Computational Physics Tutorial 1

Mogamad Rayhaan Perin
PRNMOG001

February 16, 2020

1 Dot Product

The plot showing the execution time vs the size of the arrays for the dot product is shown below:

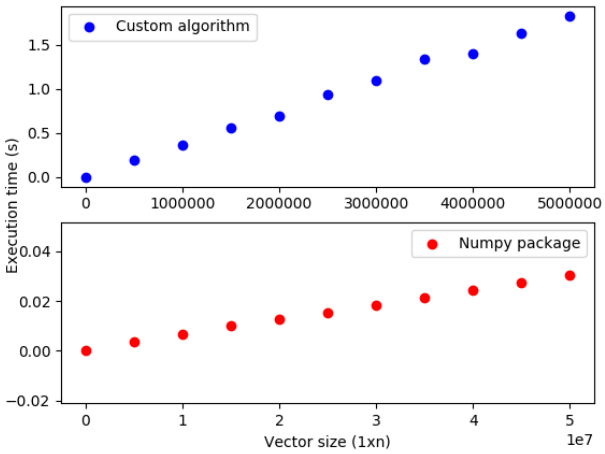

Figure 1: Plot showing the relationship between the execution time and the size of the arrays used.

The execution time of the custom algorithm is much slower than that of the dedicated package numpy which is clearly seen by the custom algorithm taking $\approx 1.55\ s$ to compute the dot product between 5×10^6 sized vectors while the numpy package took $\approx 0.025\ s$ to compute the dot product between 5×10^7 sized vectors which is significantly faster, you run into ram issues when trying to compute the dot product of large vectors whose order of magnitude is greater than the 10^7. The expected asymptotic behavior of the dot product is $\mathcal{O}(n)$ which can be seen.

2 Matrix Vector Product

The plot showing the execution time vs the size of the arrays for the matrix vector product is shown below:

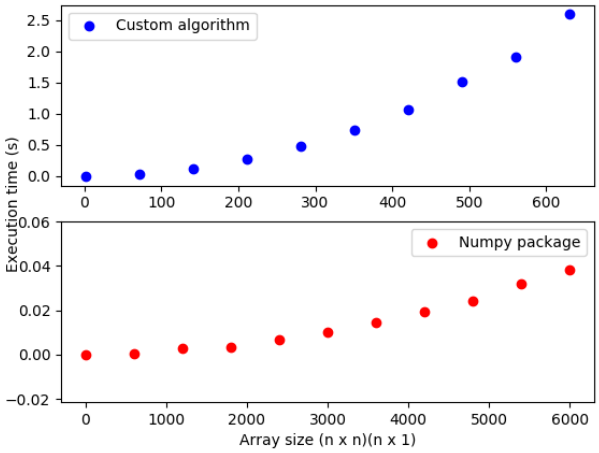

Figure 2: Plot showing the relationship between the execution time and the size of the arrays used.

The execution time of the custom algorithm is much slower than that of the dedicated package numpy which is clearly seen by the custom algorithm taking $\approx 2.0\ s$ to compute the matrix vector product between a 600 x 600 sized matrix and a 600 x 1 sized vector while the numpy package took $\approx 0.04\ s$ to compute the matrix vector product between a 6000 x 6000 sized matrix and a 6000 x 1 sized vector which is significantly faster, you run into ram issues when trying to compute the matrix vector product of large arrays whose size is greater than 6000. The expected asymptotic behavior of the matrix vector product is $\mathcal{O}\left(n^2\right)$ which can be seen.

3 Vector Matrix Vector Product

The plot showing the execution time vs the size of the arrays for the vector matrix vector product is shown below:

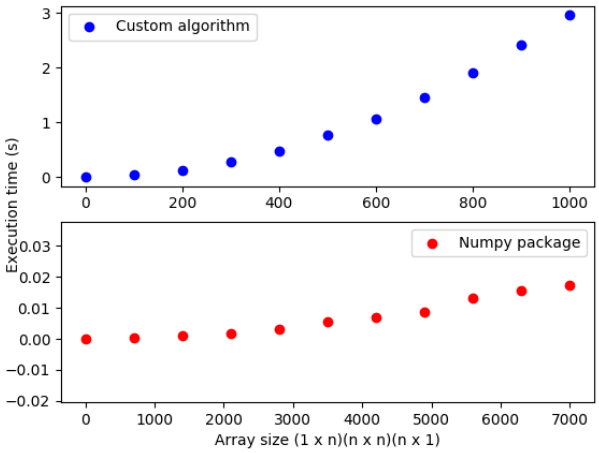

Figure 3: Plot showing the relationship between the execution time and the size of the arrays used.

The execution time of the custom algorithm is much slower than that of the dedicated package numpy which is clearly seen by the custom algorithm taking $\approx 3.0\ s$ to compute the vector matrix vector product between a 1 x 1000 vector, 1000 x 1000 sized matrix and a 1000 x 1 sized vector while the numpy package took $\approx 0.00\ s$ to compute the matrix vector product between a 1 x 1000 sized vector, 1000 x 1000 sized matrix and a 1000 x 1 sized vector which is significantly faster, you run into ram issues when trying to compute the matrix vector product of large arrays whose size is greater than 7000. The expected asymptotic behavior of the vector matrix vector product is $\mathcal{O}\left(n^2 + n\right) \approx \mathcal{O}\left(n^2\right)$ which can be seen. The approximation makes sense due to the n^2 term dominating. The same asymptotic behavior cannot be seen clearly in the numpy package plot due to the ram constraints but if you could compute the product for greater than 7000 sized matrices you will see the same asymptotic behavior as in the custom algorithm.

4 Matrix Matrix Product

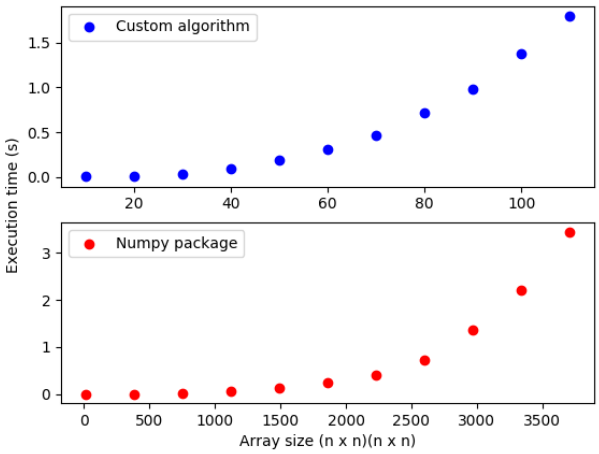

Figure 4: Plot showing the relationship between the execution time and the size of the arrays used.

The execution time of the custom algorithm is much slower than that of the dedicated package numpy which is clearly seen by the custom algorithm taking $\approx 1.5\ s$ to compute the matrix vector product between two 100 x 100 sized matrices while the numpy package took $\approx 1.3\ s$ to compute the matrix vector product between two 3000 x 3000 sized matrices which is significantly faster, you run into ram issues when trying to compute the matrix vector product of large arrays whose size is greater than 4000. The expected asymptotic behavior of the matrix matrix product is $\mathcal{O}\left(n^3\right)$ which can be seen.

5 FLOPS

In order to calculate the FLOPS of my system i used the dot product multiplied by a random number in order to turn its' entries into floats. I then made the matrix size of 5x5 which took 9 FLOP. the time for such a calculation took $\approx 0.0003\ s$ to complete. Then FLOPS = FLOP/time which gave a rough estimate of 336696969 FLOPS $\approx 3.3 \times 10^8$ FLOPS.