LE-6 Report: Monte Carlo Techniques

GSSI Astroparticle Physics PhD program
a.y. 2020/2021

Alessio Mei

Introduction

This report presents descriptions and results of the exercises proposed for the PhD course "LE-6 Monte Carlo Techniques" at Gran Sasso Science Institute. The codes used to solve the following exercises are written in Python and are available in the GitHub repository at the link:

`https://github.com/alessiomei/LE-6-Exercises`

Contents

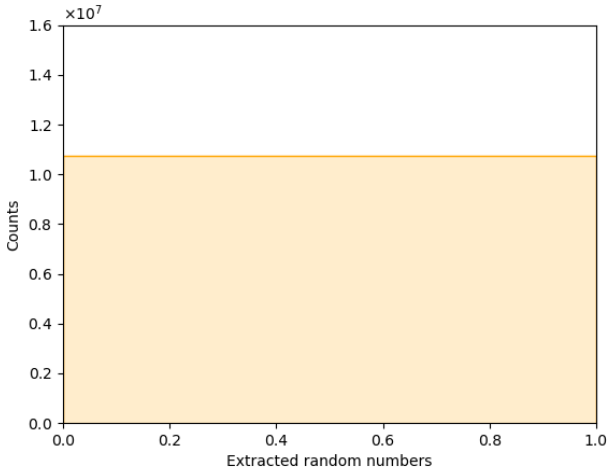


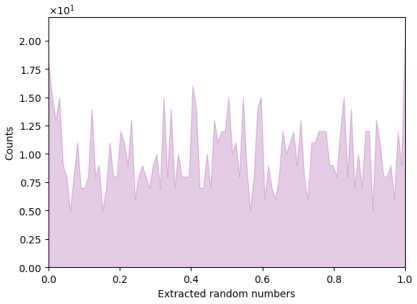
Figure 1: Distribution obtained through MCG with 2^{30} random numbers. In this case $\chi^2 = 1.7 \cdot 10^{-6}$ and $p = 1.00$, confirming that the uniformity of this distribution is statistically significant.

1 Uniform random sampling

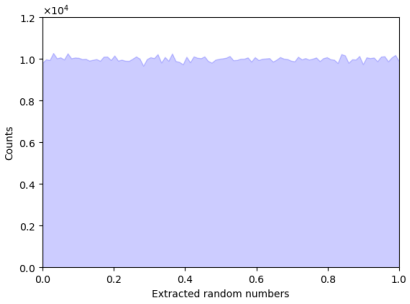
To perform a uniform random sampling, I code a simple pseudo-random number generator like a Multiplicative Congruential Generator (MCG):

$$R_n = a \cdot R_{n-1} \tag{1}$$

Where $a = 663608941$ is the multiplicative factor and $R_0 = 987654321$ is the seed. The modulo operation is not specified since it is automatically done by using only unsigned integers (in Python `uint32`), which are variables that occupy a fixed amount of memory (32 bit).
In order to obtain the maximum sequence length related to this MCG, I extract random numbers until I find again the initial seed R_0. To perform this operation in a relatively short amount of time, I have to use `Numba`, a high performance JIT compiler that translates a subset of Python and Numpy code into fast machine code. The CPU time employed for this operation is $\Delta t \simeq 244$ s, while the sequence length is $2^{30} = 1073741824$.
To obtain an uniform distribution, I normalize the random number sample between zero and one by dividing the array by its maximum. To test if this code is able to return an uniform distribution, I compute the χ^2 and relative p value of the total distribution and of the distributions obtained by considering only the first 10^3 and 10^6 elements. The results are shown in Fig. 1 and Fig. 2.



(a) $N = 10^3, \chi^2 = 95.80, p = 0.57$



(b) $N = 10^6, \chi^2 = 130.65, p = 0.02$

Figure 2: Distribution of the first 10^3 (*Left panel*) and 10^6 (*Right panel*) random numbers generated through MCG. Since the p values are far from 1, in these cases the uniformity of these distributions is not statistically significant.

1.1 MINSTD algorithm

A better choice of MCG is the so called MINSTD generator:

$$R_n = a \cdot R_{n-1} \bmod(m) \tag{2}$$

Where $a = 7^5 = 16807$ is the multiplier and $m = 2^{31} - 1$ is the modulus of the generator. I perform the same analysis of MCG, and I compute the maximum step length, which in this case is $m - 1 = 2^{31} - 2 = 2147483646$.
It is possible to test that the sequence length does not depend on the choice of the seed, which has to be defined as $0 < R_0 < m$. I compute χ^2 and p value in order to test the uniformity of this distribution. The results are shown in Fig. 3.

