An Efficient Solver for Sparse Linear Systems based on Rank-Structured Cholesky Factorization

David Bindel

Department of Computer Science
Cornell University

15 March 2016

```
u = K \ f
```

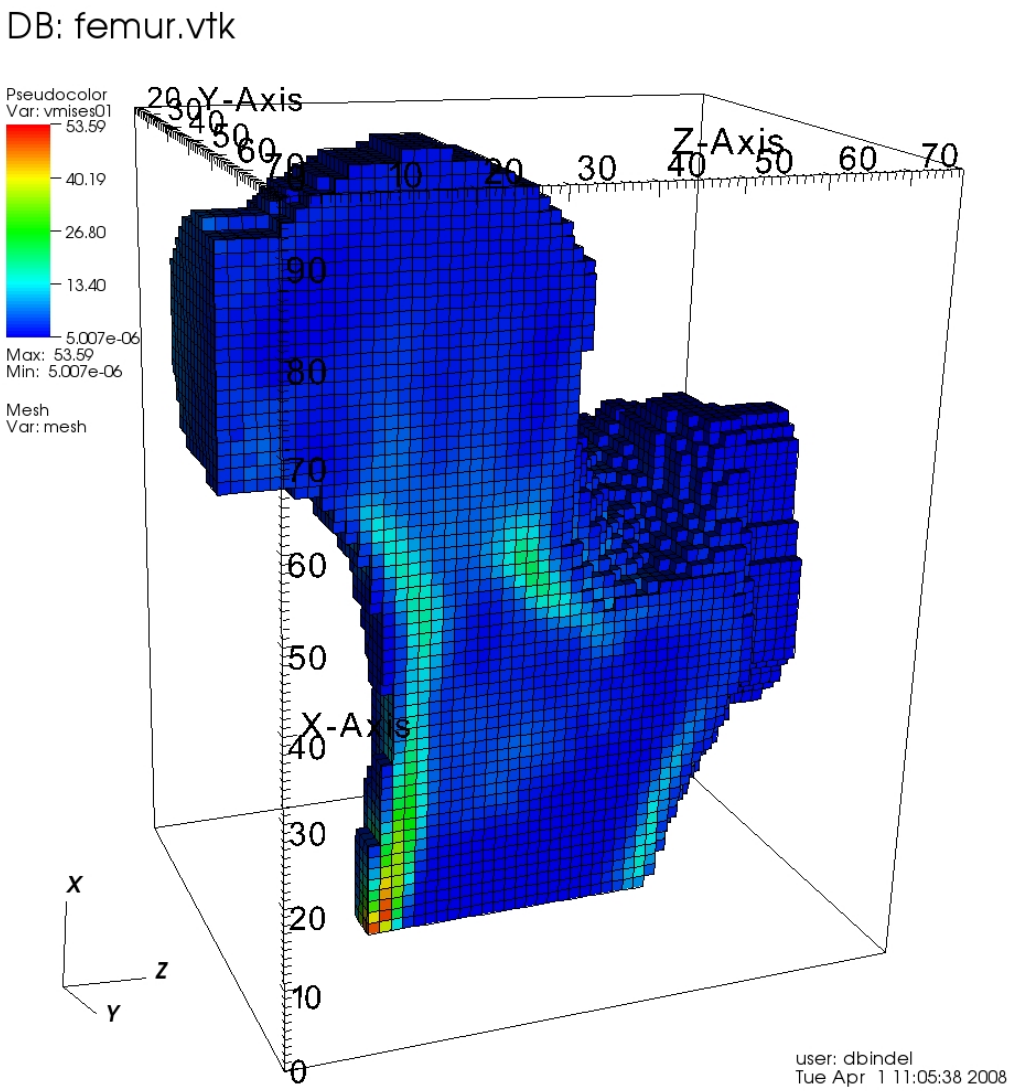
Great for circuit simulations, 1D or 2D finite elements, etc.

Standard advice to students: Just try backslash for these problems.
Standard response: What about for the 3D case?

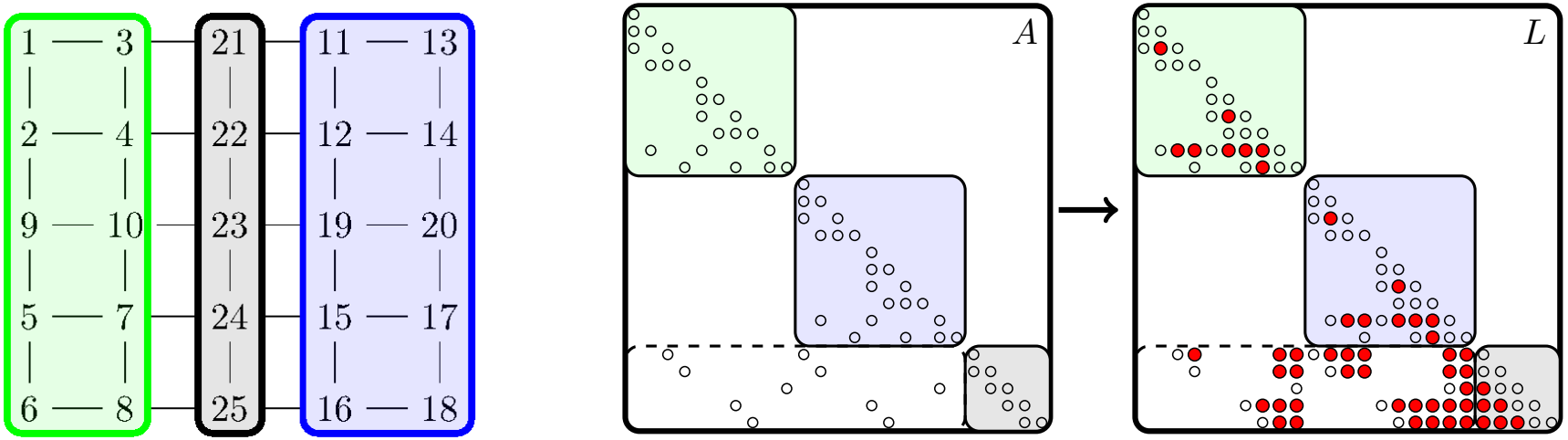
"Try PCG with a good preconditioner. Maybe start with the ones in PETSc. You've taken Matrix Computations, right? Blah blah yadda blah..."

(Not an actual student)

My own frustration (c. 2008)

