Report file - Problem Set #2

Matteo Dell'Acqua
GitHub: MatteoDellAcqua6121

September 24, 2024

Abstract

This is the report for the problem set #3. Since the problem set is composed of four exercises, we divide the report into four sections, one for each problem. The scripts (labelled as ps_3.'problem number') and the raw file of the images are in this directory.

1 Problem 1

1.1 Formulation of the problem

We are asked to compute the product of two (large) square matrices, either naively using three nested loops, or the `dot` function of NumPy and compare their speed and time complexity.

1.2 Computational methods

Explicitly, the naive function is written as:

```
for i in np.arange(N1):
        for j in np.arange(N3):
            for k in np.arange(N2):
                C[i][k]+=A[i][k]*B[k][j]
```

where N is the size of the matrices. Its complexity is expected to scale as $O(N^3)$.

In order to verify the previous claim and to compare the naive method with the `dot` function of NumPy, we test both methods with increasingly large random matrices (with entries between -1 and 1) and compute the speed using the `timeit` module.

1.3 Results

While the `dot` function is significantly faster (as one can see from fig. 1, the time elapsed differs by about two orders of magnitude), it is not clear what's its complexity. Due to their differences in speed, we chose to run the naive method for matrices of size ≤ 30 and the `dot` function up to size 300 matrices.

We can better visualize the complexity in a log-log scale (since a polynomial function $f(x) = x^\alpha$ looks like a linear function $g(x) = \alpha x$ in these coordinates). We **visually** the following complexities (see fig. 2):

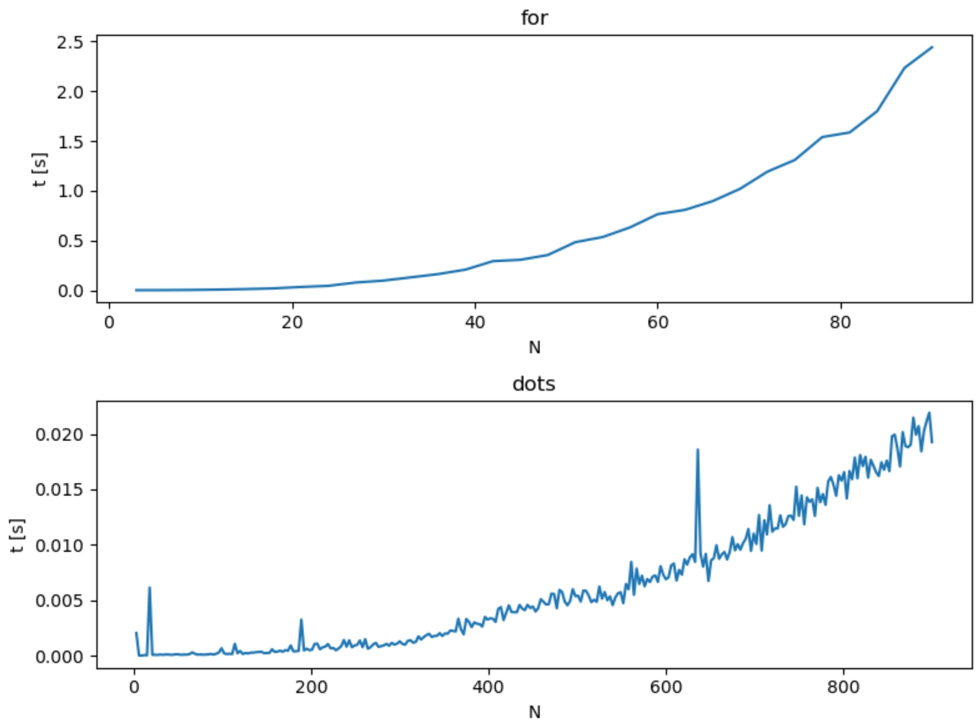

Figure 1: Time elapsed during multiplication of random square matrices of size N for the naive method and the `dot` function.

$$\text{naive} \sim \mathcal{O}(N^3), \qquad \texttt{dot} \sim \mathcal{O}(N^{2.8}), \tag{1}$$

which, for the naive function, agree with our analytical expectation.

2 Problem 2

2.1 Formulation of the problem

We are asked to simulate the chain of decays shown in fig. 3 for 20000 seconds, with initial conditions of 10000 atoms of ^{213}Bi and no other species.

