PHYS-GA2000-PS3

Ahmet Koral Aykın

September 26, 2023

1 Introduction

In the first question, we are asked to find the derivative of a function numerically. To calculate the derivative, we use the following definition of it

$$\frac{df(x)}{dx} = \lim_{\delta x \to 0} \frac{f(x+\delta x) - f(x)}{\delta x} \tag{1}$$

On computer, it is actually impossible to take the limit as δx goes to zero. However, we can approximate the derivative by making δx as small as possible. It is of great importance to notice here that, if we make δx progressively smaller, the result must deviate from the true value at some point. The reason of it is that as the δx goes to zero, the values of $f(x+\delta x)$ and $f(x)$ come closer, which implies subtraction of two nearly equal values; resulting in fractional error.

In the second question, we empirically discover how the amount of time required to perform floating point number operations rises with the number of operations. One way to visualize this is to calculate the time required for multiplication of $N \times N$ matrices in which the number of operations are proportional to N^3. In theory, since each operation (multiplication and addition) takes essentially same time, the amount of time must also be proportional to N^3.

In the third and fourth questions, we are trying to solve the problem of decay of an atom utilizing random numbers. Decay of an atom is probabilistic in nature, which means that we can only know the probability that an atom decays in a time interval of length t but not the exact time it will decay. How we attack to this kind of problems which are probabilistic in nature is to utilize random numbers.

2 Methods

To perform a matrix multiplication of size $N \times N$, we need to form 3 ***for*** loops. Accordingly, the number of operations to be performed also scales with N^3 as well as the the amount of time that it takes. We can measure the time in Python with **time.time()** function. Matrices are created using random matrix generator **random.randint()**. The indices are integer values between 0 and 9. As seen in the results, due to randomness involved in creating matrices, there are small deviations from N^3.

The probability that an atom decays in a time interval of length t is given by

$$p(t) = 1 - 2^{\frac{-t}{\tau}} \tag{2}$$

where τ is the half-life of the associated atom. By using uniformly-distributed random numbers we can decide on whether an atom will decay or not. It is easier to think of this problem as a coin flip. In both cases, we know the probability that it is heads or tails (50% tails and 50% heads, of course, assuming non-biased coin) but we do not know whether it will be heads or tails prior to flipping the coin. And using random-number-generators in Python, we can decide whether it is tails or heads by comparing the corresponding probability with the random number from an uniform distribution between 0 and 1. Similarly, the random number from uniform-distribution being smaller than the probability associated with the time interval of length t corresponds to decay of an atom. Also, it should be noted that since we perform operations atom-by-atom and it is random in nature, the order of the inner ***for*** loops does not affect the result. This is one way of modeling the decay process of atoms, which requires us to go on in equal time intervals (the time interval for the third question is 1 s, and there is a total of 20000 time steps). On the other hand, we can also model this process using the probability distribution itself which is apparently non-uniform. It is important to note that probability function and probability distribution are not the same thing. Probability distribution is defined as the the probability of decay between times t and $t + dt$. In our case, probability distribution is given by

$$P(t) = 2^{\frac{-t}{\tau}} \frac{ln2}{\tau} \tag{3}$$

and we can transform a random number from uniform-distribution to another from the associated non-uniform-distribution as follows:

$$x = \frac{-1}{\mu} ln(1 - z) \tag{4}$$

where $\mu = ln2/\tau$, x is the random number from the non-uniform distribution whereas z from uniform. The derivation of this equation can be found in [1]. Quick interpretation of Eq. 3 tells us that it is more probable for an atom to decay at earlier times. In the extreme case, when t goes to infinity, the probability of an atom (that has not decayed after infinite amount of time) decaying reduces to zero. So, the random numbers generated using this non-uniform distribution are actually the times that a decay occurs. What we do here is actually to let the computer decide that it is a decay process or not based on the probability distribution.

3 Results

The result of the first question is presented in Table 1. As can be seen, the error starts to increase after 1e-8 as δx becomes smaller.

Table 1: Corresponding δx and calculated $f'(x)$ values.

