Second Assignment for Computational Physics

Xinyu Liu

September 23, 2024

Contents

1 Problem 1: Computation time for N*N matrice mutiplication

The associated code of this problem is ps3-1.py.

We do the matrice mutiplication of a N by N matrice. Since we expect that the computation cost is growing with N in a non-linear way, i.e.:

$$t = \alpha * N^{\beta} \tag{1}$$

So, we can test this relationship by doing linear regression against $\ln t$ and lnN, since:

$$\ln t = \ln \alpha + \beta * \ln N \tag{2}$$

In order to reduce the systematic error introduced by computer system, we first try to do linear regression with equally spaced $\ln N$ with a broad range of N. The result goes as follows:

Table 1: The relationship between calculation time(lnt) and matrice size(lnN) with explicit for loop

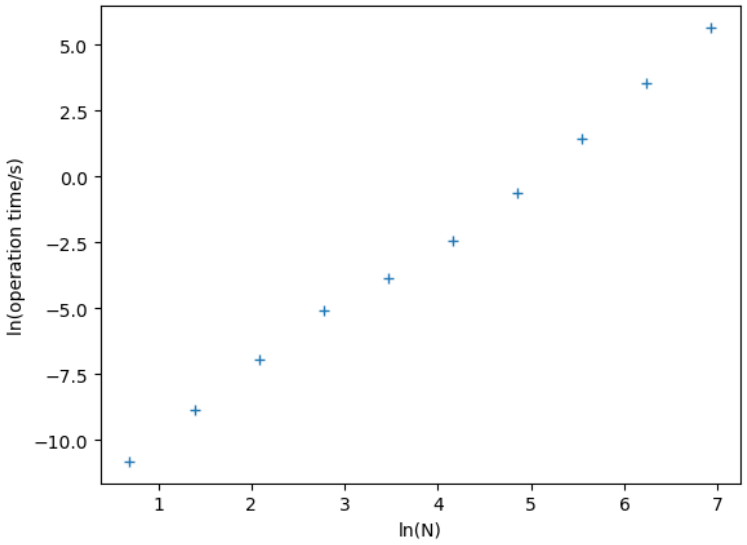
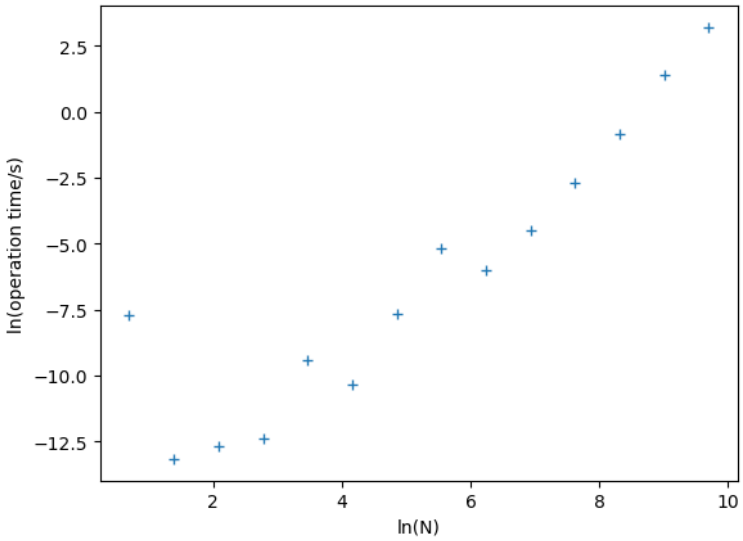


Table 2: The relationship between calculation time(lnt) and matrice size(lnN) with np.dot method



We can clearly see that, for smaller size matrice calculation, since the computation is so fast, the time cost depends more on stochastic computer system perturbation and for some random point cost more time than expected. In order to better trace the time dependence of matrice calculation itself, we decide to set N such that the calculation time is always to the order of 1 - 10 seconds. After choosing better N size for each case, we can do similar plotting and linear regression which goes as follows:

Table 3: The relationship between calculation time(lnt) and matrice size(lnN) with np.dot method

