

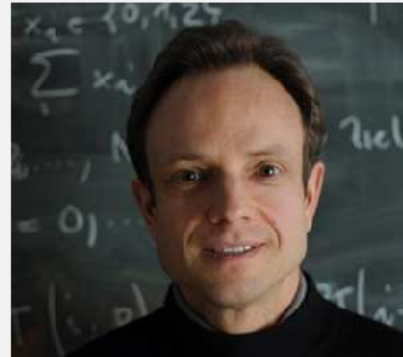
Course Video

Section 2.7.3: Direct Solution of Sparse Linear Systems of Equations

Prof. R. Hiptmair, SAM, ETH Zurich

Date: September 19, 2020

(C) Seminar für Angewandte Mathematik, ETH Zürich



Dependencies. Section 2.7.1, Section 1.4, Section 2.3

Duration: 18 minutes

II. Direct Methods for Square Linear Systems of Equations

2.7. Sparse Linear Systems

2.7.3. Direct Solution of Sparse Linear Systems of Equations

Assume: System matrix in sparse matrix format
 ↓
 tells location of zero entries
 ↓
 can be exploited by sparse elimination techniques

Eigen:

C++-code 2.7.3.1: Function for solving a sparse LSE with EIGEN → GITLAB

```

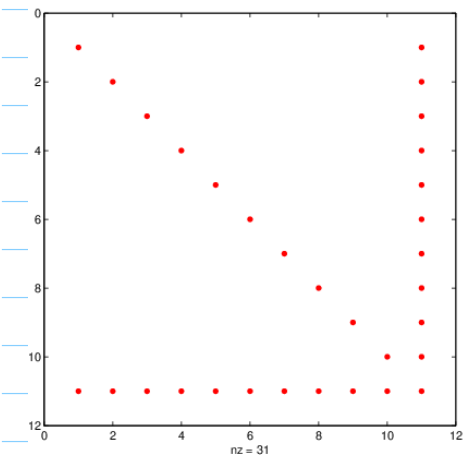
2 using SparseMatrix = Eigen::SparseMatrix<double>;
3 // Perform sparse elimination
4 void sparse_solve(const SparseMatrix &A, const VectorXd &b, VectorXd &x) {
5     Eigen::SparseLU<SparseMatrix> solver(A);
6     if (solver.info() != Eigen::Success) {
7         throw "Matrix factorization failed";
8     }
9     x = solver.solve(b);
10 }

```

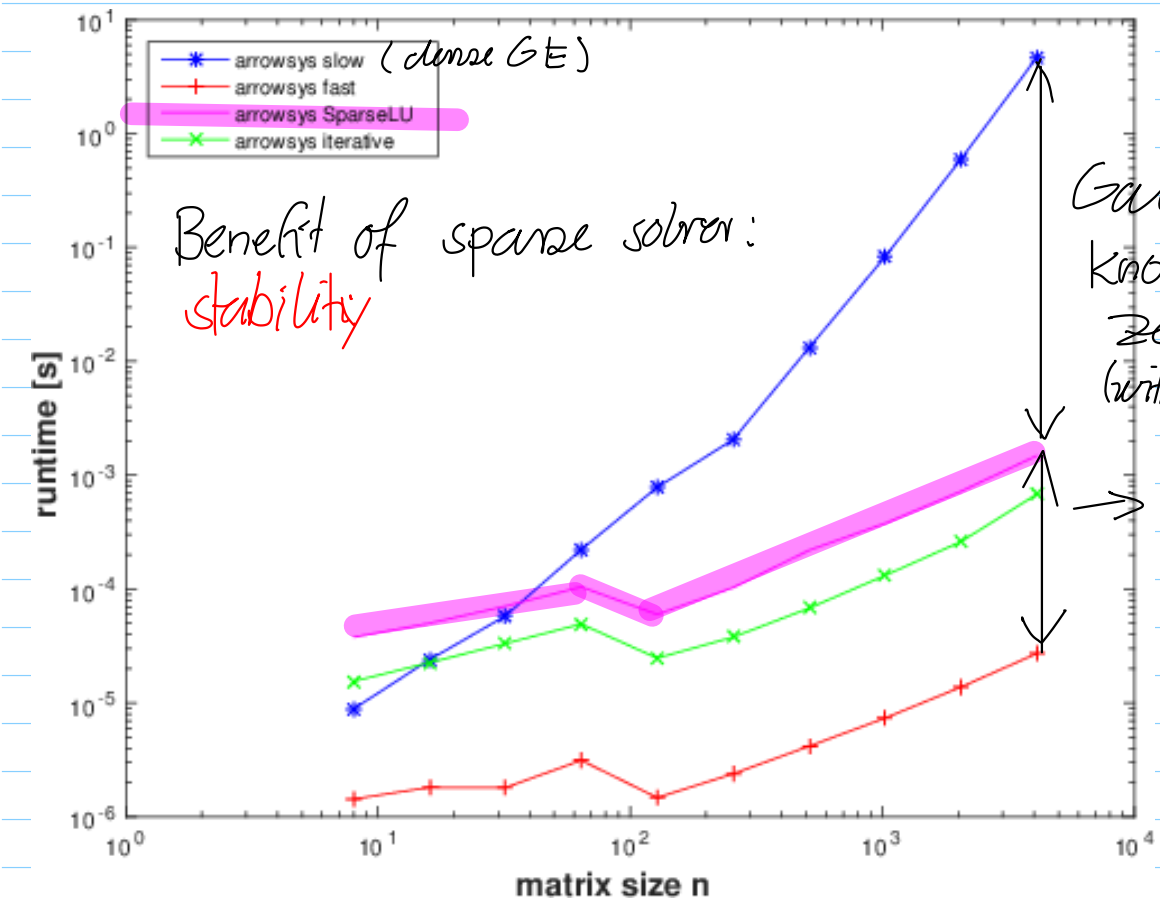
→ Check this → the expensive part
 → triangular solves: cheap