2.2 Computational methods

We discretize time in steps of 1 second and for every atom at every step, we decide whether it decays or not (and for the ^{213}Bi atoms, in which channel it decays into) with a `np.random` extraction with cut-off given by:

$$p = 1 - 2^{-\delta t/\tau}, \tag{2}$$

with $\delta t = 1s$ the size of the time step, and τ the half-life of the decaying specie.

In order to avoid multiple simultaneous decays we run the chain from the bottom up.

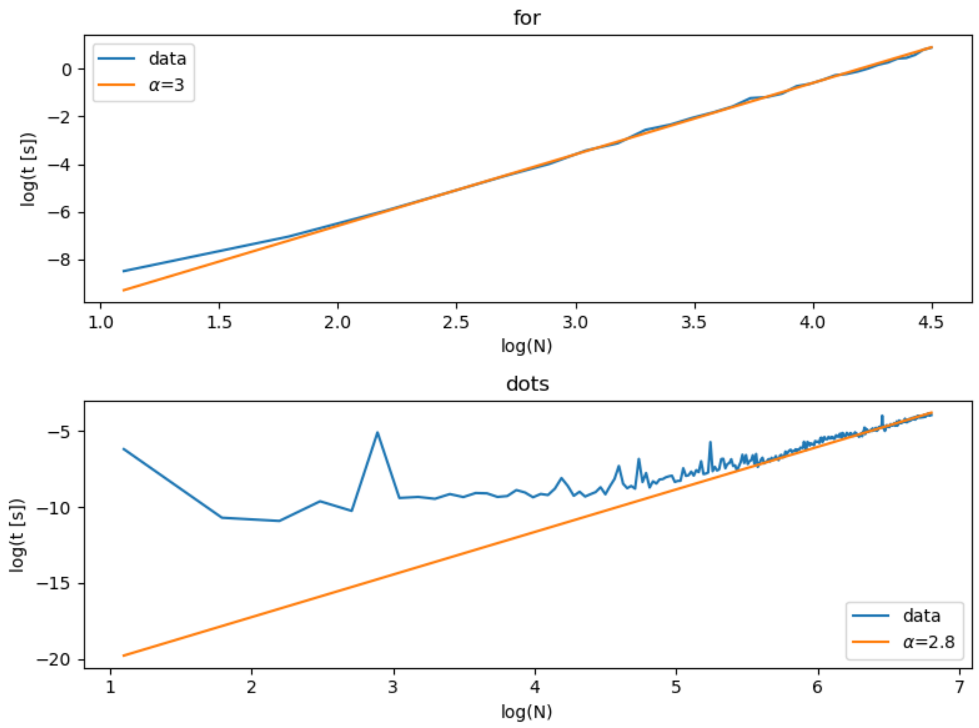

Figure 2: Time elapsed during multiplication of random square matrices of size N for the naive method and the `dot` function plotted in logarithmic scale and compared with linear functions of slope α.

2.3 Results

We report the results of our simulations (plotted using `matplotlip.pyplot`) in fig. 3.

3 Problem 3

3.1 Formulation of the problem

We are asked to simulate a simpler decay ($^{208}Tl \rightarrow {}^{208}Pb$ with half-life $\tau = 3.053$minutes) with a refined method, explained in section 3.2.

The goal of this problem is to make acquaintance with non-uniform random number generators.

Given to variables x, r such that one can be rewritten in terms of the other $x = x(r)$, their pdfs satisfy the following equations:

$$|f_r(r)\mathrm{d}r| = |f_x(x)\mathrm{d}x| \quad \Leftrightarrow \quad \left|\frac{\mathrm{d}x}{\mathrm{d}r}\right| = \frac{1}{f_x(x)}. \tag{3}$$

