Exercises

Romualdo Pastor
Departament de Fisica
Modul B4, 2a planta, Campus Nord
email: romualdo.pastor@upc.edu

Monte Carlo integration

Monte Carlo integration

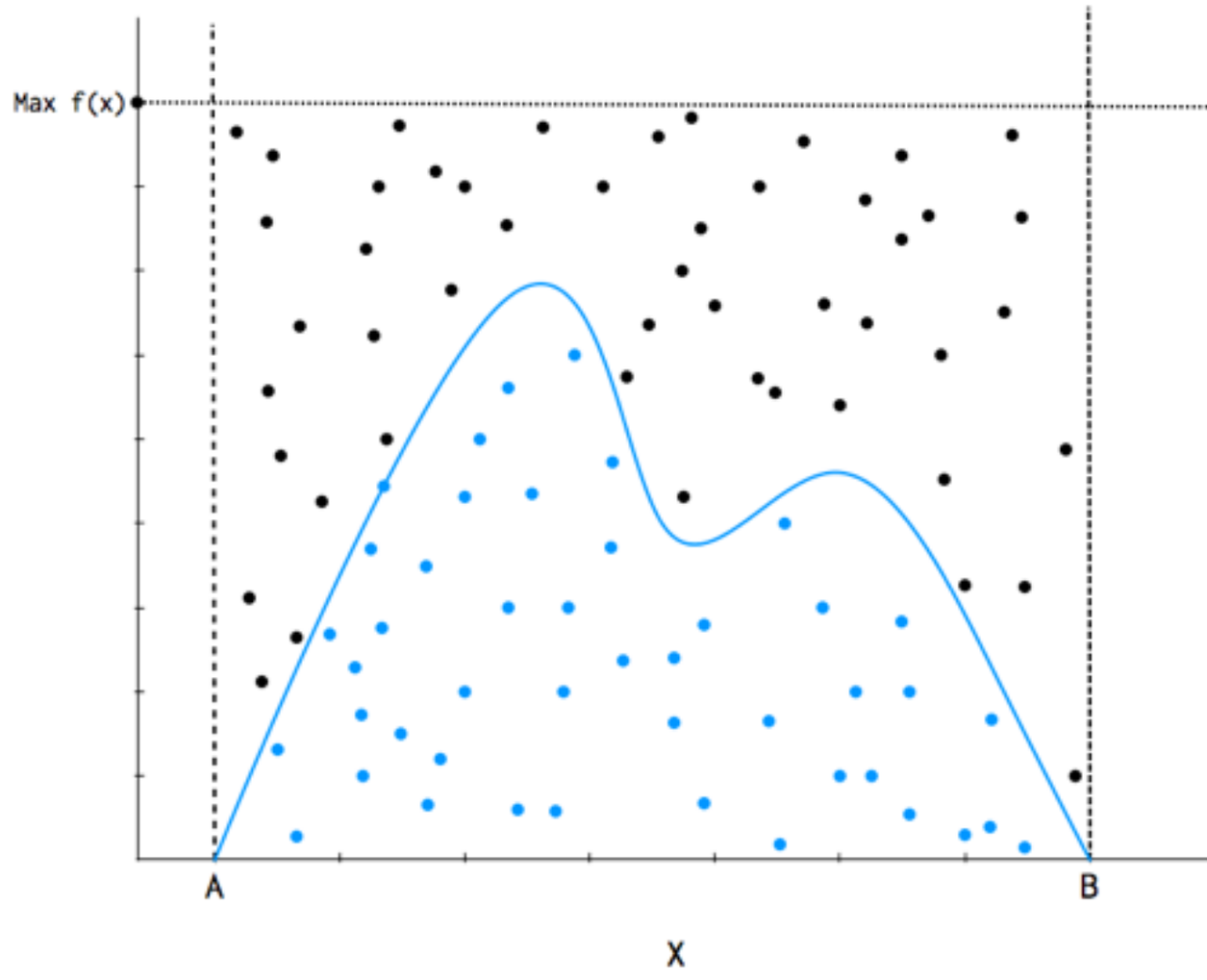
We can perform approximate integrals using random numbers.

