BENCHMARK OF AN LU DECOMPOSITION ALGORITHM IN PYTHON

MÁRTON PAPP, ÁKOS SZABÓ

ELTE FI

INTRODUCTION

Objective

- Implement & benchmark custom LU decomposition algorithm $(PA = LU)$ using Doolittle's method with partial pivoting.
- Evaluate **"pure Python"** performance (no low-level optimizations).

Motivation

- LU decomposition is a fundamental tool for solving linear systems $(Ax = b)$ efficiently.
- High computational cost: $O(n^3)$ complexity.
- Understanding Python's limitations in numerical algorithms.

METHODOLOGY

Implementation Environment

- **Hardware:** Intel Core i5-1135G6 (4 cores), 8 GB RAM.
- **Software:** Python 3.14, NumPy 2.3 (for dense matrix storage) .
- **Constraint:** Custom implementation without external optimization libraries (e.g., LAPACK).

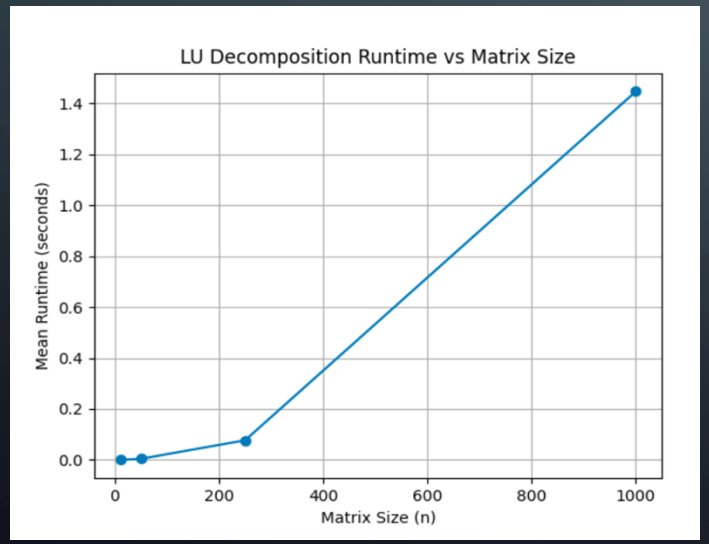
Benchmarking Protocol

- **Dataset:** Randomly generated square matrices ($A \in \mathbb{R}^{n*n}$).
- **Test Cases:** Matrix sizes n ∈ {10, 50, 250, 1000} to evaluate runtime scaling.
- **Reliability:** 100 independent trials per size to measure variability.
- **Metrics:**
  - Execution Time (high-precision timer).
  - Numerical Stability ($||PA = LU||_F$).

RESULTS

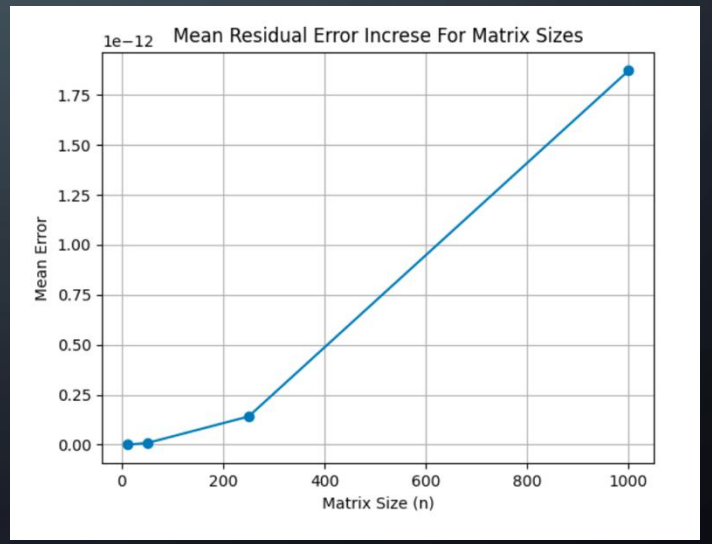
Runtime Analysis

- Shows clear cubic growth ($O(n^3)$) aligning with theory.
- Runtime rises sharply, exceeding **1.4s** for n=1000.



Numerical Stability

- Residual error ($||PA = LU||_F$) increases with size ($10^{-16} \rightarrow 10^{-12}$)
- Despite growth, error remains negligible.



CONCLUSION

Key Findings

- Confirmed theoretical $O(n^3)$ **complexity** with predictable runtime scaling.
- High numerical precision ($PA \approx LU$) achieved via **partial pivoting**.

Final Verdict

- **Educational use:** Ideal implementation for understanding algorithmic mechanics.
- **Production use:** For large scale systems, optimized compiled libraries (e.g., LAPACK) are necessary