Numerical Computing

Lecture 5: Direct Methods for Linear Systems

Francisco Richter Mendoza

Università della Svizzera Italiana
Faculty of Informatics

September 9, 2025

Overview

- **Gaussian Elimination** and LU Decomposition
- **Pivoting Strategies** for numerical stability
- **Cholesky Decomposition** for symmetric positive definite matrices
- **Stability Analysis** and growth factors
- **Special Matrix Structures** (tridiagonal, band matrices)
- **Iterative Refinement** for improved accuracy

Goal: Understand the mathematical foundations and stability properties of direct methods for solving linear systems $Ax = b$.

LU Decomposition

Definition

For matrix $A \in \mathbb{R}^{n \times n}$, the **LU decomposition** is:

$$A = LU$$

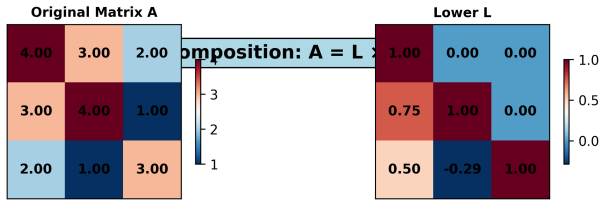
where L is lower triangular with unit diagonal and U is upper triangular.

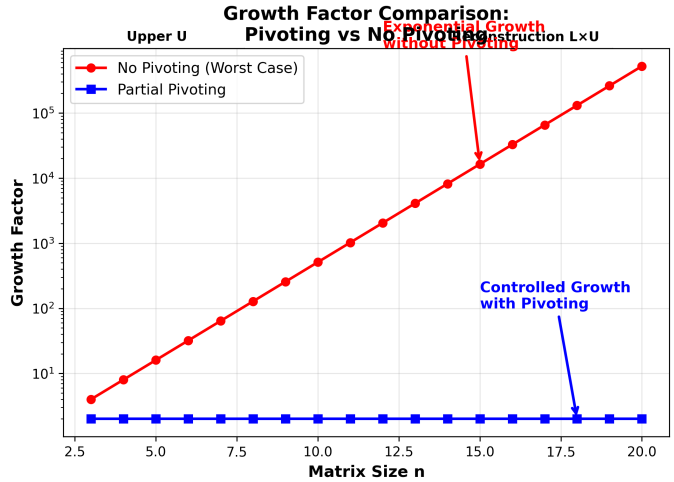
Theorem (Existence)

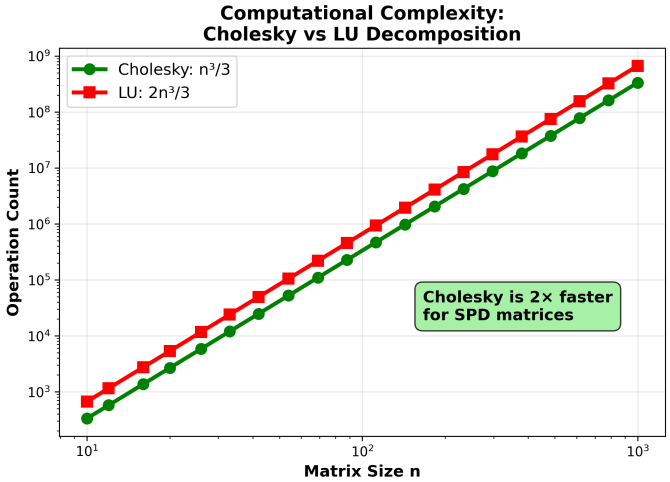
LU decomposition exists and is unique if and only if all leading principal minors of A are nonzero.

Key insight: Gaussian elimination systematically computes the LU factorization.

