# Krishna Dindial Problem Set 3

https://github.com/kdindial/phys-ga2000

September 25, 2024

## 1 Problem 1: Matrix Multiplication

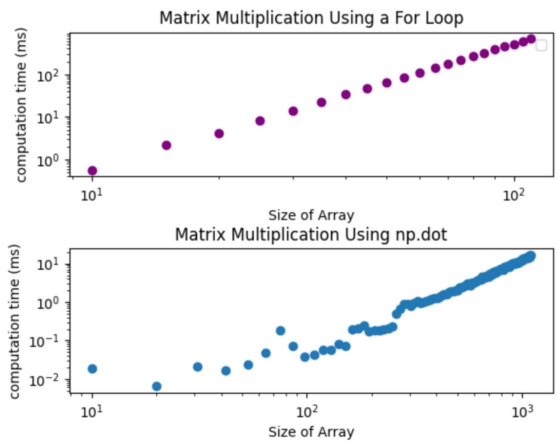


Figure 1: for loop method vs dot method

In this problem I made a plot comparing computation time of performing matrix multiplication the naive with a for loop way vs multiplying two matrices with the dot way. I generated NxN square matrices of random numbers between 0 and 1 and 1. Last time I used timit to find the time it takes for a function to run, but I passed in the function with no arguments as a string. for example, If i want to know how long the function f() takes, I passed timit "f()". Now I wanted to specifically check timit( f(A,B) ) where f is the multiplaction of matrices A and B, and I wanted to compare the time it took for different matrices A and B with different sizes.

I told chat gpt: "I have a function that takes two arguments. I want to use timeit but vary the the argument of the function." Chatgpt told me:

"To use timeit for a function with varying arguments, you can define the function with the arguments you want, and then pass the function call within a lambda to timeit.timeit. This allows you to vary the arguments each time you run timeit."

It gave me an example that you can see in the comments of my code. I understood that if i tried timit(f(a,b)) it would try to pass whatever f(a,b) returns into the function timit, but a lambda function would allow me to pass the function itself as an argument to timeit. Once I understood this, it was straightforward how to make the plots:

you can see that for arrays 100x100, the dot method takes less than a tenth of a millisecond, while the for loop method takes almost 1 second. Lastly, you can see that I matrix size vs time on a log log plot. I used numpys polynomial fit method to find the slope of the log log plot, and found that the slope of the for loop method log log plot was about 2.97. This means that the time does go by about $N^3$.

## 2 problem 2

Here is my plot for problem 2:

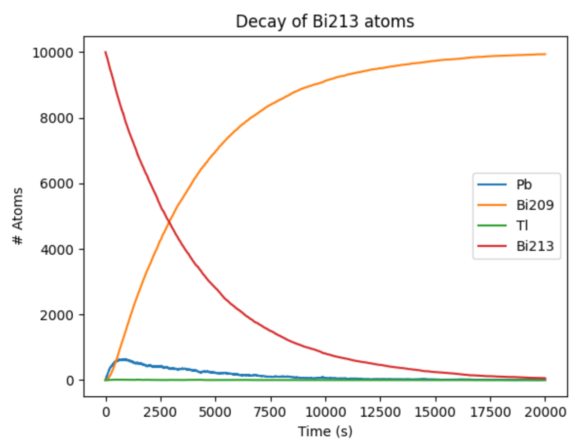


Figure 2: Radioactive decay of Bi213

## 3 Problem 3

This problem was interesting because usually when making plots, I have some f(x), then I create a linearly spaced array of x vals over some range, and then plot x vs f(x). In this problem we use the probability distribution of the decay times of atoms to generate a bunch of t values. In this case our t values are our independent variable. Then I generated a linearly spaced array of values that went 999,998,997... to 0 to represent the number of atoms we had left, because at time 1, 1 atom decays and we have 999 left, then at time 2 we have 998 atoms left etc... It was interesting at first that it was the linearly spaced array was what was going to go on the y axis. Here is the plot of the Tl atoms decaying:

