

Lab # 3 Monte Carlo Simulation & Confidence Intervals

For the next questions under Linux even if it is simple to use **rand()** for a first test (or more precisely (double) **rand()** / (double) **RAND_MAX** using **stdlib**; you now know that for scientific applications default system random number generators are often very old and statistically weak. **They have to be banned for scientific programming**. It is far better to use a generator such as Mersenne Twister proposed by Matsumoto. You can find the current implementation in C of Mersenne Twister (MT) go to “Matsumoto Home Page” – Mersenne Twister – 2002 version – explanation & C code. Download the .tar file with the source code + expected output and readme. Compile and test if you obtain the expected output (for portability & **reproducibility** reasons), then use for instance the **gen_ran1** or **gen_ran2** functions.

- 1) Compute π with the Monte Carlo method (all these lab implementations are still proposed in C). Test your code with 1 000 points, 1 000 000 points and lastly 1 000 000 000 points (meaning 2 billion pseudo random numbers drawn for the sampling of a single run (experiment)). For this case study let see how many drawings you may need to get a precision at 10^{-3} (3.141) and then at 10^{-4} (3.1415). We don't have replicates in this question.
- 2) Propose a loop on this function to compute n independent experiments (replicates). This is simply achieved if we do no reinitialize the pseudo-random number generator between two experiments (replicates). With the results of N experiments (replicates) proposes a final result which is the mean of all the estimated PI.
- 3) Computing of confidence intervals: the user inputs the number of replicates (experiments) he wants to perform before computing a confidence interval at 95% ($\alpha=0.05$) – **to do so use the technique & table given in appendix**. You can compare your result with the **M_PI** (constant often defined in `<math.h>`) and see whether the number of replicates improves your results and decreases the confidence radius. The number of random drawings (sampling) for each individual replicate (each experiment) is also a sensitive parameter. Be careful when you compare your result with **M_PI** since you have indeed very few significant numbers in a float variable (at least 7 significant numbers).

The Monte Carlo method needs a lot of random sampling to increase its precision (remember the slow convergence rate – square root of the number of drawings, but this method is the only one available to compute volumes or hyper-volumes in hyper-spaces or to achieve evenly distributed space filling in large dimensions where accurate mathematical methods are often intractable or non-existing.

Life Science model – the origins:

- 4) Propose a quick code simulating a rabbit population growth. The unit is a couple of rabbits and the time step is of a month. We consider a that a couple of young rabbits become adult in one month, when the female rabbit becomes pregnant, it takes one month to produce 2 little rabbits considered as new young rabbits that will wait for another month to become adult etc... (see figure 1). If we suppose that rabbits do not die, we obtain the diagram below (observation made by Leonardo of Pizza). <http://www.maths.surrey.ac.uk/hosted-sites/R.Knott/Fibonacci/fibnat.html>

NOTE : Prepare a written report of this lab – it will be part of your grading.

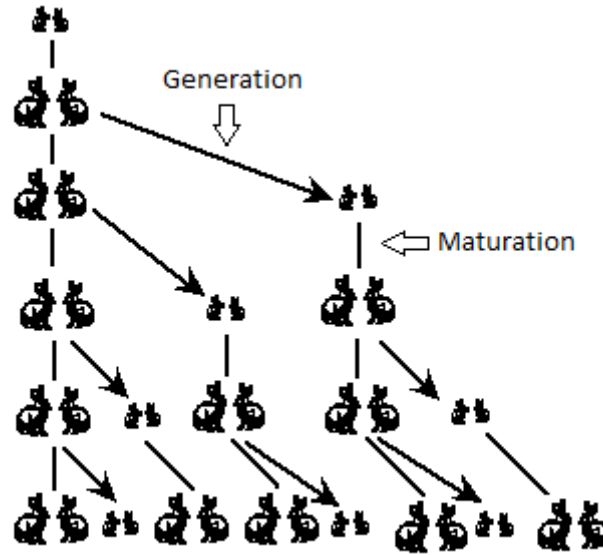


Figure 1: Rabbit population growth (empty spot for a young couple and a filled spot for a couple of adults)

Appendix: Computing confidence Intervals

Introduction : If X is a simulation result, (X_1, \dots, X_n) is the set obtained with n independent replications (experiments) of this stochastic simulation. This means that each independent stochastic simulation experiment is run (under the same conditions) but with a different (and independent) random stream. Usually the computing of confidence intervals is achieved on arithmetical means.

$$\bar{X}(n) = \sum_{i=1}^n X_i / n$$

The computing of a confidence interval is simple when we consider that the X_i variables (simulation results) have identical independent Gaussian distributions.

Principle : The confidence interval is centered around the arithmetical mean, so we just compute what is called the confidence radius. Here is the theoretical statistical hypothesis that is used to obtain this radius. If X_i have identical independent Gaussian distributions with a mean μ and a variance of σ^2 , then the following random variable:

$$T(n) = \frac{\bar{X}(n) - \mu}{\sqrt{S^2(n)/n}} \text{ is distributed according to a Student law with } n-1 \text{ degrees of liberty.}$$

$$S^2(n) = \frac{\sum_{i=1}^n [X_i - \bar{X}(n)]^2}{n-1} \text{ is an estimate without bias of the } \sigma^2 \text{ variance.}$$

The confidence radius, at the $1-\alpha$ level, is given by:

$$R = t_{n-1, 1-\alpha/2} \times \sqrt{\frac{S^2(n)}{n}}$$

Where $t_{n-1, 1-\alpha/2}$ is the quantile of order $1-\alpha/2$ of a Student law with $n-1$ degrees of liberty.

(meaning that it is the value which has a probability of $1-\alpha/2$ to be bypassed, or since the Student distribution is symmetric, we have a probability $1-\alpha$ to be bypassed in absolute value)

Table 1 below give, the values $t_{n-1,1-\alpha/2}$ for $\alpha = 0.05$ with one degree of freedom and depending on n .

n being the number of replications.

The computing of R gives the following interval $[\bar{X} - R, \bar{X} + R]$, with a $1-\alpha$ confidence.

It only necessitates to compute $\text{sqrt}(S^2(n)/n)$ and to use the table hereafter to obtain the value of $t_{n-1,1-\alpha/2}$

If $\alpha = 0.05$, the confidence interval is said at 95%,

Table 1: values of $t_{n-1,1-\alpha/2}$ with one degree of freedom and depending on n .

$1 \leq n \leq 10$	$t_{n-1,1-\alpha/2}$	$11 \leq n \leq 20$	$t_{n-1,1-\alpha/2}$	$21 \leq n \leq 30$	$t_{n-1,1-\alpha/2}$	$n > 30$	$t_{n-1,1-\alpha/2}$
1	12.706	11	2.201	21	2.080	40	2.021
2	4.303	12	2.179	22	2.074	80	2.000
3	3.182	13	2.160	23	2.069	120	1.980
4	2.776	14	2.145	24	2.064	$+\infty$	1.960
5	2.571	15	2.131	25	2.060		
6	2.447	16	2.120	26	2.056		
7	2.365	17	2.110	27	2.052		
8	2.308	18	2.101	28	2.048		
9	2.262	19	2.093	29	2.045		
10	2.228	20	2.086	30	2.042		