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 <u>often very old and statistically weak</u>. **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. If you can, prefer using separate compiling with the Matsumoto source code on the one side (with the main), and on the other side your simulation code with the main program, calling the MT initialization and the drawing of pseudo-random numbers between 0 and 1.

1) <u>Compute π with the Monte Carlo method</u> (implementations for this lab are still <u>proposed</u> 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 drawn)`). In this question you don't have to make replicates for this question (independent experiments – ie drawing the "dice" many times).

2) <u>Computing independent experiments</u>: propose a loop which will call the previous function to compute '*n*' independent experiments (replicates). This is simply achieved if we do not reinitialize the pseudo-random number generator between two experiments (replicates). The first thing to achieve, is to keep the proper Makoto initialization, the run the loop over the PI simulation function and store all the results of *n* experiments (replicates) in an array (of double). Then propose a final result which is the mean of all the 'estimated PIs' (arithmetic mean).

3) <u>Computing of confidence intervals around the mean</u>: 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 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 the main 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). However, it is the only method 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 (uniformity up to 623 dimensions for the Mersenne Twister).

Appendix: Computing confidence Intervals

<u>Introduction :</u> 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 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 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.

We just have to compute $S^2(n)$, and to use it in the following formula to obtain the confidence radius at the 1-α level :

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

We find the value of $t_{n-1,1-\alpha/2}$ in the table 1 below. It 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)

The table 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 (number of independent experiments).

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-α/2}	11≤n≤20	t_{n-1,1-α/2}	21≤n≤30	t_{n-1,1-α/2}	n>30	t_{n-1,1-α/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		