

# ECE 4822 – Numerical Libraries Report

Shahzad Khan

## Goal

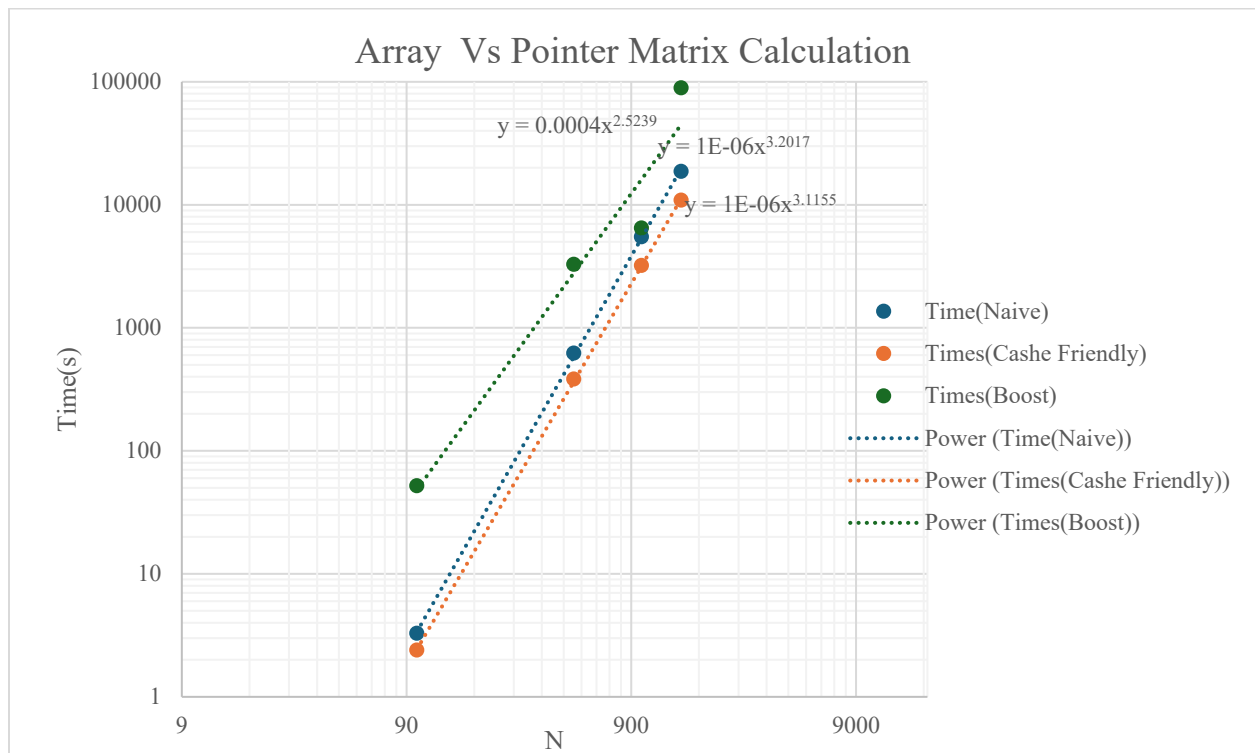
The goal of this assignment was to repeat HW\_01 (matrix multiplication) but this time utilize the power of numerical libraries in C/C++: Boost, STL, and Eigen. The objective was to evaluate performance differences, code complexity, and overall efficiency between the libraries compared to the naive triple-loop implementation.

## Problem

In the original HW\_01, matrix multiplication was implemented using three nested loops. This approach, while straightforward, had a cubic time complexity ( $O(N^3)$ ) and demonstrated poor performance for large matrices. The use of external libraries was expected to provide significant improvements in execution time and code simplicity.