2 Random sampling from other distributions

2.1 Inversion method

I want to sample the small-angle Rutherford probability density function (PDF) $p(x)$ using Monte Carlo techniques:

$$p(x) = \frac{2x}{(1+x^2)^2} \quad 0 \leq x < \infty \tag{3}$$

To do so, I compute the relative cumulative probability distribution:

$$c(x) = \int_0^x p(y)\ dy = \int_0^x \frac{2y}{(1+y^2)^2} dy = \frac{x^2}{1+x^2} \tag{4}$$

From Eq. 4 it is evident that the cumulative PDF can be inverted, thus the inversion method can be applied. If I write the cumulative PDF as $c(x) = t$,

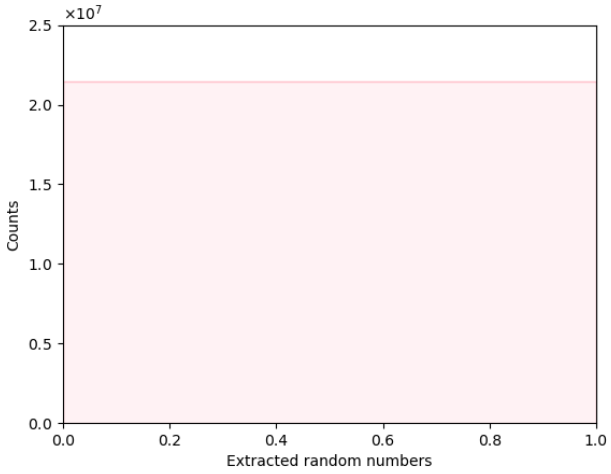


Figure 3: Distribution obtained through MINSTD generator with $2^{31}-2$ random numbers. In this case $\chi^2 = 1.16 \cdot 10^{-6}$ and $p = 1.00$, confirming that the uniformity of this distribution is statistically significant.

the inverted cumulative PDF is defined as:

$$c^{-1}(t) = x = \sqrt{\frac{t}{1-t}} \tag{5}$$

To sample the Rutherford formula I generate 10^6 random numbers t between $[0, 1]$ and calculate the inverse cumulative distribution $c^{-1}(t)$ as a function of them. The result is shown in Fig. 4.

2.2 Inversion and rejection

As discussed in the previous Section, it is straightforward to do a random sampling of a distribution $p(x)$ using the inversion method. However, it can not be used if the cumulative PDF $c(x)$ is not invertible. In these cases, it is possible to perform a sampling using the rejection method, which consist in sampling a second invertible probability distribution $\pi(x)$ which has to satisfy the condition $C\pi(x) \geq p(x)$ for any x, where C is a positive constant.
To check the differences between these two methods, I use both of them to sample a unit circle with its center in the origin of the Cartesian plane. For both cases, I extract couples (ξ_1, ξ_2) of random numbers (both between $[0, 1]$) and compute the coordinates $(x(\xi_1, \xi_2), y(\xi_1, \xi_2))$ until 10^6 couples are sampled inside the circle.

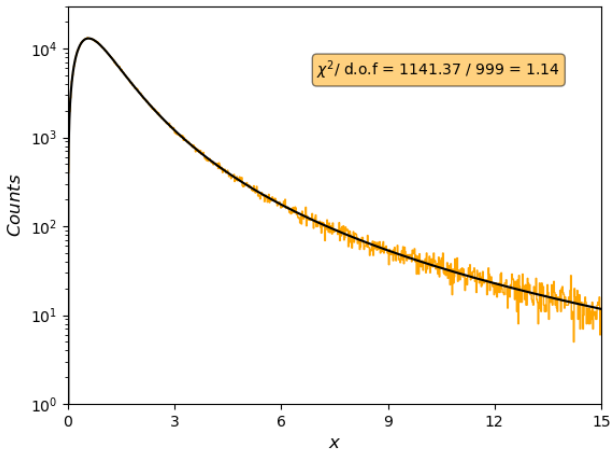


Figure 4: Sampled small-angle Rutherford distribution (in orange) compared with its best-fit function $p(x)$ (in black).

For the inversion/analytic method, the coordinates are defined as:

$$\begin{cases} x = \sqrt{\xi_1}\cos(2\pi\xi_2) \\ y = \sqrt{\xi_1}\sin(2\pi\xi_2) \end{cases} \tag{6}$$

The coordinates are directly computed from the random numbers, since they automatically satisfy the "inside the circle" condition.
For the rejection method, the coordinates are defined as:

$$\begin{cases} x = -1 + 2\xi_1 \\ y = -1 + 2\xi_2 \end{cases} \tag{7}$$

Thus, I generate random points filling a square of side 2 centered in the origin and require that they fall inside the unit circle, i.e. $x^2 + y^2 \leq 1$.
The result is that the inversion method is less efficient ($\Delta t \simeq 2.8$ s) than the rejection one ($\Delta t \simeq 1.3$ s).
This can be explained considering that, even if the inversion method samples directly the points inside the unit circle, the machine takes CPU time to compute transcendental functions like sines and cosines. In contrast, the rejection method requires only an "if" statement to sample the circle, while the computational procedure is much faster, hence this process results more efficient.

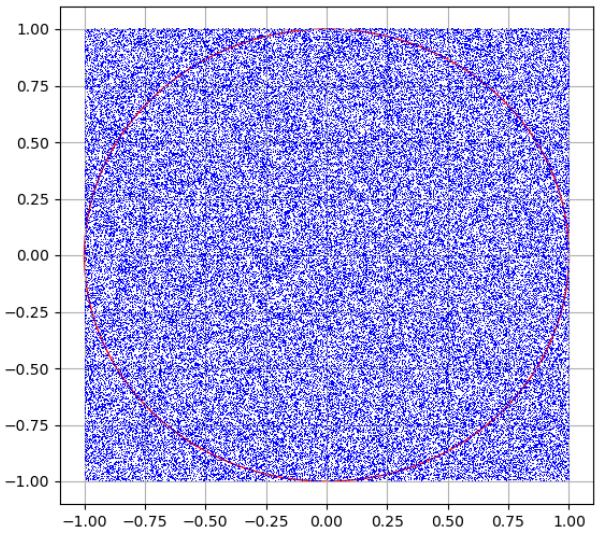


Figure 5: Plot of all the 10^5 random number couples $x, y \in [-1, 1]$ (in blue) and the circle of radius 1 and center 0 (in red).