LU Decomposition Process







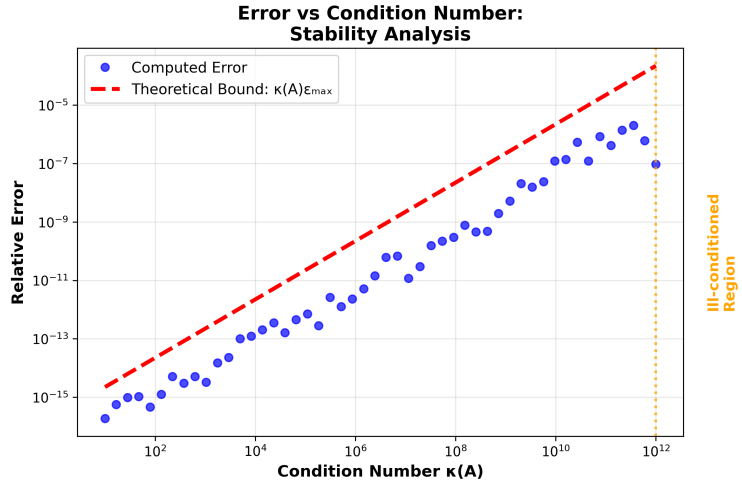


Figure: LU decomposition visualization showing the factorization $A = L \times U$, pivoting strategy comparison, Cholesky efficiency analysis, and condition number effects on stability.

Computational Complexity

Theorem (Operation Count)

LU decomposition of an $n \times n$ matrix requires:

$$\frac{2n^3}{3} + O(n^2) \text{ operations}$$

Proof sketch.

Column k elimination requires $\approx 2(n-k)^2$ operations:

$$\sum_{k=1}^{n-1} 2(n-k)^2 = 2\sum_{j=1}^{n-1} j^2 = \frac{2n^3}{3} + O(n^2)$$

□

Practical implication: Doubling matrix size increases computation by factor of 8.

Pivoting for Numerical Stability

Definition (Partial Pivoting)

At step k, choose pivot row p such that:

$$|a_{p,k}^{(k)}| = \max_{k \le i \le n} |a_{i,k}^{(k)}|$$

Definition (Growth Factor)

$$\rho = \frac{\max_{i,j,k} |a_{i,j}^{(k)}|}{\max_{i,j} |a_{i,j}|}$$

measures element growth during elimination.

Without pivoting: ρ can be 2^{n-1} (exponential growth)
With partial pivoting: ρ typically $O(n)$ in practice

Cholesky Decomposition

Definition

For symmetric positive definite A:

$$A = LL^T$$

where L is lower triangular with positive diagonal.

Theorem (Computational Advantage)

Cholesky decomposition requires:

$$\frac{n^3}{3} + O(n^2) \text{ operations}$$

Exactly half *the cost of LU decomposition!*

Algorithm: $L_{ii} = \sqrt{A_{ii} - \sum_{k=1}^{i-1} L_{ik}^2}$, $L_{ji} = \frac{1}{L_{ii}} \left(A_{ji} - \sum_{k=1}^{i-1} L_{jk} L_{ik} \right)$

