# Saudin Abdic, 11730114
# Christoforos Eseroglou, e12331470
# Elsayed Saleh Abrishq Bassam, 12302324
# Ravelet Thomas, 12302809

# Numerical Simulation and Scientific Computing: Assignment 2

## Task 2. Estimate and Benchmark the Performance of dense Matrix-Matrix Multiplication

## Task 2.1 Estimate and Modeling

### Calculate the Theoretical Single Threaded Machine Balance $B_m$

The theoretical single threaded machine balance $B_m$ for a specific CPU is given by the formula

$$B_m = \frac{memory\ bandwidth\ [\text{Bytes/s}]}{peak\ performance\ [\text{FLOPs/s}]}$$

For this task the used CPU was: Intel® Core™ i7-8565U (8M Cache, up to 4.60 GHz). All the relevant information about the processor can be found under: `https://ark.intel.com/content/www/us/en/ark/products/149091/intel-core-i7-8565u-processor-8m-cache-up-to-4-60-ghz.html`. This task requires the following data:

- total cores: 4
- memory bandwidth: 37.5 GBytes/s
- max turbo frequency: 4.60 GHz

The ´peak performance [FLOPs/s]' can be calculated by

$$\text{peak performance} = \text{max turbo frequency} \times \text{total cores} \times 2$$

where 2 is the number of achieved FLOPs per clock cycle for standard double-precision (64-bit) floating-point operations.
Therefore the machine balance $B_m$ can be computed as:

$$B_m = \frac{37.5\ \text{GBytes/s}}{4.60\ \text{GHz}\ \times 4 \times 2} = 1.019\ \text{Bytes/FLOP}$$

This means that the theoretical single-thread machine balance for the used CPU is 1.019 Bytes/FLOP.

**Calculate the Code Balance $B_c$ for a MMM of square matrices of size $N \times N$**

The code balance $B_c$ is defined as

$$B_c = \frac{data\ traffic\ [\text{Bytes}]}{floating\ point\ operations\ [\text{FLOPs}]}$$

Given that there are three matrices (two input matrices and one output matrix), each containing $N^2$ entries, and considering that each matrix element is represented as a double precision floating point requiring 8 Bytes of memory, the ´data traffic' for a MMM can be computed as follows:

$$data\ traffic = 3 \times 8 \times N^2\ [\text{Bytes}]$$

The total number of floating-point operations in a MMM can be determined by recognizing that each element in the output matrix involves $N$ multiplications and $(N-1)$ additions. Summing these operations and multiplying by the total number of entries in the output matrix, which is $N^2$, gives the overall count of floating-point operations. Therefore:

$$floating\ point\ operations = 2 \times N^3\ [FLOPs]$$

Thus the code balance $B_c$ is:

$$B_c = \frac{3 \times 8 \times N^2}{2 \times N^3} = \frac{12}{N}\ [Bytes/FLOPs]$$

**For which matrix size $N$ would a MMM utilize the peak performance on your benchmark system?**

A MMM would utilize the peak performance on the used benchmark system when $B_m = B_c$. This means:

$$B_m = 1.1019\ [\text{Bytes/FLOP}] = \frac{12}{N}\ [\text{Bytes/FLOPs}] \rightarrow N = 11.8$$

**Calculate the single threaded runtime of a MMM on your benchmark system using N=1000, N=2000, N=5000 assuming the MMM would utilizes your systems peak performance.**

The following formula computes the runtime of a MMM on the used CPU:

$$runtime = \frac{floating\ point\ operations\ [\text{FLOPs}]}{peak\ performance\ [\text{FLOPs/s}]} = \frac{2 \times N^3\ [\text{FLOPs}]}{4.60\ \text{GHz}\ \times 4 \times 2}$$

Therefore:

- for N = 1000 $\rightarrow runtime_{N=1000} = \frac{2 \times 1000^3}{4.60\ \text{GHz} \times 4 \times 2} = 0,054\ s$
- for N = 2000 $\rightarrow runtime_{N=2000} = \frac{2 \times 2000^3}{4.60\ \text{GHz} \times 4 \times 2} = 0,435\ s$
- for N = 5000 $\rightarrow runtime_{N=5000} = \frac{2 \times 5000^3}{4.60\ \text{GHz} \times 4 \times 2} = 6,793\ s$

As expected, the runtime grows as the matrix size increases.