3 Numerical estimate of π

It is possible to compute an estimate of π using Monte Carlo techniques.
I generate $N_{tot} = 10^5$ couples of random numbers (x, y), both between $[-1, 1]$, thus filling a square of side 2 centered in the origin of the Cartesian plane.
Given N_{circle} the number of couples that fall inside a circle of radius 1 centered in 0 (i.e. $x^2 + y^2 \leq 1$), it is possible to obtain an estimate of π, since the ratio between the number of points inside and the total number of points is equal to the ratio of the circle and square surfaces (Fig. 5):

$$\frac{N_{circle}}{N_{tot}} = \frac{\pi}{4} \;\rightarrow\; \pi = 4\,\frac{N_{circle}}{N_{tot}} \tag{8}$$

The value obtained using this method is $\pi_{MC} = 3.14912$. I expect that the more points I inject, the more accurate is the π estimate. To verify this hypothesis, I compute the difference between the expected and estimated values $\pi - \pi_{MC}$ in function of the number of random couple extractions N_{tot}. The result is shown in Fig. 6 , where it is possible to observe that this difference converges to zero for increasing N_{tot}.

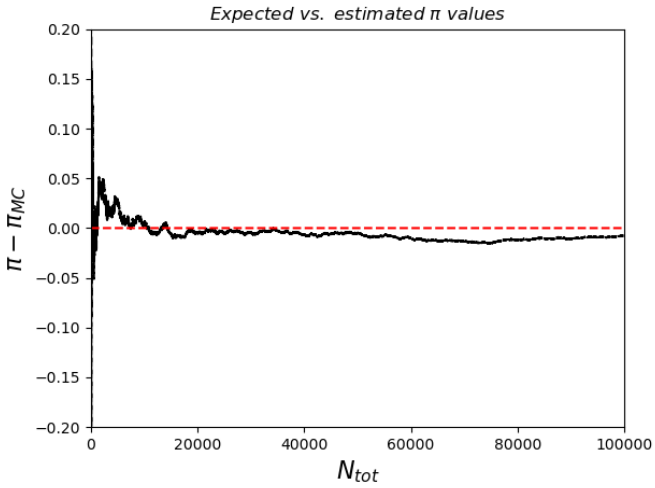


Figure 6: Difference between expected and estimated π values as a function of the number of extracted couples N_{tot}.

3.1 Uncertainty evaluation

In principle, the standard deviation σ on π estimates is expected to scale like:

$$\sigma = \frac{k}{\sqrt{N}} \tag{9}$$

Where N is the number of random couples and k is the characteristic standard deviation parameter. Since a π-distribution is expected to behave like a Gaussian function, I compute π 10^5 times using this Monte Carlo method, each time extracting $N = 100$ random couples. By fitting the resulting $\pi - \pi_{MC}$ distribution with a Gaussian function, I compute its best fit standard deviation $\sigma = 0.164$ (Fig. 7).
From this estimate, I can obtain the parameter k:

$$k = \sigma \cdot \sqrt{N} = 1.64 \tag{10}$$

Since this parameter is constant, it can be very useful for estimating how many random variable extractions are needed in order to compute π with a given precision. For example, if I want to evaluate π with an error of $\sigma \lesssim 10^{-4}$, I have to extract a number of couples:

$$N = \left(\frac{k}{\sigma}\right)^2 \gtrsim 2.7 \cdot 10^8 \tag{11}$$

Finally, to check if k is a constant parameter, I repeat the same calculation for the standard deviation σ also with $N = 1000, 5000$. The results are shown in

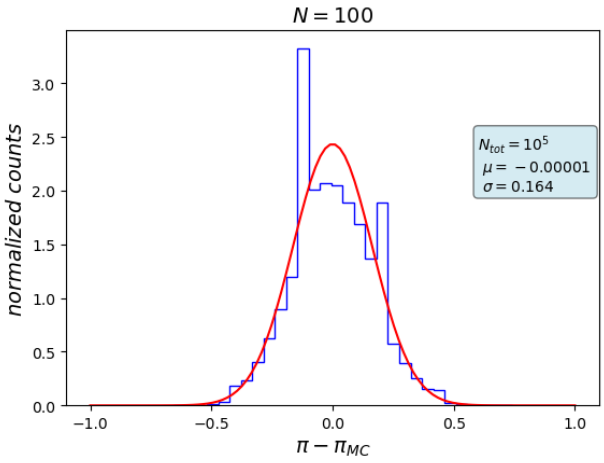


Figure 7: $\pi - \pi_{MC}$ distribution computed extracting $N = 100$ random couples $N_{tot} = 10^5$ times, in blue. The best fit Gaussian distribution is shown in red, with mean value μ and standard deviation σ.

Fig. 8, where it is plotted also the expected behavior of the standard deviation σ for the k value obtained before with $N = 100$. All the three points are distributed like the theoretical curve (Eq. 9), demonstrating that the parameter k does not depend on the number of couples N.

