Thematic Conference

ECCOMAS

Multibody Dynamics 2013
01-04 July, Zagreb, Croatia

A comparison of Algorithms for Sparse Matrix Factoring and Variable Reordering aimed at Real-time Multibody Dynamic Simulation

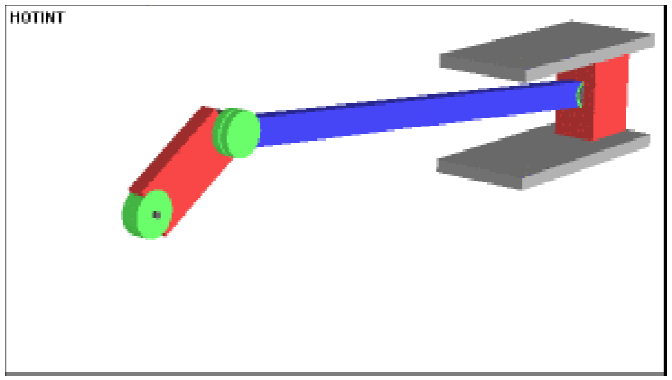
Jose-Luis Torres-Moreno,
Jose-Luis Blanco,
Javier López-Martínez,
Antonio Giménez-Fernández

1-4 July, 2013
Zagreb (Croatia) – ECCOMAS Multibody Dynamics 2013

UNIVERSIDAD DE ALMERÍA

Goal of this work

Problem
under study



Multibody dynamic
formulation:

$$\left[\begin{array}{c|c} \mathbf{M} & \mathbf{\Phi_q}^{\top} \\ \hline \mathbf{\Phi_q} & \mathbf{0} \end{array}\right] \left[\begin{array}{c} \ddot{\mathbf{q}} \\ \boldsymbol{\lambda} \end{array}\right] = \left[\begin{array}{c} \mathbf{Q} \\ \mathbf{c} \end{array}\right]$$

$$\mathbf{Ax} = \mathbf{b} \rightarrow \mathbf{x} = \mathbf{A} \backslash \mathbf{b}$$

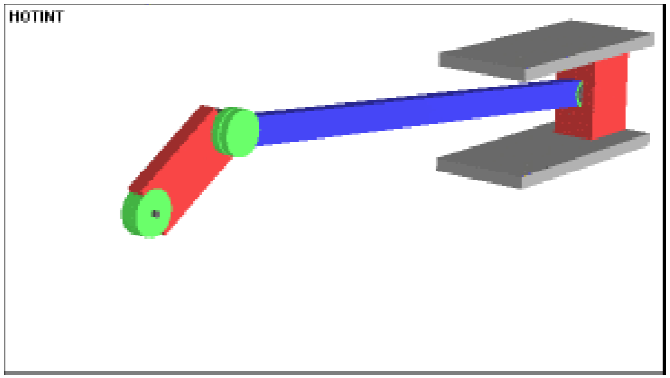
Best numerical solver algorithm?

➔ It depends…

Talk outline

1. **Problem introduction**
2. **Tested algorithms**
3. **Benchmark**
4. **Results**

Problem introduction



Modeling decisions:
coordinates, formulation, etc.

$$\ddot{\mathbf{q}}(t) = \mathbf{f}\left(\mathbf{q}(t), \dot{\mathbf{q}}(t), t\right)$$

$$\ddot{\mathbf{z}}(t) = \mathbf{f}\left(\mathbf{z}(t), \dot{\mathbf{z}}(t), t\right)$$

Our choices for this work:

Dependent coordinates vs. Independent coordinates

Coordinate type for modeling: Natural coordinates

Dynamic formulation: Index-1 Lagrange w. Baumgarte

$$\left[\begin{array}{c|c} \mathbf{M} & \boldsymbol{\Phi}_{\mathbf{q}}{}^{\top} \\ \hline \boldsymbol{\Phi}_{\mathbf{q}} & \mathbf{0} \end{array}\right] \left[\begin{array}{c} \ddot{\mathbf{q}} \\ \boldsymbol{\lambda} \end{array}\right] = \left[\begin{array}{c} \mathbf{Q} \\ \mathbf{c} \end{array}\right]$$

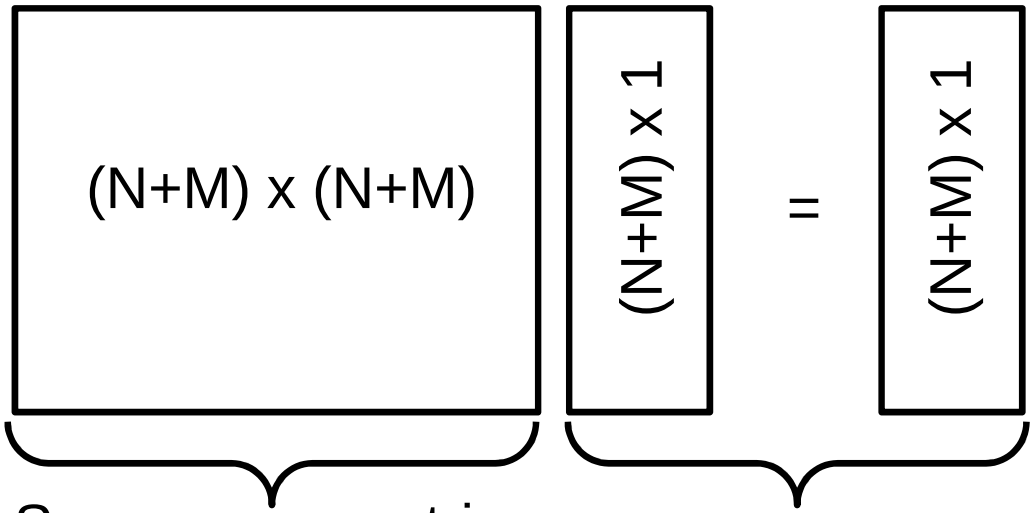
Problem introduction

$$\left[\begin{array}{c|c} \mathbf{M} & \mathbf{\Phi_q}^\top \\ \hline \mathbf{\Phi_q} & \mathbf{0} \end{array}\right] \left[\begin{array}{c} \ddot{\mathbf{q}} \\ \boldsymbol{\lambda} \end{array}\right] = \left[\begin{array}{c} \mathbf{Q} \\ \mathbf{c} \end{array}\right]$$

A **x** = **b**

Problem structure?

- *N* dependent coordinates
- *M* constraints

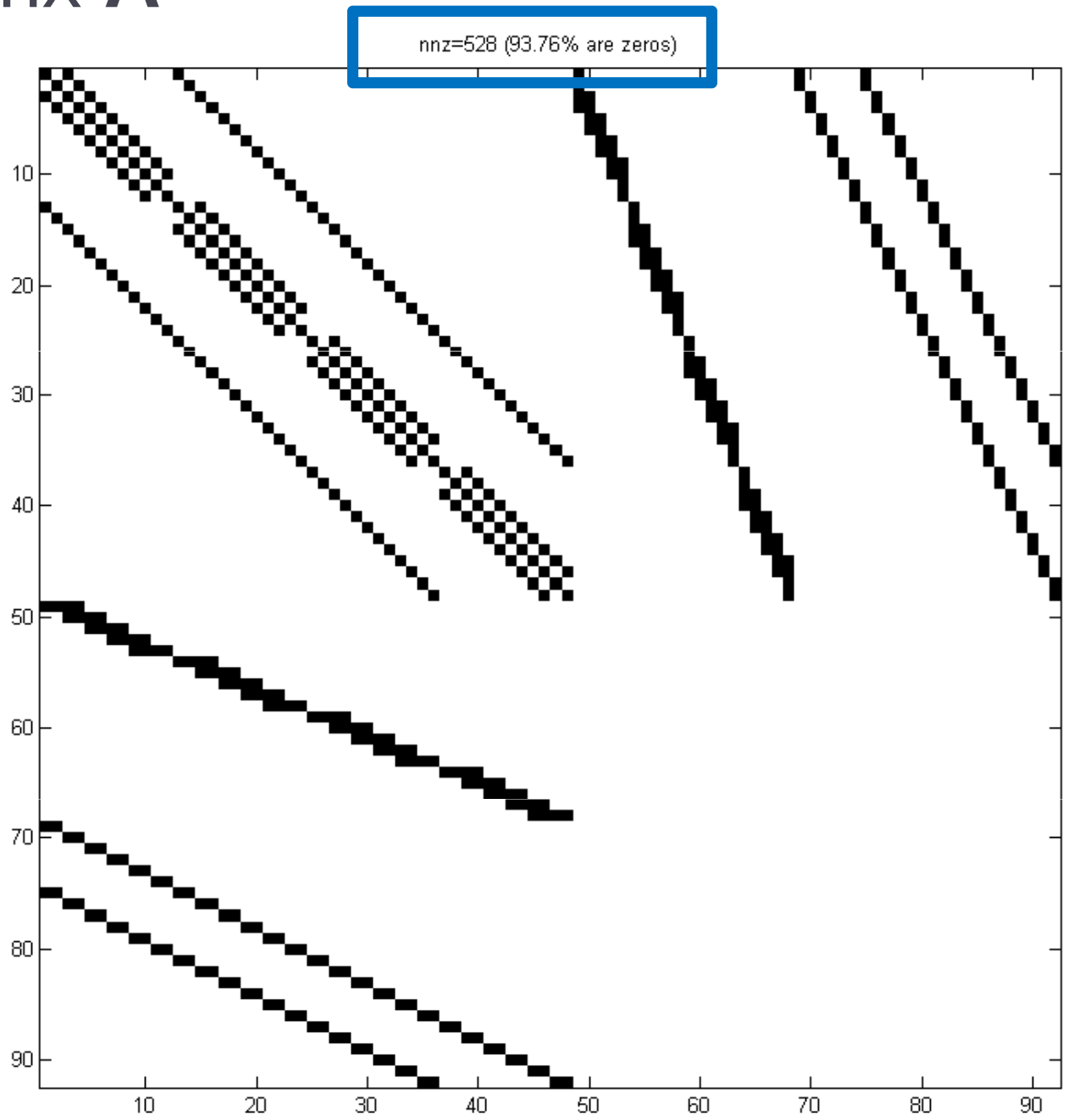


Structure of matrix **A**

$$\underbrace{\left[\begin{array}{c|c} \mathbf{M} & \mathbf{\Phi_q}^\top \\ \hline \mathbf{\Phi_q} & \mathbf{0} \end{array}\right]}_{\mathbf{A}}$$

