package is intended for Monte Carlo simulations of physical processes involving radioactive sources and the interaction of the emitted radiation with scatterers and detectors. The functions in the package intended to be run by external scripts are `source`, `source_compton`, `scatterer` and `detector`.

1 Description of main functions

1.1 `source` and `source_compton`

The function `source` takes as inputs:

- `act`, the activity of the source in μCi. Default is `act=1` μCi.
- `Tf`, the time the source emits radiation for. Default is `Tf=1` s.
- `axis` and `cone`, which allow for emitted rays to be accepted only if they fall inside a cone around an axis. `cone` should be an angle between 0 and π, describing the maximum angle between an emitted ray and the indicated axis. If these are not given, rays will come out in every direction. If only `cone` is given, the angle will be relative to the z axis.
- `posn`, the position of the source. Default is $[0, 0, 0]$.
- `isotope`, the radioactive isotope indicated as element followed by mass number, for example `'Na22'`. If the isotope indicated has not been preprogrammed or no isotope is given, default is Na-22.

The output is a list of events, organised as an array. Each element of this array corresponds to an emitted γ ray, and contains the information of time elapsed between the beginning of the information and emission of this ray, energy, position and direction of propagation.

The function `source_compton` has the same inputs and outputs, but increased efficiency when there is only one decay mode and we are interested in γ rays emitted in a small cone, such as for simulation of a Compton scattering experiment. This is achieved by rescaling the activity by

$$\texttt{act} \to \frac{1-\cos(\texttt{cone})}{2}\texttt{act}, \tag{1}$$

and generating propagating directions only inside the intended cone instead of checking after generating the direction. As currently implemented, `source_compton` does not work for sources with more than one decay mode.

1.2 `scatterer`

This function takes the following inputs:

- `solid`, indicating the shape of the scatterer. Default (and at the moment only supported option) is `'cylinder'`.
- `center` and `axis`, two vectors indicating the position of the cylinder. Defaults are [0,0,5] and [0,0,1].
- `radius` and `length`, two scalars indicating the size of the cylinder. Defaults are 1 cm for both.
- `eventsin`, the list of incoming events, an array with the same description as the output of `source`.
- `material`, a string indicating the chemical symbol of the material the scatterer is made of. Default is `'Al'`. A text file with the corresponding interaction cross sections should be present in the same folder as the scripts, as detailed in section 2.1.
- `rho`, the density of the scattering material. Necessary if the material is not Al or Pb.

For each γ ray in `eventsin`, we start by adding it to an empty array named `liverays`. We then find the intersection of the path of that γ with the volume of the scatterer. If there is no intersection, the γ ray is simply added to `eventsout`. If there is an intersection, the program finds the length of the intersection d and obtains the cross section σ for the required energy from the input data.

A uniform random number $u \in (0,1)$ is then drawn. If $u < e^{-\sigma\rho d}$, there is no interaction, the γ is propagated to the final intersection point and added to `eventsout`. Otherwise, an interaction happens after travelling a distance of $-\frac{\log u}{\sigma\rho}$ inside the scatterer. The γ is propagated to that point.

A new random number is then drawn to determine if the interaction is Compton, photoelectric or pair production. If it is photoelectric, the γ is absorbed and we move on to the next ray. If Compton, see section 2.2 for an explanation of the algorithm followed. The outgoing ray is added back to `liverays`. If pair production, we draw a random direction and add two rays of energy 511 keV moving in that direction and the opposite to `liverays`.

Once `liverays` is empty, we move on to the next element of `eventsin`. Once all γ rays have been scattered, the array `eventsout` is the result.

It should be noted that before any other calculations, we find the transformation from the lab frame to a frame where the axis of the scatterer points along the z axis and the center of the detector is at the origin, and transform all positions and velocities from the lab to the scatterer frame,

$$p_S = R(p_L - x_C), \qquad\qquad v_S = Rv_L. \tag{2}$$

Before returning `eventsout`, we perform the inverse transformation on all the vectors back to the lab frame,

$$p_L = R^{-1}p_S + x_C, \qquad\qquad v_L = R^{-1}v_S. \tag{3}$$

1.3 `detector`

This function has a similar structure to `scatterer`. The inputs it takes are exactly the same, with the difference that the default `material` is NaI.

