Sparse Linear System Solvers Comparison

Ferri Marco - 807130
Habbash Nassim - 808292

Università degli Studi di Milano-Bicocca

Course: *Methods of Scientific Computation*

1. INTRODUCTION

PERFORMANCE ANALYSIS OF SPARSE LINEAR SYSTEMS SOLVERS USING **CHOLESKY DECOMPOSITION**

- **Operative Systems:**
  - Linux
  - Windows
- **Libraries/Programming Environments:**
  - MATLAB
  - Eigen (C++)
  - Scikit-Sparse (Python)

2. LIBRARIES

MATLAB

MATLAB is a programming platform designed for matrix-based computation, allowing for easy and efficient implementations of computational mathematics.

MATLAB is well documented, supported and has a big community.

MATLAB is not open-source and specially not free: the cost of the license varies wildly depending on the usage. Outside of academia, for individuals, it ranges from 800€ per year to a 2000€ one-time purchase.

MATLAB is easy to learn and use, but is also a whole programming environment, making it less portable than other choices. Its code is translatable automatically to C++, enabling an ample range of hardware support.

MATLAB

MATLAB has built-in capabilities to solve sparse linear systems using the Cholesky factorization.

As described here, whenever x = A \ B (mldivide operator) is computed, MATLAB chooses internally the optimal solver based on matrices shapes and symmetries, minimizing computation time.

For sparse matrices, Cholesky factorization is used when the matrix is square, not diagonal, is not triangular nor permuted triangular and is Hermitian.

Eigen (C++)

Eigen is an open-source C++ library for linear algebra, containing various related utilities and algorithms, ranging from fast matrices operations to computational geometry.

Eigen presents an extensive documentation, stable release cycles and is used by a plethora of projects (e.g. Google's Tensorflow).

Eigen, being a header-only library, is quite easy to include in a project, and does not require additional dependencies - although it is possible to extend it, more on it in the conclusions.

Eigen (C++)

Eigen implements different methods for solving and factoring sparse linear systems. Depending on the properties of the matrix, one method might be preferred over another.

Optimization choices are left over to the user, which might require a certain degree of knowledge over the matter.

The solver used for this project is SimplicialLDLT, which provides a LDL^T Cholesky factorizations without square root of sparse matrices that are self-adjoint and positive definite.

In order to reduce the fill-in, a symmetric permutation P is applied prior to the factorization.

Scikit-Sparse (Python)

Scikit-Sparse is an open-source Scipy.Sparse companion library for sparse matrix manipulation which extends the Scipy library with routines not included due to conflicting licenses, essentially acting as a Python wrapper for CHOLMOD.

Scikit-Sparse is documented (although is lacking in the descriptions of the methods used), but is no longer being actively worked on, making it a less suitable choice than the others.

On most Linux distributions the already-compiled binaries are offered in the distro's repository. For Windows, the compilation of SuiteSparse/Cholmod or use of some dependency manager is necessary.

Scikit-Sparse (Python)

Scikit-Sparse allows for the computation of both LDL^T and LL^T Cholesky factorizations.

It also offers the ability to perform the costly fill-reduction analysis once, and then re-use it to efficiently decompose many matrices with the same pattern of non-zero entries.

3.
COMPARISON TESTS

OPERATIONS

All tests are conducted by computing the equation Ax = b, where b is chosen for having the exact solution composed as xe = [1 1 1 1 1...].

In other words, b = A * xe.

SAMPLES

All input sample are positive-definite and symmetric matrices that come from the SuiteSparse Matrix Collection.

In particular, the following matrices have been used:

- ex15
- shallow_water1
- cfd1
- cfd2
- parabolic_fem
- apache2
- StocF-1465
- Flan_1565
- G3_circuit

PERFORMANCES

For each library and platform, three parameters have been measured for the comparison:

- Resolution Time
- Used Memory
- Relative Error

ENVIRONMENTS

The tests have been run on Azure Virtual Machines.