Most important features

- M is constant
- Static non-zero structure
- **SPARSITY**



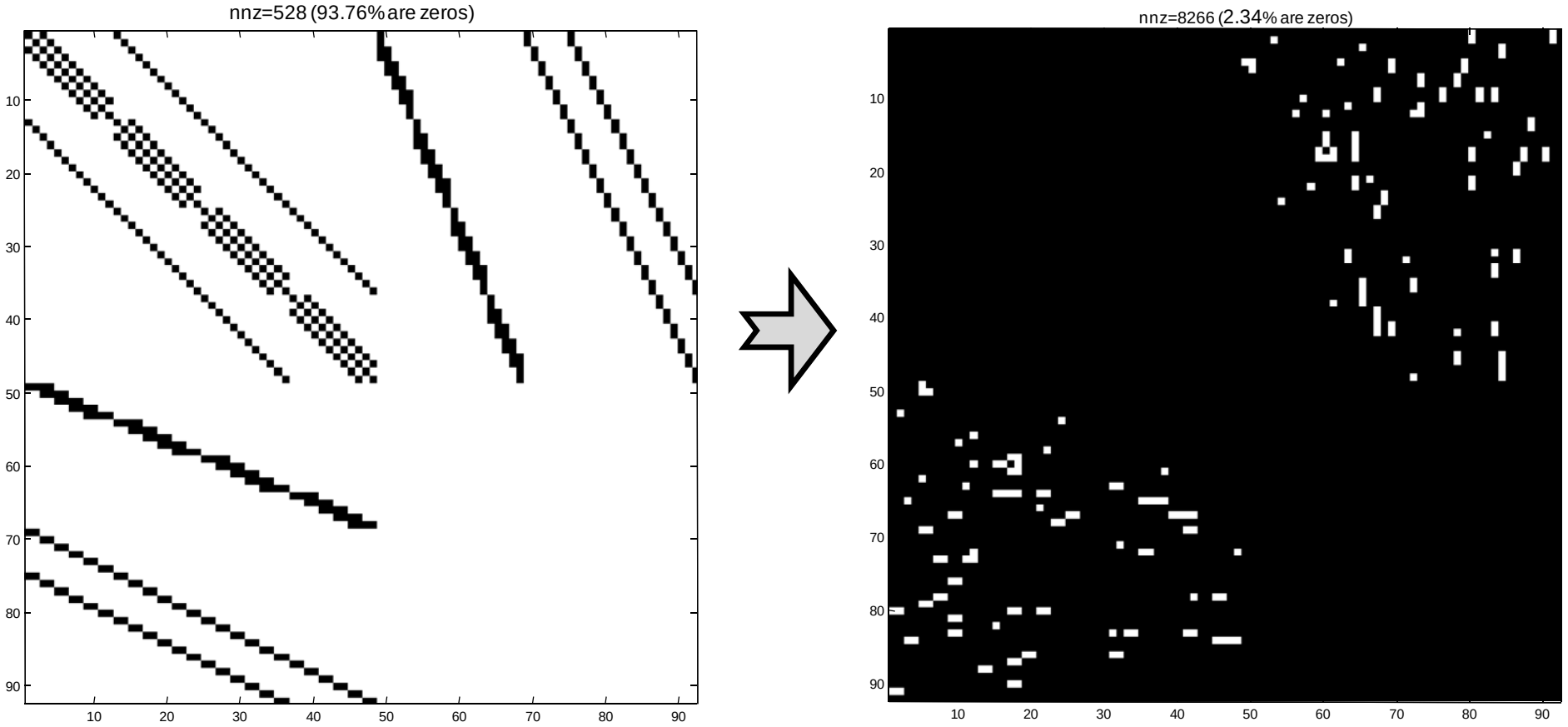
Why sparsity matters?

Naive matrix inversion:

$$\underbrace{\begin{bmatrix} \mathbf{M} & \phi_{\mathbf{q}}^{T} \\ \phi_{\mathbf{q}} & \mathbf{0} \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} \ddot{\mathbf{q}} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{Q} \\ \mathbf{c} \end{bmatrix}$$

$$\begin{bmatrix} \ddot{\mathbf{q}} \\ \lambda \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{M} & \phi_{\mathbf{q}}^{T} \\ \phi_{\mathbf{q}} & \mathbf{0} \end{bmatrix}^{-1}}_{\mathbf{A}^{-1}} \begin{bmatrix} \mathbf{Q} \\ \mathbf{c} \end{bmatrix}$$

Why sparsity matters?



$$\begin{bmatrix} \mathbf{M} & \phi_{\mathbf{q}}^{T} \\ \phi_{\mathbf{q}} & \mathbf{0} \end{bmatrix} \Rightarrow \begin{bmatrix} \mathbf{M} & \phi_{\mathbf{q}}^{T} \\ \phi_{\mathbf{q}} & \mathbf{0} \end{bmatrix}^{-1}$$

In general, the inverse of a sparse matrix is NOT sparse!

Cost: $O\left(\left(N+M\right)^{3}\right)$

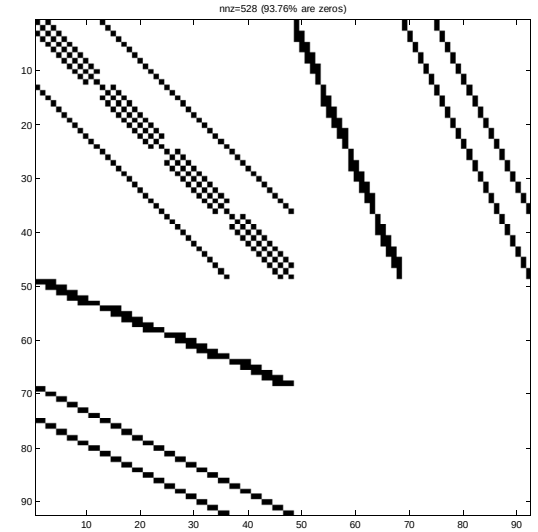
Why sparsity matters?

How to keep matrices sparse? → Avoid direct matrix inversion.

Alternatives? → Sparse matrix factorization algorithms:

- 1) **Method**: LU, Cholesky.
- 2) **Variable ordering**

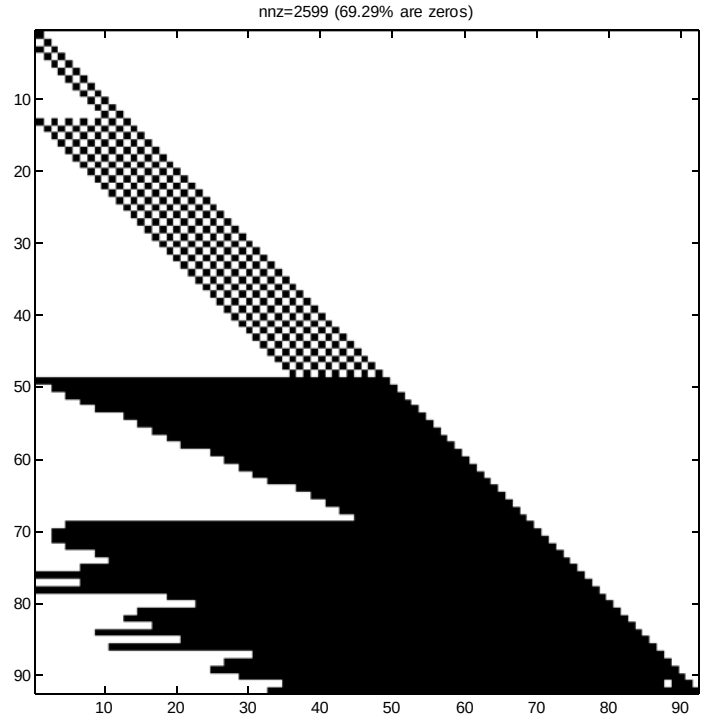
Why variable reordering matters?