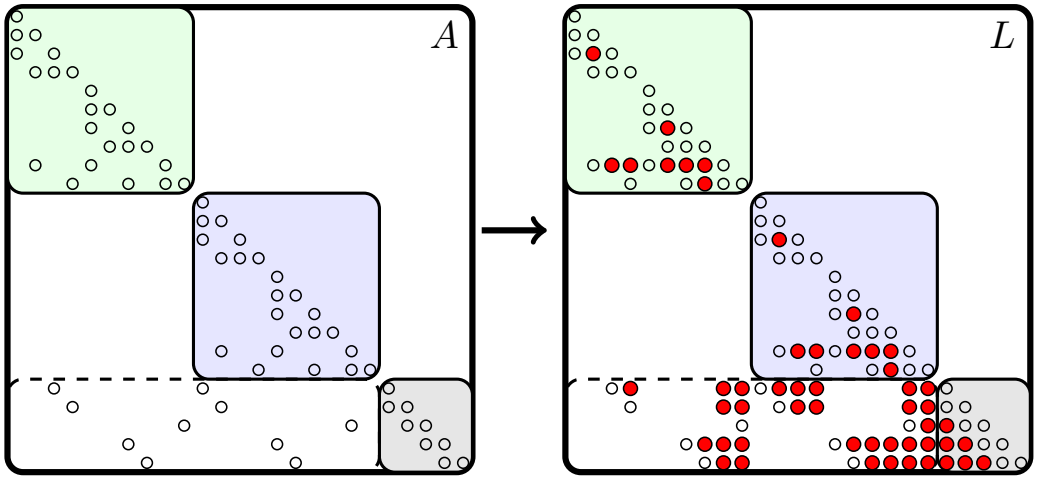


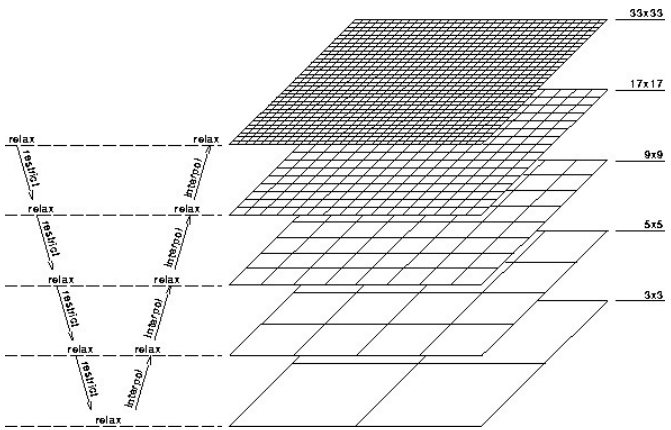
Factorization with nested dissection



- Cholesky: $A = LL^T$, L is lower triangular.
- L may have more nonzeros than A (fill)
- Order controls fill – still a *lot* of memory in 3D

Direct or iterative?





CW: Gaussian elimination scales poorly. Iterate instead!

- **Pro:** Less memory, potentially better complexity
- **Con:** Less robust, potentially worse memory patterns

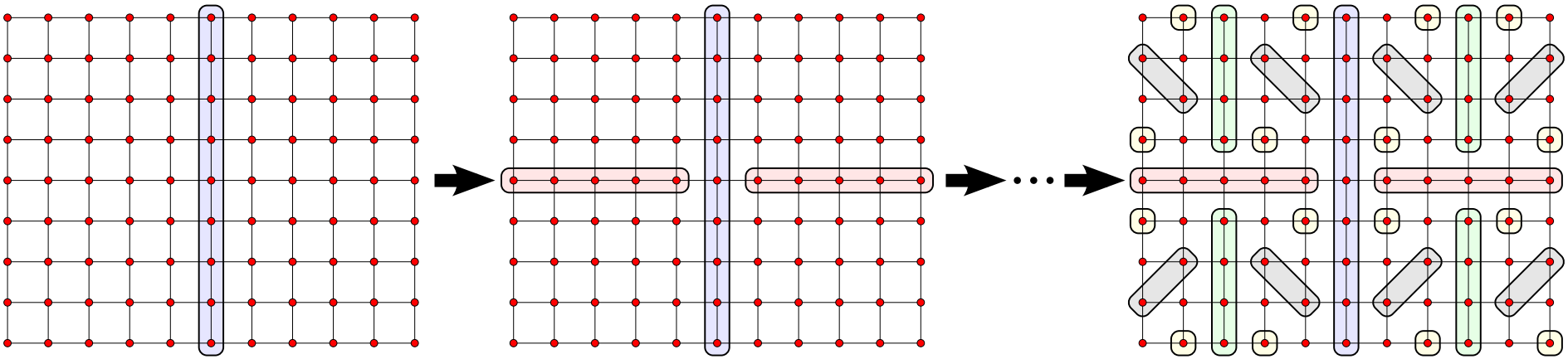
Commercial finite element codes still use (out-of-core) Cholesky.
Longer compute times, but fewer tech support hours.

Desiderata

I want a code for sparse Cholesky ($\mathbf{A} = \mathbf{L}\mathbf{L}^T$) that

- Handles modest problems on a desktop (or laptop?)
  - Inside a loop, without trying my patience
  - $\implies$ Does not need gobs of memory
  - $\implies$ Makes effective use of level 3 BLAS
- Requires little parameter fiddling / hand-holding
- Works with general elliptic problems (esp. elasticity)

From ND to “superfast” ND



ND gets performance using just *graph* structure:

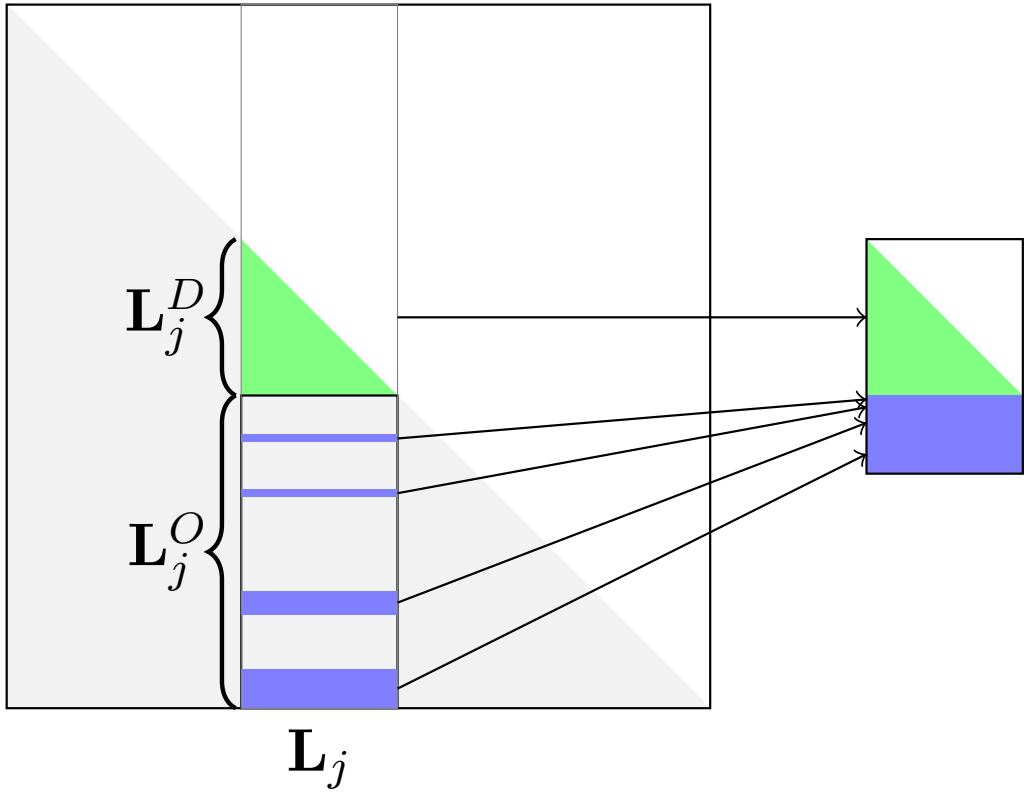
2D: $O(N^{3/2})$ time, $O(N \log N)$ space.
3D: $O(N^2)$ time, $O(N^{4/3})$ space.

Superfast ND reduces space/time complexity via *low-rank* structure.