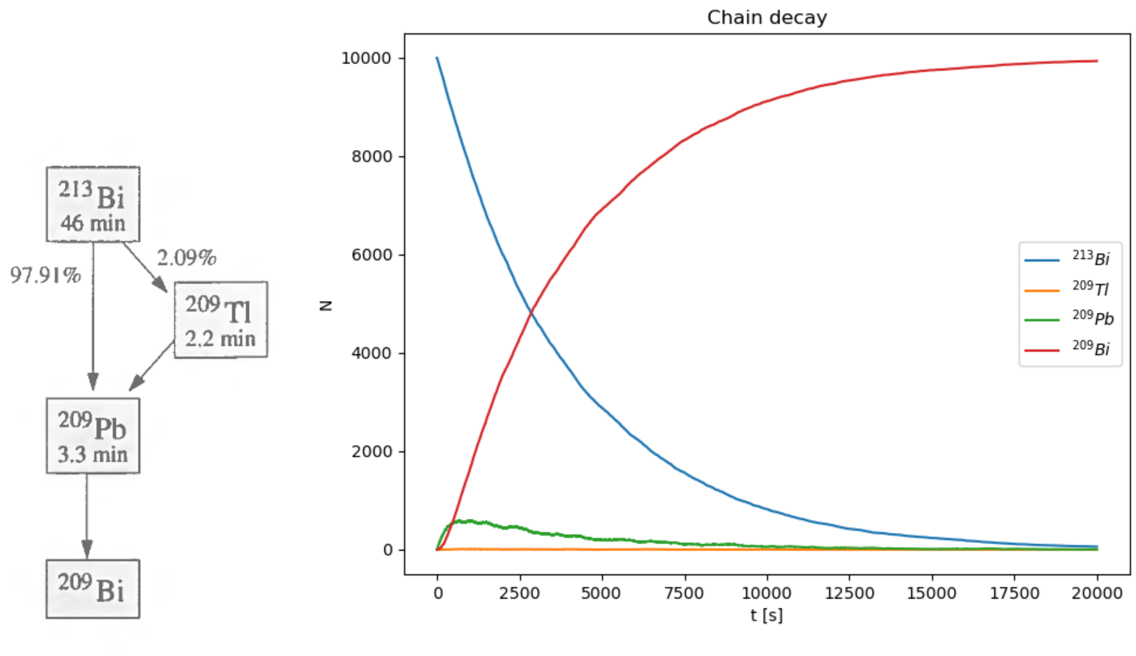

Figure 3: On the left the chain decay we are going to simulate. On the right, the results of the simulations: we started with 10000 atoms of ^{213}Bi and we run the system for 20000 seconds, with incremental steps of 1 second. We plot the population size for each species over time.

3.2 Computational methods

The decay process is governed by the pdf:

$$P(t)\mathrm{d}t = 2^{-t/\tau}\frac{\ln 2}{\tau}\mathrm{d}t. \tag{4}$$

that computes the probability of an atom to decay between t and $t + \mathrm{d}t$.

Thus, in order to understand the decay of $N = 1000$ atoms of ^{208}Tl, we extract N times of decay from the distribution 4. By solving the eq. (3) (with $t \leftrightarrow x$), this can be done by applying the transformation:

$$t = \frac{\tau}{\ln 2}\ln 1 - r \tag{5}$$

to a uniformly distributed variable r generated using the `random` function of NumPy. We store the extracted times of decay into an array `decay`

In order to plot the population size over time, we first create an array containing the time coordinates of the plotted points by dividing into $M = 10000$ intervals the segment $t = [0, \texttt{decay.max}]s$, then (by first sorting `decay` using the `sort` attribute of a `np.array`) we compute the number of decays happening between two such times and finally we update the population sizes `Tl` and `Pb` taking the decays into account:

```
M=10000
T=0
Tl[0]=1000

for i in np.arange(M):
if i>0:
    Tl[i]=Tl[i-1]
    Pb[i]=Pb[i-1]
dec=0
while decays[T]<(i+1)*decays.max/M:
    dec+=1
    T+=1
Tl[i]-=dec
Pb[i]+=dec
```

3.3 Results

We report the results of our simulations (plotted using `matplotlip.pyplot`) in fig. 4.

4 Problem 4

4.1 Formulation of the problem

We are required to experimentally verify the validity of the central limit theorem (CLT)[1]:

Theorem 1 *Suppose $x_1.x_2, \ldots, x_N$ is a sequence of i.i.d. random variables with $\mathbf{E}[x_i] = \mu$ and $Var[x_i] = \sigma^2 < \infty$. Then, as N approaches infinity, the random variables $\sqrt{N}(y - \mu)$ converge in distribution to a normal one $\mathcal{N}(0, \sigma^2)$,*

by looking at the case where x is distributed as $exp(-x)$.

