Faculty of Computer Science

19.06.2025

LinAlgTools

Numerical Algorithms of Linear Algebra

Zinkin Zakhar, 232
Supervisor: Dmitriy Trushin, associate professor

Goal & Motivation

- Main goal
  - Implement a matrix decomposition library in C++
- Why this project?
  - Existing libraries, like Eigen and Armadillo, prioritize performance over readability.

Tasks

1. Study matrix decomposition algorithms (discussed below).
2. Write down the necessary theory.
3. Implement a C++ library (from scratch)
4. Validate correctness
5. Benchmark performance.

Functional requirements

- Implement *Matrix*, *SubMatrix* and *ConstSubMatrix* classes.
- Implement *RandomGenerator* class.
- Implement supplementary transformations.
- Implement decompositions.

Key Algorithms Implemented

1. *QR Decompositions*
2. *Real Schur Decomposition for real matrices*
3. *Singular Value Decomposition (SVD)*

Key Algorithms implemented

- QR Decomposition ($A \in \mathbb{C}^{m \times n}$):
  $A = QR$
  $Q \in \mathbb{C}^{m \times m}$ is unitary, $R \in \mathbb{R}^{m \times n}$ is upper-triangular.

- Real Schur Decomposition ($A \in \mathbb{R}^{n \times n}$):
  $A = USU^*$
  $U \in \mathbb{C}^{n \times n}$ is unitary, $S \in \mathbb{C}^{n \times n}$ is block upper-triangular.

- SVD ($A \in \mathbb{C}^{m \times n}$):
  $A = U\Sigma V^*$
  $U \in \mathbb{C}^{m \times m}$, $V \in \mathbb{C}^{n \times n}$ are unitary, $\Sigma \in \mathbb{C}^{m \times n}$ is diagonal.

Structure of tests

```
LinAlgTools/
└─ tests/
   ├─ validation_tests/
   └─ benchmark_tests/
```

Validation tests

Validation happens at two levels:

- Precondition and postcondition asserts.
- At least 1 hand-written test for all methods and functions.

Performance tests

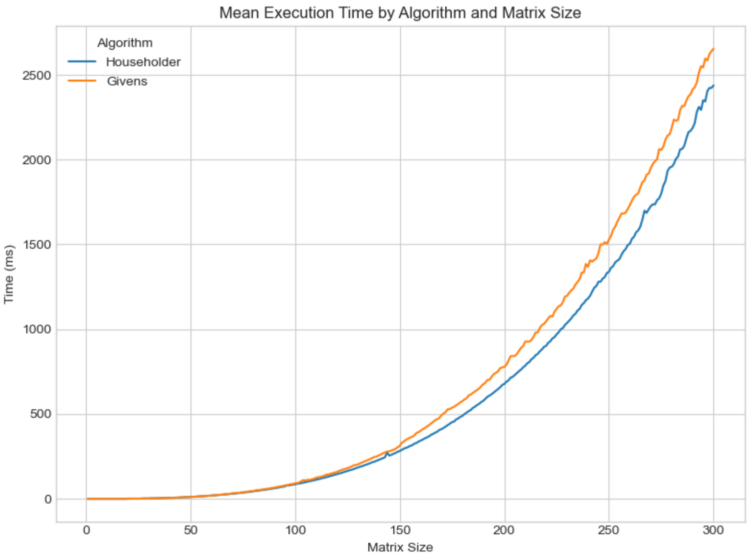
Deterministic algorithms (QR):

- Stop when all transformations are applied.

Iterative algorithms (Real Schur, SVD):

- Iterate until matrices achieve desired form (ε precision).

QR Decomposition performance



Results

- Theoretical presentation of the material.
- Developed C++ library (3500 lines of code):
    1. *Matrix class*
    2. *RandomGenerator class*
    3. *QR Decompositions*
        - 3.1 *Householder Reflections*
        - 3.2 *Givens Rotations*
    4. *Real Schur Decomposition*
        - 4.1 *Hessenberg Form*
        - 4.2 *Wilkinson Shift*
    5. *Singular Value Decomposition (SVD)*
        - 5.1 *Bidiagonalization*

Potential improvements

- Implement Golub-Kahan SVD.
- Implement Schur Decomposition for complex matrices.
- Handle sparse and dense matrices efficiently.