Strategy

- Start with CHOLMOD (a good supernodal left-looking Cholesky)
  - Supernodal data structures are compact
  - Algorithm + data layout $\implies$ most work in level 3 BLAS
  - Widely used already (so re-use the API!)
- Incorporate compact representations for low-rank blocks
  - Outer product for off-diagonal blocks
  - HSS-style representations for diagonal blocks
- Optimize, test, swear, fix, repeat

Supernodal storage structure



$\mathbf{L}(\mathcal{C}_j, \mathcal{C}_j) \equiv \mathbf{L}_j^D$

$\mathbf{L}(\mathcal{C}_j, \mathcal{R}_j) \equiv \mathbf{L}_j^O$ collapsed

Supernode factorization

$\mathcal{U}_j^D \longleftarrow \mathbf{A}(\mathcal{C}_j, \mathcal{C}_j)$
$\mathcal{U}_j^O \longleftarrow \mathbf{A}(\mathcal{R}_j, \mathcal{C}_j)$

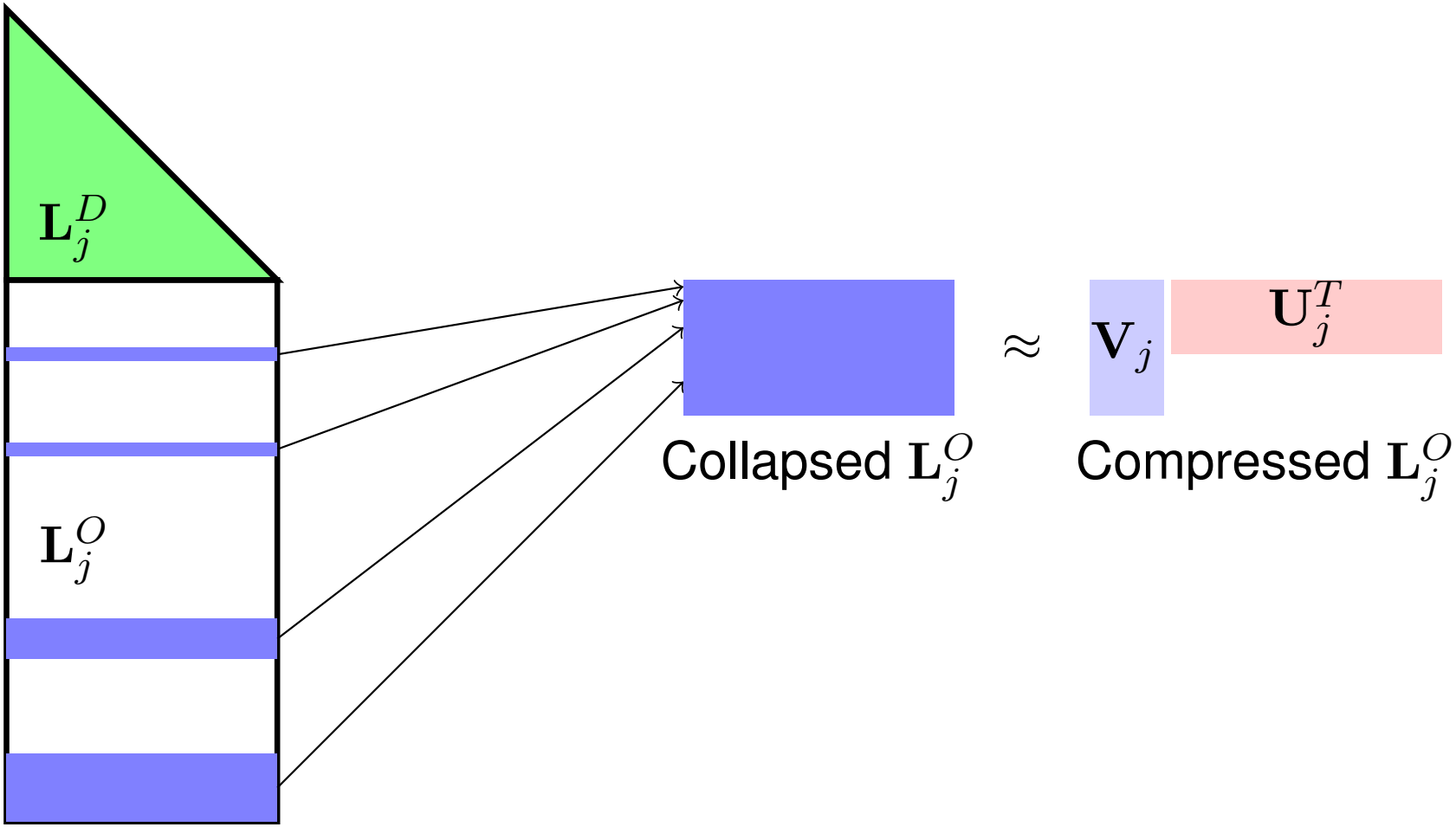
for each $k \in \mathbb{D}_j$ **do**
  Build dense updates from $\mathbf{L}_k^O$
  Scatter updates to $\mathcal{U}_j^D$ and $\mathcal{U}_j^O$

$\mathbf{L}_j^D \longleftarrow \mathrm{cholesky}(\mathcal{U}_j^D)$
$\mathbf{L}_j^O \longleftarrow \mathcal{U}_j^O (\mathbf{L}_j^D)^{-T}$

- Initialize storage
- Pull Schur contributions
- Finish forming $\mathbf{L}_j^D$

What changes in the rank-structured Cholesky?

Off-diagonal block compression



Collapsed off-diagonal block is a (nearly low-rank) dense matrix

Off-diagonal block compression

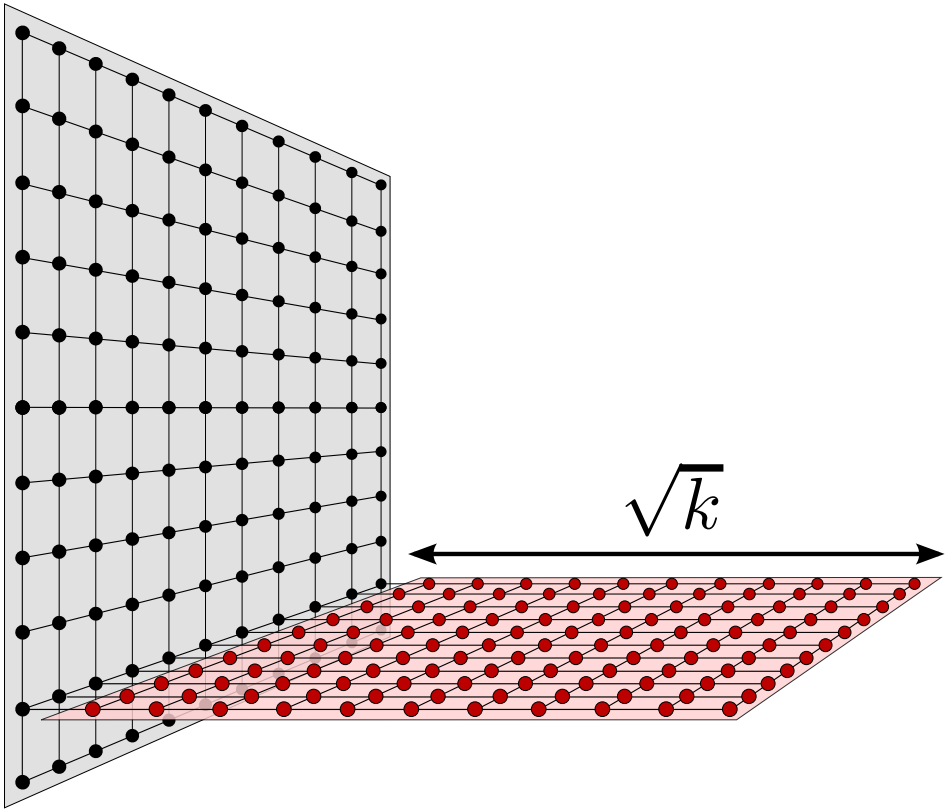
$\mathbf{G} \longleftarrow \text{rand}(|\mathcal{C}_j|, r+p)$
$\mathbf{C} \longleftarrow (\mathbf{L}_j^O)^T \mathbf{G}$
for $i = 1, \ldots, s$ **do**
  $\mathbf{C} \longleftarrow (\mathbf{L}_j^O)\mathbf{C}$
  $\mathbf{C} \longleftarrow (\mathbf{L}_j^O)^T \mathbf{C}$
$\mathbf{U}_j = \text{orth}(\mathbf{C})$
$\mathbf{V}_j = \mathbf{L}_j^O \mathbf{U}_j$