The algorithm is also similar, with the difference that for each ray in `eventsin` we create a variable corresponding to how much energy is dumped into the detector. In a photoelectric interaction we suppose all energy from the γ ray is absorbed, in Compton we take all the energy difference to be absorbed, and for pair production we assume $E - (1.022 \text{ MeV})$ is absorbed.

For each γ ray which dumps any amount of energy, we now calculate the energy detected depending on properties of the detector. For a Ge solid state detector, energy deposited in the crystal goes into the production of electron-hole pairs. Each electron hole pair takes an energy ϵ to create, so if an amount of energy E is deposited in the crystal, on average $N = E/\epsilon$ pairs are created. In scintillation detectors such as NaI, energy goes into producing scintillation light, which is composed of photons of energy ϵ. Once again, $N = E/\epsilon$ photons are created. Note these ϵ are different quantities, but that will not be important in our derivation.

Since these processes are approximately Poissonian, the variance in N is proportional to N, $\sigma_N^2 = kN$ for some constant k which depends on the specific detector (note the process in solid state detectors is strictly speaking *not* Poissonian, for which $k = 1$, but the variance has the same N-dependence with a different k). The standard deviation of the distribution of detected energy, given the amount of deposited energy E, then depends on E as

$$\frac{\sigma_E}{E} = \frac{\sigma_N}{N} = \frac{\sqrt{kE/\epsilon}}{E/\epsilon} = \sigma_0\sqrt{\frac{E_0}{E}}. \tag{4}$$

We can then measure σ_0 at some energy E_0 experimentally, and from there get the general distribution. The program uses values obtained for $E_0 = 662$ keV of $\sigma_0 = 3.18\%$ for NaI and $\sigma_0 = 0.0592\%$ for Ge.

The detected energy is then obtained from the deposited energy by multiplying E by a draw of a Gaussian with mean 1 and σ obtained as described above. In the end a histogram is made out of the array of detected energies. The histograms have 512 bars, in order to emulate the behaviour of the PHA.

2 Other input assumptions and equations used

2.1 Cross Sections

The program draws the cross sections for each interaction process of a γ ray with matter from a table provided. A good source for these tables is NIST. The program assumes the order of the columns is:

- Energy in MeV
- Cross section for Rayleigh scattering in cm^2/g
- Cross section for Compton scattering in cm^2/g
- Cross section for photoelectric effect in cm^2/g
- Cross section for pair production in nuclear field in cm^2/g
- Cross section for pair production in electron field in cm^2/g
- Total interaction cross section including Rayleigh scattering in cm^2/g
- Total interaction cross section not including Rayleigh scattering in cm^2/g

At the moment, the two different pair production cross sections are simply summed, and Rayleigh scattering is ignored. The program interpolates linearly between the energies on the table, so the table should be filled closely enough for this to be accurate.

The input file should be called `x.txt`, where `‘x’` is given to the `scatterer` and `detector` functions with the `material` option. For example, if `scatterer` is called with `material=‘Al’`, the program will look for the cross section data in the file `Al.txt`. Unless the density of the material is preprogrammed, the function should also be called with the `rho` option specifying the density of the material in g/cm^3. If the density is not preprogrammed and not specified, a warning will be printed. Note the program skips the first ten lines of the input file, so the data should start on the eleventh line of the file.

If a γ ray is ever analysed with energy below the energy of the first line in the data file, the program prints a warning:

```
Warning:  Energy is lower than the lowest value on the table provided.
```

If this happens often, more data should be added for lower energies. If the data starts at low enough energies, this should never happen, as the photoelectric cross section dominates before we get to a very low energy γ ray.

