Università degli studi di Trieste
Corso di laurea magistrale in Fisica della materia

Laboratorio di Fisica Computazionale

Unit V
Numerical integration in 1D

Davide Bernocco
Matricola s285296

Programming language used:
Python

Contents

1 Equispaced points: comparison between "trapezoidal" and "Simpson deterministic methods

If we were asked to evaluate the integral of a certain function over a specific interval, the first effort usually consist in trying to solve the problem in an analytical way. But what if this task turned out to be extremely difficult or even worse impossible? We would like anyway to give an answer as most accurate as possible. This is where numerical integration comes into play.

Beside the algorithms themselves, in what will follow in this report, the attention will be posed on the behaviour of the numerical error. In particular, starting from examples with known exact solutions, we would like to find a reasonable way to estimate the error we can associate to our outcomes.

We are firstly going to show in a qualitative way the difference between two easy deterministic methods used in 1D numerical integration: the *trapezoidal* one and the *Simpson* one. In particular, here the error we are referring to is the *(absolute) actual difference* $\Delta_n = |F_n - I|$ with I exact result and F_n numerical estimate.

According to well known theoretical results, calling n the fixed number of discretization intervals, we expect the actual errors to follow two different trends for the two different methods:

$$\text{Trapezoidal method: } \Delta_n \approx 1/n^2$$
$$\text{Trapezoidal method: } \Delta_n \approx 1/n^4$$

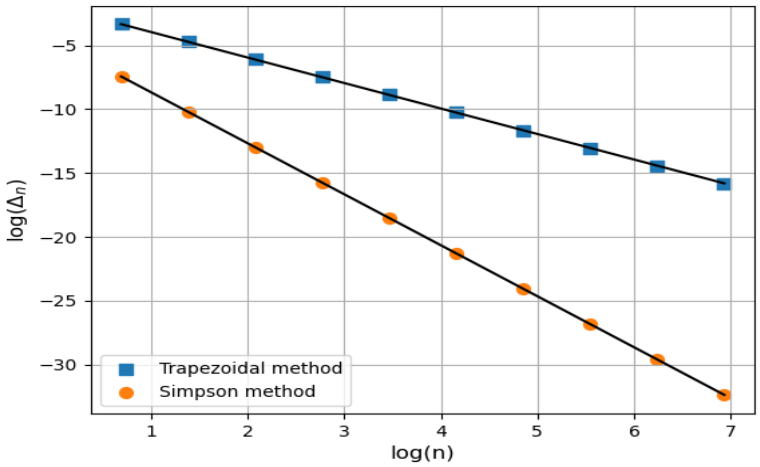

Fig 1.1 Actual errors for trapezoidal and Simpson rule, log-log scale

The linear interpolations give the accordance with the expected values:

$$\text{Slope - trapezoidal method} = -1.9996 \pm 0.0002$$
$$\text{Slope - Simpson method} = -4.000 \pm 0.002$$

2 MC methods: "generic sample mean" and "importance sampling"

We leave behind the deterministic methods and start studying the more versatile *Monte Carlo statistical methods*. The focus will be here on a method based upon a fundamental result in calculus[1], and its natural evolution for less smooth functions.

In this specific example, we are asked to evaluate the integral: $I = \int_0^1 e^{-x^2} dx$

- with MC *sample mean* algorithm that uses a set of (pseudo) random points uniformly distributed in $[0, 1[$
- with MC *importance sampling* algorithm based upon the generation of a set of points distributed according to a specific probability function that suits the best the integrated function. Here we will use $p(x) = Ae^{-x}$

It has been demonstrated that typical MC algorithm techniques display error trends that go as $1/\sqrt{n}$. Now, an effective way we have at disposal to give a reasonable estimate of the error, first requires us to calculate the numerical variance from the data $\sigma_n^2 = \langle f^2 \rangle - \langle f \rangle^2$ and then divide it by $\sqrt{n}$. Graphically we find something that confirms our expectations:

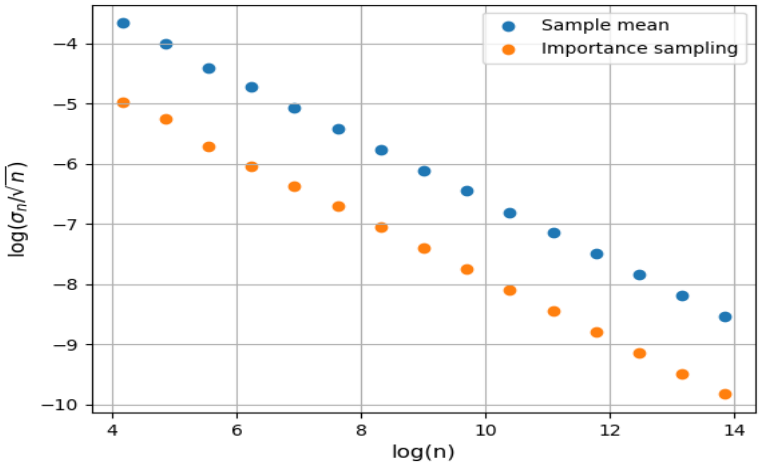

Fig 2.1 Error estimate $\sigma_n/\sqrt{n}$ associated to the sample mean and importance sampling algorithms, log-log scale

As for the speed efficiency, it is worth noting that the overall required time for the importance sampling algorithm to run is greater. However, if we looked at the time necessary for the sample mean algorithm to obtain the same error magnitude as the second algorithm, it would take more time. This clearly emerge when looking at the intercepts of the two data series showed above.

[1] $\int_a^b f(x)dx = (b-a)\langle f \rangle$ where $\langle f \rangle$ is statistically evaluated with MC techniques

3 MC method: "acceptance - rejection"

We are going to briefly see what happens with the famous algorithm of *acceptance - rejection*. The integral we would like to evaluate is $I = \int_0^1 \sqrt{1-x^2}dx$.
The numerical estimation of I is provided by the ratio N_{in}/N_{tot}[2] and, since the analytical result is known, we will have a look at what happens to the actual error. We expect here as well a typical $\approx 1/\sqrt{n}$ behaviour.
However, because of the poor efficiency of this method, to get reliable information out of the large stochastic fluctuations we performed 10^4 runs per each n. The outcome is in accordance with the expectations:

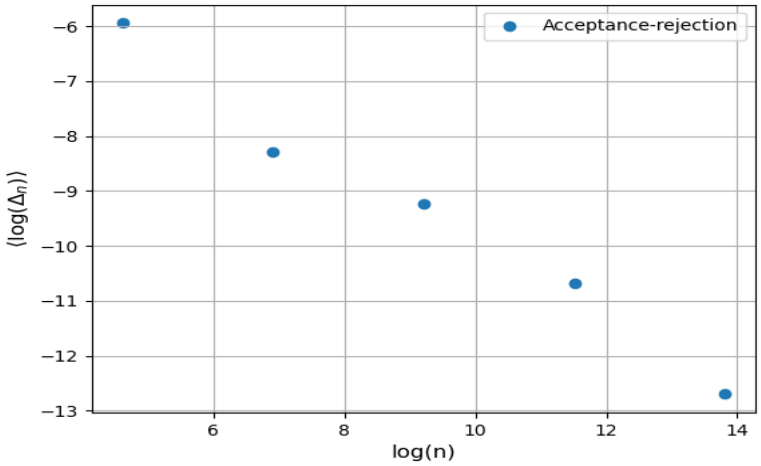

Fig 3.1 Actual error for the acceptance - rejection method, log-log scale (averages over $N = 10^4$ runs)

[2] where N_{in} is the number of (pseudo) randomly generated points that fall under the curve

4 MC sample mean: error analysis using "average of the averages and "block average"

We are going to analyze two different ways of improving the error estimate through *variance reduction*. What we will find out here will set the bases for the error estimation procedure in all the application of numerical stochastic algorithms!

