Laboratory 1. Getting Started: Environment, Command Line, Compiling, Debugging, and Programming Basics

Beatriz Rosell Cortés
María Peribáñez Tafalla

September 2024

1 Function Summary

- **Standard Matrix Multiplication** This algorithm multiplies two N×N matrices by computing the product of each row in the first matrix with each column in the second, resulting in cubic complexity. While simple, it becomes inefficient for large matrices. Our implementation uses static memory allocation, which avoids runtime overhead but is inflexible for varying sizes.
- **Eigen Math Library** Eigen optimizes matrix multiplication using cache-friendly access patterns, SIMD (parallel processing), and block matrix multiplication, dividing large matrices into smaller blocks to minimize memory latency. Though dynamic memory allocation introduces some overhead, the performance gains from these optimizations make Eigen significantly faster, especially for large matrices.

Both methods share $O(N^3)$ complexity, but Eigen's advanced optimizations yield better practical performance for large-scale computations.

2 Performance Evaluation

2.1 Analysis of Execution Times

When evaluating the efficiency of multiplying matrices, it's essential to consider the right timing metric.
- **Real Time**: This is the elapsed time from start to finish of the operation, as perceived by the user.
- **User Time**: This measures the amount of time the CPU spends in user mode executing the code. It does not include time spent waiting for I/O operations or system calls.
- **System Time**: This reflects the amount of time the CPU spends in kernel mode (system calls) on behalf of your program.

The user time directly represents the time the CPU is executing the operations that matter for the performance of the algorithm. When comparing different implementations of matrix multiplication, user time will provide a good indication of the efficiency and effectiveness of each method in terms of CPU usage.

Normal	Real (s)	User (s)	Sys (s)
2000x2000	41.032	39.624	0.078
1000x1000	4.851	4.571	0.025
500x500	0.664	0.5781	0.008
100x100	0.017	0.007	0.001
5x5	0.003	0.000	0.001

(a) Standard Matrix Multiplication

Eigen	Real (s)	User (s)	Sys (s)
2000x2000	3.568	2.139	0.085
1000x1000	0.708	0.328	0.025
500x500	0.113	0.058	0.005
100x100	0.014	0.002	0.001
5x5	0.003	0.000	0.001

(b) Eigen Math Library

Figure 1: Execution times for both coded versions varying the NxN matrix sizes.

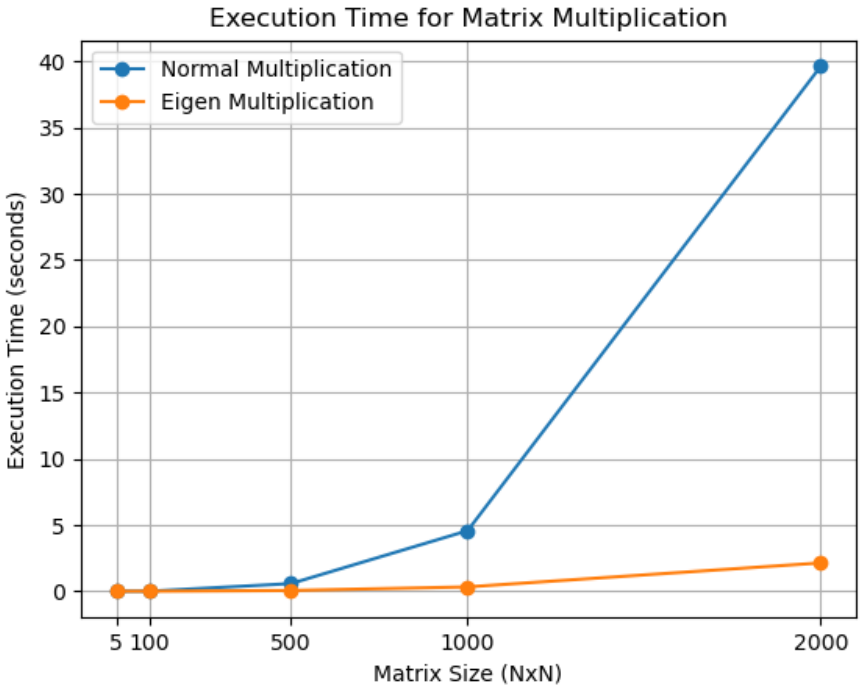

Figure 2: Graphical representation of execution times for both coded versions varying the NxN matrix sizes.

2.2 Statistical Analysis and Discussion

Both the standard matrix multiplication and Eigen's optimized multiplication have a computational complexity of $O(N^3)$. However, Eigen's optimizations make it significantly faster in practice.

The following tables summarize the mean and standard deviation of execution times across 10 runs for both implementations. We can see how the Eigen Library is not only faster, but more consistent as well. Eigen's optimizations reduce the impact of system-level variations, such as cache misses or memory access delays, resulting in lower standard deviation in processing times across repeated runs.

Matrix Size	Standard Multiplication		Eigen Math Library	
	Mean (s)	Std. Dev (s)	Mean (s)	Std. Dev (s)
2000x2000	66.2029	1.85622	3.80938	0.0314799
1000x1000	7.565	0.00823104	0.631375	0.015644
500x500	0.26025	0.00345507	0.119375	0.00360338
100x100	0.004625	0.000856957	0.003875	0.00105327
5x5	0.000875	0.000599479	0.000875	0.000599479

Table 1: Mean and Standard Deviation of Execution Times for both versions.

2.3 Additional Considerations

- **Memory Allocation Policy Justification:** Static memory allocation, used in the standard matrix multiplication, is suitable for fixed-size matrices but lacks flexibility. Dynamic memory allocation, used in Eigen, provides flexibility and is essential for handling large and varying matrix sizes efficiently.

- **Compiler Optimizations and CPU Load:** We also experimented with different compiler optimization flags, reducing the no-flag binary execution of the Standard Matrix Multiplication of approximately 6 minutes to only 1 minute using the `-O2` flag, a necessary adjustment to obtain the times for the Statistical Analysis.