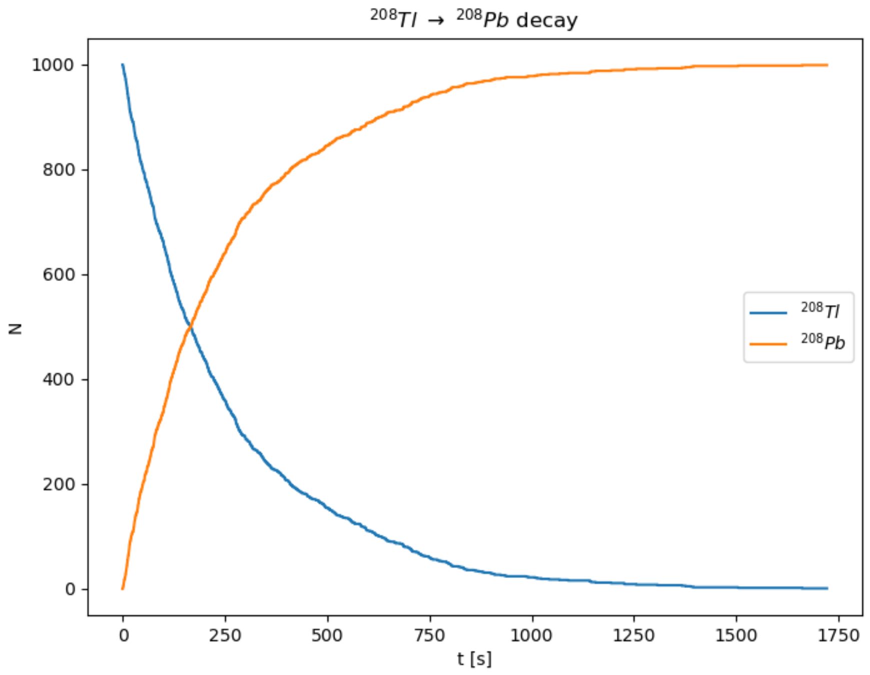

Figure 4: Population over time for the two species involved in the $^{208}Tl \rightarrow {}^{208}Pb$ decay, simulated using a non-uniform random number generator.

4.2 Computational methods

In the following, we are going to denote with $M(= 10000)$ the number of extractions of the random variable y (and $N(= 100)$ as above).

First, we extract y M times (by creating an array of iid x variables and then using the `sum` attribute of a NumPy array to efficiently take their average):

```
for i in np.arange(M):
    x=np.random.standard_exponential(N)
    y[i]=y_temp.sum()/N
```

We visually prove the CLT by plotting a histogram of the resulting $\sqrt{N}(y-1)$s against a (properly normalized) standard Gaussian (see fig. 5).

We then repeat the same sampling for different values of N and compute the sample unbiased mean, variance[2], skewness[3] and kurtosis[4]:

$$\hat{\mu} = \frac{1}{M}\sum_{i=1}^{M} y_i, \tag{6}$$

$$\hat{\sigma}^2 = \frac{1}{M-1}\sum_{i=1}^{M}(y_i-\hat{\mu})^2, \tag{7}$$

$$G_1 = \frac{\frac{1}{M}\sum_{i=1}^{M}(y_i-\hat{\mu})^3}{(\hat{\sigma}^2)^{3/2}}, \tag{8}$$

$$G_2 = \frac{(M+1)M}{(M-1)(M-2)(M-3)}\frac{\frac{1}{M}\sum_{i=1}^{M}(y_i-\hat{\mu})^4}{(\hat{\sigma}^2)^2} - 3\frac{(M-1)^2}{(M-2)(M-3)}. \tag{9}$$

4.3 Results

We report the results of our simulations (plotted using `matplotlip.pyplot`) in figs. 5 and 6.

For $N = 1$ the sample y is distributed exactly as x (i.e. a standard exponential), whose mean, variance, skewness and kurtosis are given by[5]:

$$\mu = 1, \quad \sigma^2 = 1, \quad g_1 = 2, \quad g_2 = 6. \tag{10}$$

From the plot 6 one chan see that the sample quantities quickly approach those of the Gaussian predicted by the CLT.

Visually, we estimate that the skewness and kurtosis of y have reached about 1% of their value for $N = 1$ at:

$$N_{\text{skewness}} = 300, \quad N_{\text{kurtosis}} = 30. \tag{11}$$

References

[1] *Wikipedia: Central Limit Theorem.* `https://en.wikipedia.org/wiki/Central_limit_theorem`. Accessed: 2024-09-24.

[2] *Wikipedia: variance.* `https://en.wikipedia.org/wiki/Variance`. Accessed: 2024-09-24.