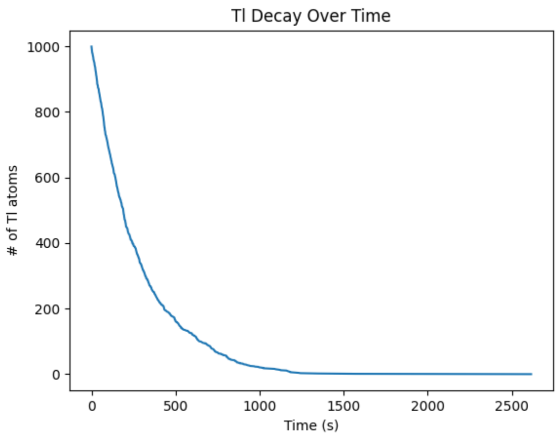


Figure 3: Radioactive decay of Tl Atoms

# 4 Problem 4

Y= $\frac{1}{N}\sum_{i=0}^{N} x_i$ where $x_i$ is a random variable distributed as $e^{-x}$ for x between 0 and infinity. The expectation value of Y would be:

$$\int_0^\infty Y(x)p(x)dx \tag{1}$$

The probability of x is $e^{-x}$ so the expectation value of y is:

$$\int_0^\infty \frac{1}{N}\sum_{i=0}^{N} xe^{-x}dx. \tag{2}$$

The integral $\int_0^\infty xe^{-x} = 1$. Therefore the expectation value as a function of N is:

$$\frac{1}{N}\sum_{i=0}^{N} 1 = \frac{N+1}{N} \tag{3}$$

The variance for independent random variables is:

$$< Y^2 > - < Y >^2 \tag{4}$$

From our last calculation, we know:

$$< Y >^2 = (\frac{N+1}{N})^2 \tag{5}$$

To find $< Y^2 >$ We would take the integral

$$\int_0^\infty Y(x)^2 p(x) dx \tag{6}$$

which is

$$\int_0^\infty \frac{1}{N^2} \sum_{i=0}^{N} x^2 e^{-x} dx \tag{7}$$

The integral $\int_0^\infty x^2 e^{-x} dx = 2$ so,

$$< Y^2 >= \frac{1}{N} \sum_{i=0}^{N} 2 \tag{8}$$

$$< Y^2 >= \frac{2(N+1)}{N^2} \tag{9}$$

finally, we have that the variance is equal to:

$$\frac{2(N+1)}{N^2} - \frac{N+1}{N^2} = \frac{N+1}{N^2} \tag{10}$$

Below I plotted how the mean, variance, skew and kurtosis evolve with increasing n:

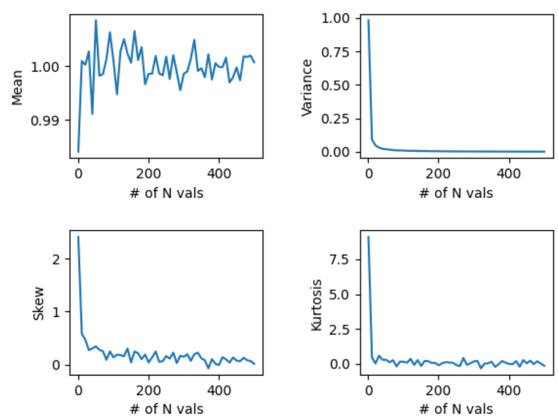


Figure 4: Mean, Variance, Skew and Kurtosis vs N

We are also asked to find a value of N where the skewness and kurtosis have reached about 1 of their value for N = 1. I did this by making a while loop that itterated over the kurtosis or skew until the kurtosis or skew was less than 1 percent of the initial kurtosis or skew. This is not the best way to do it and it is pretty lazy because it returns a different answer every time. What I can say is that N is usually to be between 200-300 for the skew to reach 1 percent of its initial value and N needs to be at least 70 to get the kurtosis to to 1 percent of its initial value.