4 Monte Carlo integration

4.1 Unidimensional integration

I solve the following 1D integral using a Monte Carlo integration method:

$$I_n = \int_0^1 x^n dx = \frac{1}{1+n} \tag{12}$$

Where $n \in [1, 5]$. I extract $N_{ex} = 10^5$ random numbers and compute an approximate value using the mean integral theorem, which states that it is possible to approximate an integral by uniformly sampling its integration range with random numbers x_i. Thus, the integral can be written as:

$$I \simeq V \cdot \frac{1}{N_{ex}} \sum_{i=1}^{N_{ex}} f(x_i) \tag{13}$$

Where in this case the integration volume is simply $V = 1$.
The value obtained with this numerical approach has been compared with its

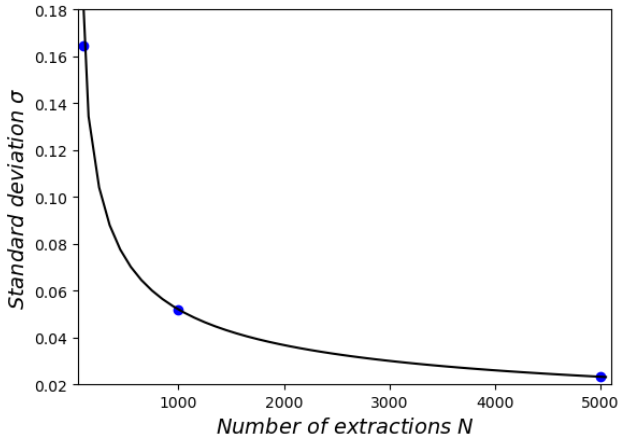


Figure 8: Standard deviation of the π estimate as a function of the extraction number N. The blue points are the σ values obtained for $N = 100, 1000, 5000$.

exact one. The results are shown in Fig. 9.
I repeat the same calculation $N = 100$ times with a number of extractions $N_{ex} = 10^4$ for each slope n, in order to build distributions of I_n estimates. Knowing the distribution, it is possible to compute the standard deviation σ and the mean value μ of the discrepancy between the analytical and numerical estimate of I_n.
Since the standard deviation scales like $\sigma = \frac{k}{\sqrt{N}}$, I can anticipate how many extractions are required in order to obtain an estimate with an error smaller than 10^{-3}, for each slope n. The results are shown in Tab. 1.

4.2 Integration in N dimensions

It is possible to use Monte Carlo also to solve a multi-dimensional integral like:

$$I_D = \prod_{i=1}^{D} \int_0^1 e^{-x_i} dx_i = \left(1 - \frac{1}{e}\right)^D \tag{14}$$

Where $D \in [1, 8]$ is the dimensionality of the integral. It is interesting to analyze how fast an integral estimate through Monte Carlo is in converging to the analytical value as a function of the dimensionality D with respect to a standard numerical method like the "mid-point" summation.
I perform this analysis by computing I_D with both these methods by requiring the number of histories (i.e. extractions) of the Monte Carlo N_h to be equal to the number of cells N_c employed for the mid-point calculation. These values vary with D according to Tab. 2.
Since the Monte Carlo error follows the central limit theorem, its relative error