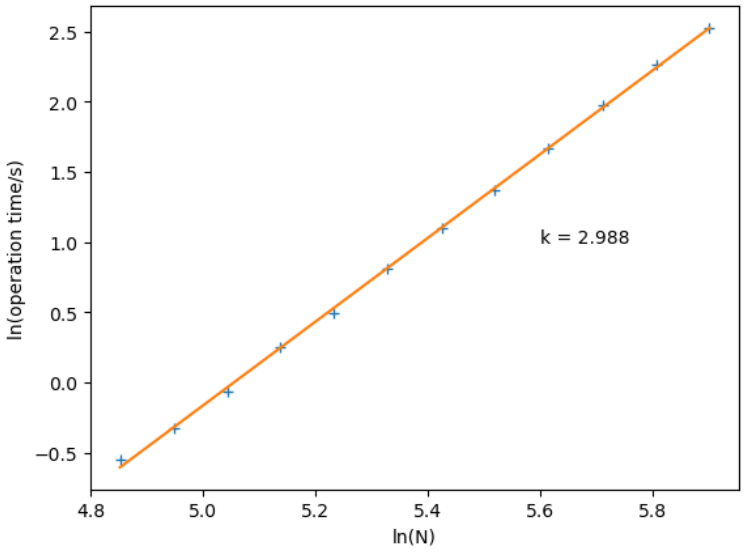
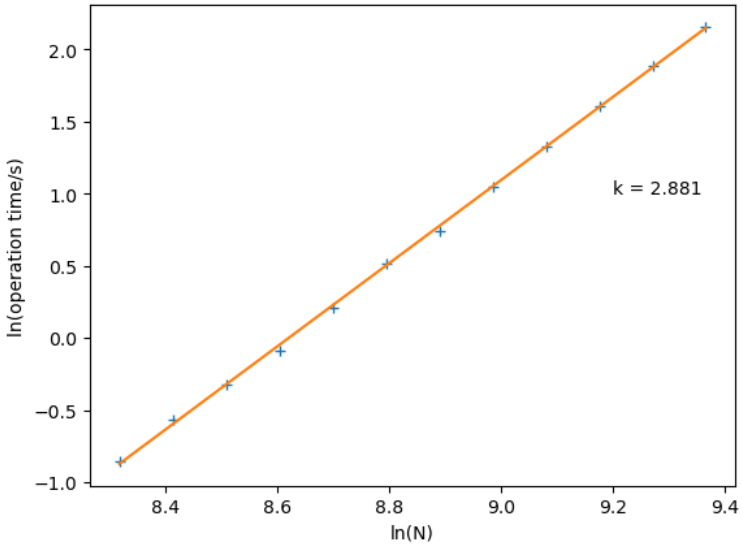


Table 4: The relationship between calculation time(lnt) and matrice size(lnN) with np.dot method



We see that after optimizing the choice of N, the linearity between $\ln N$ and $\ln t$ is clearer. The slope of either case is 2.988 and 2.881, which is close to the expected value of 3. This shows that, for a considerable N(where the calculation time is around the order of seconds), the calculation time approximately grows linearly with N^3.

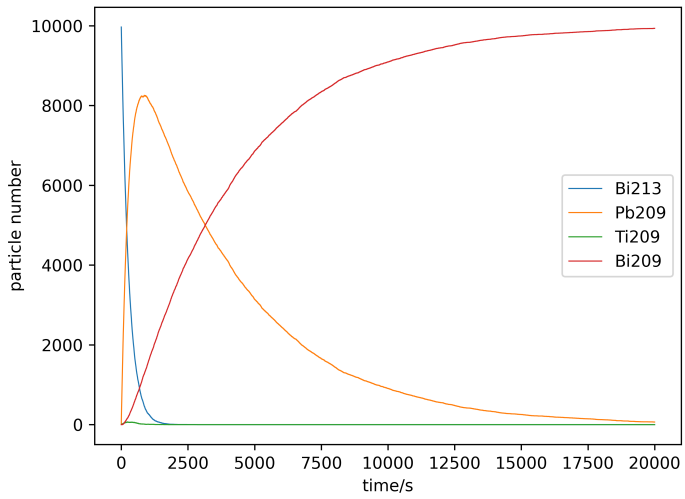
Also, it's evident that the implicit np.dot calculation is much faster than the explicit for loop calculation. The explicit for loop takes more than 200s to calculation the mutiplication of 1000*1000 matrice however for np.dot method it only takes 20s for a 16384*16384 matrice mutiplication. This is because the np.dot method is closer to hardware level and possibly also accelerated by GPU.

2 Problem 2: Exercise 10.2

The associated code of this problem is ps3-2.py.

We do the exercise by following the instruction. We use numpy to generate 10000 different random number and compare it with decay possibility at the same time to speed up. This can be seen in our code. The calculation result goes as follows:

Table 5: The number of different particles under the decay of ^{213}Bi in 20000 second



3 Problem 3: Exercise 10.4

The associated code of this problem is ps3-3.py.