Now Im going to include some plots to show the distribution of Y values gets more gaussian as N increases:

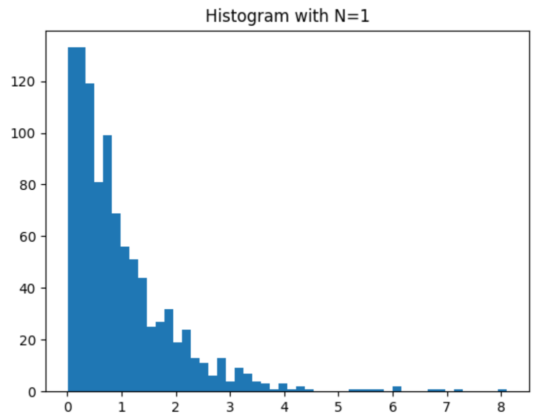


Figure 5: N=1

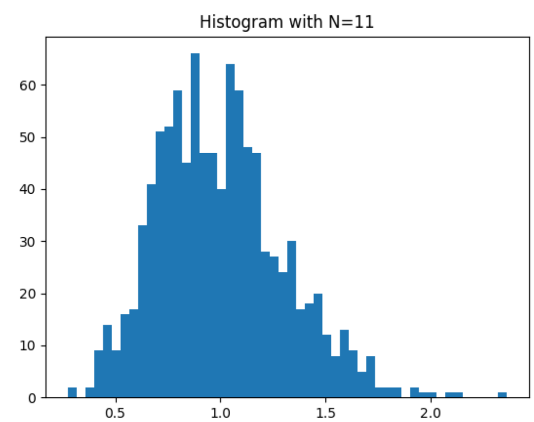


Figure 6: N=11

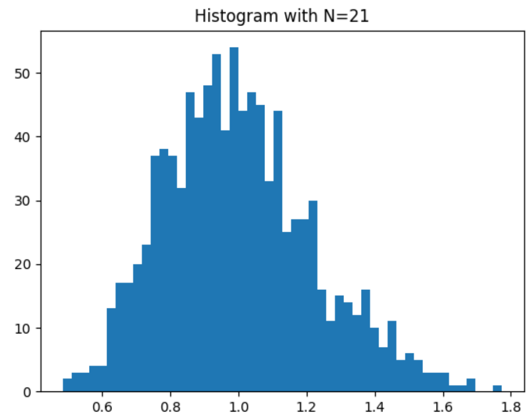


Figure 7: N=21

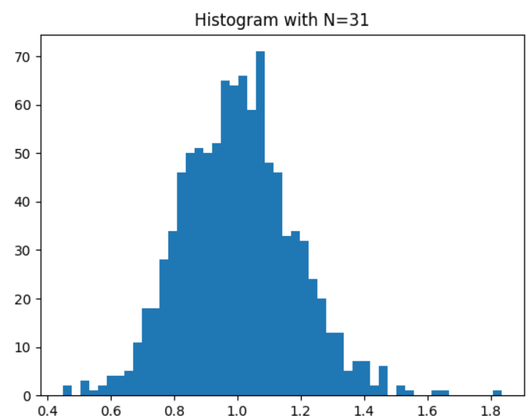


Figure 8: N=31

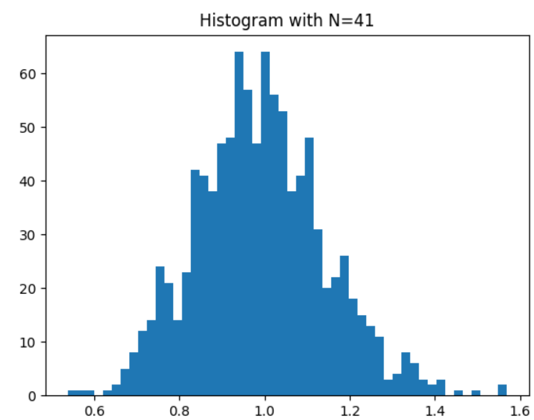


Figure 9: Caption