The idea is that the integral of a function is equal to the area under it

We can compute this area approximately by generating points at random in the box [A, B] x [0, max f(x)]

If we generate N total points and n points fall below the function curve (blue points), the area under the function (the integral) is

$$\int_A^B f(x)dx = (B-A)[\max f(x)]\frac{n}{N}$$



To do:

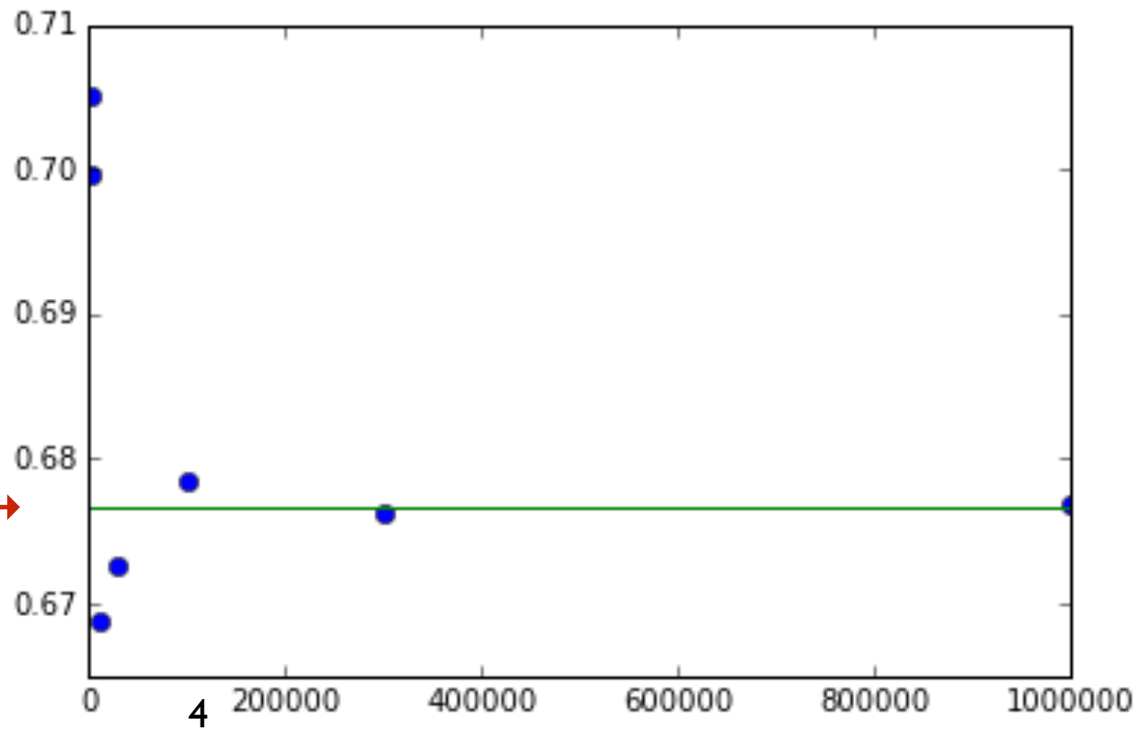
Implement a Monte Carlo integration in Python in a generic way, using modules

- ➡ One module for the function that is going to be integrated
- ➡ One module for a function that performs the integration and a function that determines the maximum and minimum
- ➡ Use the random number generator from the built-in module `random`

Check the code with the integral $\int_0^1 \frac{x}{\cos(x)} dx$

You should obtain:

Exact value: 0.6765535 →



Random walk in 1 dimension

Random walk in one dimension

We consider a drunkard that walks by performing one step to the right with probability p and to the left with probability q = 1-p

We are interested in knowing what will be his position after n steps

Of course, it will be described by a probability distribution, P(x, n), of being at position x after n steps

Mathematically, we can compute

$$\langle x \rangle_n = (p-q)n, \qquad \mathrm{Var}(x)_n = \langle x^2 \rangle_n - \langle x \rangle_n^2 = 4pqn$$

We want to check these results by mean of numerical simulations

To do:

Implement a random walk in one dimension in Python

- ➡ Make general functions to compute the position of the walker up to a total number of steps N, for general p and q
- ➡ From these data, compute $\langle x \rangle_n$ and $Var(x)_n$
- ➡ Check the validity of the previous results plotting this quantities as a function of n, for some values of p and q
- ➡ Make a first version in pure python, and then another one using bumpy
  - ❖ Try to exploit at the maximum the capabilities of numpy…