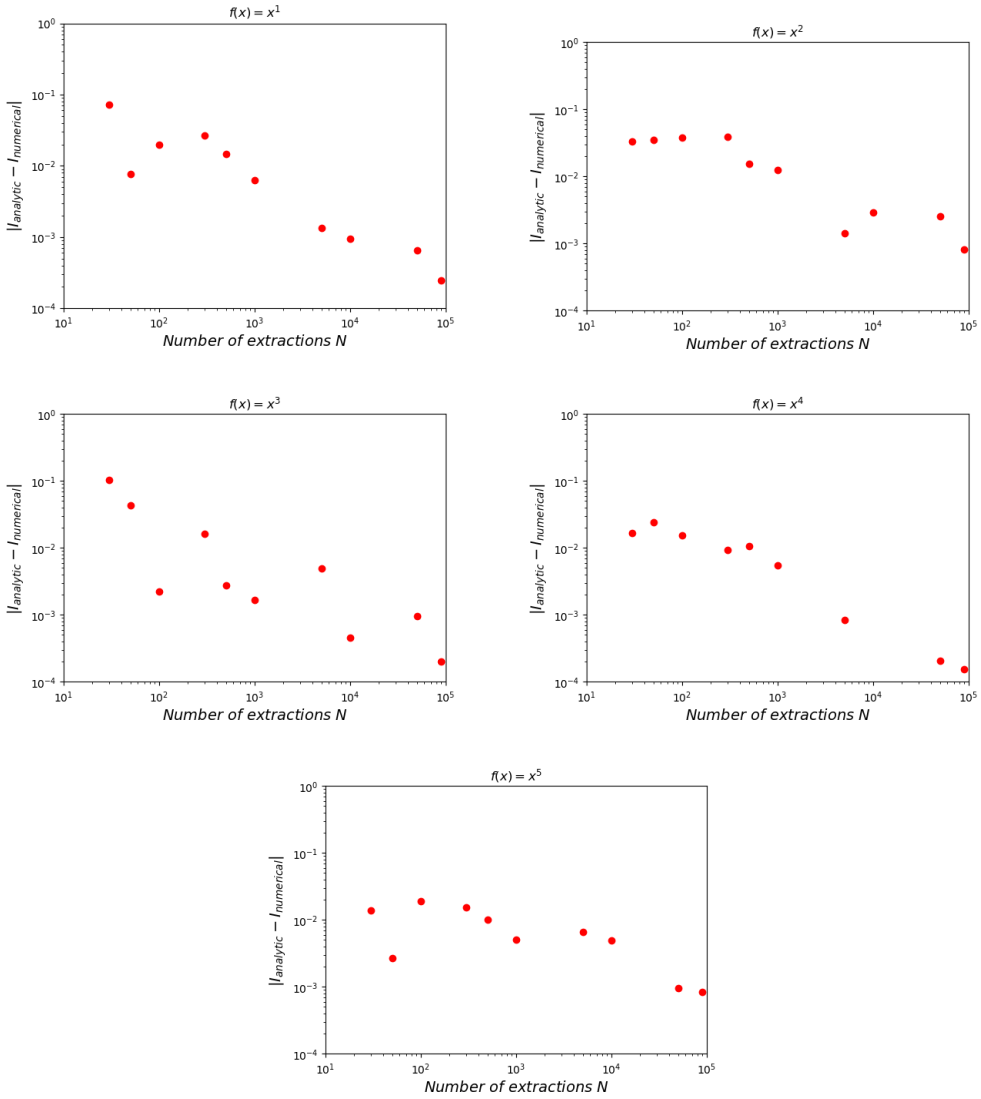


Figure 9: Absolute value of the difference between the integral computed analytically and the one computed numerically using Monte Carlo techniques, as a function of the extraction number N_{ex} for the 5 different slopes $n \in [1, 5]$ of the function $f(x) = x^n$.

n	μ	σ	$N_{ex}(\sigma < 10^{-3})$
1	$0.3 \cdot 10^{-4}$	0.0029	81395
2	$1.1 \cdot 10^{-4}$	0.0029	84825
3	$4.3 \cdot 10^{-4}$	0.0029	82508
4	$1.6 \cdot 10^{-4}$	0.0028	80405
5	$1.3 \cdot 10^{-4}$	0.0026	68090

Table 1: Mean value μ, standard deviation σ and extraction number N_{ex} required to have and error less than 0.001, as a function of the various slopes n. These values are obtained from a distribution of $N = 100$ estimates of the difference $I_{analytical} - I_{numerical}$ through Monte Carlo integration with $N = 10^4$ extractions.

D	1	2	3	4	5	6	7	8
$N_h = N_c$	65536	256^2	40^3	16^4	9^5	6^6	5^7	4^8

Table 2: Number of MC histories N_h and cells N_c for the various dimensionality D.

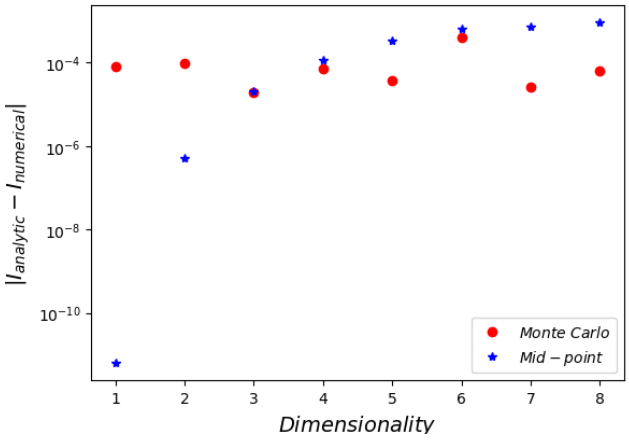


Figure 10: Discrepancies obtained through Monte Carlo (red dots) and mid-point (blue stars) methods as a function of the dimensionality.

scales like $\Delta I/I \propto N^{-1/2}$ and depends only on the number of histories N_h.
On the other hand, the mid-point relative error scales like $\Delta I/I \propto N^{-2/D}$, thus for $D > 4$ the Monte Carlo method is faster than the mid-point one.
In Fig. 10 I plot the absolute value of the difference between analytical and numerical integrals $|I_{analytic} - I_{numerical}|$ as a function of the dimensionality D, clearly supporting this behavior.

5 Truncation errors

I numerically compute the sum:

$$X = \sum_{i=1}^{N} \frac{1}{N} = 1 \tag{15}$$

Where $N = [10^4,\ 10^5,\ 10^6]$. I store X and $1/N$ as `float` and `double` variable types (i.e. variables which occupy 32 and 64 bit, in Python `float32` and `float64`, respectively). Since these values can occupy a limited amount of memory, I expect that the variable X will not be exactly 1, instead it will have a truncation error which will be more severe (i.e. higher) the less number of bit it occupies. The truncation errors are shown in Tab. 3.

N	`float`	`double`
10^4	$2.5 \cdot 10^{-8}$	$9.4 \cdot 10^{-14}$
10^5	$2.5 \cdot 10^{-8}$	$1.9 \cdot 10^{-12}$
10^6	$2.5 \cdot 10^{-9}$	$7.9 \cdot 10^{-12}$

Table 3: Differences between the analytical and numerical estimate of the sum $\sum_1^N 1/N$ storing `float` and `double` variables for different N.

