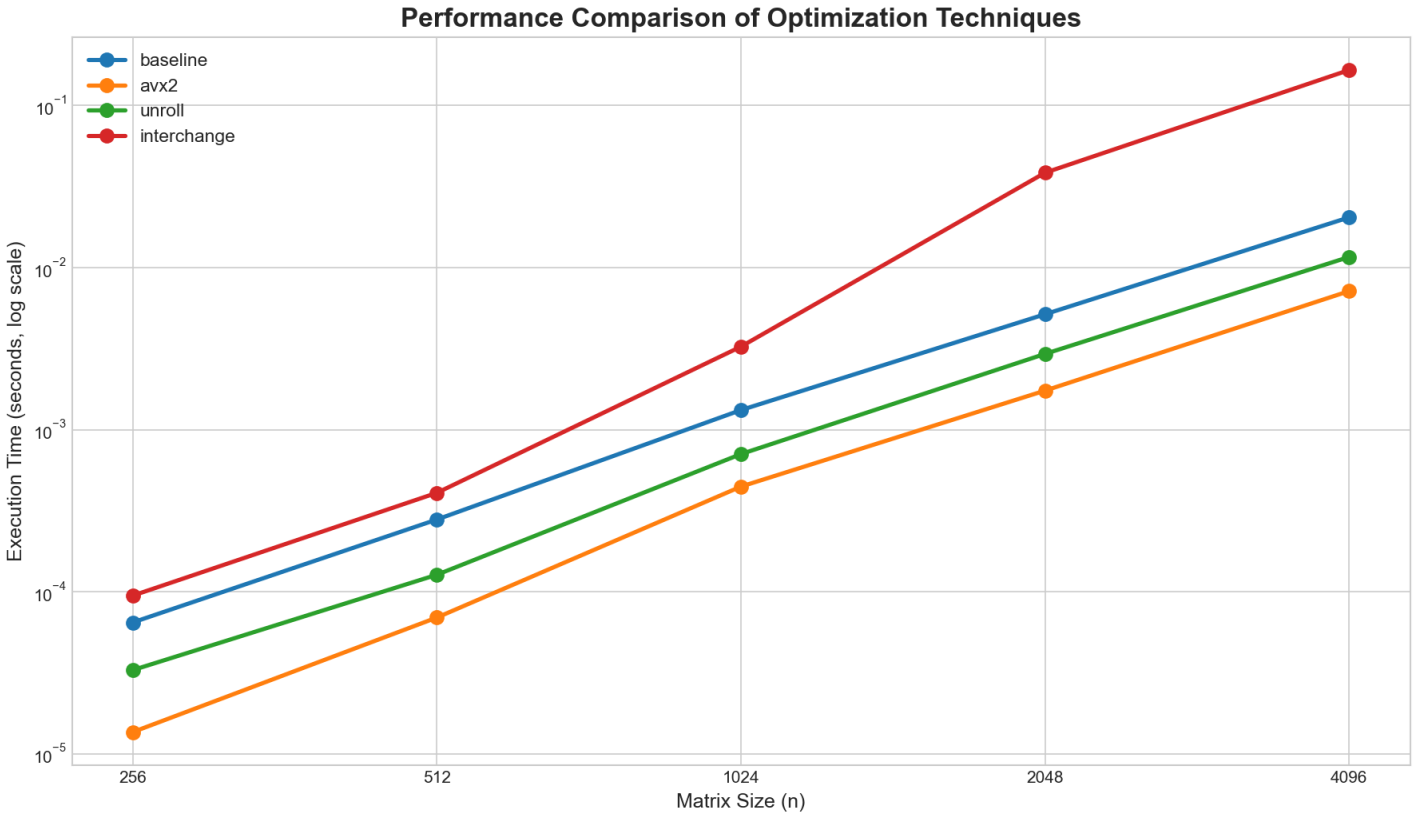
HW1 Performance Report

1. Optimization Techniques

This report compares several optimization techniques for matrix-vector multiplication against a baseline compiled with g++ -O3.

1. AVX2 SIMD: Leverages data parallelism by processing 4 doubles simultaneously using Advanced Vector Extensions.
2. Loop Unrolling: Reduces loop overhead by manually processing 8 elements per inner-loop iteration.
3. Loop Interchange: Swaps the inner and outer loops to demonstrate the negative impact of poor memory access patterns (cache performance).

2. Performance Chart



3. AI Usage Log

AI Tool Used: Google Gemini

How I used the AI as a programming tool:
1. Code Scaffolding: I prompted the AI to generate initial boilerplate for the C++ programs (argument parsing, timing), which I then reviewed and adapted for the project's specific needs.
2. Syntax and Concept Lookup: I used the AI to look up the specific syntax for AVX2 intrinsics (`_mm256_fmadd_pd`, etc.) and to get a template for the horizontal sum logic, which I then integrated and validated within my `avx2.cpp` implementation.
3. Makefile Structure: I requested a basic `makefile` structure for handling multiple C++ targets, which I then

customized with the specific compiler flags (`-march=native`) and targets required for this assignment.
4. Automation Scripting: The AI significantly accelerated the creation of the report generator. I had it generate individual functions for running benchmarks, creating plots, and building tables. I then assembled these components, debugged the integration, and refined the final report layout.

Where the AI tool was useful:
The tool excelled at accelerating development by handling repetitive or syntactically complex tasks. It was invaluable for quickly generating templates and looking up function calls, which allowed me to focus more on the high-level optimization strategy and the analysis of the results.

Where the AI tool fell short:
The tool required careful guidance and validation. For instance, its initial C++ code failed to compile due to a linker error (using `gcc` instead of `g++`), which I had to diagnose and correct. It also did not proactively suggest the crucial `-march=native` flag, reinforcing the need for the programmer to possess the underlying domain knowledge.

Impact on my role as a programmer:
Using the AI shifted my role from a traditional coder to that of a technical director and quality assurance engineer. My primary tasks became defining the problem with precision, breaking it down into components the AI could handle, and then critically reviewing, debugging, and integrating the generated code.