Compress *without* explicit $\mathbf{L}_j^O$:

- Probe $(\mathbf{L}_j^O)^T$ with random $\mathbf{G}$
- Extract orth. row basis $\mathbf{U}_j$
- $\mathbf{L}_j^O = \mathbf{V}_j \mathbf{U}_j^T \implies \mathbf{V}_j = \mathbf{L}_j^O \mathbf{U}_j$

Where do we get the estimated rank bound r?

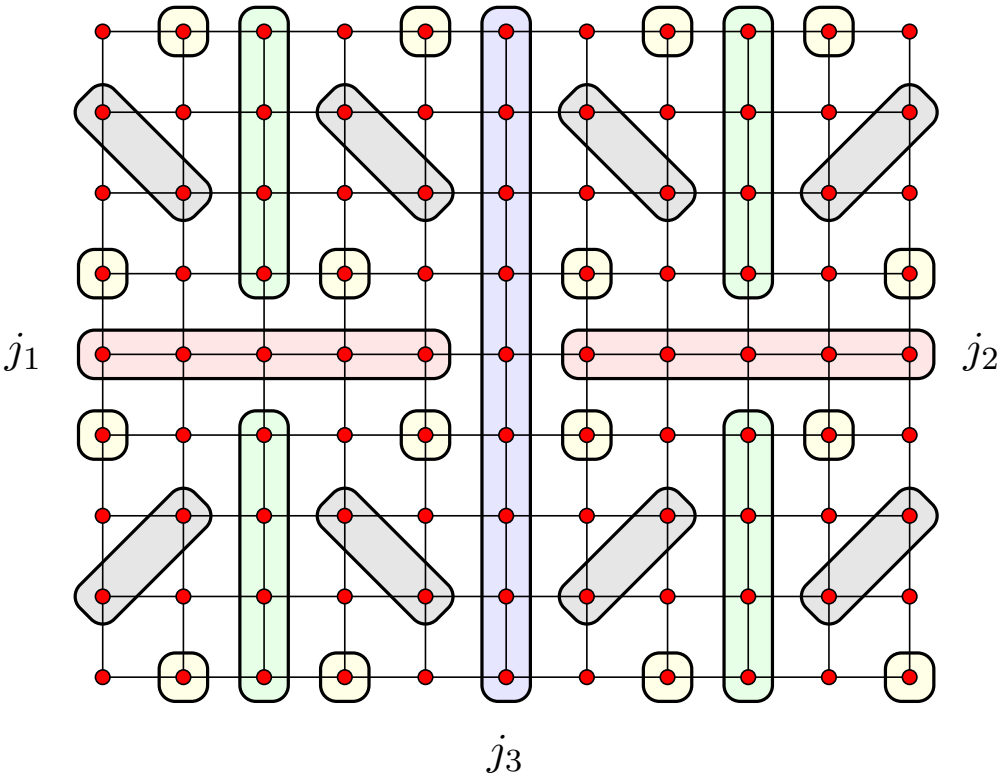
Interaction rank

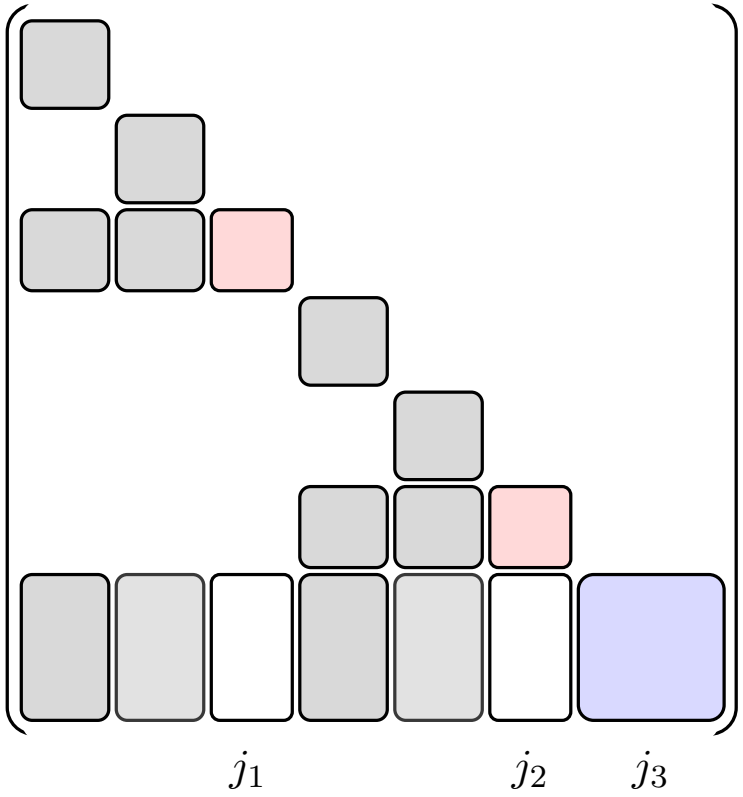


Could dynamically estimate the rank of L_j^O.

Practice: empirical rank bound $\approx \alpha\sqrt{k}\log(k)$.

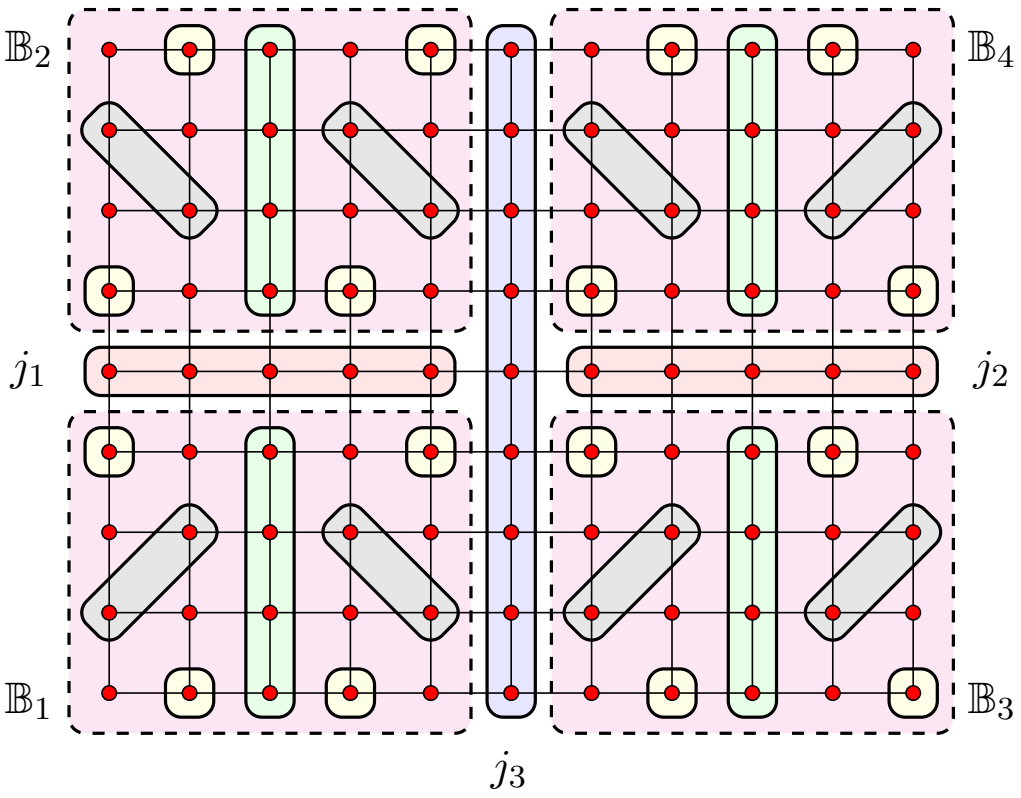
Optimization: Selective off-diagonal compression