2.2 Compton and Klein-Nishina

In this section we will describe how the script implements the Klein-Nishina formula. First recall the relevant formulas for Compton scattering: when photons of energy E are scattered by an angle

θ, the ratio between the final and initial energies is given by

$$r(E,\theta) = \frac{1}{1 + \frac{E}{m_e c^2}(1-\cos\theta)}, \tag{5}$$

where $m_e c^2 = 511$ keV is the electron rest mass. The differential cross section for scattering is given by

$$\begin{aligned} \frac{dP}{d\Omega} &= \frac{\alpha^2\hbar^2}{2m_e^2c^2} r(E,\theta)^2 \left(r(E,\theta) + r(E,\theta)^{-1} - 1 + \cos^2\theta \right) \\ &= A r(E,\theta)^2 \left(r(E,\theta) + r(E,\theta)^{-1} - 1 + \cos^2\theta \right), \end{aligned} \tag{6}$$

where in the second line we just grouped all the multiplicative constants into a normalisation A that we will not need (therefore we will not be too careful about keeping the meaning of A invariant from equation to equation – it is simply the required normalisation constant). For low energies $E \ll m_e c^2$, $r(E,\theta) \simeq 1$ and the differential cross section simplifies into

$$\frac{dP}{d\Omega} = A(1+\cos^2\theta). \tag{7}$$

In this package, we take the total Compton cross section from existing data, and we only need to use the equations above to draw an angle θ according to the correct probability distribution and from θ calculate the correct final energy E_f. We will do this by obtaining a θ_L obeying the distribution from the low-energy approximation (7) by *inverse transform sampling* and then using that θ_L and *rejection sampling* to end with a θ drawn from the full Klein-Nishina distribution (6).

The principle of inverse transform sampling is that given a probability distribution $P(x)$, we start by calculating the cumulative distribution function $f(x) = P(X < x)$. We then invert $f(x)$ to obtain $f^{-1}(u)$. We can now use this to transform a uniform distribution in the interval $(0,1)$ to a variable with the distribution P by setting $x = f^{-1}(u)$. In brief, this works because the probability of this algorithm giving an x smaller than any given value x_0 is given by $f(x_0)$, which is the correct distribution. Note that we cannot use this method directly on the Klein-Nishina formula because we cannot invert the cdf in closed form.

To use this method, the first step is therefore to find the cdf corresponding to (7) as a probability distribution for the scattering angle θ. Note that $d\Omega = \sin\theta d\theta d\phi$, so the probability distribution as a function of θ is given by

$$P(\theta) = A(1+\cos^2\theta)\sin\theta. \tag{8}$$

Integrating this distribution to find the cdf, we find

$$f(\theta) = \frac{\int_0^\theta P(\theta')d\theta'}{\int_0^\pi P(\theta')d\theta'} = \frac{1}{8}\left(4 - 3\cos\theta - \cos^3\theta\right). \tag{9}$$

We will take this as the cdf for $\cos\theta$, $f(x) = \frac{1}{8}(4 - 3x - x^3)$. Inverting this function (and keeping in mind that x should be real) we get

$$f^{-1}(u) = y(u) - \frac{1}{y(u)}, \qquad y(u) = \left(2 - 4u + \sqrt{1 + (2-4u)^2}\right)^{1/3}. \tag{10}$$

We can then implement the inverse transform sampling for the distribution (7) by drawing a uniform random number $u \in (0,1)$ and setting $\theta_L = f^{-1}(u)$.

Given this θ_L, we now use rejection sampling to arrive at a θ drawn from (6). This method transforms between a random variable drawn from a distribution $P_1(x)$ to a random variable drawn from a distribution $P_2(x)$ by adding an additional step where we accept the drawing with probability $P_2(x)/(kP_1(x))$, where k is a constant such that $P_2(x) < kP_1(x)$ for all x. If the drawing is rejected, we draw a new sample until we the drawing is accepted. Clearly this method is more efficient if the distributions P_1 and kP_2 are close to each other.

In this instance, the probability of acceptance for an angle θ_L drawn from (7) is

$$P_{\text{accept}} = \frac{r(E,\theta_L)^2 \left(r(E,\theta_L) + r(E,\theta_L)^{-1} - 1 + \cos^2\theta_L\right)}{1 + \cos^2\theta_L}. \tag{11}$$

Note that for any E and θ this is smaller than 1, as required.