Special Matrix Structures

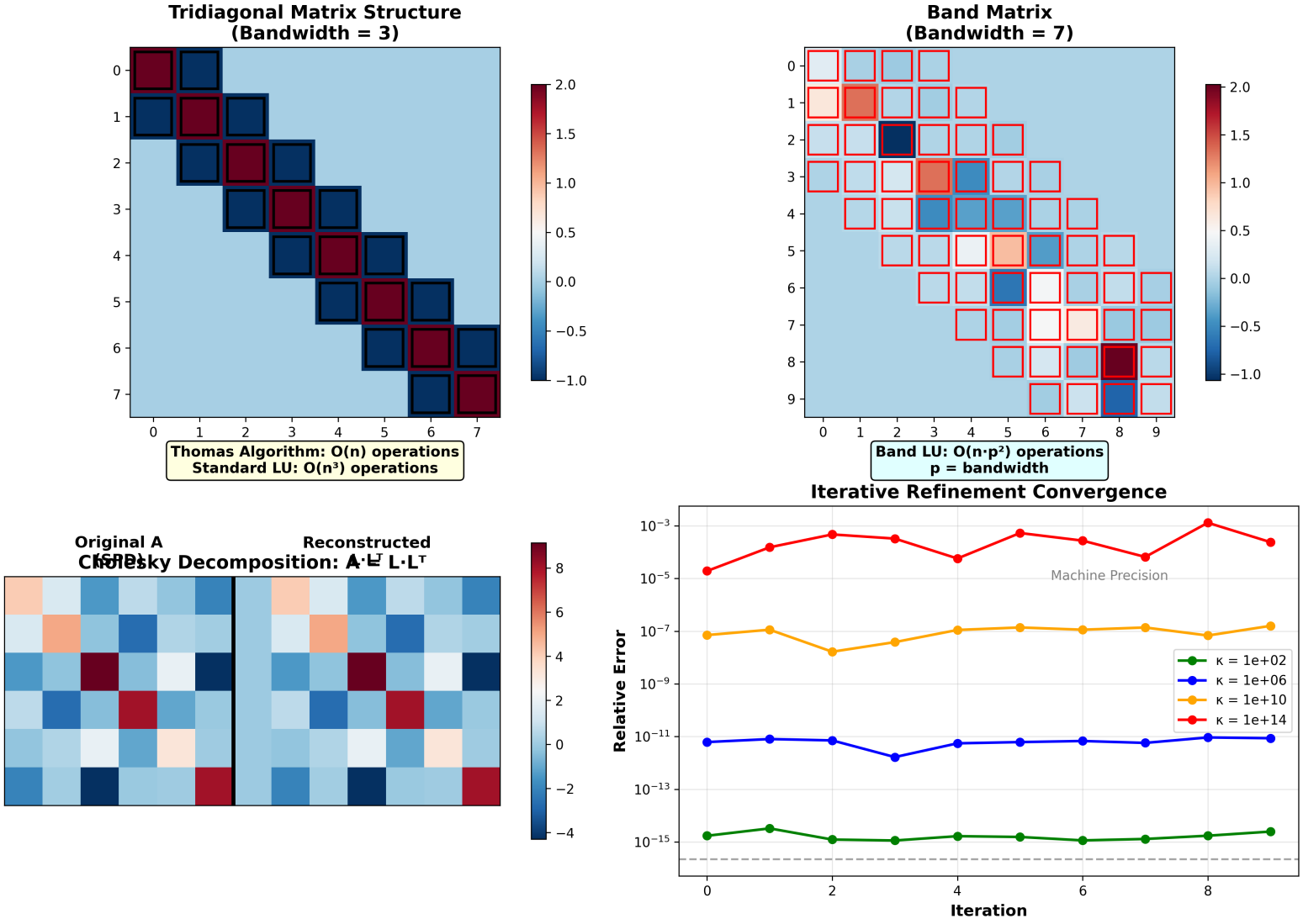


Figure: Special matrix structures: tridiagonal systems with Thomas algorithm O(n) complexity, band matrices, Cholesky decomposition visualization, and iterative refinement convergence analysis.

Thomas Algorithm for Tridiagonal Systems

For tridiagonal matrix:

$$\begin{pmatrix} b_1 & c_1 & & \\ a_2 & b_2 & c_2 & \\ & \ddots & \ddots & \ddots \\ & & a_n & b_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{pmatrix}$$

Forward elimination:

for $i = 2$ to n **do**

$c'_i = c_i/(b_i - a_i c'_{i-1})$

$d'_i = (d_i - a_i d'_{i-1})/(b_i - a_i c'_{i-1})$

end for

Back substitution:

$x_n = d'_n$

for $i = n - 1$ down to 1 **do**

$x_i = d'_i - c'_i x_{i+1}$

end for

Backward Stability Analysis

Theorem (Backward Stability)

Gaussian elimination with partial pivoting is backward stable:

$$(A + \Delta A)\tilde{x} = b$$

where $\|\Delta A\|_\infty \leq \gamma_n \rho \|A\|_\infty$ *with* $\gamma_n = O(n\epsilon_{mach})$.