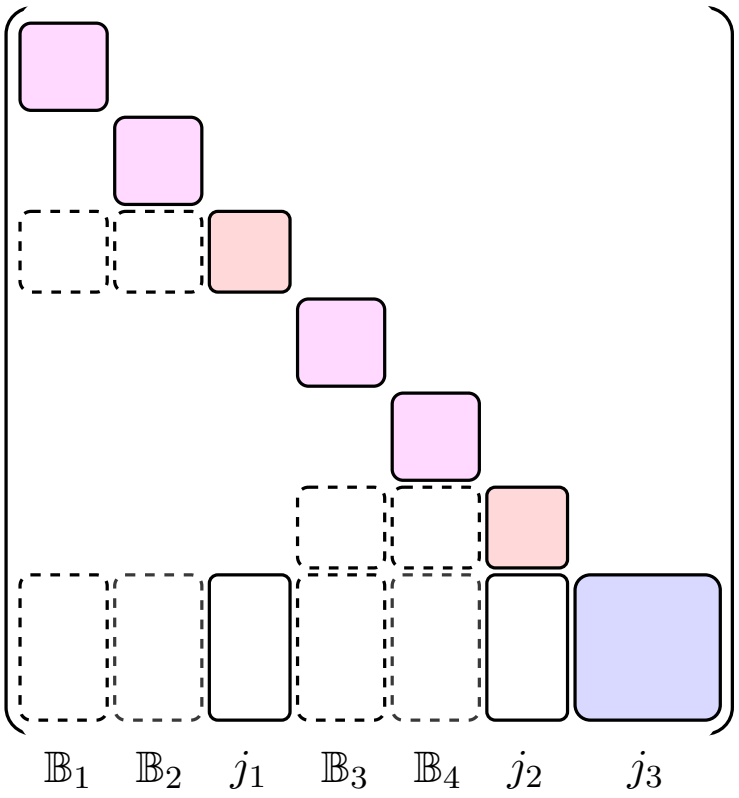




Compress off-diagonal blocks of sufficiently large supernodes (j_1, j_2).

Optimization: Interior blocks





Don't store *any* of $\mathbf{L}_j^O$ for "interior" blocks
(Represent as $\mathbf{L}_j^O = \mathbf{A}_j^O(\mathbf{L}_j^D)^{-1}$ when needed)

Diagonal block compression

$$\mathbf{L}_j^D = \left(\begin{array}{cc|cc} \mathbf{L}_{j,1}^D & \mathbf{0} & \multicolumn{2}{c}{\multirow{2}{*}{$\mathbf{0}$}} \\ \mathbf{L}_{j,2}^D & \mathbf{L}_{j,3}^D & & \\ \hline \multicolumn{2}{c|}{\multirow{2}{*}{$\mathbf{L}_{j,4}^D$}} & \mathbf{L}_{j,5}^D & \mathbf{0} \\ & & \mathbf{L}_{j,6}^D & \mathbf{L}_{j,7}^D \end{array}\right)$$

Basic observation: off-diagonal blocks are *low-rank*.
(H-matrix, semiseparable structure, quasiseparable structure, ...)

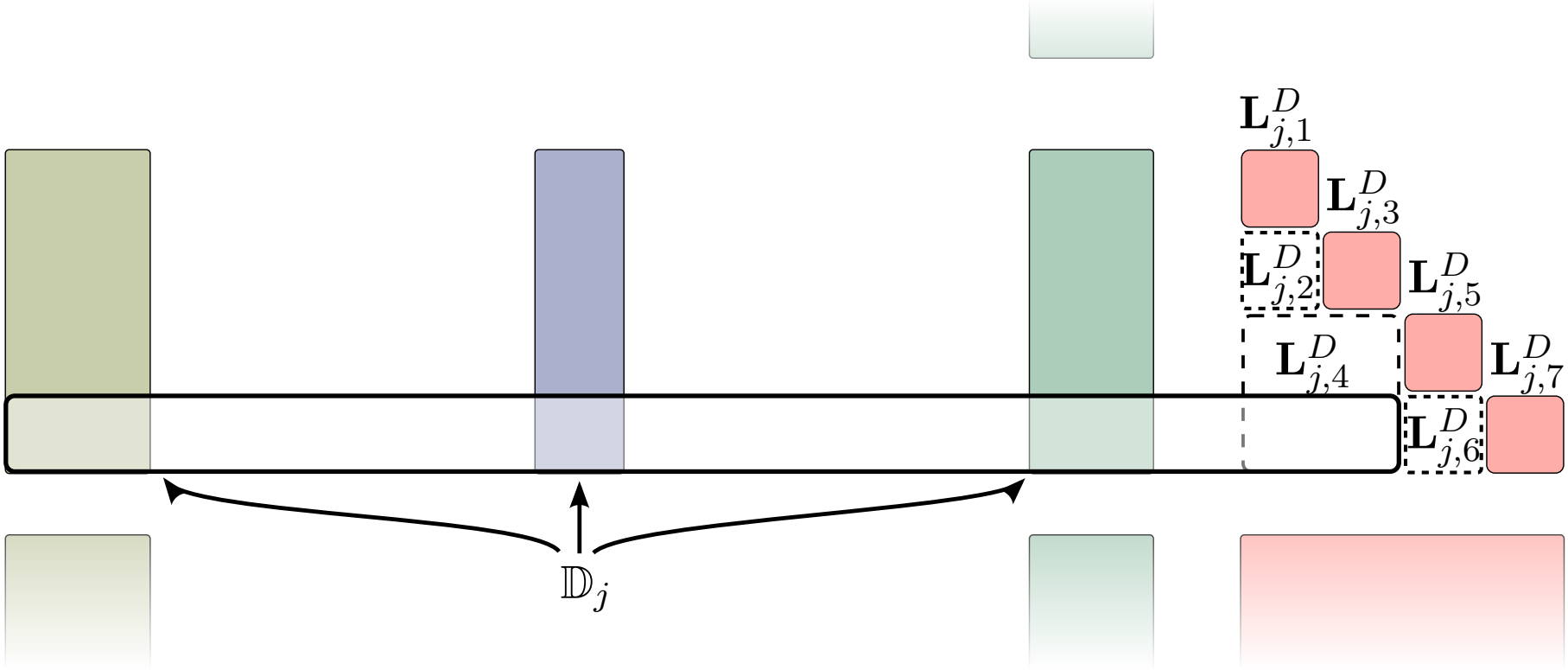
Assumes reasonable ordering of unknowns!