## Solutions

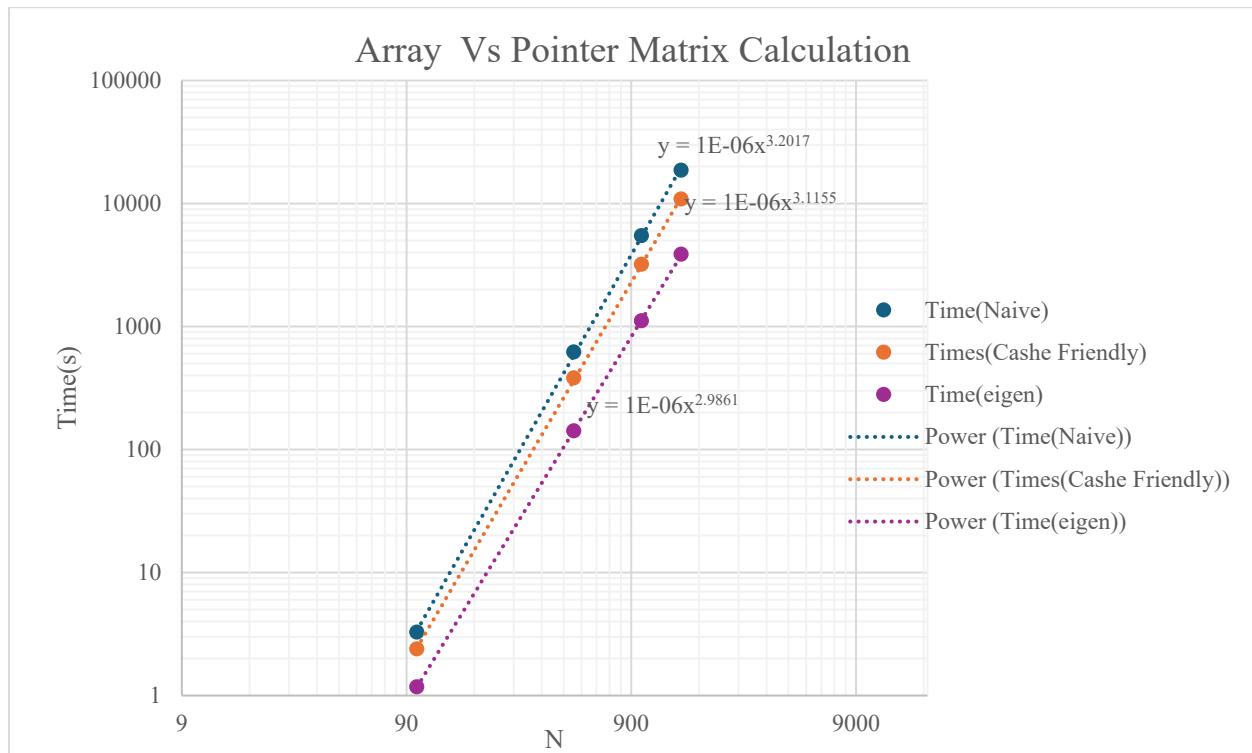
### Boost



Boost provides a wide range of high-level abstractions, including support for matrix operations. By switching to Boost, I was able to drastically reduce code complexity—nested loops were replaced with cleaner matrix manipulation syntax. However, performance results did not meet expectations:

- **Execution times:** Boost was consistently slower than the manual implementation, even for large matrices (e.g., 1500×1500).
- **Time complexity:** Theoretical complexity was measured at approximately  $O(N^{2.5})$ , compared to the naive  $O(N^{3.2})$ . Despite this, execution time was worse.

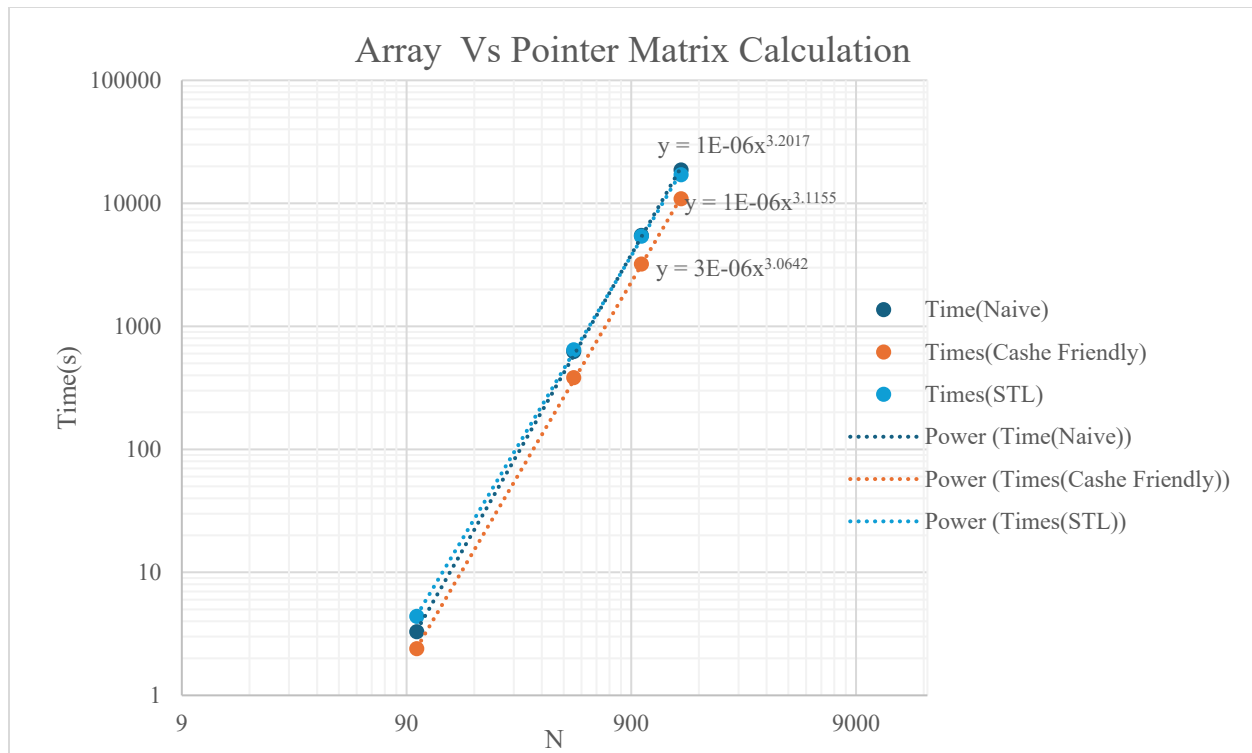
## Eigen



Eigen proved to be the most efficient and reliable library for matrix multiplication:

- **Performance:** Eigen drastically reduced computation times. For  $N = 1500$ , Eigen was nearly **5× faster** than the naive triple-loop method.
- **Time complexity:** Measured at approximately  $O(N^{2.9})$ , an improvement over both naive and Boost implementations.

## STL

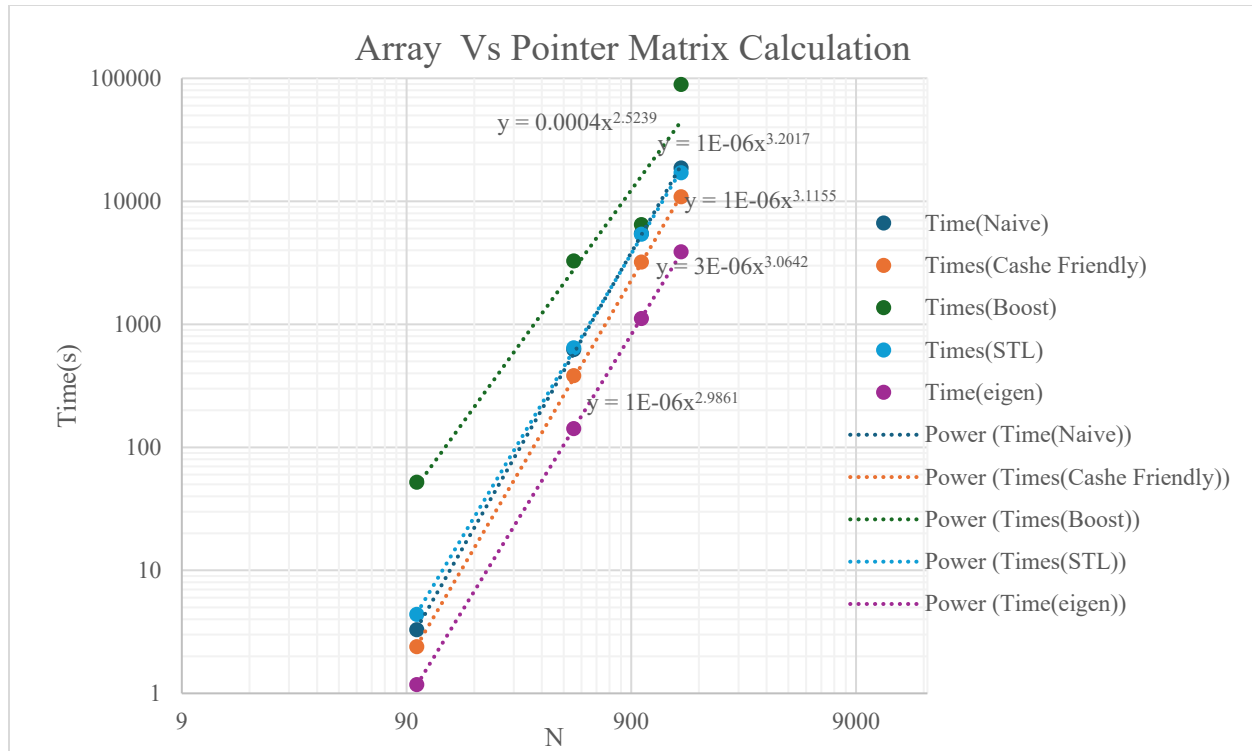


The STL-based implementation closely resembled the naive triple-loop version, but used `std::vector` instead of arrays:

- **Execution times:** Nearly identical to the original HW\_01 results.

**Conclusion:** While STL improved code readability and memory safety, it provided no significant performance gains for matrix multiplication. The similarity to the naive method explains the comparable results.

## Overall Conclusion



This experiment highlighted how different libraries impact both performance and code structure:

- **Boost:** Improved code readability but underperformed in execution speed due to abstraction overhead.
- **STL:** Provided safety and structure but offered no performance gains compared to the naive triple-loop approach.
- **Eigen:** Combined simple syntax with deep low-level optimizations, delivering exceptional performance and scalability.

**Key takeaway:** Writing efficient numerical code requires more than choosing the right algorithm it also depends on how libraries interact with hardware and memory. Eigen's hardware-aware design and optimizations made it the best choice for large-scale numerical tasks in this study.