We do the exercice by following the instruction and generate the variable with correct distribution function of the time a particle decays. In python numpy module we can first generate a random number with even distribution between 0-1(by numpy.random.random())method. We name this stochastic variable z. So the associated PDE for z is:

$$p(z) = 1 \tag{3}$$

What we want in our problem is the decay time of a particle t which have the half lifetime τ, whose PDE satisfy:

$$f(t) = \frac{\ln 2}{\tau} 2^{-t/\tau} \tag{4}$$

So the transfer function between t and z satisfy:

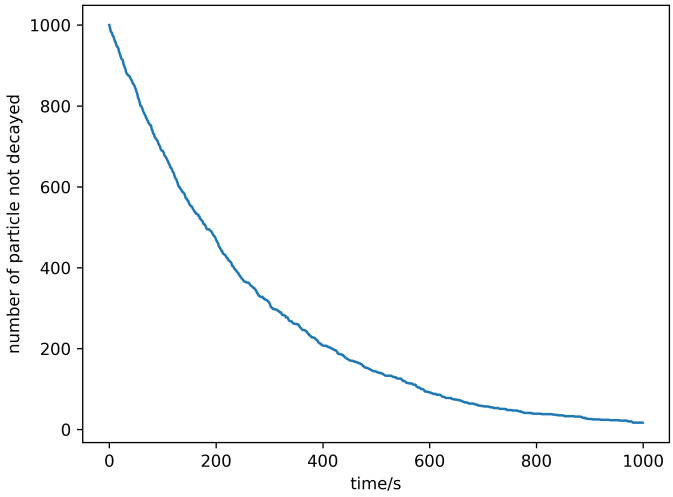
$$\int_0^{t(x)} \frac{\ln 2}{\tau} 2^{-t/\tau} dt = z \tag{5}$$

The transfer function is:

$$t(z) = -\tau \log_2(1 - z) \tag{6}$$

We can thus generate the decay time by this function and plot a diagram for the number of particle undecayed under certain time.

Table 6: the number of particle undecayed with time



4 Problem 4: Central limit theorem demonstration

The associated code of this problem is ps3-4.py.

4.1 Theoretical derivation

We have N independent variable x_i which each satisfy exponential distribution $p(x_i) = e^{-x_i}$. The variable y is defined as $y = \frac{1}{N}\sum_{i=1}^{N} x_i$. So that the mean value and the variation of y satisfies:

$$\bar{y} = \overline{\frac{1}{N}\sum_{i=1}^{N} x_i} = \frac{1}{N}\sum_{i=1}^{N} \bar{x_i} \tag{7}$$

Since $\bar{x_i} = \int_0^{+\infty} xe^{-x}dx = \Gamma[2] = 1$, where Γ represent Gamma function. The mean value of y satisfies:

$$\bar{y} = \frac{1}{N}\sum_{i=1}^{N} 1 = 1 \tag{8}$$

The variance of y satisfies:

$$\overline{(\Delta y)^2} = \overline{(\frac{1}{N}\sum_{i=1}^{N}\Delta x_i)^2} \tag{9}$$

Since different variable is independen and all variables satisfy same distribution function, the variance of y satisfies:

$$\overline{(\Delta y)^2} = \frac{1}{N^2}\overline{\sum_{i=1}^{N}(\Delta x_i)^2} = \frac{1}{N}\overline{(\Delta x_1)^2} \tag{10}$$

The variance of a single variable can be calculated by:

$$\overline{(\Delta x_1)^2} = \int_0^{+\infty}(x-1)^2 e^{-x}dx = \Gamma[3] - 2\Gamma[2] + \Gamma[1] = 2 - 2 + 1 = 1 \tag{11}$$

So that the variance of y is $\frac{1}{N}$.

4.2 Computational calculation

We first generate 10^5 different y based on $N = 1, 10, 100, 1000$. We draw a histogram to visualize the distribution of 10^5 different y.