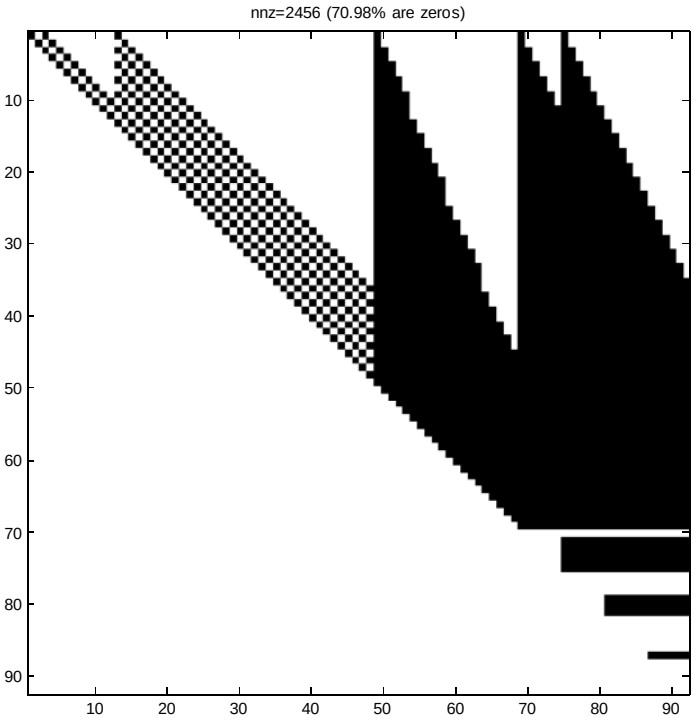
$\mathbf{A} =$ 

~ 94% zeros

“Natural” ordering

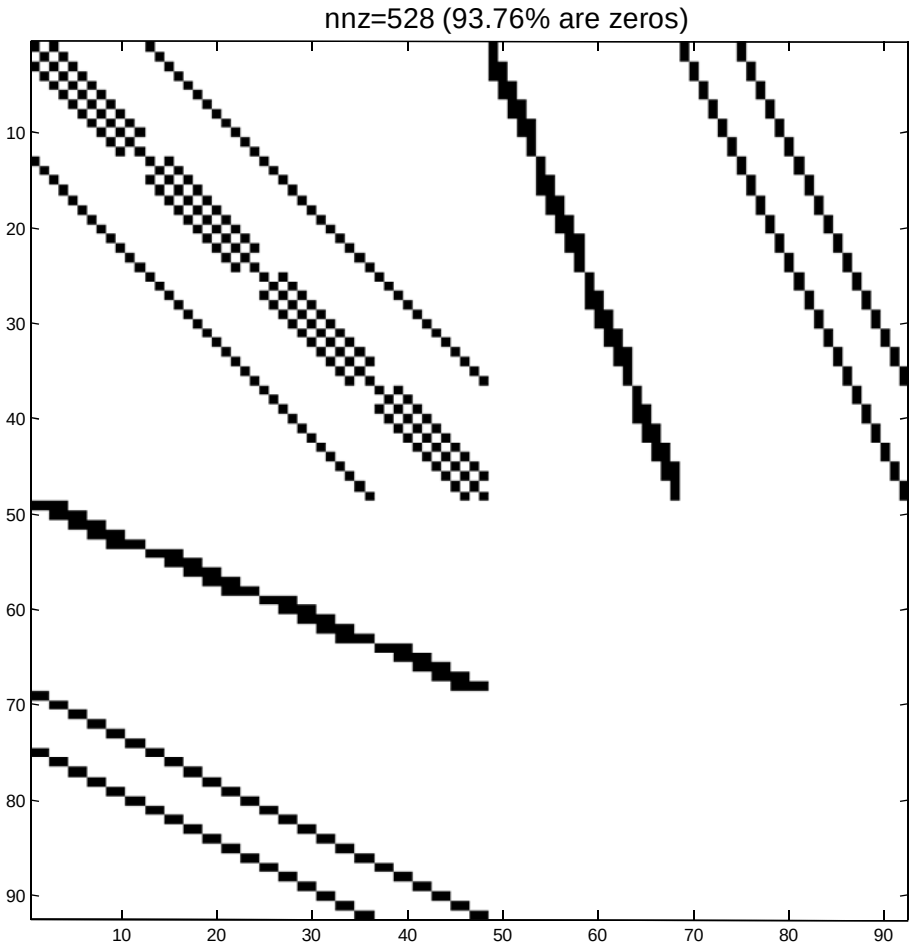
~ 70% zeros

$\mathbf{A} = \mathbf{L}\,\mathbf{U} =$ 

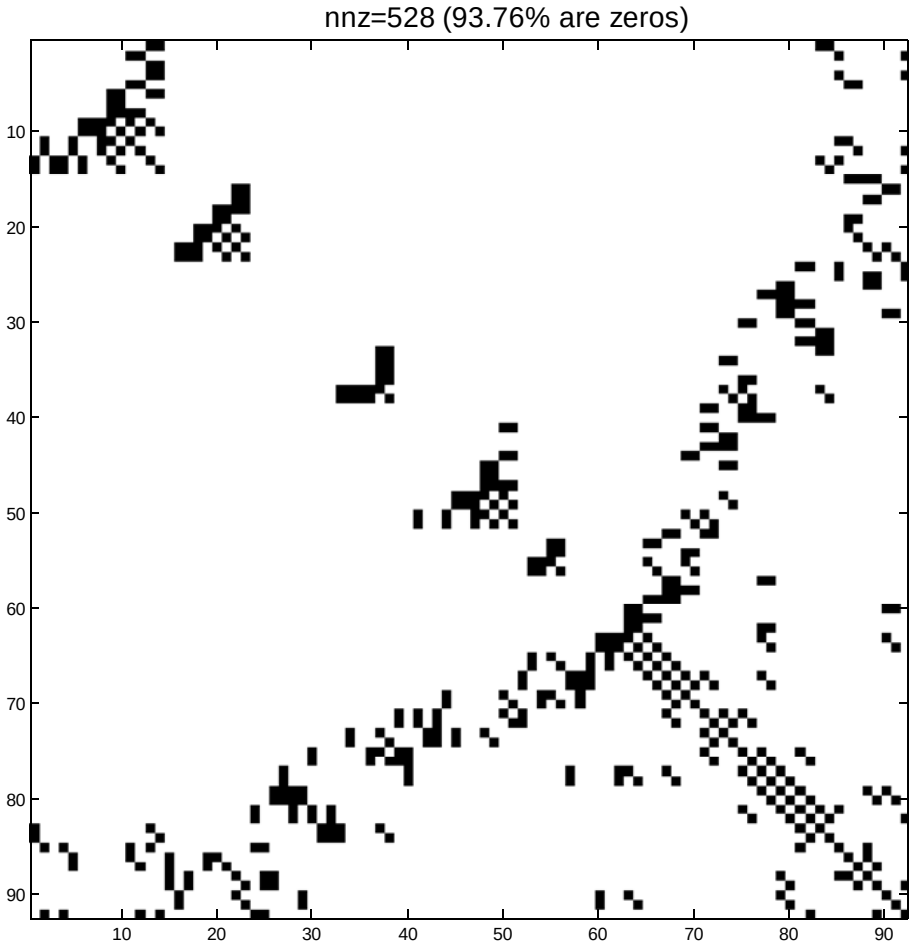


Why variable reordering matters?

Reordering = permutation of variable indices

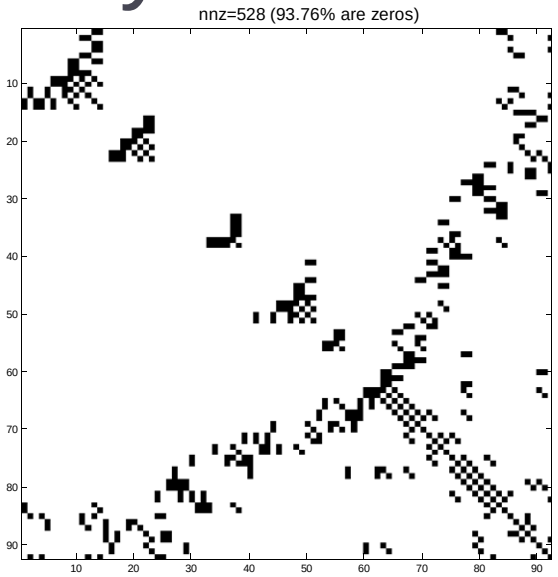


$\mathbf{A}(:,:)$



$\mathbf{A}(\mathbf{p},\mathbf{p})$

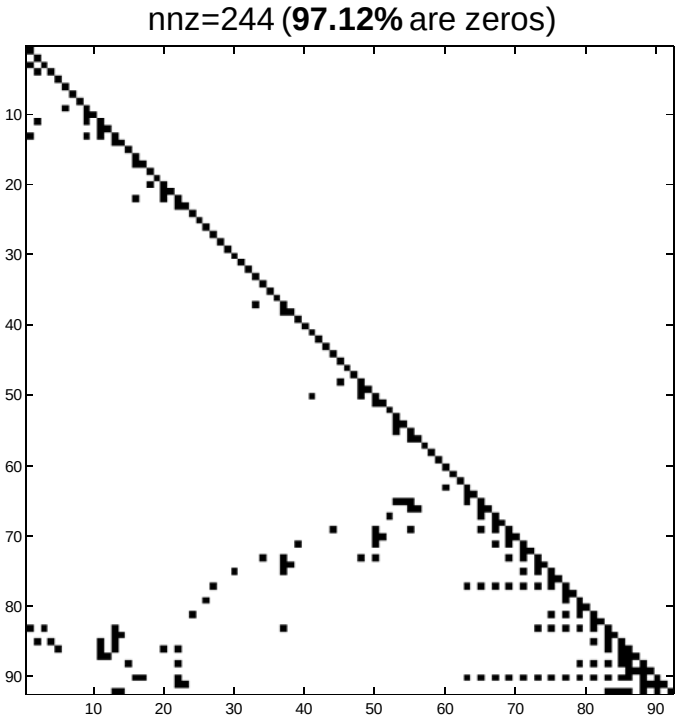
Why variable reordering matters?

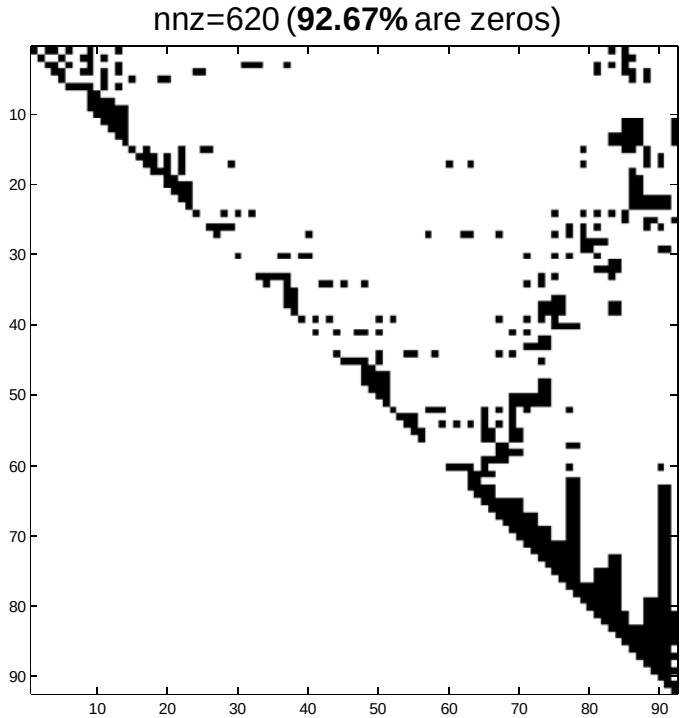


~ 94% zeros

Optimized ordering

~97% and ~93% zeros





Talk outline

1. Problem introduction
2. **Tested algorithms**
3. Benchmark
4. Results

Tested algorithms

Approaches:

- LU decomposition
- Cholesky decomposition

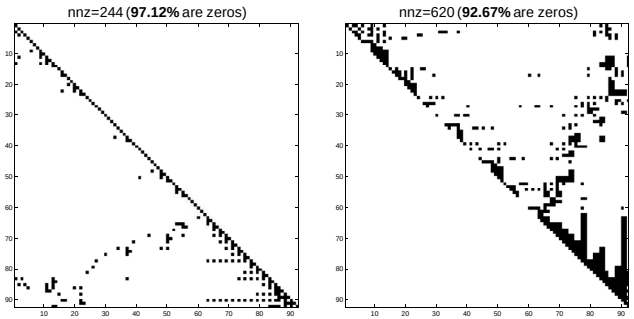
Tested algorithms

Approaches:

- **LU decomposition**
- Cholesky decomposition

$$\mathbf{Ax} = \mathbf{b}$$

$$\overbrace{(\mathbf{LU})}^{\mathbf{A}}\mathbf{x} = \mathbf{b} \rightarrow \begin{cases} 1)\mathbf{L}\underbrace{\mathbf{Ux}}_{=\mathbf{y}} = \mathbf{L}\,\mathbf{y} = \mathbf{b} \quad \text{, solve for } \mathbf{y} \\ 2)\ \mathbf{Ux} = \mathbf{y} \quad \text{, solve for } \mathbf{x} \end{cases}$$



Tested algorithms

Benchmarked LU implementations (6 variations):

1) **Dense LU** – Eigen C++ library
2) **Sparse LU** – Suitesparse's **KLU**
   a) AMD ordering
   b) COLAMD ordering
3) **Sparse LU** – Suitesparse's **UMFPACK**
   a) Natural ordering
   b) AMD ordering
   c) METIS ordering

Symbolic vs. **numeric** factorization!

Tested algorithms

Approaches:

- LU decomposition
- **Cholesky decomposition** (only when M is definite positive)

$$\underbrace{\begin{bmatrix} \mathbf{M} & \phi_{\mathbf{q}}^{T} \\ \phi_{\mathbf{q}} & \mathbf{0} \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} \ddot{\mathbf{q}} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{Q} \\ \mathbf{c} \end{bmatrix}$$

- Apply Cholesky to M.
- Use Schür complement to build two reduced systems of equations.

