Lab # 3 Monte Carlo Simulation & Confidence Intervals

For the questions of lab n°3 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**. Such generators have to be banned for scientific programming**. It is far better to use a generator such as Mersenne Twister proposed by Matsumoto and discovered in lab 2.

1) Compute π with the Monte Carlo method (implementations for this lab 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^{-2} (3.14) and then at 10^{-4} (3.1415) – remember that the convergence rate is very slow (`sqrt(of the total numbers)`). You don't have to make replicates for 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). Store the results of *n* experiments (replicates) in an array and propose a final result which is the mean of all the 'estimated PIs'.

3) Computing of confidence intervals around the mean: the user inputs the number of replicates (experiments) he wants to perform before computing a confidence interval at 95% (α=0.05) – **to do so use the technique & table given in appendix of this lab**. 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 results with M_PI since you have indeed very few significant numbers in a float variable (only 7 significant numbers for floats). 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 (up to 623 dimensions with Mersenne Twister).

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 hereafter (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.

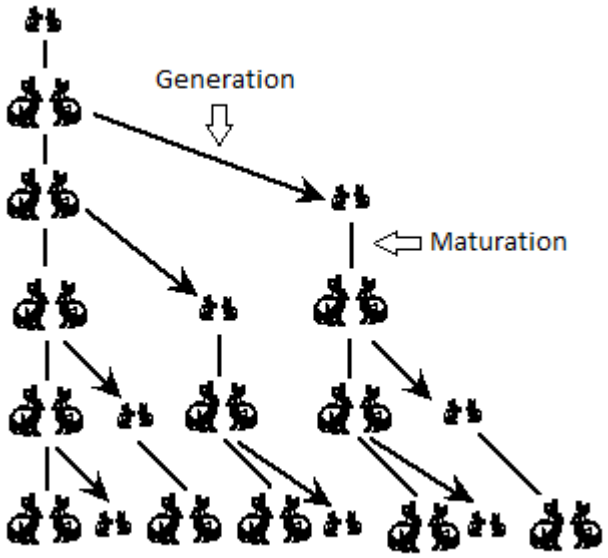

Figure 1: Rabbit population growth (empty spot for a young couple and a filled spot for a couple of adults)

Appendix: Computing confidence Intervals

Introduction : If *X* is a simulation result, (X_1 , ... , 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.

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

The computing of a confidence interval is simple when we consider that the X_i variables (simulation result++++s) 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 theoretical mean μ and a variance of σ^2, then the following random variable :

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

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

The confidence radius, at the 1-α 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-α 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 $\left[\overline{X} - R, \overline{X} + R\right]$, with a 1-$\alpha$ confidence.
It only necessitates to compute 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≤n≤10	$t_{n-1,1-\alpha/2}$	11≤n≤20	$t_{n-1,1-\alpha/2}$	21≤n≤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**	**+∞**	**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**		