6 Tracking algorithms

Tracking algorithms are used to simulate the propagation of particles across a medium, taking into account the dynamics of the system and all the peculiar interactions that can influence it. The medium can be divided in regions where path lengths between two subsequent interactions are sampled from the PDF independently (Markov process).
For tracking algorithms, the PDF $p(s)$ has the form:

$$p(s) = \mu e^{-\mu s} \tag{16}$$

Where s is the step length and μ is the so-called interaction or attenuation coefficient, which encloses the information about variations of the survival probability of the particle.
There are two equivalent ways to carry out the step length sampling.
The first one, employed in Geant4, is to propose an individual step length s_i, sampled from $p(s_i) = \mu_i e^{-\mu_i s_i}$, where $i = 1, ..., N$ with N number of interactions. The final step length is chosen as $s = \min\{s_1, ..., s_N\}$.
The second one, employed in Penelope, is to directly sample s according to the exponential $\mu e^{-\mu s}$, where $\mu = \sum_i \mu_i$ is the sum of all the N attenuation coefficients.
To prove that these two methods are equivalent, I use the one in Geant4, simulating a set of s_i by sampling random numbers and taking the minimum. In this case $i \in \{1, 2\}$, $s = \min\{s_1, s_2\}$ and for simplicity $\mu_1 = 1$ and $\mu_2 = 2$.
If the two methods are equivalent, by fitting the distribution of s obtained using the Geant4 method with the function $p(s) = \mu e^{-\mu s}$ and keeping μ as a free parameter, I should find the attenuation coefficient used in Penelope's PDF as best fit, in this case $\mu = \mu_1 + \mu_2 = 3$. The result is shown in Fig. 11.
This equivalence can be proved also analytically, by defining the cumulative probability that an exponentially distributed random variable s is greater than

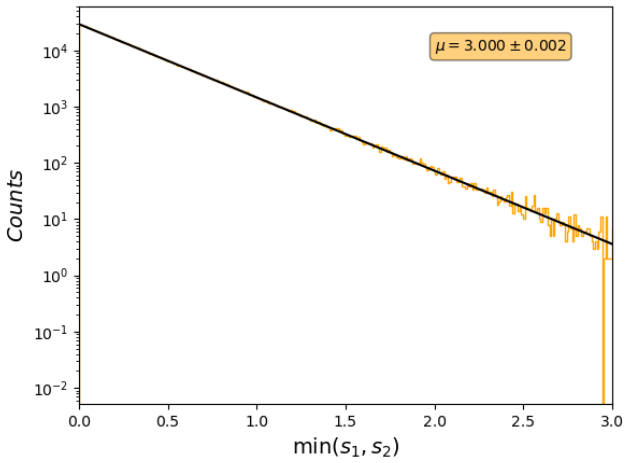


Figure 11: Simulated step length distribution (in orange) vs. best-fit model (in black). The best-fit value of the attenuation parameter is $\mu \simeq 3$, which is the one expected by Penelope.

$\min\{s_1, s_2\}$ as:

$$\begin{aligned} P(s > \min\{s_1, s_2\}) &= 1 - P(s < \min\{s_1, s_2\}) = \\ &= 1 - P(s < s_1) \cdot P(s < s_2) = \\ &= 1 - e^{-\mu_1 s} \cdot e^{-\mu_2 s} = 1 - e^{-(\mu_1+\mu_2)s} \end{aligned} \tag{17}$$

Where I used the fact that s_1 and s_2 are independent variables and the definition of cumulative probability distribution as integral of the PDF, i.e. :

$$P(s < s_{1,2}) = \int_s^\infty dx \ \mu_{1,2} e^{-\mu_{1,2} x} = e^{-\mu_{1,2} s} \tag{18}$$

The PDF associated to this cumulative probability distribution $P(s > \min\{s_1, s_2\})$ can be written as:

$$\begin{aligned} p(\min\{s_1, s_2\}) = \frac{dP(s > \min\{s_1, s_2\})}{ds} = \frac{d}{ds}\left(1 - e^{-(\mu_1+\mu_2)s}\right) = \\ = (\mu_1 + \mu_2) e^{-(\mu_1+\mu_2)s} = p(s) \end{aligned} \tag{19}$$

Which demonstrates that sampling from a PDF with μ given by the sum over all the possible interactions (like Penelope) is the same of sampling from different interactions and take the minimum step length (like Geant4).
From this discussion, one can also calculate the probability that s_i is the minimum, thus it will be sampled according to the Geant4 method. For example,

let's define $P(s_1 = \min\{s_1, s_2\})$ as:

$$\begin{aligned} P(s_1 = \min\{s_1, s_2\}) &= \int_0^\infty ds\ P(s = s_1) \cdot P(s > s_2) = \\ &= \int_0^\infty ds\ \mu_1\ e^{-\mu_1 s} \cdot e^{-\mu_2 s} = \\ &= \mu_1 \int_0^\infty ds\ e^{-(\mu_1+\mu_2)s} = \\ &= \frac{\mu_1}{\mu_1 + \mu_2} \end{aligned} \tag{20}$$

Similarly, it can be shown that $P(s_2 = \min\{s_1, s_2\}) = \mu_2/(\mu_1 + \mu_2)$.
These probabilities define the sampling fractions associated to a given process with attenuation coefficient μ_i, given by the ratio between μ_i and the sum over all the interactions $\sum_i \mu_i$.
In the scenario of this exercise, I expect that s_1 and s_2 will be sampled 1/3 and 2/3 of the times, respectively. By counting how many times s_1 has been selected from the code, I computed the sampling fractions, which are 0.33 and 0.66 for s_1 and s_2, respectively, hence supporting this theoretical calculation.
All these mathematical demonstrations can be easily generalized for the general case in which the number of processes is N instead of 2.

7 Sampling of an interaction

I want to sample the final state of particles following a photoelectric interaction of a gamma-ray of energy $E_\gamma = 1$ MeV with the K-shell of a Pb atom, with atomic number $Z = 82$ and binding energy $U = 88.01$ keV.