Theorem (Forward Error Bound)

$$\frac{\|\tilde{x} - x\|}{\|x\|} \leq \gamma_n \rho \kappa(A) + O(\epsilon_{mach}^2)$$

Key insight: Error depends on both algorithm stability (ρ) and problem conditioning ($\kappa(A)$).

Stability Analysis

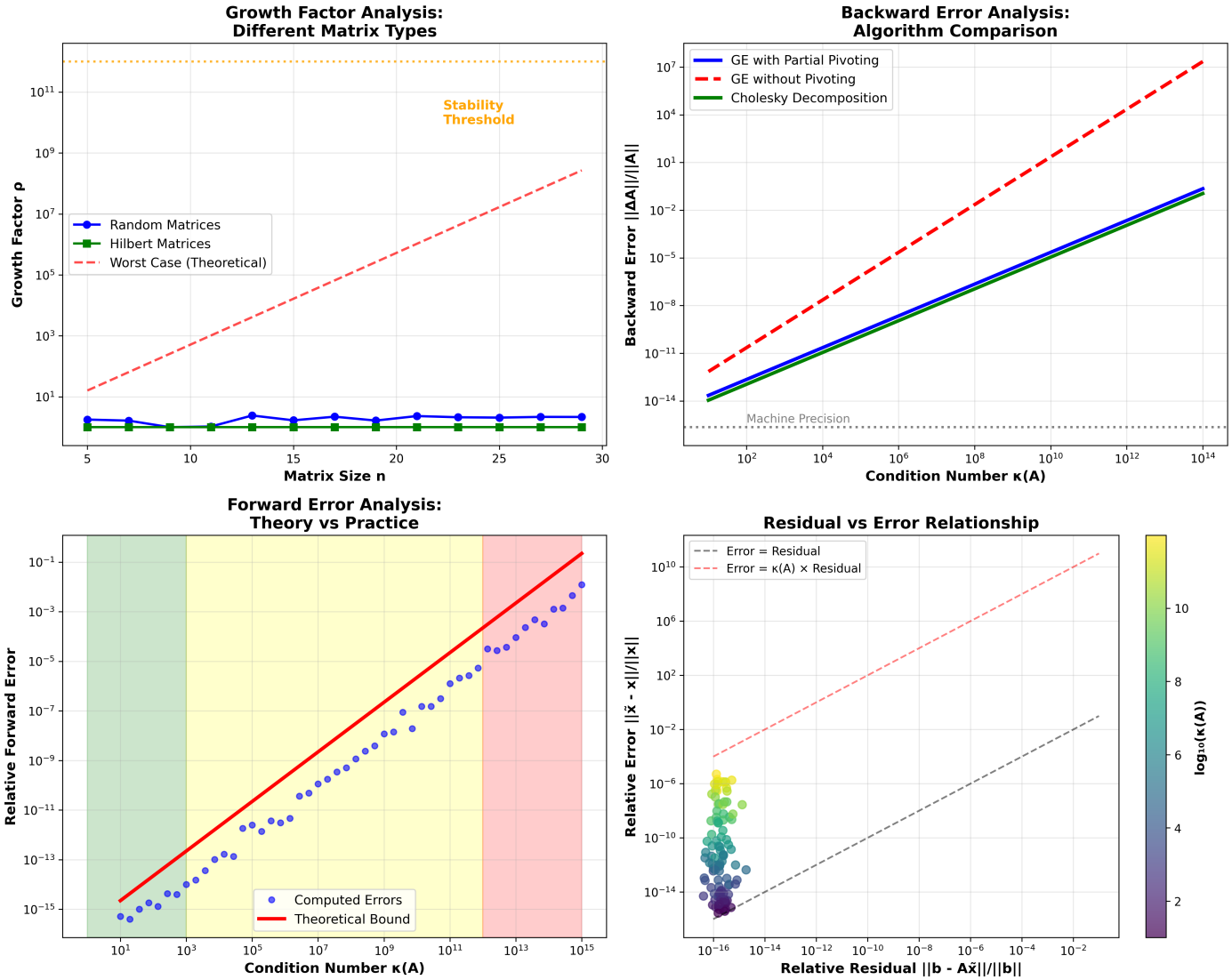


Figure: Comprehensive stability analysis: growth factor comparison across matrix types, backward error analysis for different algorithms, forward error vs condition number relationship, and residual-error correlation.

Iterative Refinement

Algorithm 1 Iterative Refinement

Given: A, b, initial solution $x^{(0)}$
for $k = 0, 1, 2, \ldots$ **do**
  Compute residual: $r^{(k)} = b - Ax^{(k)}$
  Solve correction: $A\delta^{(k)} = r^{(k)}$
  Update solution: $x^{(k+1)} = x^{(k)} + \delta^{(k)}$
end for

Theorem (Convergence)

If $\epsilon\kappa(A) < 1$ where ϵ is the backward error of LU, then iterative refinement converges and can recover full machine precision.

Practical benefit: Can improve accuracy from $O(\kappa(A)\epsilon_{\text{mach}})$ to $O(\epsilon_{\text{mach}})$.