The following are both the Windows and Linux virtual machines technical specifications:

- 8 VCPUs
- 56 GB of RAM

We used Ubuntu 18.04 and Windows 10 Pro 1803.

4.
SOURCE CODE

Also available on *https://github.com/dodicin/slss-comparison-test*

MATLAB

```
load(inputfile);
[rows, columns] = size(Problem.A);

xe = ones(columns, 1);
b = Problem.A * xe;
x = Problem.A \ b;

error = norm(x - xe) / norm(xe);
```

Eigen (C++)

```
typedef SparseMatrix<double, ColMajor, long long> SpMat;

SpMat A;
loadMarket(A, argv[1]);
VectorXd xe = VectorXd::Constant(A.cols(), 1);
MatrixXd b = A.selfadjointView<Lower>() * xe;

SimplicialLDLT<SpMat> chol(A.selfadjointView<Lower>());
VectorXd x = chol.solve(b);
double rel_error = ((VectorXd)(x - xe)).norm()/xe.norm();
```

Scikit-Sparse (Python)

```
mat = io.mmread(sys.argv[1])

A = sp.sparse.csc_matrix(mat)
xe = np.ones(A.shape[0])
b = A.dot(xe)

factor = cholesky(A)
x = factor.solve_A(b)

rel_error = np.linalg.norm(x-xe)/np.linalg.norm(xe)
```