δx	Calculated $f'(x)$
1e-2	1.010000000000001
1e-4	1.0000999999998899
1e-6	1.0000009999177333
1e-8	1.0000000039225287
1e-10	1.000000082840371
1e-12	1.0000889005833413
1e-14	0.9992007221626509

The result of the second question is presented in Figure 1. The green curve represents the function $f(N) = N^3$, first normalized by dividing the whole array by the highest number in that array (which is the last element) and then multiplied by the the highest number in time array for a better visualization. As seen **dot()** function takes no time in comparison to explicit one. For $N = 100$, while the time explicit method took is 0.295711 s, **dot()** took only 0.000323 s. It

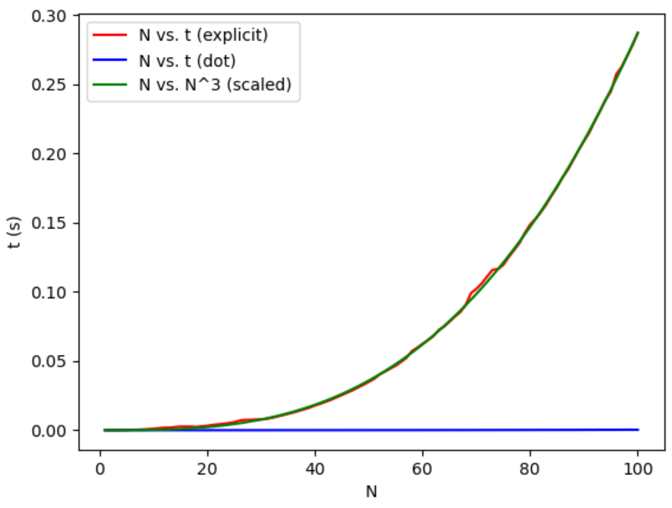

Figure 1: The amount of time required to perform floating-point operations as a function of the matrix size N.

The results of the question 3 and 4 are presented in Figure 2 and 3, respectively.

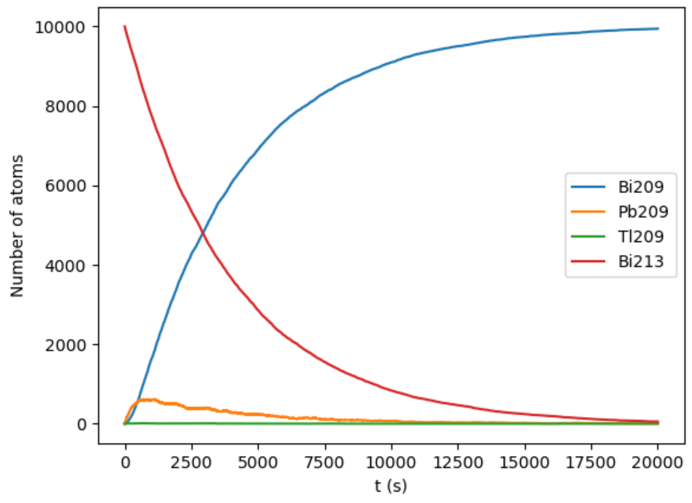

Figure 2: The number of atoms in decay chain with respect to time.

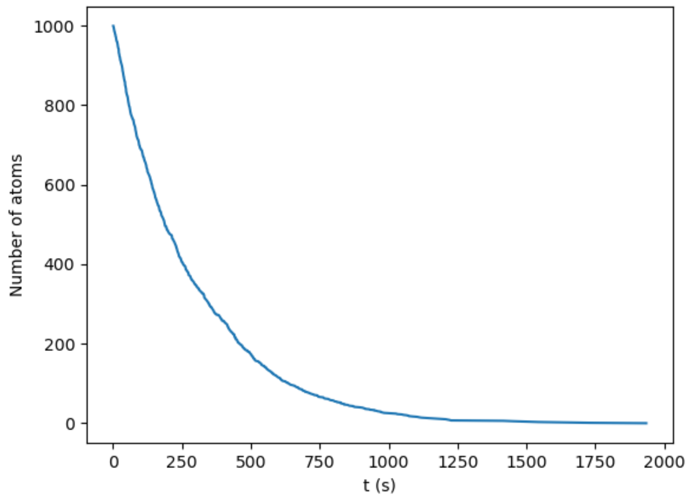

Figure 3: The number of atoms that has not decayed with respect to time.

My GitHub repository is ***akoralaykin/phys-ga2000***

References

[1] M.E.J. Newman. *Computational Physics*. CreateSpace Independent Publishing Platform, 2013.