[R. von Schwerin, *Mathematical Methods for MBS in Descriptor Form*, 1999]

Tested algorithms

Approaches:

- LU decomposition
- **Cholesky decomposition** (only when M is definite positive)

Benchmarked Cholesky implementations (3 variations):

1) Sparse Cholesky – Suitesparse's **CHOLMOD**
   a) AMD ordering
   b) COLAMD ordering
   c) METIS ordering

Talk outline

1. Problem introduction
2. Tested algorithms
3. **Benchmark**
4. Results

Our benchmark model

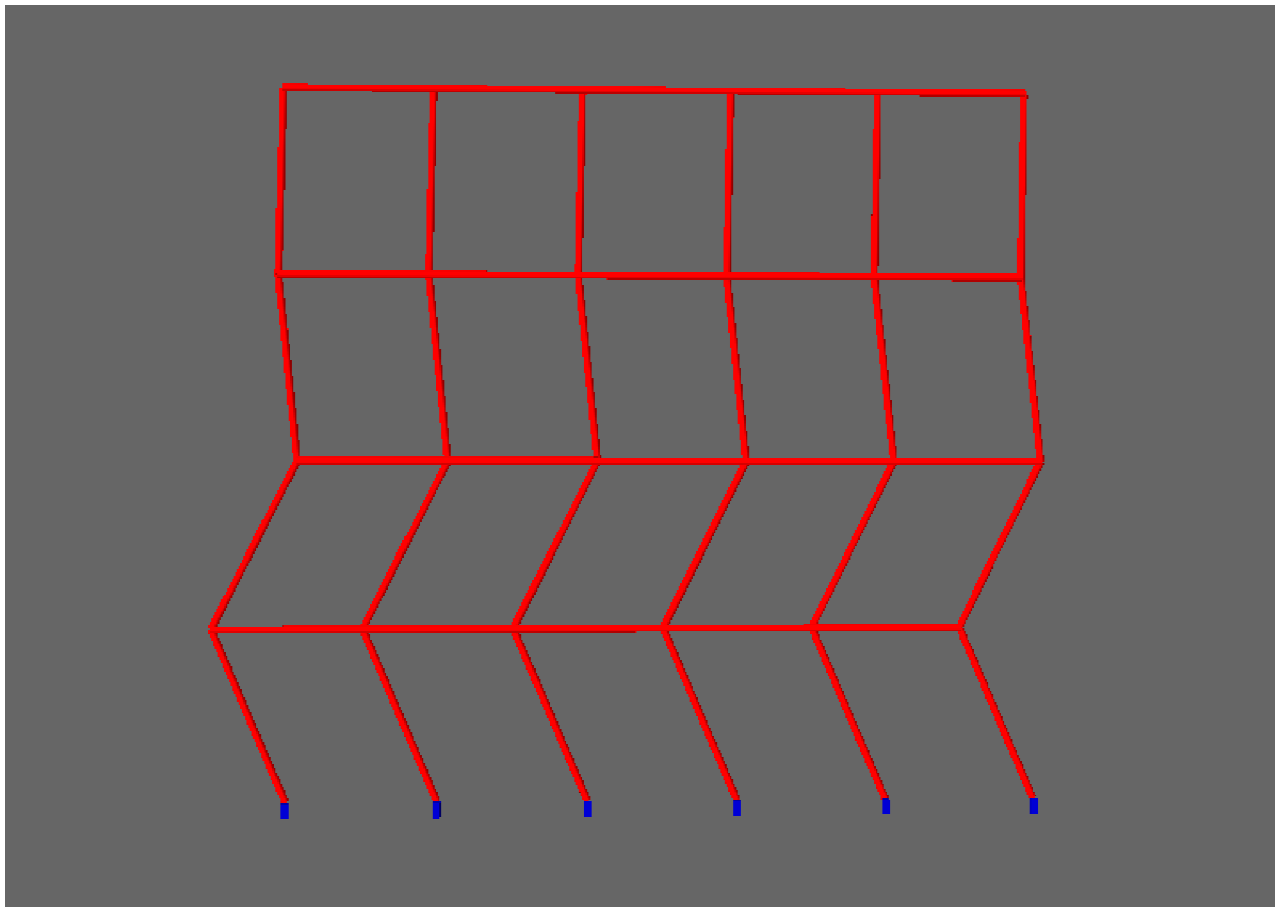
Test mechanism:

Parameterized 2D lattice of $N_x \times N_y$ articulated quadrangles.

Example:

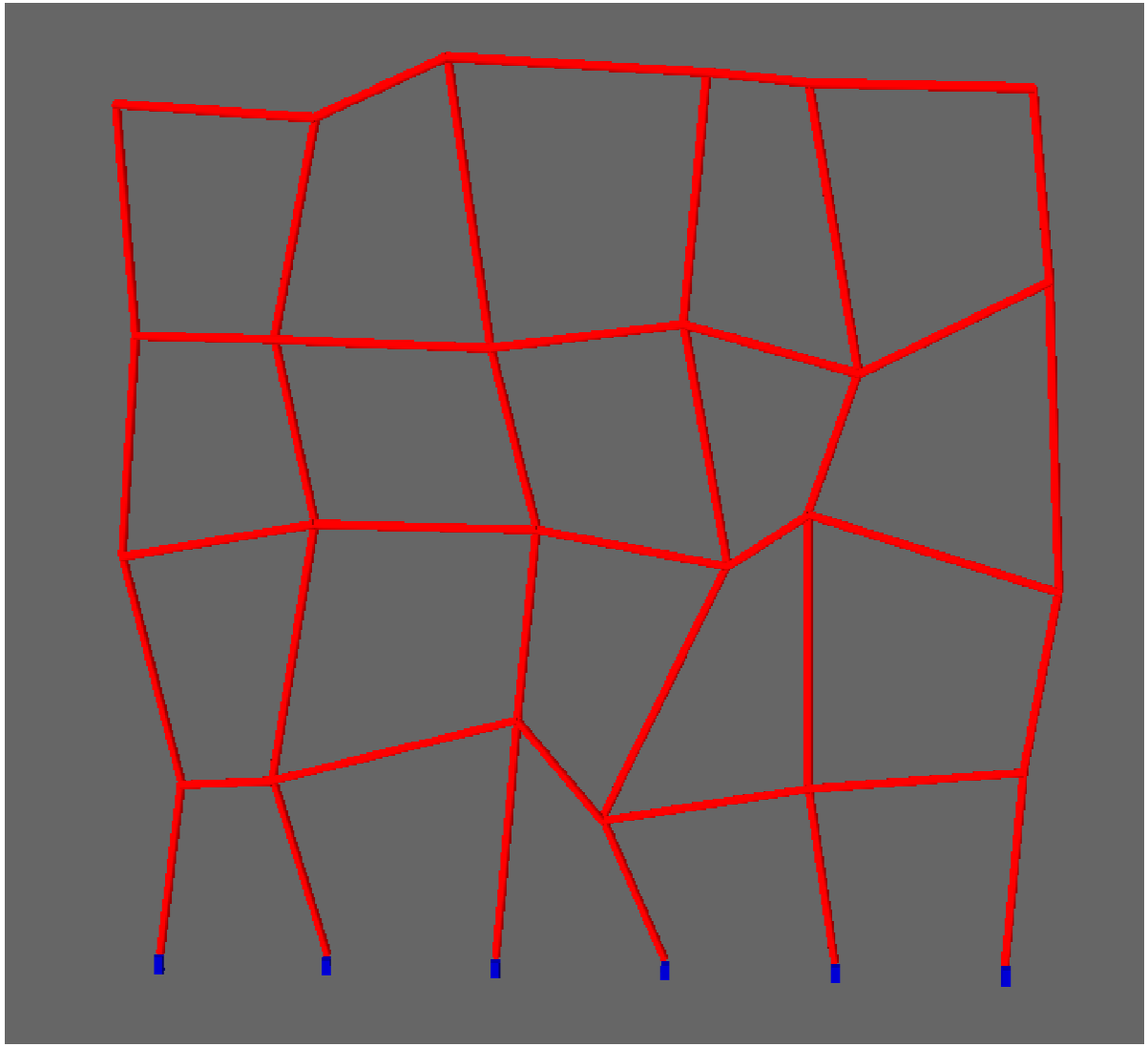
Nx = 5

Ny = 4

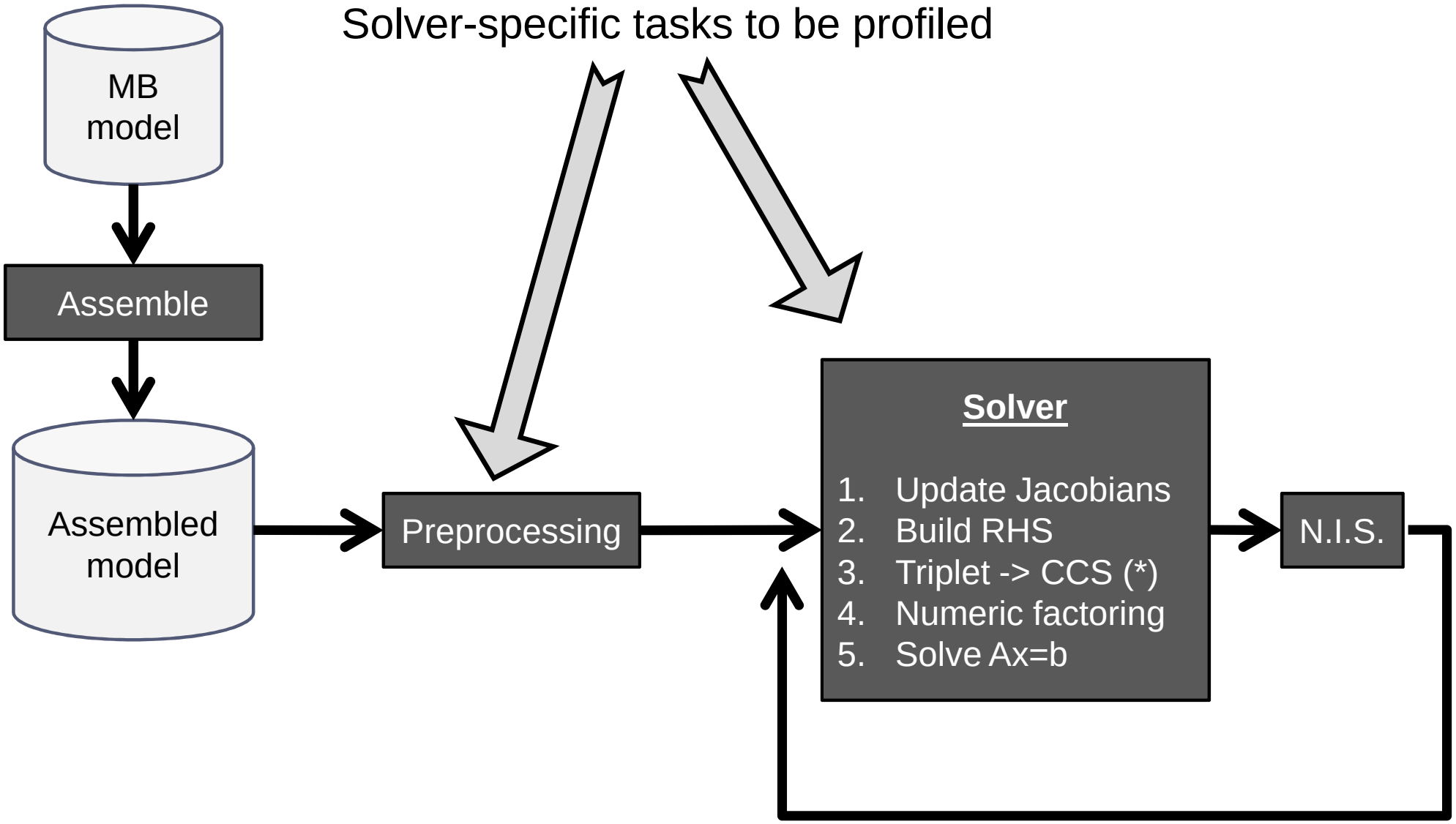


Our benchmark model

Irregular node locations to avoid singular configurations:



Benchmark C++ implementation

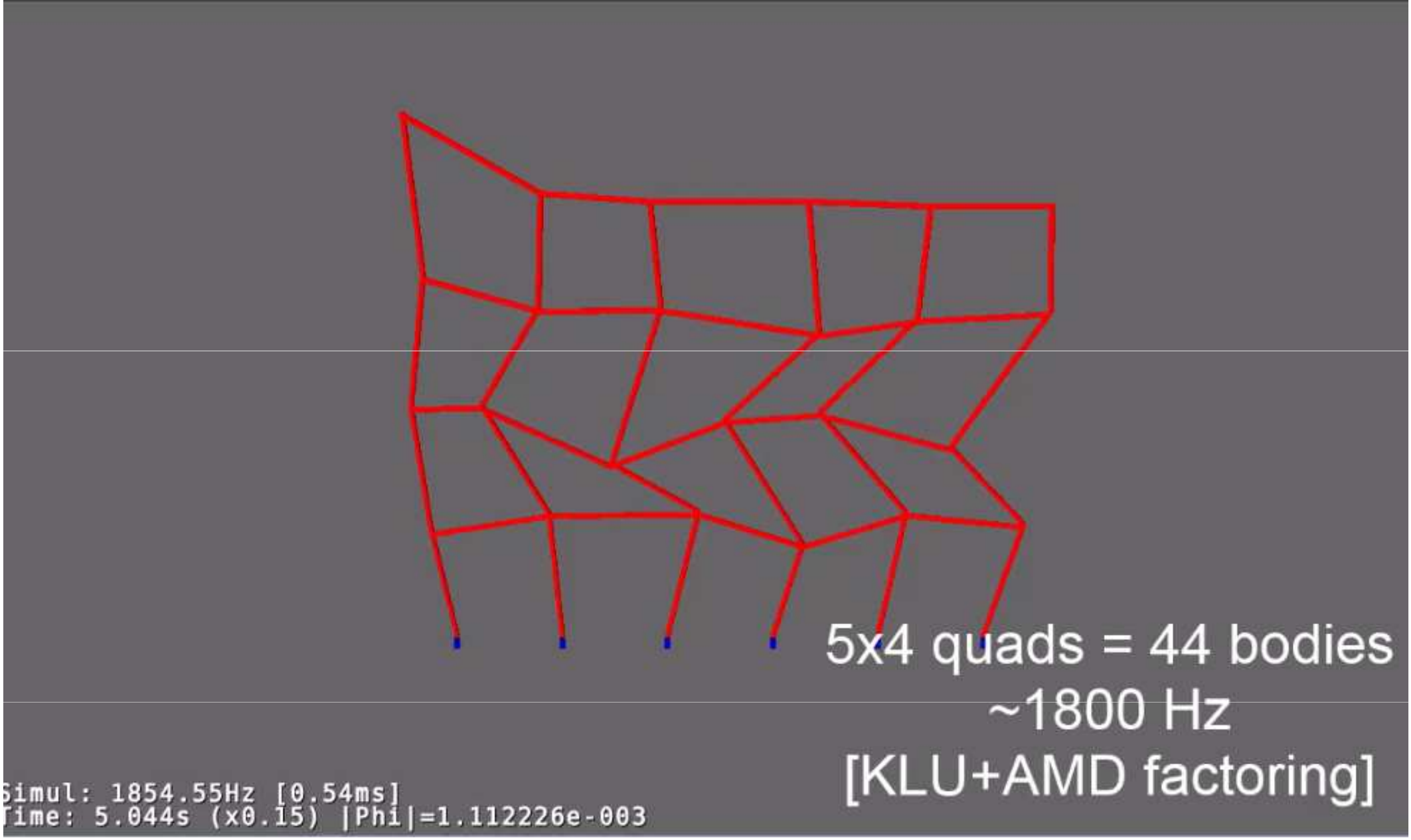


The benchmark

- For each solver
  - For each ordering
    - For N=1…32
      - Build mechanism with N_x = N, N_y =N.
      - Run thousands of time steps for each configuration.
      - Measure **average execution time** of each calculation.