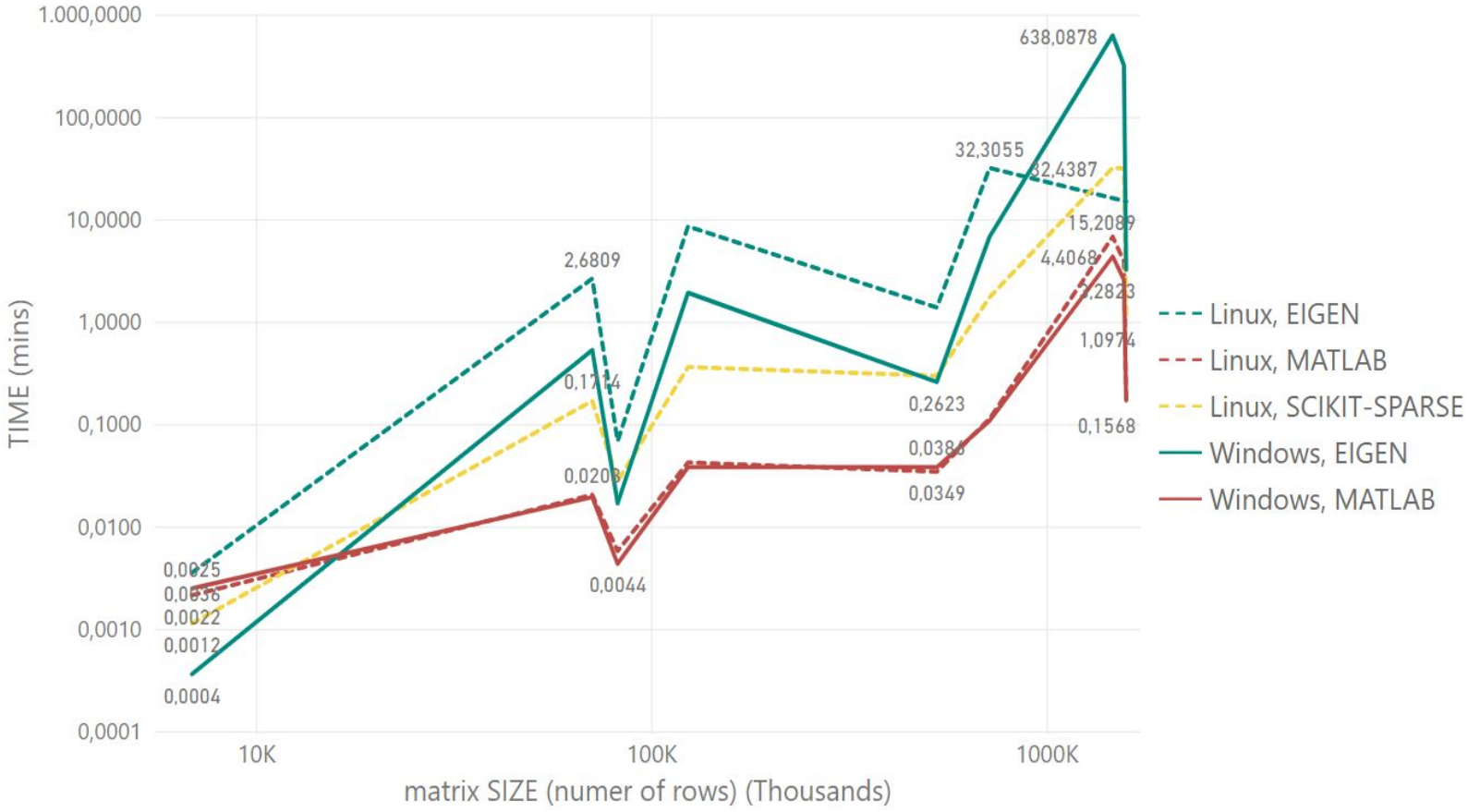
5.
RESULTS

SOME NOTES

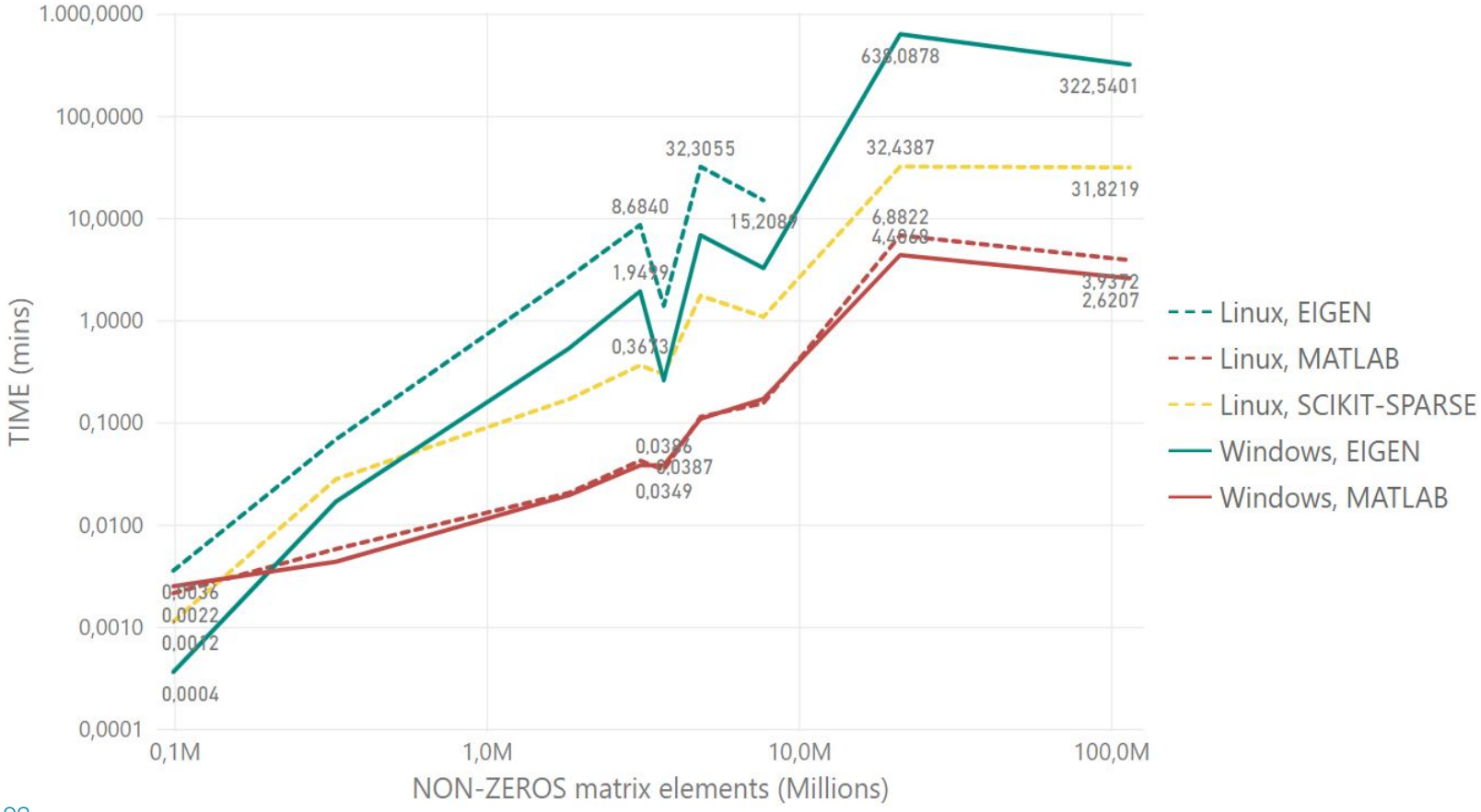
MATLAB and Eigen scripts have been ran on both Windows and Linux, while Scikit-Sparse was only tested on Linux, due to arising issues with its installation on Windows. Specifically, obtaining the SuiteSparse library and compiling it to run it with Python has failed.