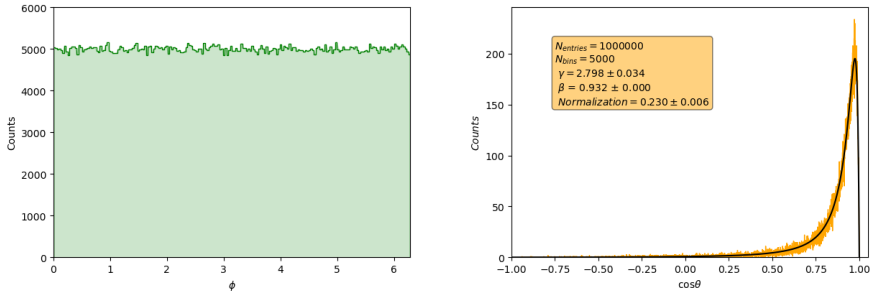
7.1 Photo-electron

A photon can be absorbed by lead for photoelectric effect, exciting the electrons that compose its atoms. If the photon is energetic enough, it can transfer its energy to the electrons and unbind them from the nucleus, leading to the emission of a free electron, called "photo-electron", with an energy E_e given by:

$$E_e = E_\gamma - U \tag{21}$$

The emission direction is isotropic in the azimuthal angle $\phi \in [0, 2\pi]$, while the polar angle $\theta \in [0, \pi]$ is given by the differential cross section $d\sigma/d\Omega$ described by the Sauter's formula:

$$\frac{d\sigma}{d\Omega} \propto \frac{\sin^2\theta}{(1 - \beta\ \cos\theta)^4} \left[1 + \frac{1}{2}\gamma(\gamma - 1)(\gamma - 2)(1 - \beta\cos\theta)\right] \tag{22}$$



(a) (b)

Figure 12: *Left panel:* Uniform distribution of the azimuthal angle ϕ obtained through differential cross section sampling.
Right panel: $\cos\theta$ distribution (in orange) and its best-fit curve (in black). By fitting this distribution with the Sauter's formula, I computed the best-fit values $\gamma \simeq 2.798$ and $\beta \simeq 0.932$, very close to the exact values $\gamma = 2.79$ and $\beta = 0.93$.

Where γ and β are parameters which depend on E_e, defined as:

$$\begin{aligned} \beta =& \frac{\sqrt{E_e(E_e + 2m_e c^2)}}{E_e + m_e c^2} \sim 0.93 \\ \gamma =& 1 + \frac{E_e}{m_e c^2} \sim 2.79 \end{aligned} \tag{23}$$

I want to know the emission direction of the photo-electron after the photoelectric interaction. Supposing that the initial photon was traveling along the z-axis, with coordinates (0, 0, 1), I can obtain information on the electron propagation direction by sampling the Sauter's formula using Monte Carlo techniques.
Eq. 22 shows a very peculiar shape of this function, thus the inversion sampling method can not be applied in this case. I decide to employ the rejection method for sampling this distribution as a function of $\cos\theta$, using as auxiliary PDF an uniform distribution.
I simulate $N_{ex} = 10^6$ electrons by extracting N_{ex} random number couples (x, y), with $x \in [-1, 1]$ and $y \in [0, 900]$, considering that the maximum of the distribution is $\sim$ 894. Of all these couples, I keep only the ones that satisfy the acceptance criterion, i.e. the ones below the curve described by Eq. 22. The results are shown in Fig. 12.
This is a perfect example of how useful the rejection method is, allowing the sampling of any kind of distribution. Nonetheless, depending on the shape of the sampled function, this method can be very inefficient. It is possible to define an efficiency $\epsilon = S_p/S_\pi$, where S_p and S_π are the areas below the primary distribution p and the auxiliary one π, respectively.
Using this particular choice of auxiliary distribution, The cross section sampling efficiency is $\epsilon \sim 6\%$, clearly showing that a high fraction of the simulated elec-

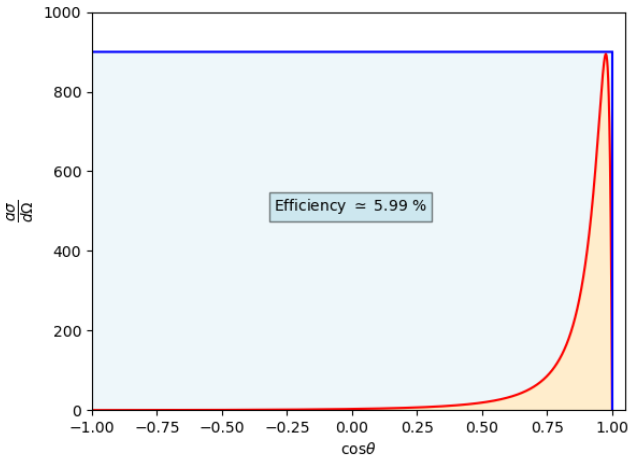


Figure 13: Plot of the differential cross section as a function of $\cos\theta$ (in red) and the rectangle used to sample it through rejection method (in blue). The light-blue and orange shaded areas illustrate the surfaces below the rectangle and Sauter's curve, respectively, showing how inefficient this method is in sampling this particular distribution.

trons is rejected by the acceptance criterion (Fig. 13).
Once the polar and azimuth angles are sampled from the distribution, it is possible to understand the photo-electron emission direction. To better visualize the results, I plot the sampled points on a sphere of unitary radius, with the hypothesis that the initial photon was traveling along the z-axis. If instead the gamma-ray was initially propagating along another direction, for example the y-axis (0, 1, 0), the sampling process would be the same, with the caveat of rotating the sampled coordinates according to the new initial direction. The results are shown in Fig. 14.