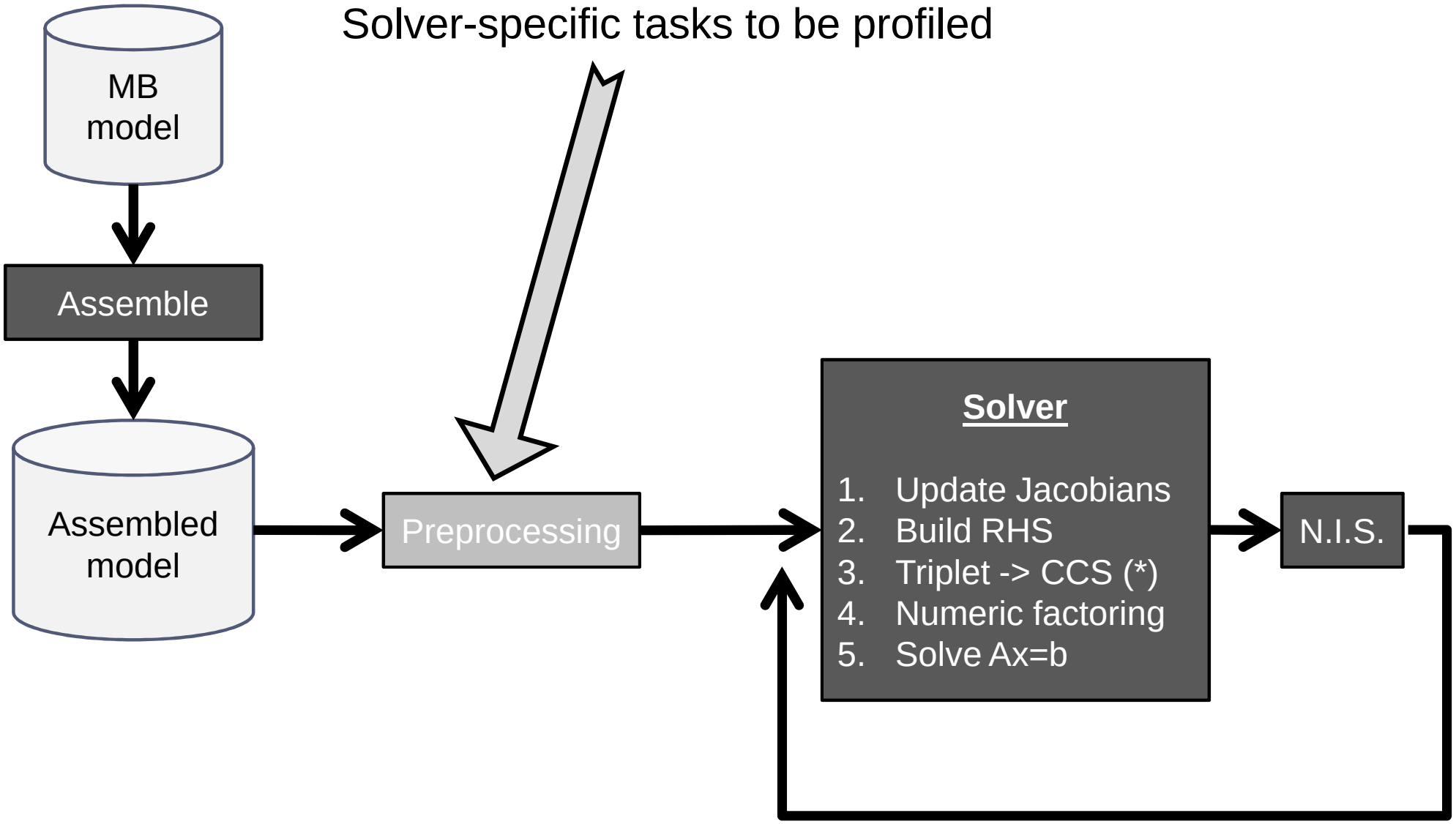
Talk outline

1. Problem introduction
2. Tested algorithms
3. Benchmark
4. **Results**

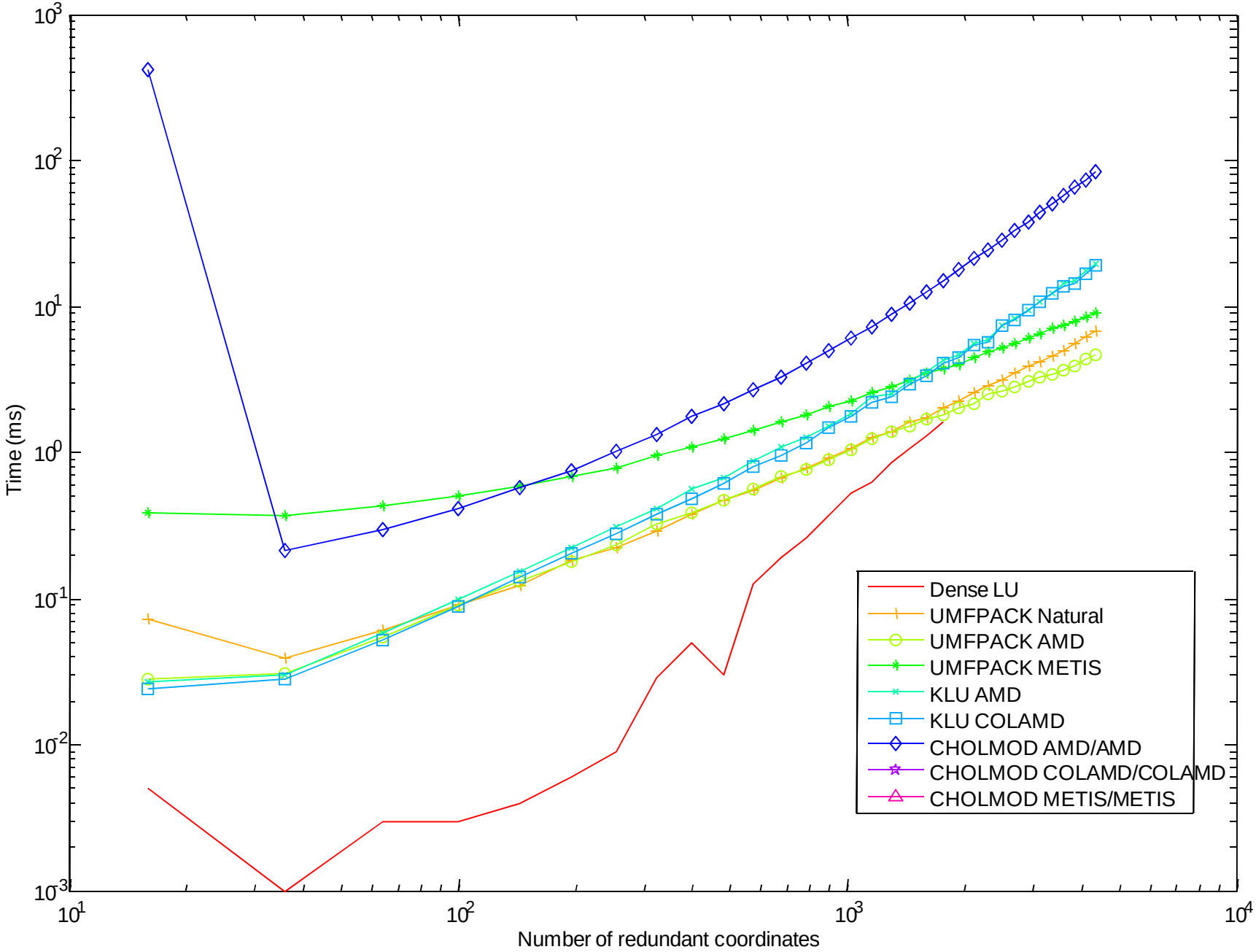


http://www.youtube.com/watch?v=1REd2bQbnys

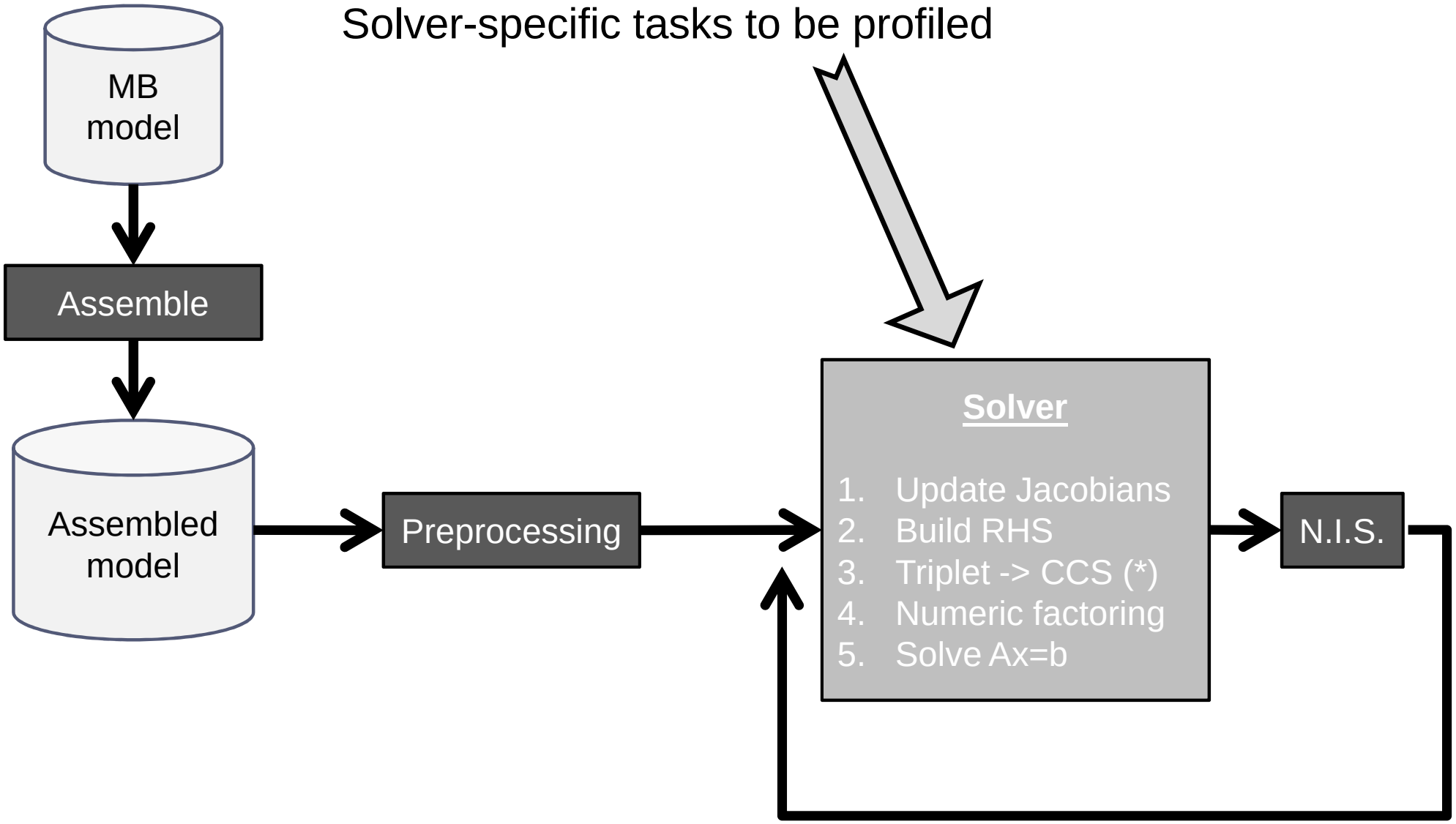
Benchmark C++ implementation



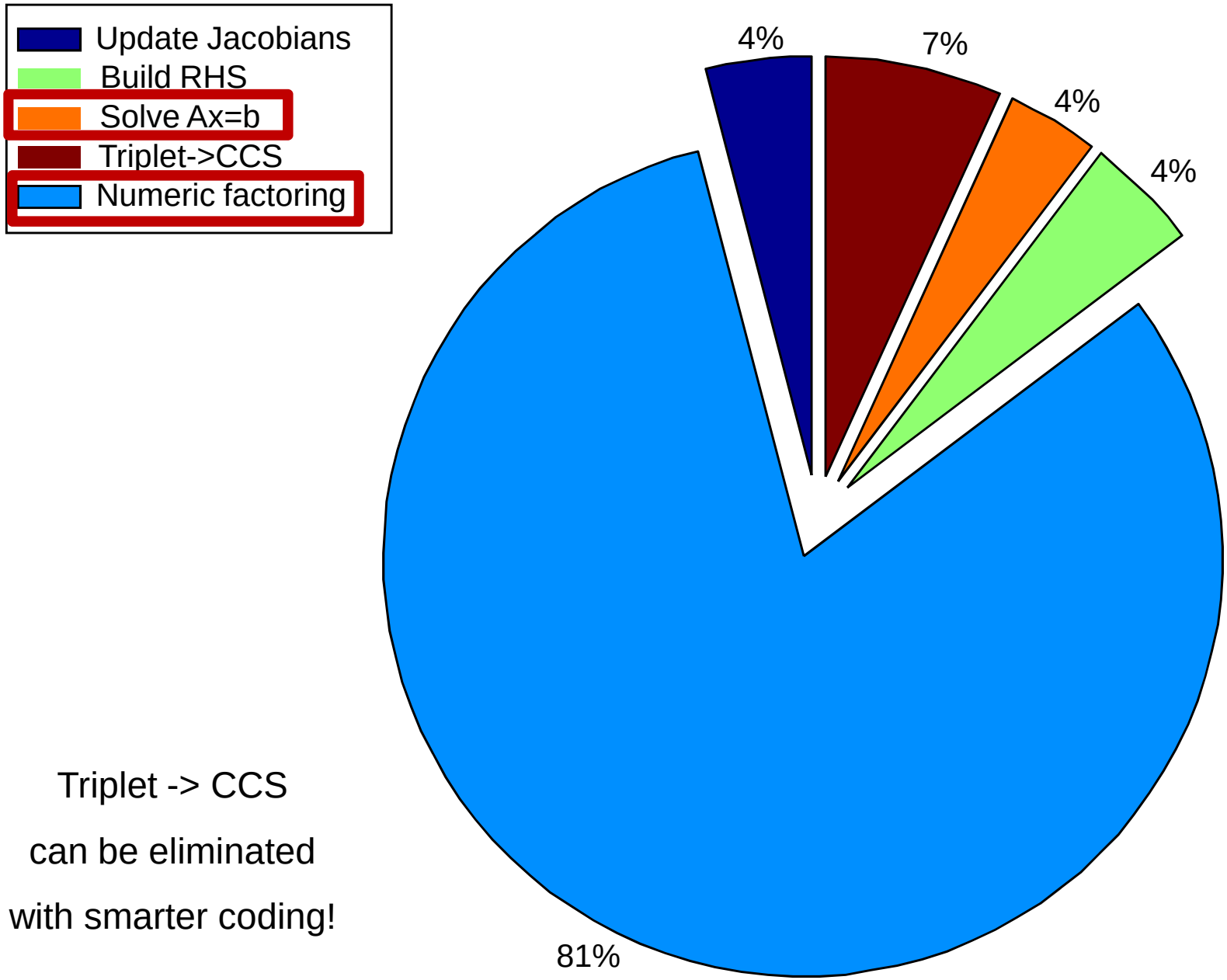
Results: Preprocessing (symbolic factorization)



Benchmark C++ implementation



Results: Distribution of time (KLU+COLAMD)