Moreover, not all matrices managed to be factorized and computed (the two largest matrices, StocF-1465 and Flan_1565 failed with Eigen).

RESOLUTION TIME BY MATRIX SIZE



RESOLUTION TIME BY MATRIX NON-ZEROS ELEMENTS

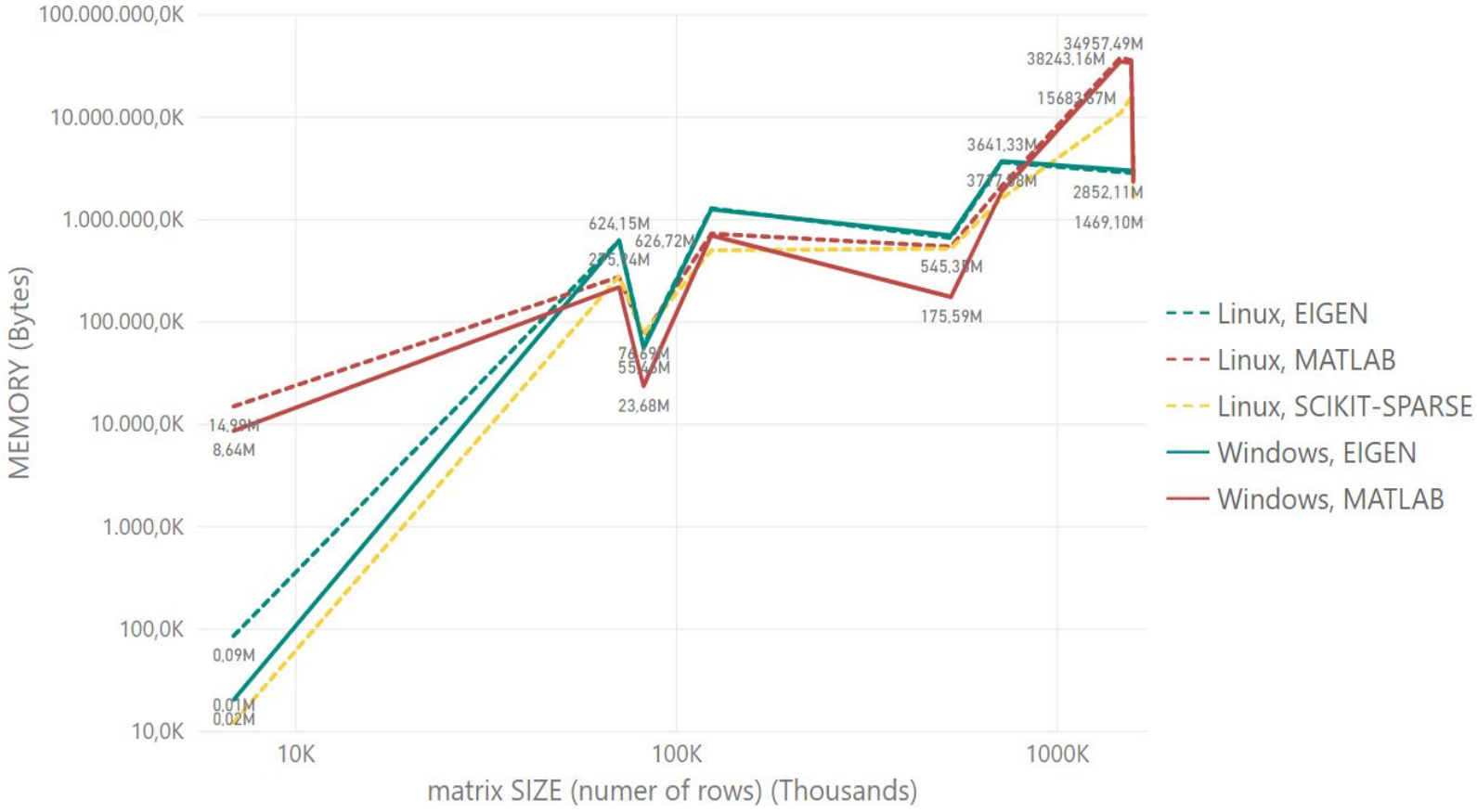


