GSSI LE-6 Course
Monte Carlo Techniques
Exercises

Luciano Pandola
Academic Year 2020-2021

1 Uniform random sampling

Write a computer code that simply performs the following:

```
:Initialization:   i = 987654321
:Iteration:        i = i*663608941
```

Verify that the sequence length is 2^{30} by seeing that `i` returns to its original seed. How much CPU time does it take to go through the whole sequence? Normalize the value `i` so to generate pseudo-random numbers between 0 and 1.
Notice: if you use an unsigned integer for the variable `i`, the allowed maximum for `i` will be the content of 4-bytes (i.e. 2^{32}-1). If you use a signed integer, one bit is taken by the sign.
Does the distribution look flat? Test the assumption that the distribution is uniform between 0 and 1 using a χ^2 test (build a histogram with $N = 100$ bins). What if you take only a part of the sequence (i.e. the first 10^3, or the first 10^6 events) ?

1.1 MINSTD algorithm

Try to code the MINSTD generator

$$R_n = 7^5 R_{n-1} \ \mathrm{mod}(2^{31} - 1)$$
$$\xi_n = \frac{R_n}{2^{31} - 1} \tag{1}$$

with $m = 2^{31} - 1$ and $a = 7^5 = 16807$.
How long is the sequence? How does the sequence lenght compare to m? Does the period depend on the value of the seed R_0?

2 Random sampling from other distributions

2.1 Inversion method

Form the cumulative probability distribution for the small-angle Rutherford distribution

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

Invert it to indicate how x would be determined from a uniform random number ξ.

2.2 Inversion and rejection

Implement a computer code which makes a uniform sampling (x, y) from a unit circle. Try both the analytic method:

$$x = \sqrt{\xi}_1 \cos(2\pi\xi_2) \tag{3}$$

$$y = \sqrt{\xi}_1 \sin(2\pi\xi_2) \tag{4}$$

and the rejection method.

$$x = -1 + 2\xi_1 \tag{5}$$

$$y = -1 + 2\xi_2 \tag{6}$$

$$\text{provided } (x^2 + y^2 \leq 1) \tag{7}$$

Which one is the fastest? Is it better to have higher efficiency (= analytic) or to save on trascendent functions (= rejection) ?

Notice: if you perform this exercise in the ROOT environment, results might be affected by the compiler optimization options (which are given automatically by ROOT).

3 Numerical estimate of π

Write a computer code to estimate the value of π by Monte Carlo. Shoot unifomly points (x, y) on a square $-1 < x < 1$ and $-1 < y < 1$ and count of many of them fall into the circle. Plot the difference between the estimate and the true value of π vs. the number of generated couples, to check the convergence ($\pi = 3.1415926535$).

3.1 Uncertainty evaluation

Evaluate also the uncertainty of the estimate: in principle, the standard deviation on π is expected to scale as $\frac{k}{\sqrt{N}}$. Since the estimator in this case is not based on a simple mean, use a brute force approach: repeat many times the evaluation of π with $N = 100$ and build the distribution of the estimates (a Gaussian, hopefully). [Notice: you need *independent* random sequences, so pay attention to the definition of the random seed.].
How wide the distribution is? Use the sigma of the distribution to estimate the parameter k.
Try the same with $N = 1000$ and $N = 5000$. Do the points distribute on the curve $\sigma = \frac{k}{\sqrt{N}}$?
Using the estimate of k, how many couples N you need to estimate π at better than 10^{-4} (i.e. $\sigma < 10^{-4}$) ?

4 (Optional) A toy Monte Carlo, RisiKo!