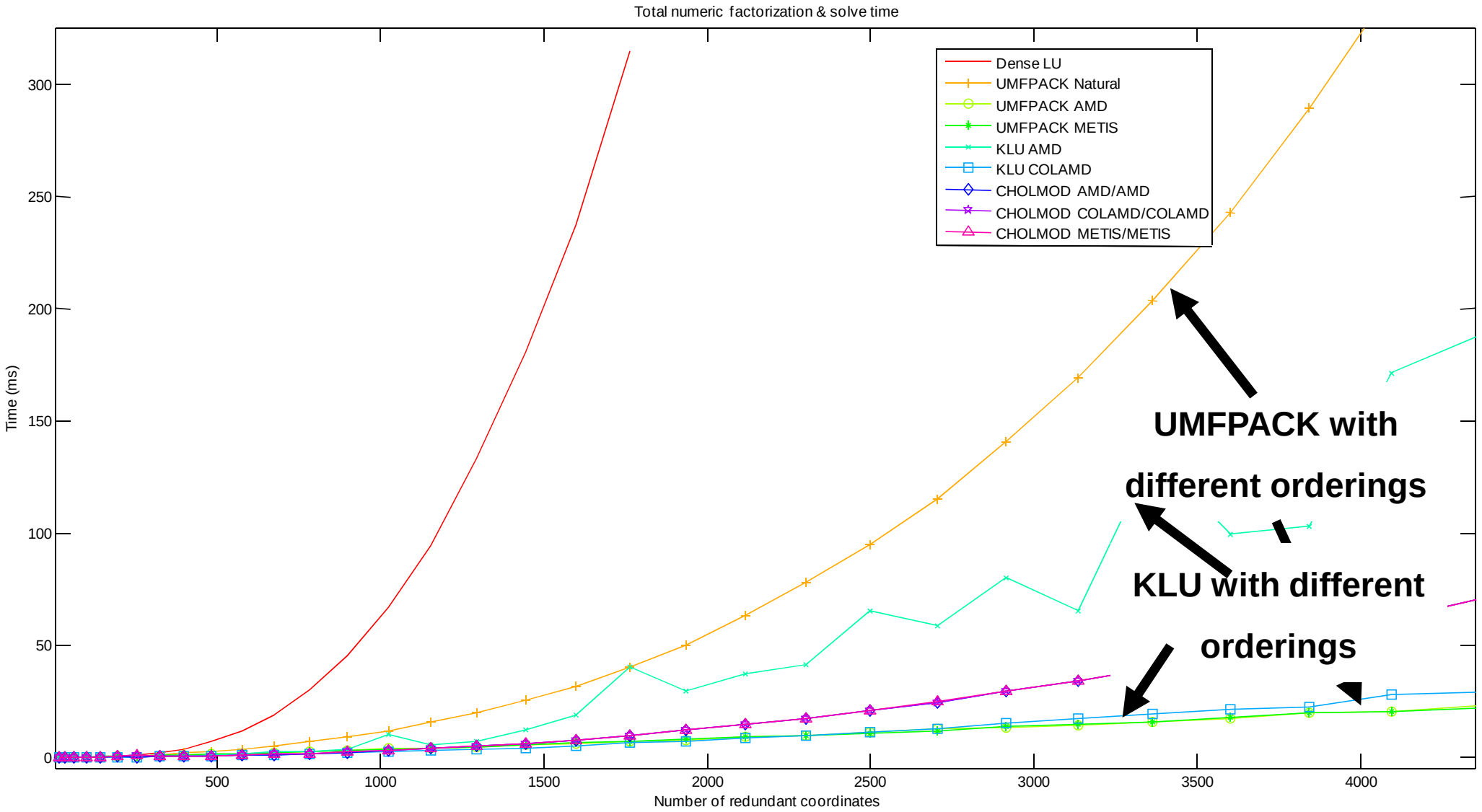


Triplet -> CCS

can be eliminated

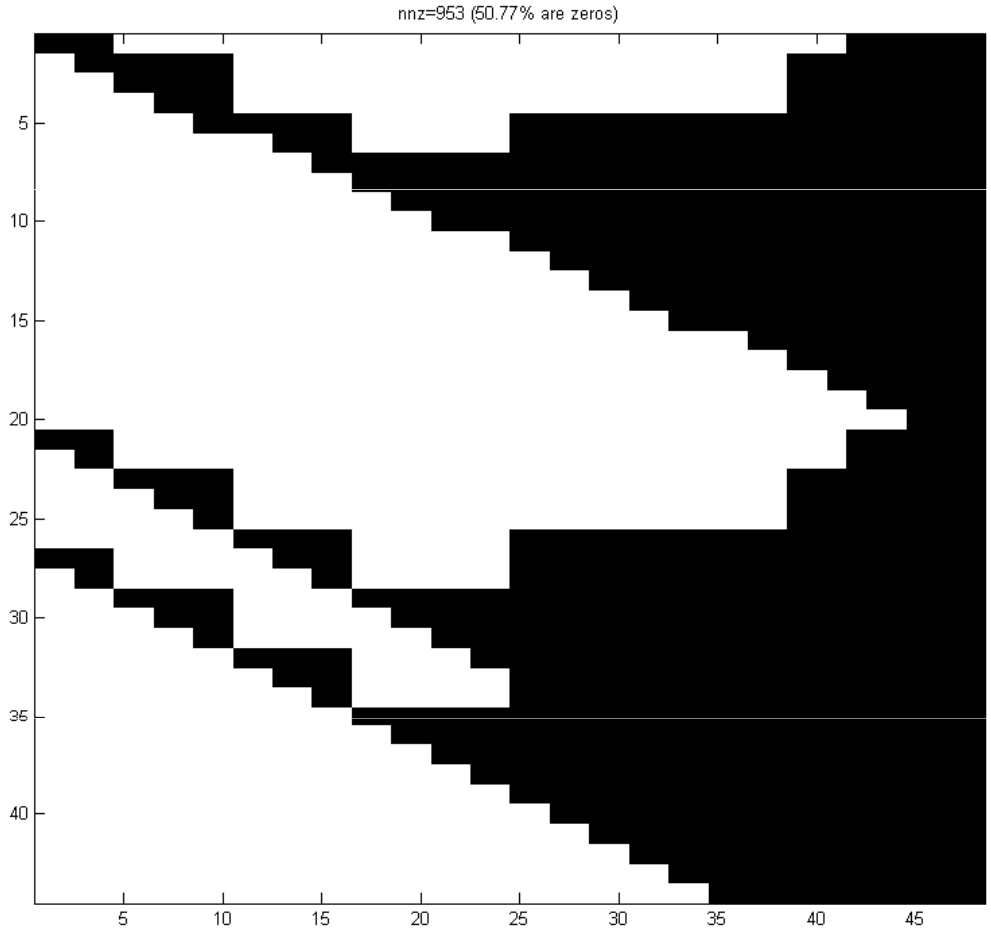
with smarter coding!