## Task 2.2 Benchmarking

**Plot the results of the benchmark test for N = {64, 128, 256, 500, 512, 1000, 1024, 1500} using the three different implementations: CUSTOM, BLAS, EIGEN. Additionally, plot the runtimes if the peak performance of your system would be utilized and provide information about the CPU. Briefly discuss your results.**

Figure 1 displays the runtime-results for the MMM using different implementations and the theoretical runtime for the CPU peak performance case.

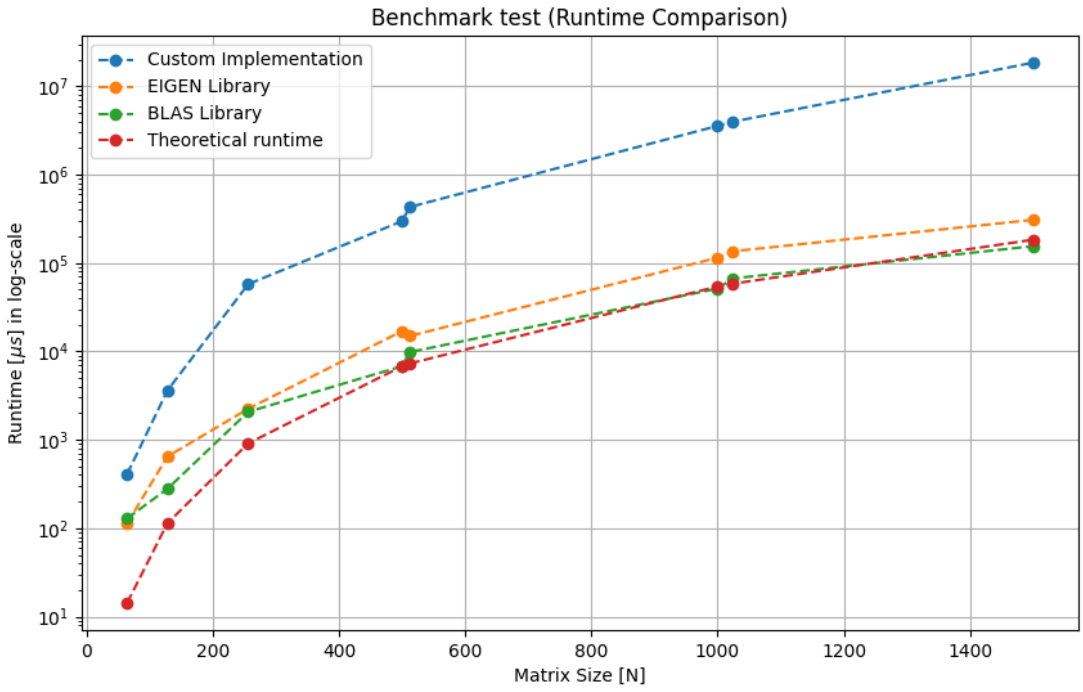


Figure 1: Benchmark test visualization

Discussion:
The CPU used for this task was the Intel® Core™ i7-8565U (8M Cache, up to 4.60 GHz). The runtime behavior aligns well with theoretical calculations, except for the BLAS implementation for matrix size N = 1000 and N = 1500, where the BLAS implementation is slightly faster. It's possible that BLAS still employs multithreading despite being disabled through provided

environment variables. As expected, the CUSTOM implementation exhibits the slowest performance due to its nested for-loop structure, leading to noticeable runtime increases with larger matrix sizes (note that the y-axis is logarithmically scaled). EIGEN and BLAS implementations demonstrate similar runtime patterns, with BLAS marginally faster. The subtle runtime distinction between the two may stem from variations in optimization strategies within their respective libraries.