Monte Carlo simulations are not used only for particle tracking, but also for solving problems which cannot be handled analytically in an efficient way. This includes, for instance, the calculation of expected distribution for test statistics (e.g. in the cases when the usual χ^2 theorems do not apply).
Let us now use the random numbers to determine a best strategy in a game of RisiKo! As all Italians know, RisiKo! is a board game which depicts a war: one has to conquer territories and the outcome of a battle (win/loss of armies) is determined by the launch of three dice. Attacker and defender will roll three dice each and the outcoming values are compared one-by-one (highest vs. highest, middle vs. middle, smallest vs. smallest), to determine the destiny of one army. In case of tie, the defender wins. So, each roll of dice with decide 3 armies (possible outcomes are: 3-0, 2-1, 1-2 and 0-3). There are additional rules (e.g. if the defender has less than $N < 3$ armies left, he/she can roll only N dice, etc.), but I assume that you are familiar with them. If not (especially for non-Italians), please ask your Italian friends ;-)

4.1 Basic probability

Simulate many 3-dice attacks, and build the distribution of possible outcomes. Calculate the average number $\langle A \rangle$ of armies that are won by the attacker for each 3-dice roll and the average number per die $\langle A_d \rangle = \langle A \rangle / 3$. Due to the tie rule, of course it will be $\langle A_d \rangle < 0.5$, i.e. the defender has an advantage. Strictly speaking, you do not need a Monte Carlo simulation for this task: you can consider the $6^3 \times 6^3 = 46656$ possible combinations, if you wish.

4.2 Planning an attack

You want to conquer Kamchatka, which is defended by $N_D = 3$ armies. You want to be fairly sure that your attack is successful. So, run a toy Monte Carlo simulation to estimate:

- what is the minimum number of armies N_A that you have to deploy, if you want to have a probability $> 80\%$ to get the territory. Do not forget to implement the additional rules that apply when the defender has less than 3 armies left (or the attacker has less than 4). How does N_A compare against $N_D / \langle A_d \rangle$?
- for the N_A which guarantees a 80% success, build the distribution of armies that are left at the end of the attack. What is the probability that you are left with at least six armies, i.e. the minimum required to keep decently both territories? (3 armies on the original one, 3 armies on the conquered Kamchatka)
- how many armies N_A you have to deploy if you want that in 80% of the outcomes you win the terroritory with at least 6 armies left?

5 Monte Carlo integration

5.1 Unidimensional integration

Solve the 1-dimensional integral by Monte Carlo integration

$$I_n = \int_0^1 x^n dx \tag{8}$$

with $n = 1 \ldots 5$. Compare the result of the Monte Carlo integration with the exact mathematical value $I_n = \frac{1}{n+1}$. Once fixed a number N, repeat many times the calculation (by using different random seeds) and build the distribution of the results I_n.
Anticipate (= estimate in advance) how many random points do you need to sample in order to evaluate the integral I_n at better than 0.001.

5.2 Integration in N dimensions

Solve the following integral both by Monte Carlo (seeding a D-dimensional unit cube) and also by simple mid-point summation:

$$dx^D I = \prod_{i=1}^{D} \int_0^1 dx_i e^{-x_i} \tag{9}$$

for $D = 1, 2, 3, 4, 5, 6, 7, 8$. D is the dimensionality of the problem. Make the number of cells N_c and the number of histories N_h equal in each case, so employ the following table:

D	1	2	3	4	5	6	7	8
N_c	65536	256^2	40^3	16^4	9^5	6^6	5^7	4^8
N_h	65536	65536	64000	65536	59049	46656	78125	65536

Compare with the simple mathematical result $(1 - 1/e)^D$. What conclusion can you make regarding the computational efficiency of either technique as the dimensionality increases?
Notice: you are *not* allowed to transform

$$I = \prod_{i=1}^{D} \int_0^1 dx_i e^{-x_i} \quad \text{into} \quad \int_0^1 dx_i e^{\sum -x_i}$$

Please calculate the product explicitly ;-).

5.3 Extra exercise

Try the same with the function

$$I = \int_0^1 \int_0^1 \ldots \int_0^1 (x_1^2 + x_2^2 + \ldots + x_D^2) dx^D \tag{10}$$

for $D = 1 \ldots 8$ (use the same N_c and N_h as above in Sect. 5.2). Compare against the exact mathematical result $I = D/3$.

6 Truncation errors

Try to calculate

$$X = \sum_{i=1}^{N} \frac{1}{N} \tag{11}$$

by storing $\frac{1}{N}$ and X as `float` (i.e. single-precision) variable. Which result do you get for $N = 10\ 000$, $N = 100\ 000$ and $N = 1\ 000\ 000$?
Now try to use `double` (i.e. double precision) variables. Does it change the outcome?