Diagonal block compression

$$\mathbf{L}_j^D \approx \left(\begin{array}{c|c|cc} \mathbf{L}_{j,1}^D & \mathbf{0} & \multicolumn{2}{c}{\multirow{2}{*}{$\mathbf{0}$}} \\ \mathbf{V}_{j,2}^D(\mathbf{U}_{j,2}^D)^T & \mathbf{L}_{j,3}^D & & \\ \hline \multicolumn{2}{c|}{\multirow{2}{*}{$\mathbf{V}_{j,4}^D(\mathbf{U}_{j,4}^D)^T$}} & \multicolumn{1}{c|}{\mathbf{L}_{j,5}^D} & \mathbf{0} \\ & & \multicolumn{1}{c|}{\mathbf{V}_{j,6}^D(\mathbf{U}_{j,6}^D)^T} & \mathbf{L}_{j,7}^D \end{array}\right).$$

How do we get *directly* to this without forming $\mathcal{U}_j^D$ explicitly?

Forming compressed updates



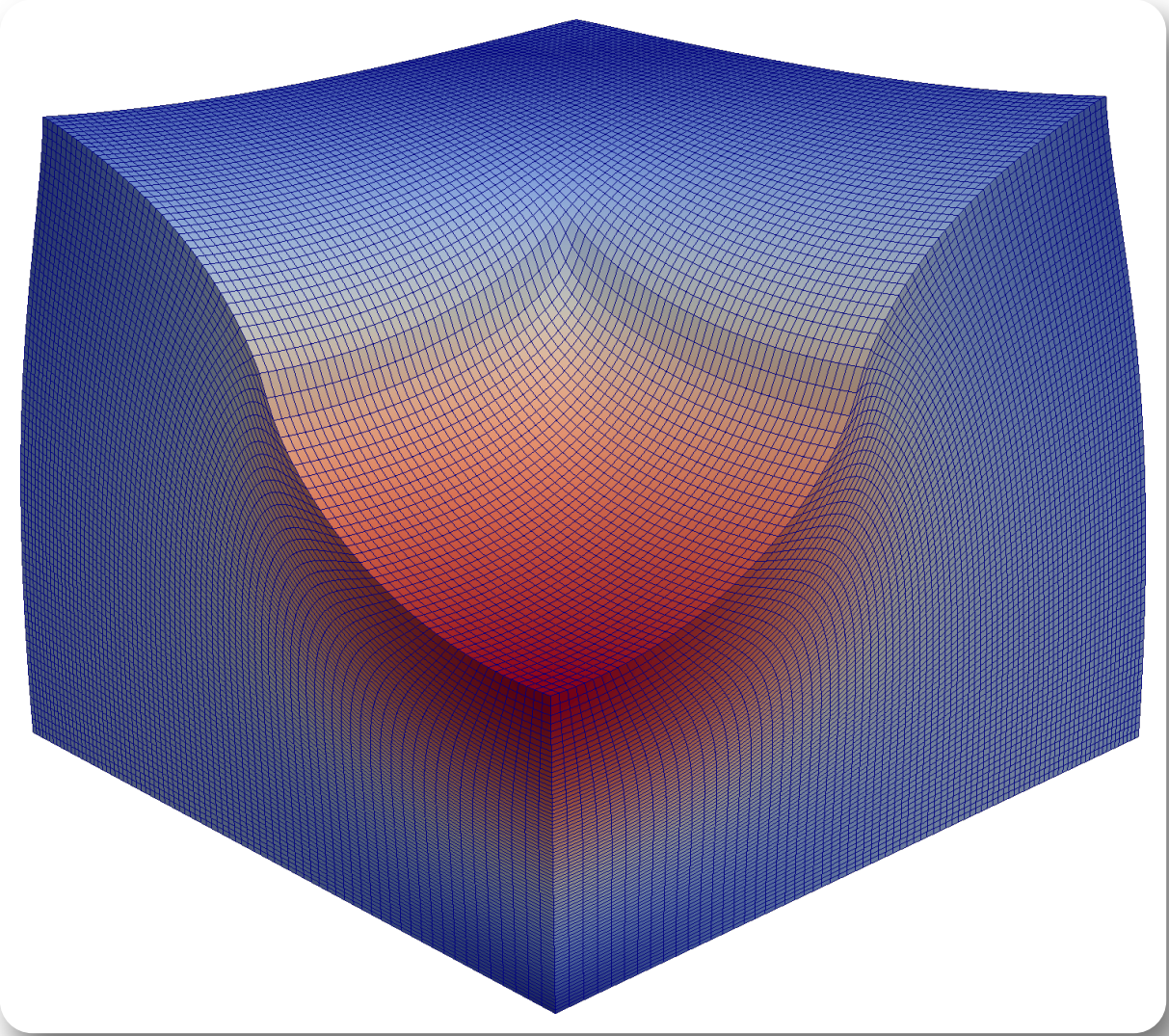
Rank-structured supernode factorization

Basic ingredients:

- Randomized algorithms form $\mathcal{U}_j^D$
- Rank-structured factorization of $\mathcal{U}_j^D$
- Randomized algorithm forms $\mathbf{L}_j^O$ (involves solves with $\mathbf{L}_j^D$)

Plus various optimizations.

Example: Large deformation of an elastic block



Example: Large deformation of an elastic block

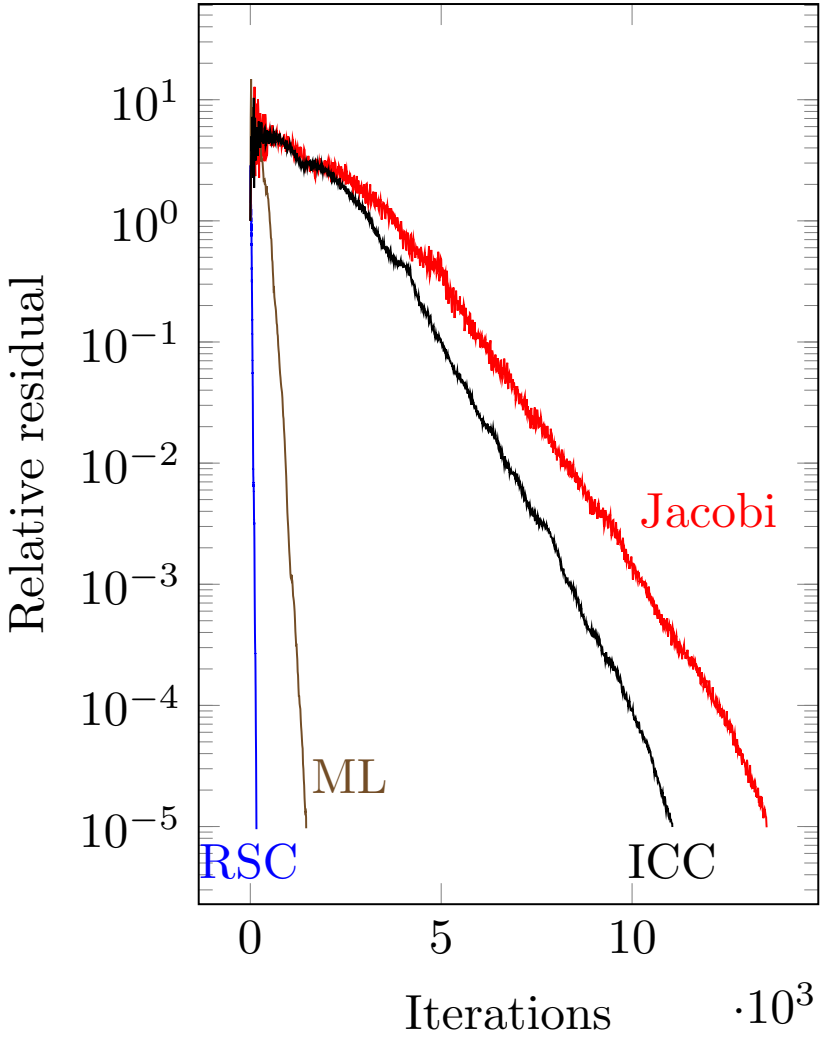
Benchmark based on example from deal.II:

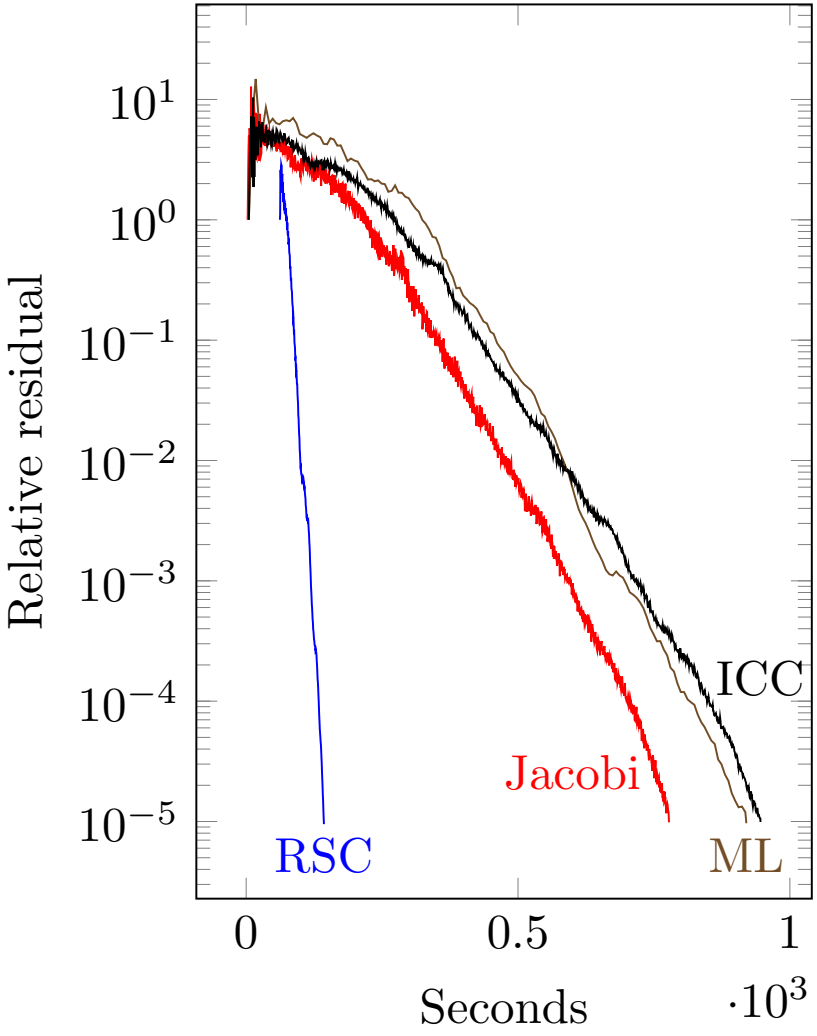
- Nearly-incompressible hyperelastic block under compression
- Mixed FE formulation (pressure and dilation condensed out)
- Tried both $p = 1$ and $p = 2$ finite elements
- Two load steps, Newton on each (14-15 steps)

Experimental setup:

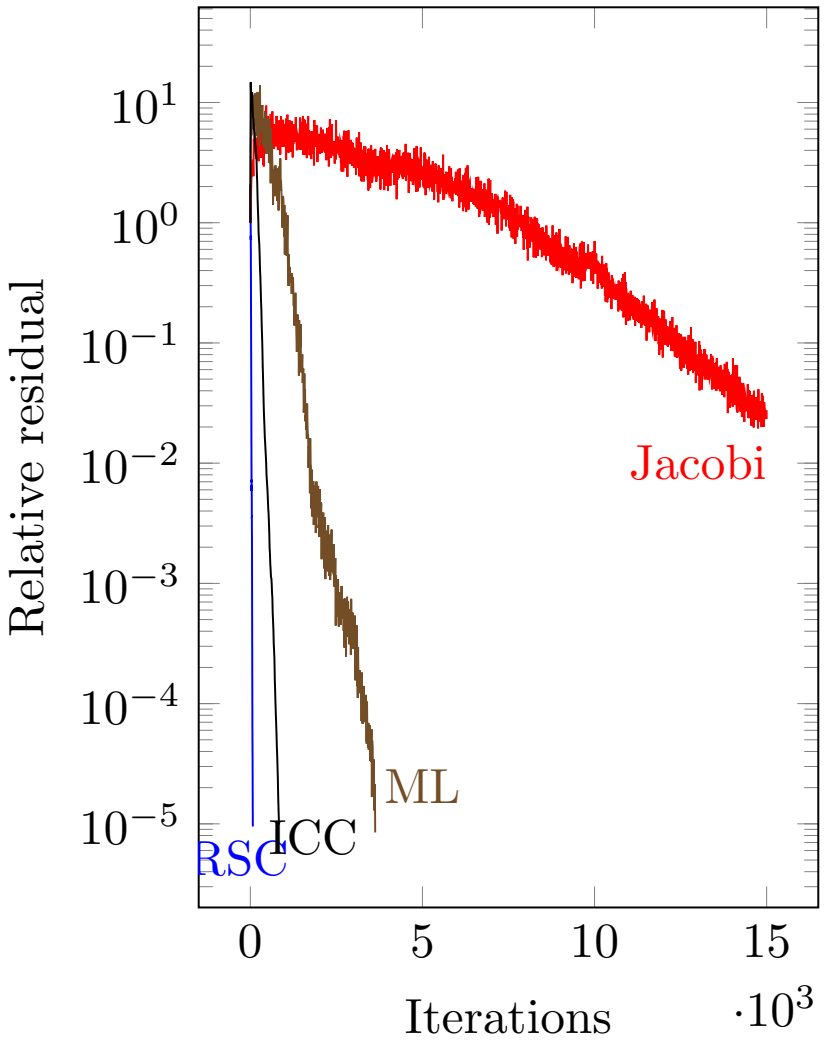
- 8-core Xeon X5570 with 48 GB RAM
- LAPACK/BLAS from MKL 11.0
- PCG + preconditioners from Trilinos