Table 7: histogram of 100000 y with N = 1

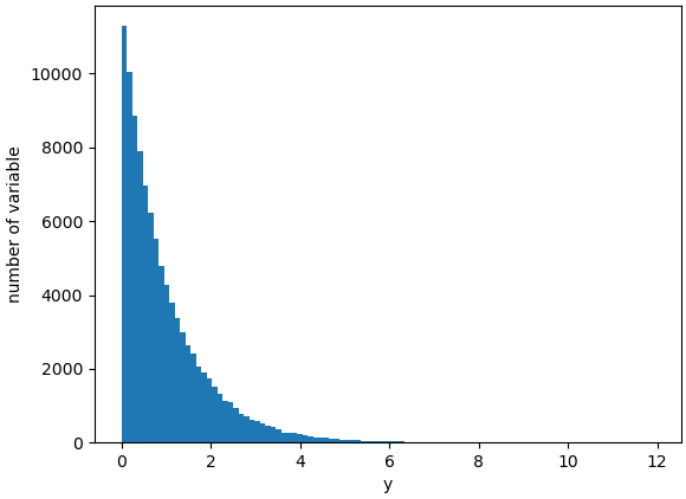
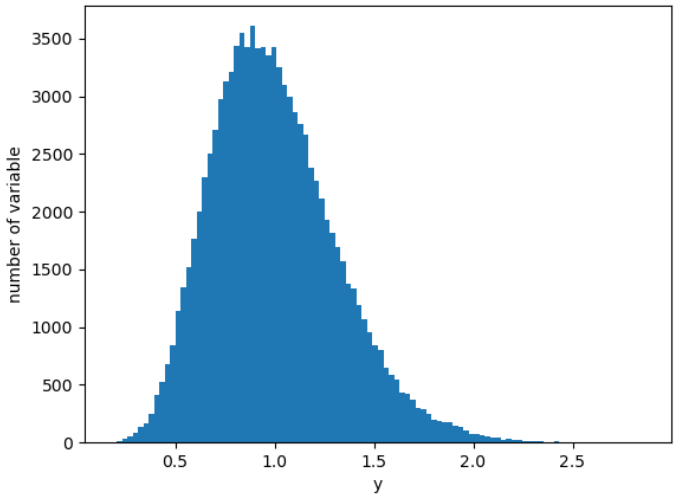


Table 8: histogram of 100000 y with N = 10



From these four plots78910, we can clearly see that the larger the N, the closer the distribution of y is to gaussian. This visualize intuitively how central limit works.

Table 9: histogram of 100000 y with N = 100

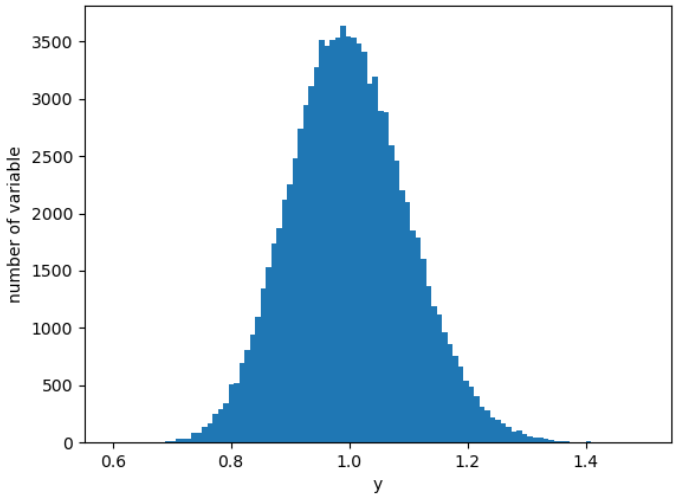
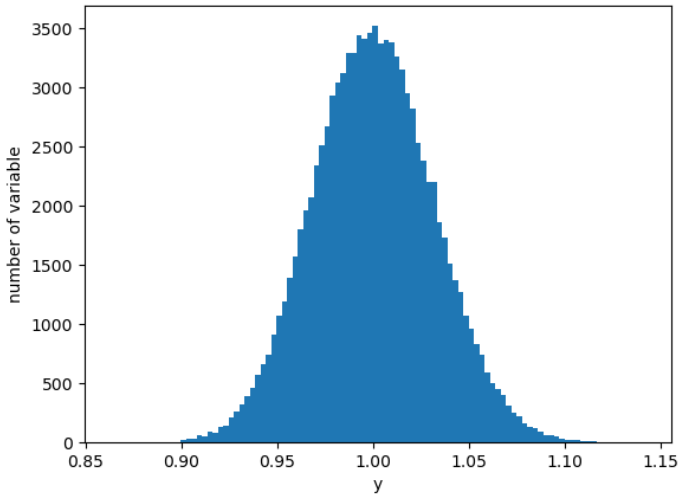
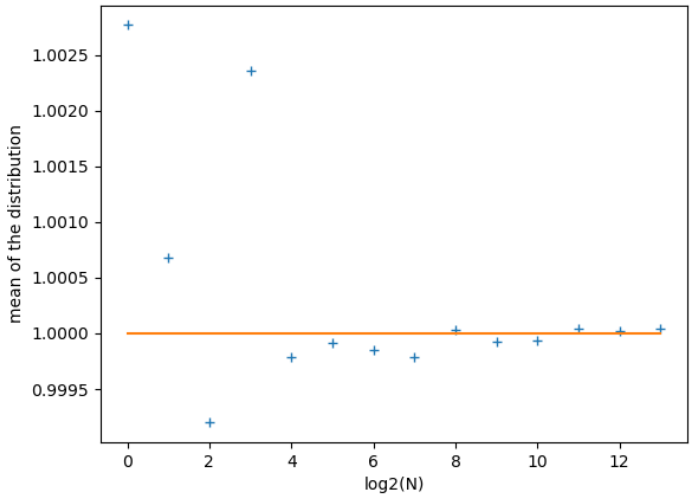