②

Ex 2.7.3.2: Sparse elimination for arrow matrix



← sparse arrow matrix



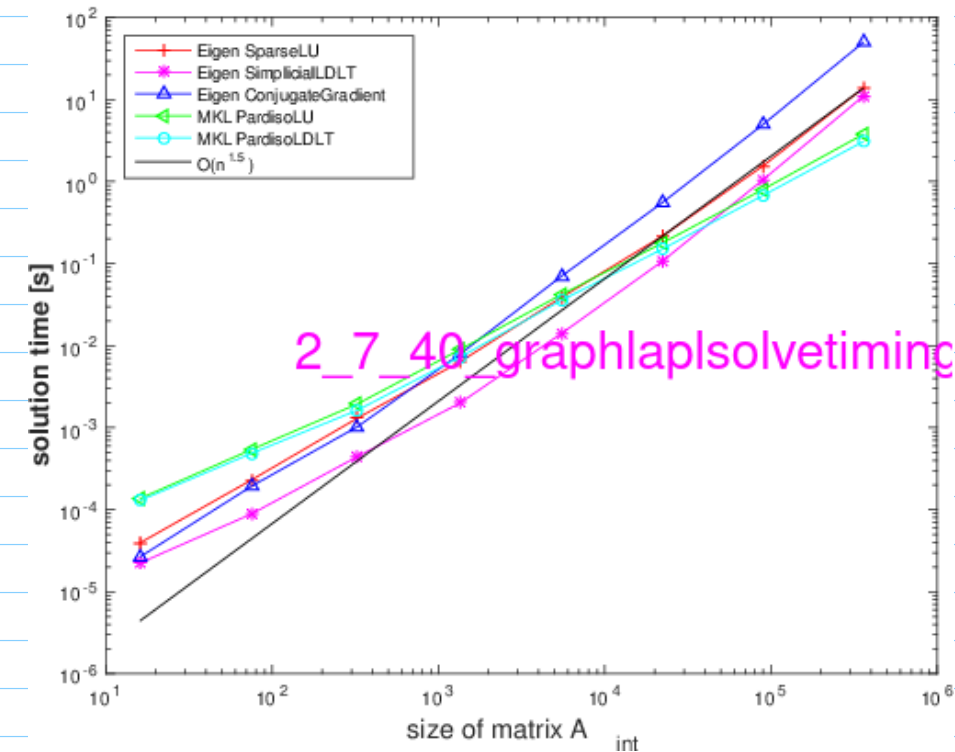
Benefit of sparse solver:
stability

Gain by exploiting
knowledge about
zero entries
(without loss of quality)

→ Cast for
• matrix analysis
• pivoting

Exp. 2.7.3.4.: Sparse elimination for combinatorial graph Laplacian

↳ sparse s.p.d., a few non-zero-entries per row/col.



Asymptotic runtime
 $O(n^{1.5})$

2_7_40 graphlapi solvetiming

In practice: Cast (sparse solution of $Ax = b$)
 $= O((nnz(A))^2) \approx 7.5-2.5$

When solving linear systems of equations directly **dedicated sparse elimination solvers** from numerical libraries have to be used!

System matrices are passed to these algorithms in sparse storage formats (→ Section 2.7.1) to convey information about zero entries.

③

4

5

7