Computational Complexity Comparison

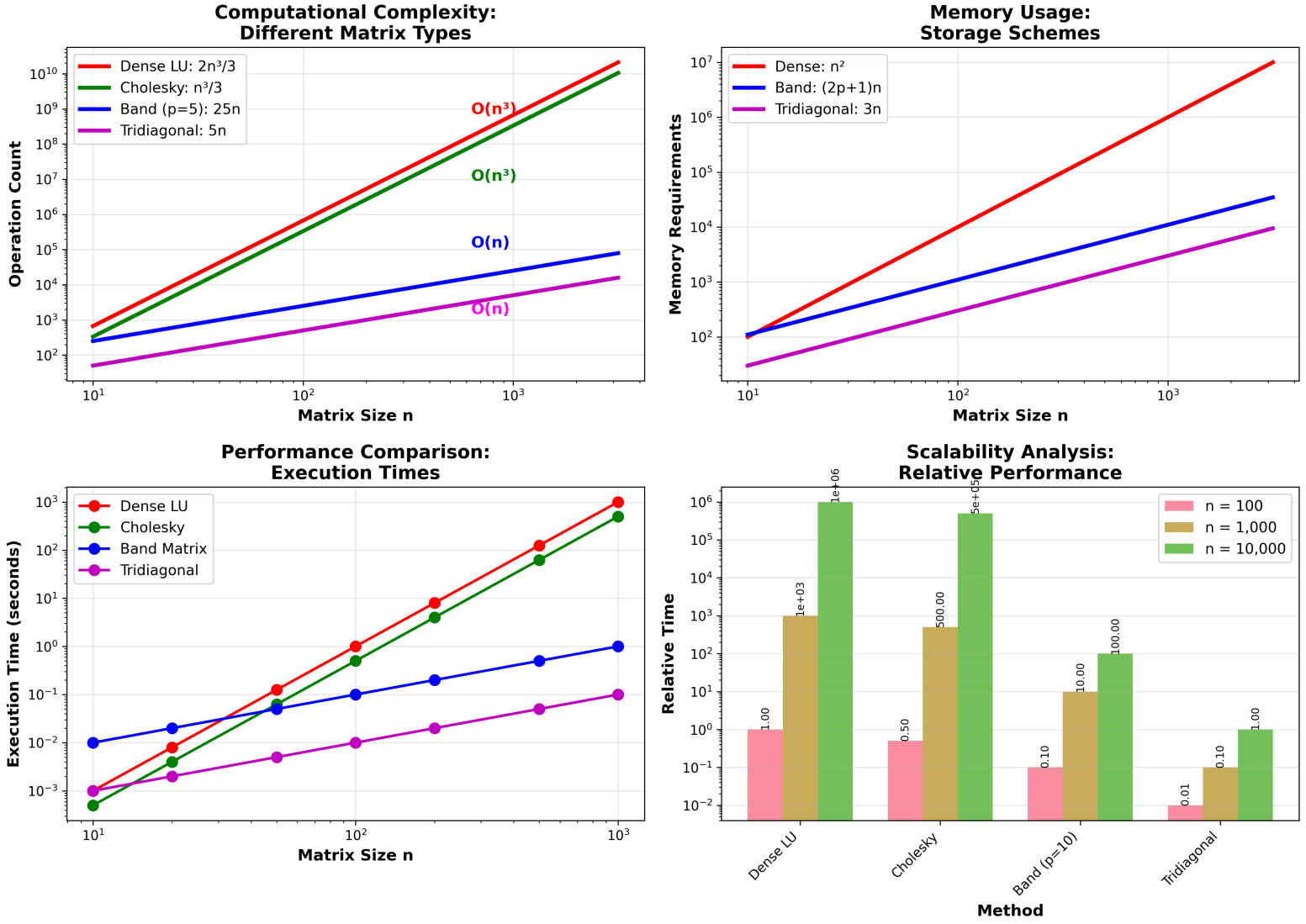


Figure: Computational complexity analysis: operation count comparison for different matrix types, memory requirements for various storage schemes, execution time comparisons, and scalability analysis showing

Practical Implementation

```
# LU decomposition with partial pivoting
from scipy.linalg import lu_factor, lu_solve
import numpy as np

# Create system
A = np.random.randn(100, 100)
b = np.random.randn(100)

# Factor once, solve many times
lu, piv = lu_factor(A)
x = lu_solve((lu, piv), b)

# For SPD matrices, use Cholesky
from scipy.linalg import cholesky, solve_triangular
A_spd = A.T @ A + np.eye(100)  # Make SPD
L = cholesky(A_spd, lower=True)
y = solve_triangular(L, b, lower=True)
x = solve_triangular(L.T, y, lower=False)
```

Key principle: Factor once, solve many times for multiple right-hand sides.

Rule of Thumb: Conditioning and Accuracy

- **Well-conditioned:** $\kappa(A) \leq 10^3$ (lose ≤ 3 digits)
- **Moderately conditioned:** $10^3 < \kappa(A) \leq 10^{12}$ (lose 3-12 digits)
- **Ill-conditioned:** $\kappa(A) > 10^{12}$ (lose > 12 digits)

Theorem (Accuracy Estimate)

For condition number $\kappa(A)$, expect to lose approximately $\log_{10}(\kappa(A))$ decimal digits of accuracy.

Example: If $\kappa(A) = 10^6$ and you start with 16 digits precision, solution has ≈ 10 accurate digits.

When to Use Direct Methods

- **Advantages:**
  - Exact solution (in exact arithmetic)
  - Predictable computational cost: $O(n^3)$
  - Robust for well-conditioned systems
  - Factor once, solve for multiple RHS
- **Disadvantages:**
  - High memory requirements: $O(n^2)$
  - Expensive for large sparse systems
  - Sensitive to ill-conditioning
  - Fill-in destroys sparsity structure

Best for: Dense systems, multiple RHS, well-conditioned problems, $n \lesssim 10^4$

Key Takeaways

1. **LU decomposition** is the foundation of direct methods
2. **Pivoting** is essential for numerical stability
3. **Cholesky** is $2\times$ faster for symmetric positive definite matrices
4. **Special structures** (tridiagonal, band) enable $O(n)$ or $O(np^2)$ algorithms
5. **Stability** depends on growth factor and condition number
6. **Iterative refinement** can recover lost accuracy

Next lecture: Least Squares Problems and QR Decomposition