Results: num. factoring + solve

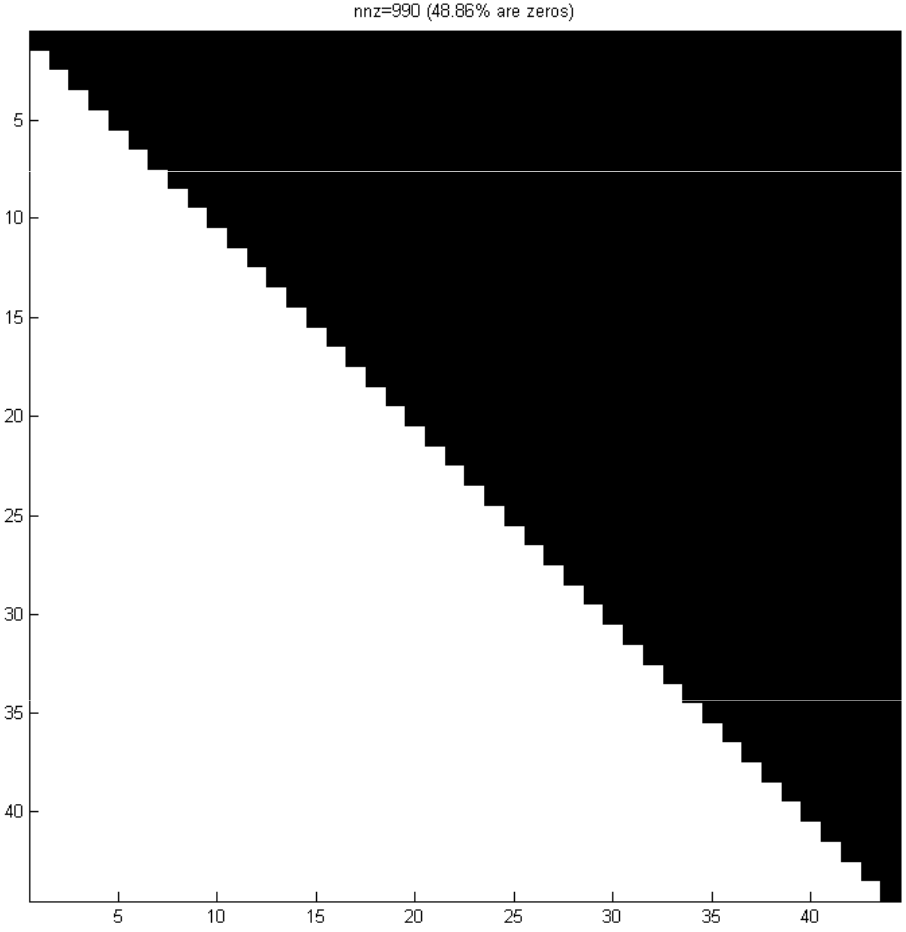


A comment on the Cholesky approach…

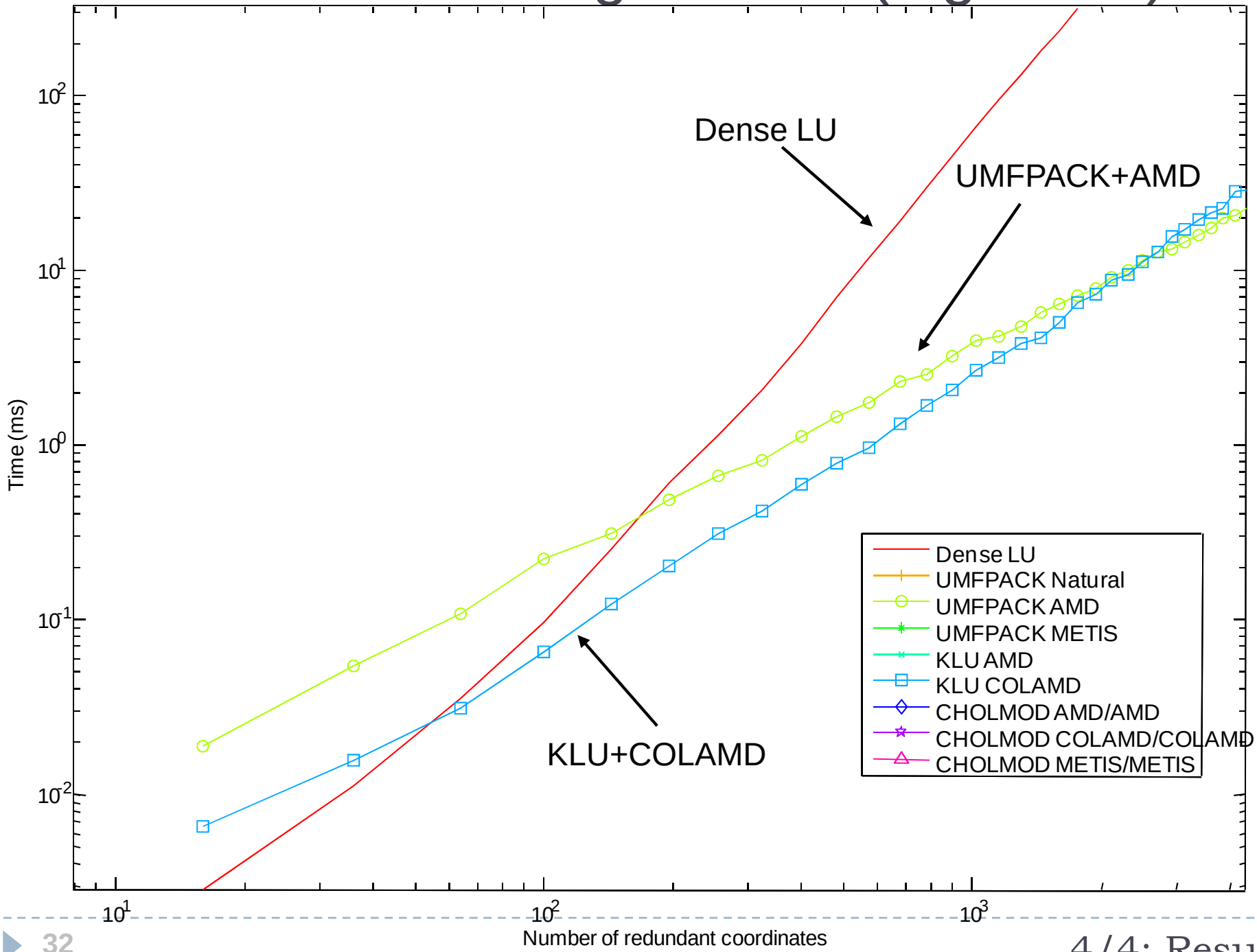
"E" matrix



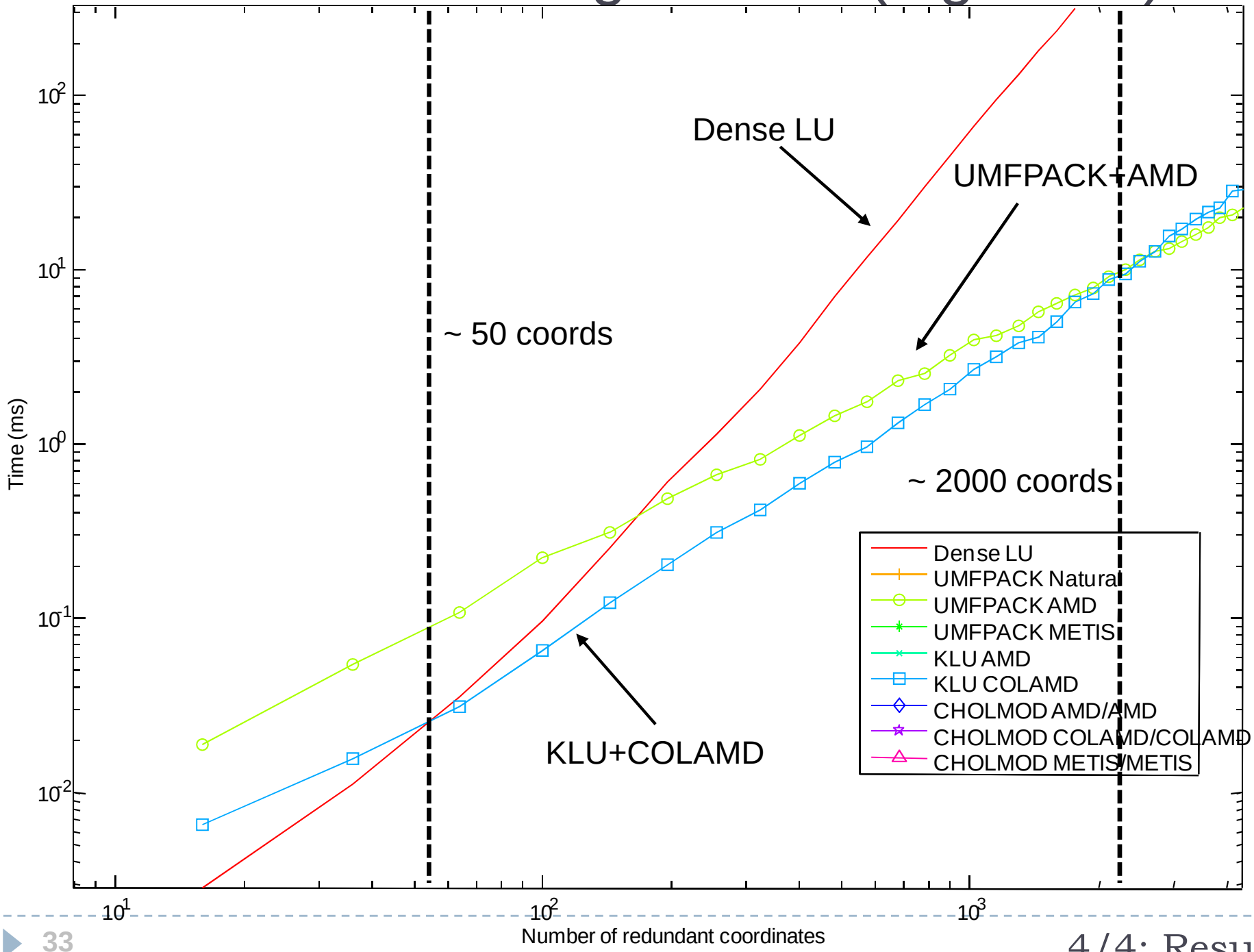
chol(E * E') → dense!!



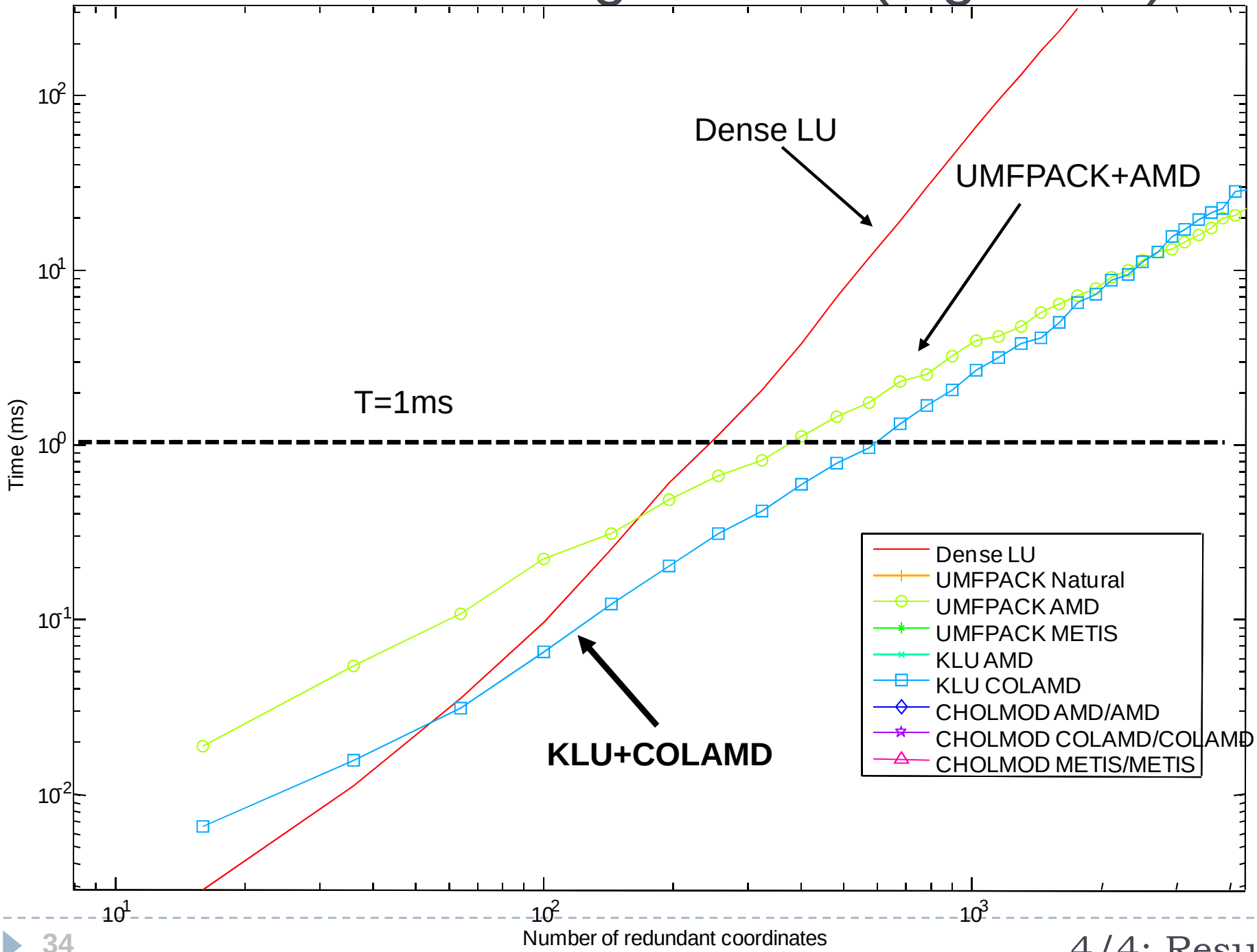
Results: num. factoring + solve (log scale)



Results: num. factoring + solve (log scale)



Results: num. factoring + solve (log scale)



Conclusions

- MBS dynamics leads to very sparse systems.
- Exploiting sparsity is crucial for performance…

  …unless N<50 approx.
- 50<N<2000 → Use KLU+COLAMD
- 2000<N → Use UMFPACK+AMD

- In any case: reuse symbolic factorizations!

Slides available online:
`http://www.ual.es/~jlblanco/` → Publications

A comparison of Algorithms for Sparse Matrix Factoring and Variable Reordering aimed at Real-time Multibody Dynamic Simulation

Thanks for your attention!

Slides available online:
`http://www.ual.es/~jlblanco/` → Publications