Benchmark of an LU decomposition algorithm in Python

Márton Papp, Ákos Szabó
ELTE FI

Introduction

The LU decomposition ($PA = LU$) is a cornerstone of numerical linear algebra, essential for solving systems of linear equations ($Ax = b$) and inverting matrices.

- **Importance:** It allows efficient solving for multiple right-hand sides compared to direct inversion. [1]
- **Problem:** The algorithm has a cubic time complexity ($O(n^3)$), making performance critical for large datasets.
- **Goal:** This study benchmarks a custom "pure Python" implementation (Doolittle's method with partial pivoting) to evaluate runtime scaling and numerical stability without low-level optimizations.

Materials and Methods

Implementation Environment

- **Hardware:** Intel Core i5-1135G6 (2.4 GHz, 4 cores), 8GB RAM (Standard Workstation). [cite: 35-37]
- **Software:** Python 3.14 using NumPy 2.3 for dense matrix operations. [cite: 39, 42]
- **Algorithm:** Custom implementation of Doolittle's method with partial pivoting. No external optimization libraries were used. [cite: 6, 44]

Benchmarking Protocol

- **Dataset:** Randomly generated square matrices ($A \in \mathbb{R}^{n \times n}$). [cite: 61]
- **Matrix Sizes:** $n \in \{10, 50, 250, 1000\}$ to capture execution scaling.
- **Trials:** 100 independent runs per matrix size to measure variability.
- **Error Metric:** Numerical stability assessed via Frobenius-norm residual: $||PA - LU||_F$

Results

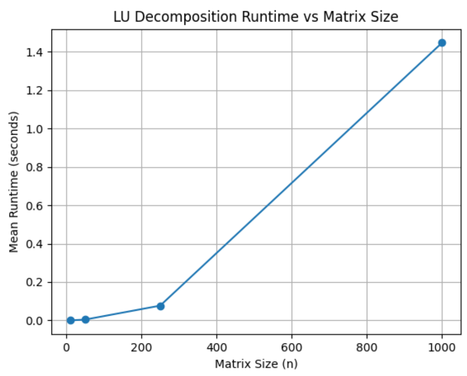

Figure 1: Mean runtime vs. Matrix Size. The execution time follows the theoretical cubic growth ($O(n^3)$), exceeding 1s at $n = 1000$.

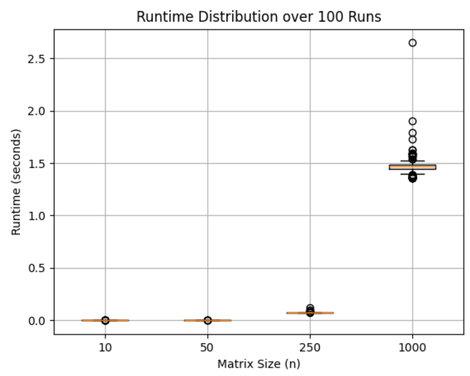

Figure 2: Runtime variability (100 runs). Significant outliers appear at larger matrix sizes ($n = 1000$) due to OS scheduling and caching effects.

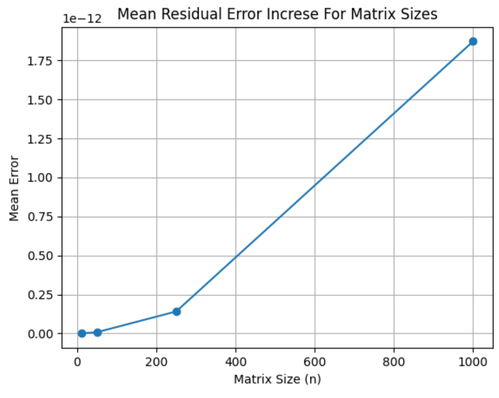

Figure 3: Residual Error ($||PA - LU||_F$) scaling. Although error grows ($10^{-16} \rightarrow 10^{-12}$), it remains negligible.

Conclusions

The implementation confirms the expected $O(n^3)$ complexity while maintaining high numerical accuracy ($PA \approx LU$). Although suitable for educational benchmarking, production environments require optimized libraries for large-scale performance.

References

[1] Dogan Kaya and Ken Wright. Parallel algorithms for LU decomposition on a shared memory multiprocessor. *Applied Mathematics and Computation*, 163(1):179–191, 2005.

[2] Salza Nur Bandiyah and Yoni Marine. Implementing LU decomposition to improve computer network performance. *International Journal of Technology and Modeling*, 4(2):82–90, 2025.

[3] Dan Simon. Kalman filtering. *Embedded Systems Programming*, 14(6):72–79, 2001.