Table 10: histogram of 100000 y with N = 1000



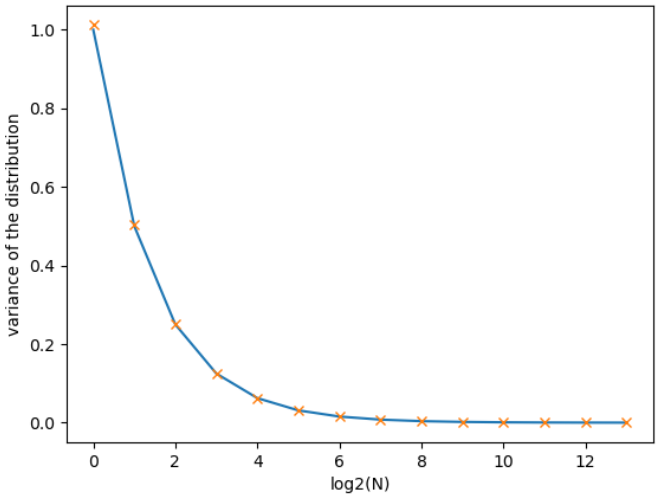
We will further try to plot how the mean, variance, skewness, and kurtosis of the distribution change with respected to N. Here's our result:

Table 11: mean of y w.r.t. N



Note that in this diagram the orange line represent the theoretical value of mean we've calculated. The result show that our theoretical calculation is correct. The mean is closer and closer to 1 for larger system(large N)

Table 12: variation of y w.r.t. N



Note that in this diagram the blue line represent the theoretical value of variation we've calculated. The result show that our theoretical calculation is correct.

Table 13: skewness of y w.r.t. N

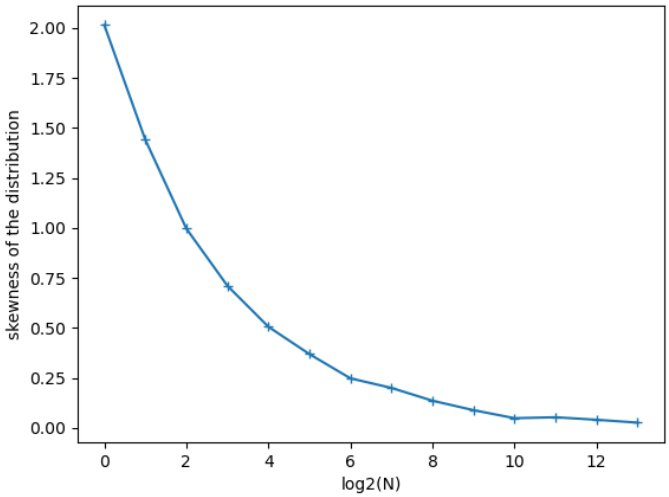
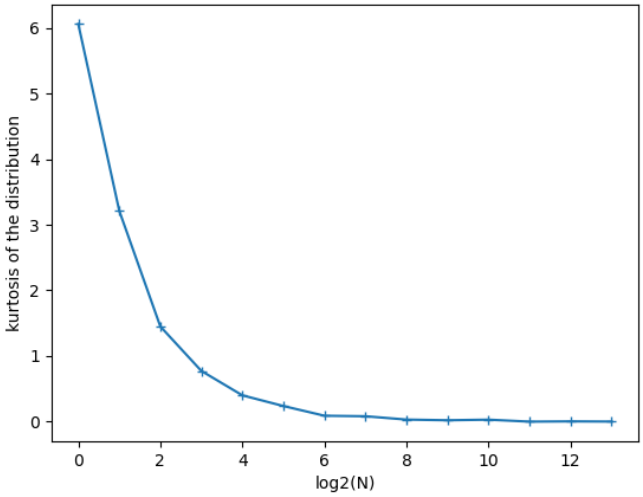


Table 14: kurtosis of y w.r.t. N



We found that in our calculation, when N equals 8192, the kurtosis of the distribution of y reach 0.01 time the kurtosis of the distribution when N

= 1. when N equals 65536, the skewness of the distribution of y reach 0.01 time the skewness of the distribution when N = 1.