Report Lab Session 1

Diego Lanaspa, Jorge Armas

October 4, 2025

1 Introduction

The goal of this report is to establish a comparison in performance between two different implementations of the same problem. We have measured the execution time of matrix multiplications implemented with base C++ and using the Eigen library and compared them.

1.1 Hardware Specifications

Aiming to make the most objective comparison we have used the same computer for all the tests ran. This machine is a personal laptop with average components and performance, as shown in . All the computations have been ran on the CPU.

Components
AMD ryzen 5 7535HS 3301 Mhz
6 cores, 12 threads
16GB RAM DDR5
NVIDIA GeForce RTX 2050

Table 1: Working computer components

2 Experimental work

This section details the experimental procedure and presents the results obtained from the performance comparison between the classical and Eigen implementations.

2.1 Methodology

In order to have the most objective measurements possible we created two different C++ programs with varying matrix sizes. The first program multiplies two NxN matrices using the classical algorithm and without any aditional libraries. For the second program, however, the Eigen library and it's matrix utils were used.

For simplicity all the matrices were constrained to be NxN dimensions. In order to look for significant differences between the two programs we decided to have different N values, those being: 100, 1000, 2000, 3000 and 5000.

In order to extract information from the data asides from measuring time the average and the standard deviation were calculated for every N value and implementation by executing them 5 times each.

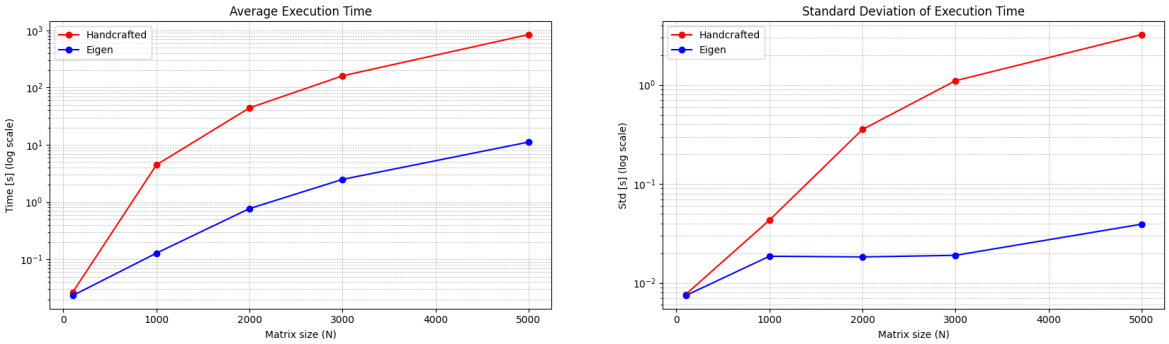
2.2 Results

Here are presented the results of the matrix multiplication experiments. The performance of the iterative implementation is compared with the Eigen library in terms of average execution time and standard deviation for the different matrix sizes.

Table 2: Results of matrix multiplication experiments

Matrix size (N)	Classical (avg [s])	Classical (std [s])	Eigen (avg [s])	Eigen (std [s])
100	0.0266	0.00765	0.0233	0.00745
1000	4.468	0.0431	0.1283	0.0186
2000	44.134	0.3550	0.77	0.0183
3000	160.224	1.103	2.49	0.0190
5000	842.56	3.227	11.168	0.0392

Table 3: Average execution time and standard deviation for matrix multiplication with classical implementation and Eigen



(a) Average execution time

(b) Standard deviation of execution time

Figure 1: Comparison of average and standard deviation execution times for matrix multiplication using iterative and Eigen implementations.

Table 2 and Figure 1 compare the execution time and variability of iterative matrix multiplication against Eigen for different matrix sizes. For small matrices, both methods perform similarly, but as the size increases, the iterative implementation becomes significantly slower. At N=5000, it takes about 843 seconds, while Eigen finishes in 11 seconds, nearly two orders of magnitude faster.

The Eigen implementation also shows much lower standard deviations, indicating more consistent performance. In contrast, the iterative implementation exhibits increasing variability for larger matrices. Overall, Eigen clearly outperforms the iterative approach in both speed and reliability, and the performance gap grows significantly with matrix size.

The difference in performance can be explained by the fact Eigen takes advantage of the matrix product property of every row by column dot product being able to be computed idependently, to paralellize , which allows for better use of the CPU architecture [1]. In contrast, the classical implementation is based on a nested triple-loop algorithm without any optimization. This behavior is consistent with the theoretical time complexity of the classical method, which is $O(N^3)$ [2], meaning that execution time increases rapidly as the matrix size grows. Although Eigen is still affected by this cubic growth, its internal optimizations significantly reduce the impact, leading to the large performance gap observed for bigger matrices.

References

[1] Eigen library documentation. Eigen and multithreading. https://libeigen.gitlab.io/eigen/docs-nightly/TopicMultiThreading.html

[2] Wikipedia. "Computational complexity of matrix multiplication." *Wikipedia, the free encyclopedia*, 2023. https://en.wikipedia.org/wiki/Computational_complexity_of_matrix_multiplication