The integral we will deal with is the same as §3, where the knowledge of the exact result will help us in comparing the results case by case.
We start thinking about what has already emerged in the first chapters: it seems a good actual error estimator cannot be σ_n, which stays constant as n grows, contrary to $\sigma_n/\sqrt{n}$. This can easily grasped looking at the following graph:

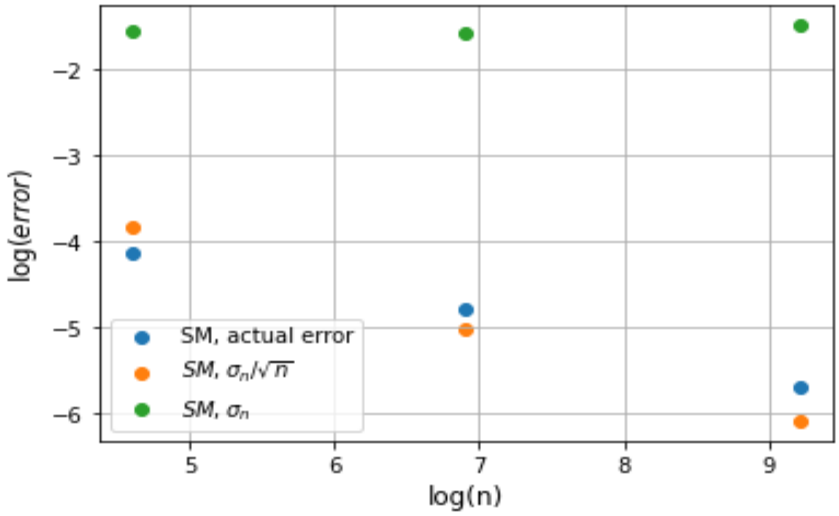

Fig 4.1 Actual error, standard deviation and $\sigma_n/\sqrt{n}$ for the sample mean method, log-log scale

4.1 Average of the averages

The first algorithm we can implement to improve the error estimate is what we could call a "brute force" one. Once fixed the sample size of our data, we just calculate the averages for a certain number of samples of the same size and then the average of the averages and its variance:

$$\sigma_n^2 = \langle M^2 \rangle - \langle M \rangle^2$$

From a simulation with $m = 10$ samples of length 10^4 we obtain (remember $\Delta_{n=10^4} = 0.0073$) :

$$\sigma_m = 0.0069 \approx \sigma_n/\sqrt{n}$$

4.2 Block average

Not only this second algorithm is for sure more clever, but it will proof the more efficient in estimating the error. Instead of taking the averages over several run of length n, we now divide the single starting sample of size n into s subsets and consider the averages m_s within each individual subset and the variance over them:

$$\sigma_s^2 = \langle m_s^2 \rangle - \langle m_s \rangle^2$$

From a simulation with $s = 10$ subsets built out of a single run of length 10^4 we obtain (remember $\Delta_{n=10^4} = 0.0073$) :

$$\sigma_s/\sqrt{s} = 0.0075 \approx \sigma_m \approx \sigma_n/\sqrt{n}$$

This confirms what has been mathematically demonstrated in literature, that is these three quantities tends to be equivalent for uncorrelated data[3]. Furthermore, because of the intrinsic way it is constructed, the *block average* method requires *less computational cost* and so it will be preferred to the average of averages method whenever we will need to determine and improve error estimate from stochastic data, reducing the impact of random fluctuations.

[3]When talking about *correlated* data, what the central limit would tell us about the error on the average - i.e. $\sigma_m \approx \sigma_n/\sqrt{n}$ - is no more valid! An easy way of handling the problem is passing through the block average method which provide a good estimate of σ_m as $\approx \sigma_s/\sqrt{s}$