RESOLUTION TIME CONSIDERATIONS

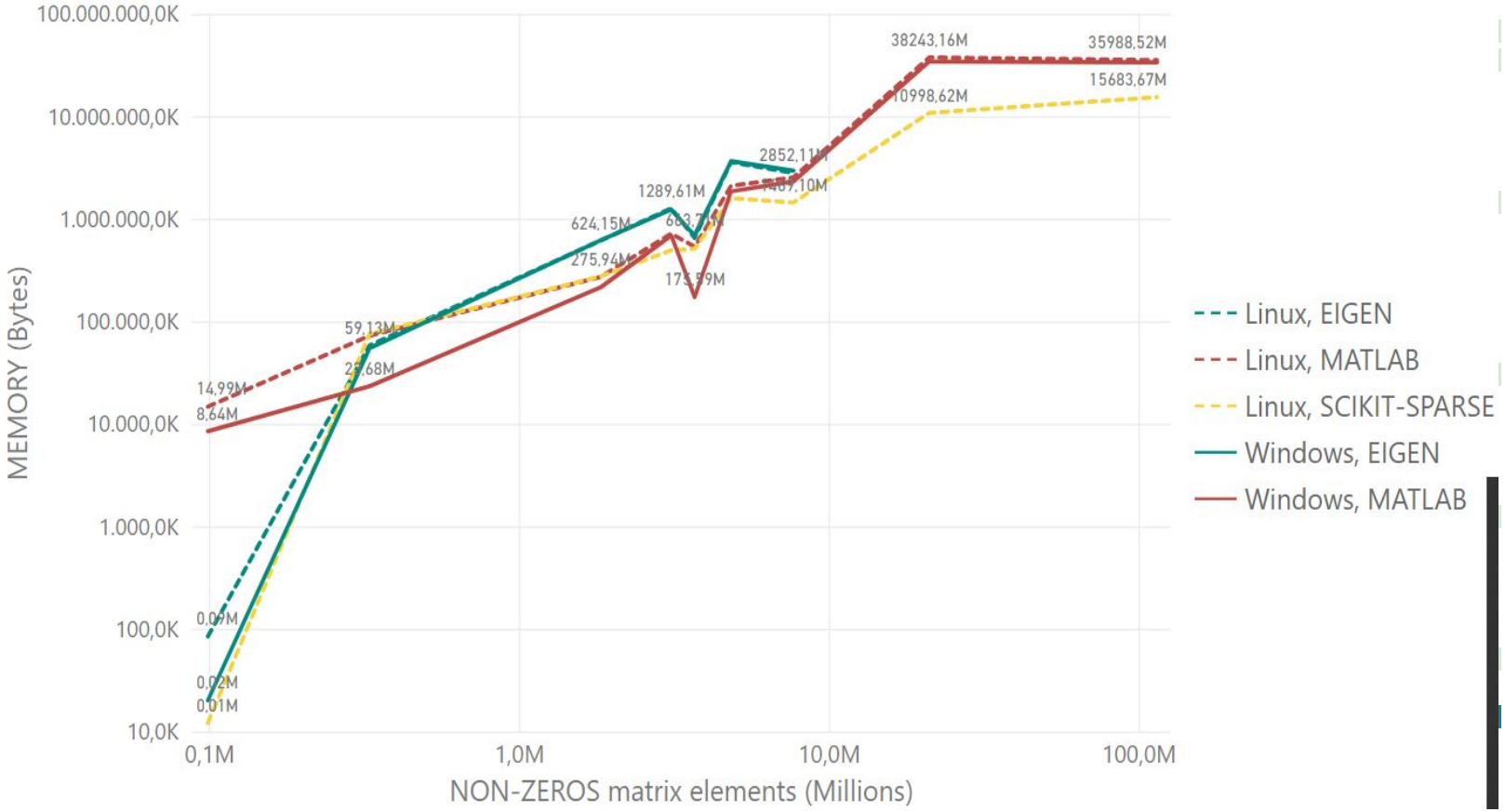
MATLAB is significantly faster than the others. It manages to solve the most complex matrix in just 6 minutes, versus 32 minutes for Scikit and more than 10 hours for Eigen.