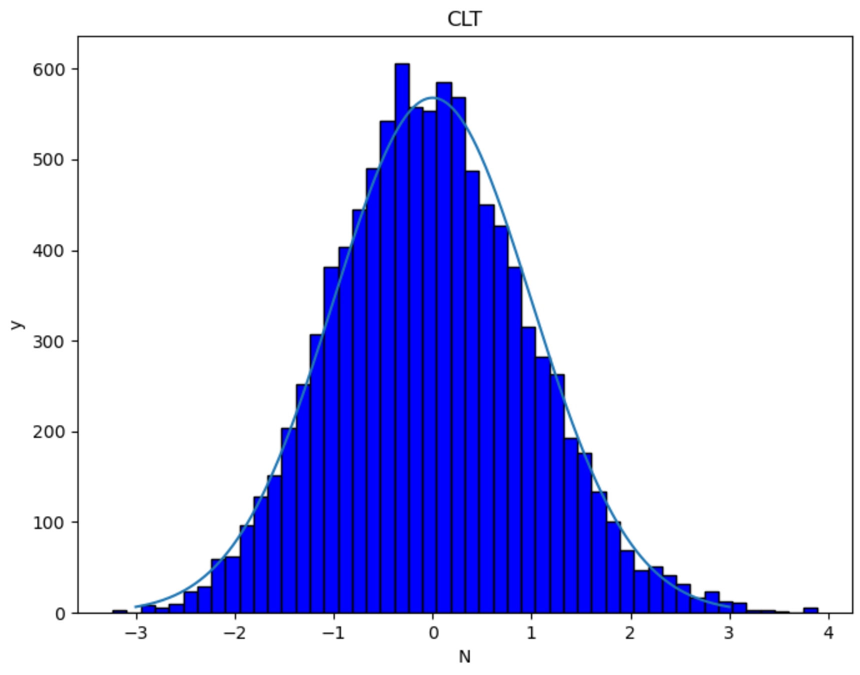

Figure 5: Comparison between histogram (of 10 bins) resulting from the extraction of $M = 10000$ copies of a random variable obtained as the mean of $N = 100$ iid (exponentially distributed) variables and a (properly normalized) standard Gaussian.

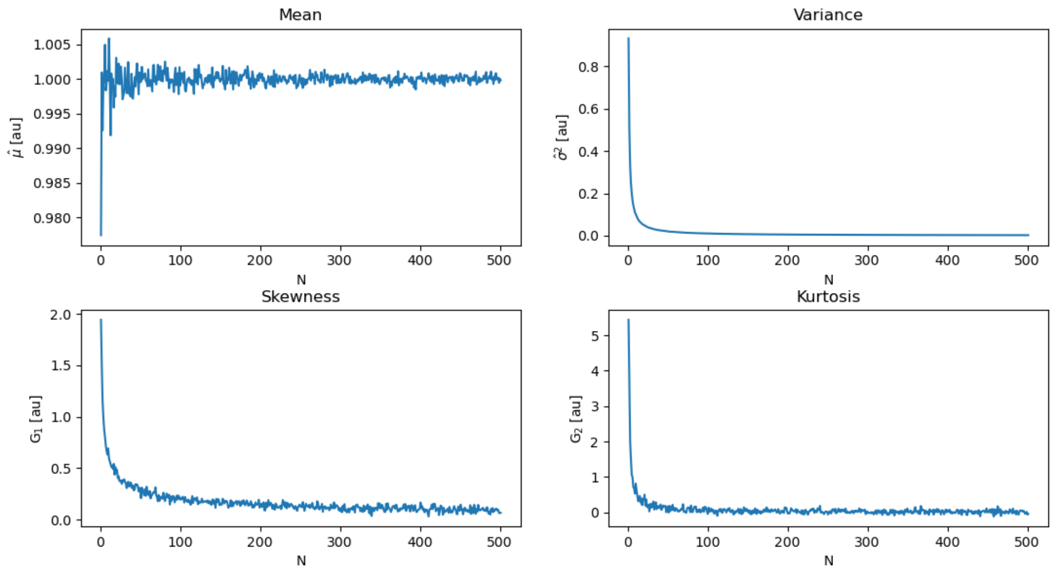

Figure 6: Dependence on N of (unbiased sample) mean, variance, skewness and kurtosis of the mean y of N iid (exponentially distributed) variables. Results were obtained from a sample size of $M = 10000$.

[3] *Wikipedia: skewness.* `https://en.wikipedia.org/wiki/Skewness`. Accessed: 2024-09-24.

[4] *Wikipedia: kurtosis.* `https://en.wikipedia.org/wiki/Kurtosis#Standard_unbiased_estimator`. Accessed: 2024-09-24.

[5] *Wikipedia: Exponential distribution.* `https://en.wikipedia.org/wiki/Exponential_distribution`. Accessed: 2024-09-24.