7.2 Fluorescence

The vacancy in the K-shell formed by the emission of the photo-electron can be filled by an electron previously occupying an higher energy level. The most probable one is called L-shell, and it can be divided in three energy levels (L_I, L_{II} and L_{III}). Each level has an energy U_L of 15.86 keV, 15.20 keV and 13.04 keV and is populated by a 2, 2 and 4 orbital electrons, respectively.
To move from the L-shell to the K-shell, the orbital electron loses its energy by emitting a K_α X-ray photon, with an energy $E_{K_\alpha} = U - U_L$. The remaining energy is used to emit a second, less energetic photon.
In order to analyze this interaction, I sampled $N_{ex} = 10^5$ times the polar and azimuthal angles ($\cos\theta$ and ϕ) of the X-ray photons, which are emitted isotrop-

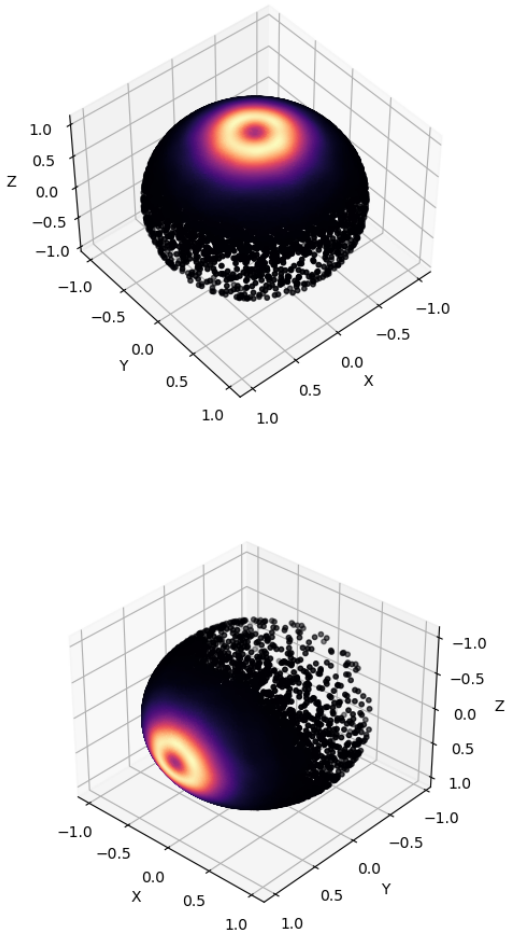


Figure 14: 3D plots of the sampled coordinates of the electron directions after a photoelectric interaction with a photon initially propagating along the z-axis (upper panel) and y-axis (lower panel). The points are symmetric around the initial propagation axis, since the differential cross section does not depend on the azimuthal angle ϕ. The increasing point density is displayed through colors, from cold to hot tones. The point density resembles the most probable emission direction, and it clearly reflects the Sauter's formula behavior, since the most probable polar emission angle is small, producing the yellow rings around the photon propagation direction.

ically.
Together with the emission direction, I sampled the energies of the two emitted photons, considering that an atomic transition is more probable if the starting energy level is more populated. The results are shown in Fig. 15.

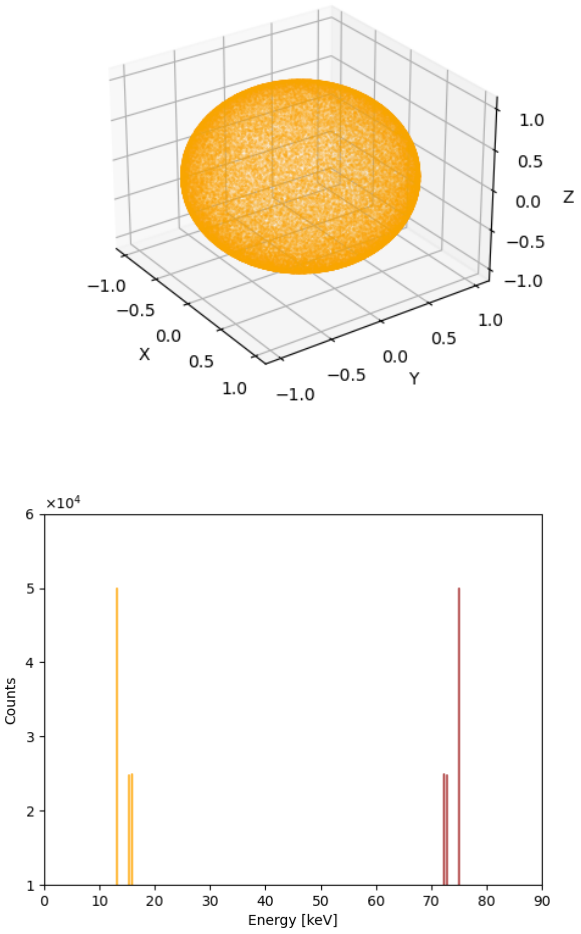


Figure 15: *Upper panel:* Isotropically distributed emission directions of the two X-ray photons.
Lower panel: Energy spectrum of the X-ray photons emitted through fluorescence. The K_α photons are shown in brown, while the secondary X-ray photons in orange.