Platform-wise, no notable difference has been detected between **Linux** and **Windows**.

MEMORY BY MATRIX SIZE



MEMORY BY MATRIX NON-ZEROS ELEMENTS



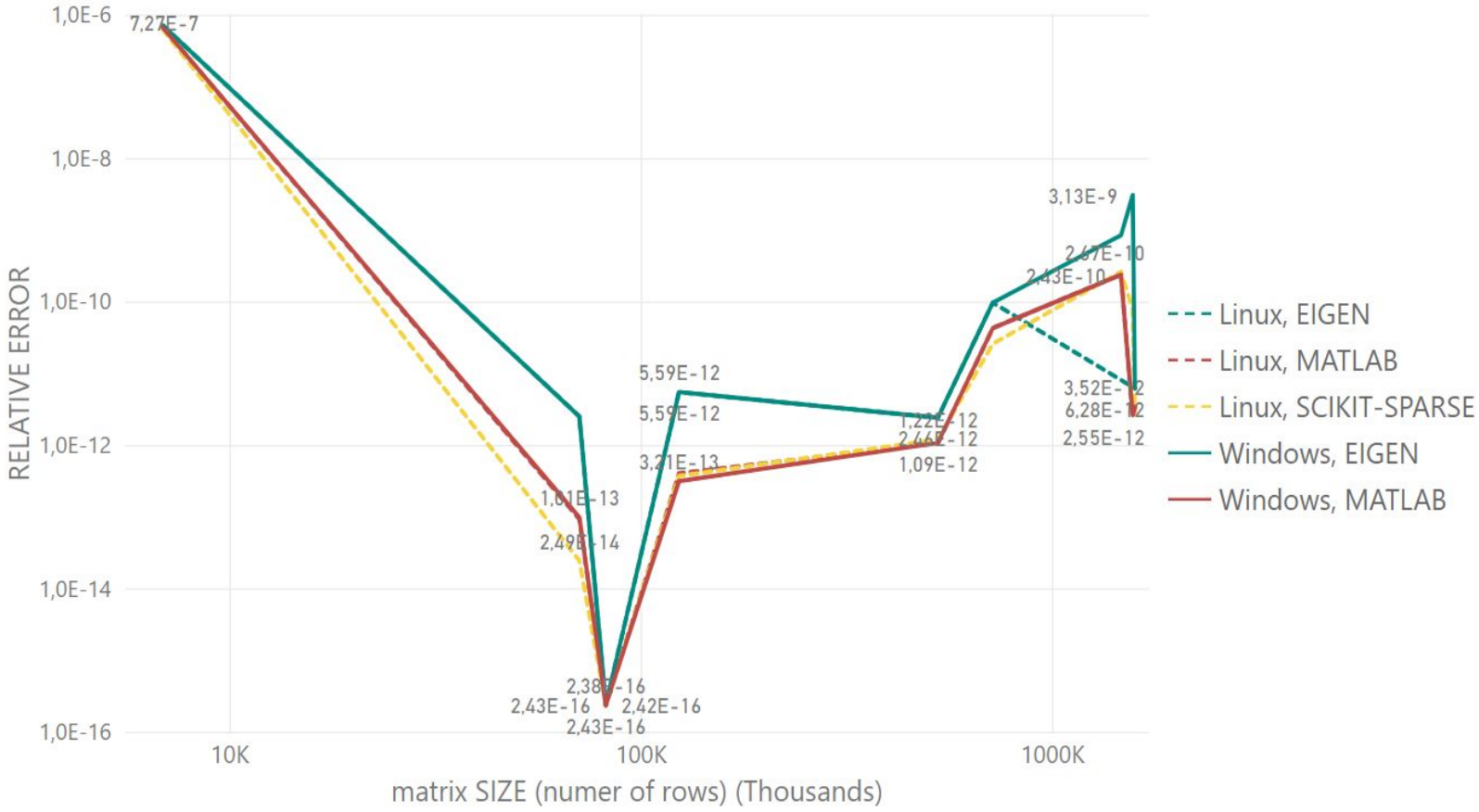
MEMORY CONSIDERATIONS

Memory usage behaves similarly between all the computed solvers, until matrices with more than 1 million non-zeros elements are reached.