7 Tracking algorithms

Prove that the two ways of sampling of the step length s employed by Geant4 and Penelope are equivalent. In other words, demonstrate that sampling the individual step lenghts s_i from $\mu_i e^{-\mu_i s_i}$ and then taking $s = \min\{s_1 \dots s_N\}$ is the same as sampling s from $\mu e^{\mu s}$, with $\mu = \sum_i \mu_i$.
Do it in two ways:

1. Pragmatically (= brute force). Simulate a set of s_i with $\mu_1 = 1$ and $\mu_2 = 2$, by sampling random numbers. Take $s = \min\{s_1, s_2\}$ and show that s is distributed as $e^{-\mu s}$, with $\mu = \mu_1 + \mu_2 = 3$.

2. Mathematically: Build the distribution $p(s)$ starting from s_1 and s_2. Do not forget the normalization constants of the pdf.

3. (Optional): Generalize the proof of (2) to the case of N competing processes $\mu_1 \dots \mu_N$.

Also, prove that the sampling fraction of process μ_i is $\frac{\mu_i}{\sum \mu_i}$. Again, this can either verified either by brute force Monte Carlo, or demonstrated in a polished mathematical proof.

8 Sampling of an interaction

Sample the final state (direction and final energy of the outgoing particles) following the photo-eletric effect of a γ-ray of energy $E = 1$ MeV on the K-shell of Pb (binding energy $U = 88.01$ keV).

8.1 Photo-electron

The primary photon here is absorbed and a photo-electron is released from the inner atomic shell. The energy of the electron is

$$E_e = E - U \tag{12}$$

i.e. the difference between the initial photon energy and the binding energy of the electron.

The direction $\cos\theta$ can be described according to the Sauter's formula:

$$\frac{d\sigma}{d\Omega} = \alpha^4 \frac{r_e^2}{2} \left(\frac{Z}{\kappa}\right)^5 \frac{\beta^3}{\gamma} \frac{\sin^2\theta}{(1-\beta\cos\theta)^4} \left[1 + \frac{1}{2}\gamma(\gamma-1)(\gamma-2)(1-\beta\cos\theta)\right] \tag{13}$$

($r_e = 2.815$ fm is the electron classical radius, α is the fine-structure constant and $\kappa = E/(m_e c^2)$ is the photon energy in units of electron mass). One has

$$\beta = \frac{\sqrt{E_e(E_e + 2m_e c^2)}}{E_e + m_e c^2} \tag{14}$$

and

$$\gamma = 1 + \frac{E_e}{m_e c^2} \tag{15}$$

Notice that the term

$$\alpha^4 \frac{r_e^2}{2} \left(\frac{Z}{\kappa}\right)^5 \frac{\beta^3}{\gamma} \tag{16}$$

is just a global normalization factor which does not depend on $\cos\theta$, so that it can ignored in the present problem: it is required to derive μ, but not to sample the final state.

Sample $\cos\theta$ and ϕ appropriately in order to calculate the direction $\vec{u}$ of the outgoing electron, in the assumption that the incoming photon travels along the z-direction, i.e. $(0, 0, 1)$. Produce a distribution of $\cos\theta$ and check that it matches the shape of the differential cross section.

Hint: plot/build the cross section as a function of $\cos\theta$ and sample it using the rejection method.

What if the photon were traveling along the y-axis $(0, 1, 0)$? Plot the distribution of $\cos\theta$ in this case (notice: θ is here the polar angle referred to the global reference frame).

8.2 Fluorescence

How would the residual energy U be released?

As a zero-th order assumption, release a K_α x-ray (i.e. move the vacancy from the K to the L shell) and then release the residual energy in one more photon. The three L-levels (L_I, L_{II} and L_{III}) have energy 15.86 keV, 15.20 keV and 13.04 keV, and they are populated by 2, 2 and 4 orbital electrons, respectively.

Sample randomly the intermediate level (L_I, L_{II} or L_{III}) according to the respective number of orbital electrons, then calculate the energy of the two x-rays of the cascade (K_α + the rest) and finally sample their directions. For simplicity, we are not considering here the possibility that a fluorescence x-ray can be converted in a Auger electron.