M2 Phys. of Complex Systems MSA
TD 2014-2015 of David LACOSTE

Numerical Simulation in Statistical Physics
TD n° 2

On sampling and Monte Carlo simulations

Some questions can be better understood by looking at and running small Python programs, which can be found at `http://www.pct.espci.fr/~david/teach.html`. With Python 2.6 or 2.7 installed, these codes can be run simply with the command: python code.py

1 The Box-Muller method

To generate numerically random gaussian variables, a popular algorithm is that of Box-Muller which is based on the following principle: Let u, v be two random independent variables, uniformly distributed on $[0, 1]$, then the random variables x et y defined by

$$x = \sqrt{-2\ln(u)}\cos(2\pi v),$$
$$y = \sqrt{-2\ln(u)}\sin(2\pi v),$$

are random gaussian variables of zero mean and of unit variance.

▷ **Q. 1-1** Prove this result analytically.

▷ **Q. 1-2** Test the implementation with the program box-muller.py.

2 Direct sampling versus Markov sampling

In this section, we analyze some practical aspects in the numerical implementation of Direct vs. Markov sampling using an example borrowed from the reference: Algorithms and computations, by W. Krauth.

Consider a square and inscribe a circle of radius $1/2$ in it. The area of the circle is then $\pi/4$. If we draw the construction on the floor, then we can throm pebbles into the square and if we distribute the peebles randomly the probability of being within the circle is $\pi/4$. We call this algorithm direct sampling.

▷ **Q. 2-1** The algorithm can be generalized to evaluate any integral of arbitrary dimension N. It is unfortunately not a very efficient algorithm for large N, why ?

In order to improve the precision in the position of the pebbles, we enlarge the previous system, by considering a very large circle in a square stadium. This time we generate trials by taking a stone and throwing it randomly, we then move to the position of the stone and repeat the process. We call this Markov sampling and we propose to introduce a set of rules, so that we converge again with the correct value of the integral. We want to generate a dynamic process that has as its average density a constant. We call the density at point r by $\Pi(r)$.

▷ **Q. 2-2** Write an equation for $\Pi(a)$ in terms of transition probabilities to go from point a to point b, T_{ab} and for staying in a, namely T_{aa}.

▷ **Q. 2-3** By introducing another condition between T_{ab} and T_{aa}, obtain an equation of balance of probability. Explain the difference between this condition and the famous detailed balance condition. In the following, we construct an algorithm which satisfies the detailed balance condition for a constant density.

▷ **Q. 2-4** Explain how to proceed to satisfy this condition both in the center of the stadium but also in its corners. What is the typical value of T_{aa} in both cases ?

▷ **Q. 2-5** In order to test the efficiency of the algorithm, one introduces the notion of acceptance ratio, which counts how many moves are accepted with respect to the total attempted moves. How do you adapt this definition to the present case ? Let us call δ the range by which the peebles are thrown at each trial. What is the expected dependance of the acceptance as function of δ ? This curve is obtained in the program acceptance ratio.py.

▷ **Q. 2-6** The program error evaluation.py also calculates the error in the estimate of π from the algorithm as function of δ. Compare this curve to the previous one, and comment in particular on the behavior at small and large δ.

3 A little more on detailed balance

▷ **Q. 3-1** Prove that if the detailed balance condition is satisfied, and a given state a is reached by two or more different paths, the resulting evaluated probabilities will be the same.

▷ **Q. 3-2** Further show that if the detailed balance condition holds, the probability of any path is equal to the probability of the time-reversed path, where each state is visited in the opposite order as in the direct path.