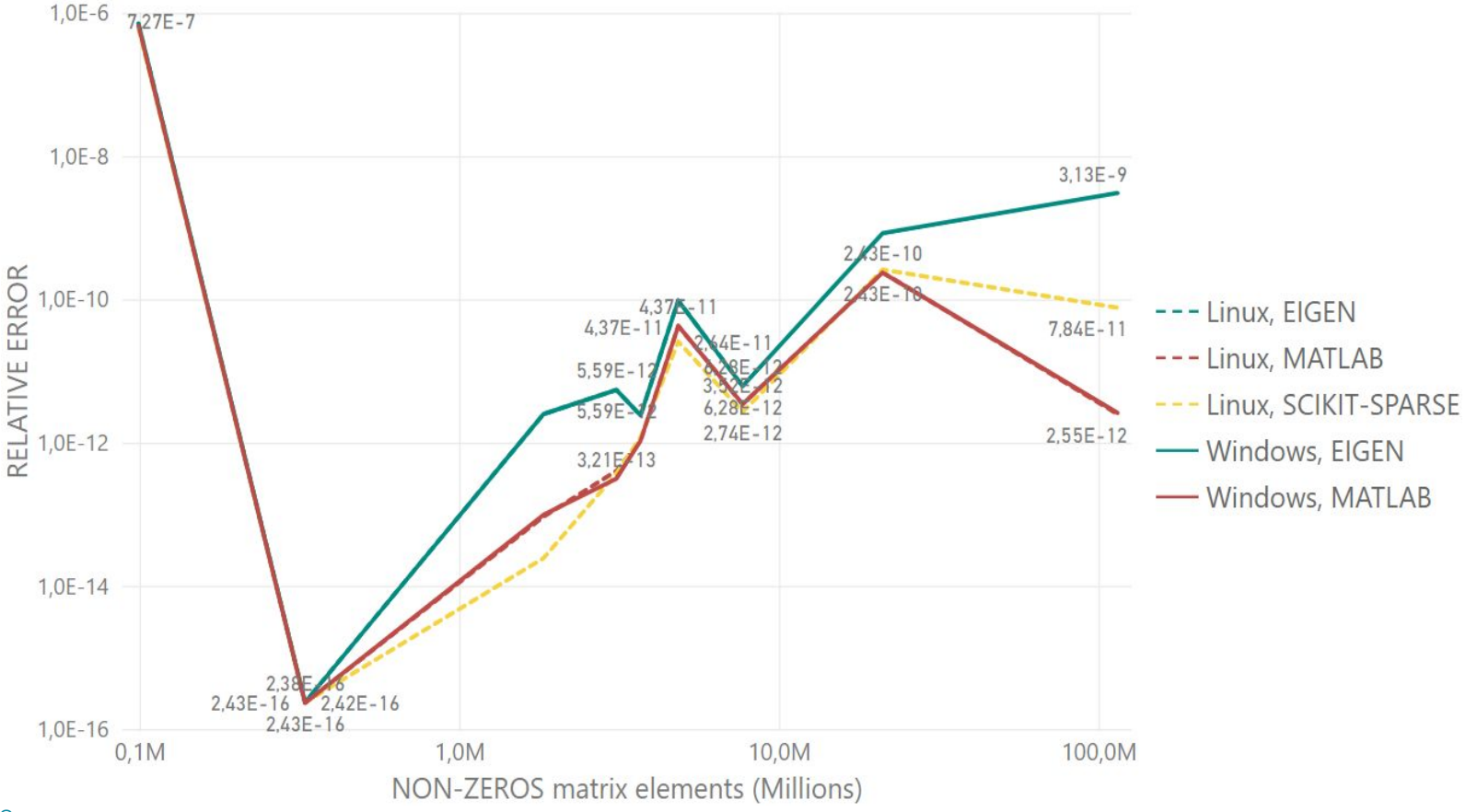
Over that threshold, **Scikit uses 4 times less memory than MATLAB**.

Unfortunately, we haven't been able to retrieve results for Eigen for bigger matrices, but the trend from previous data entry-points seemed to place it above the others in term of memory consumption.

RELATIVE ERROR BY MATRIX SIZE



RELATIVE ERROR BY MATRIX NON-ZEROS ELEMENTS



RELATIVE ERROR CONSIDERATIONS

Regarding relative error, each algorithm behaves similarly, with a maximum difference of 1-2 orders of magnitude between any set of relative error tests.

A relevant difference can be noticed in the last matrix, where Eigen and MATLAB's relative errors differ by 3 orders of magnitude; the latter is overall the most accurate among tested libraries.

6. CONCLUSIONS

CONCLUSIONS

As we've seen in the previous slides, **MATLAB seems to be the fastest tool** for solving linear sparse systems with Cholesky Decomposition. However, on large matrices it needs a significant amount of memory to work properly.

It also produces the minimum relative error compared with the other ones.

CONCLUSIONS

Scikit-Sparse didn't fare bad in the conducted tests. Although five times slower than MATLAB, it uses just one third of its memory.

As such, Scikit-Sparse might be a good pick for the resolution of smaller matrices, which can still be computed in a reasonable time.

CONCLUSIONS

Eigen fared the worst in terms of time, memory and relative error.

A note on its usage: Eigen has been used in this project as a barebone library, with no extensions. Being popular and widely supported, it might be interesting exploring the usage of BLAS libraries, such as Intel MLK, as a back-end to Eigen, or CHOLMOD's patch-in. Time consumption might improve significantly due to their low-level optimization.

THANKS!

Any questions?

You can find us at:

m.ferri17@campus.unimib.it

n.habbash@campus.unimib.it

https://github.com/dodicin/slss-comparison-test