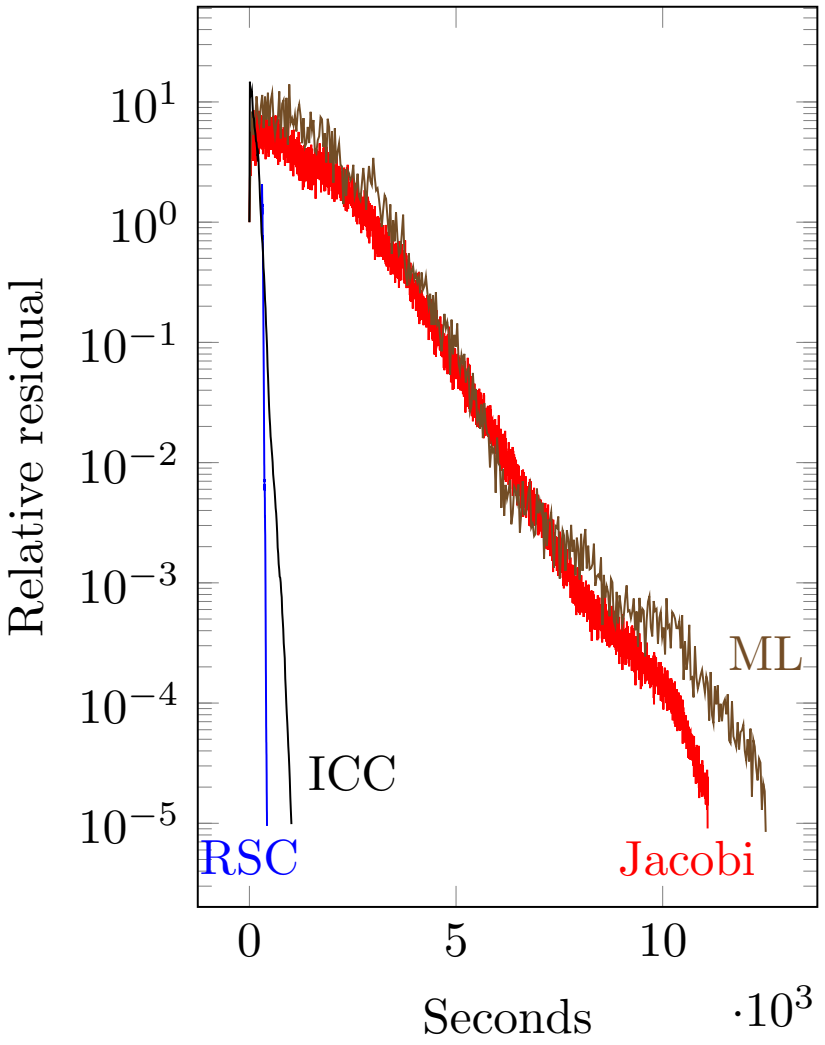
RSC vs standard preconditioners ($p = 1, N = 50$)



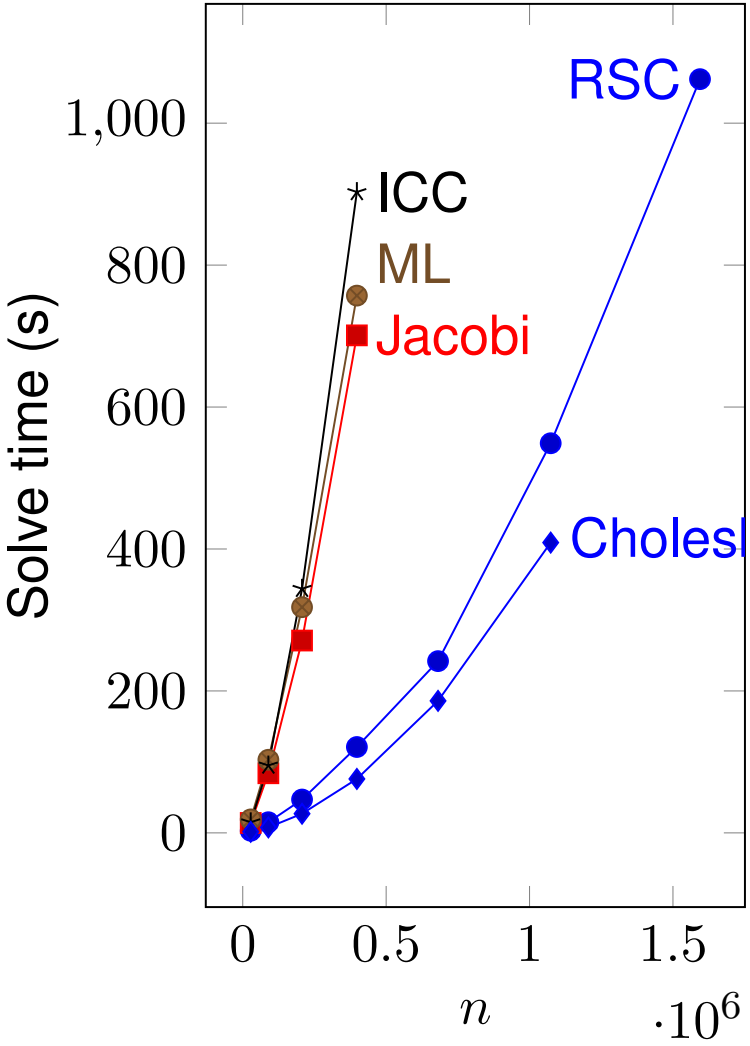


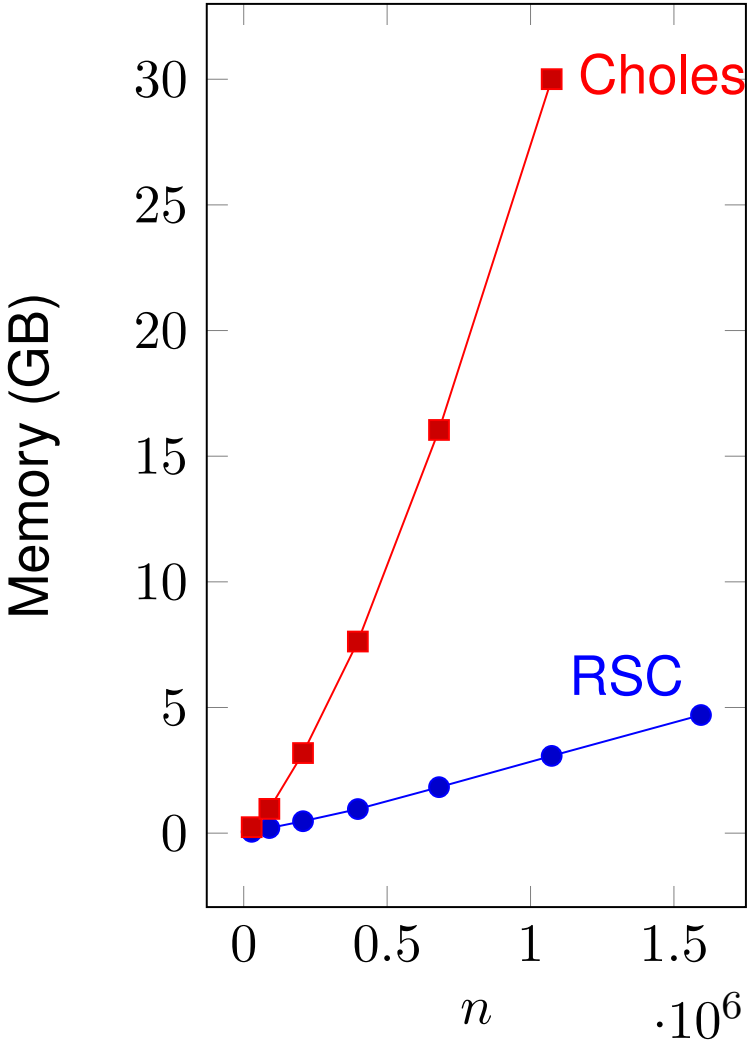
RSC vs standard preconditioners ($p = 2, N = 35$)



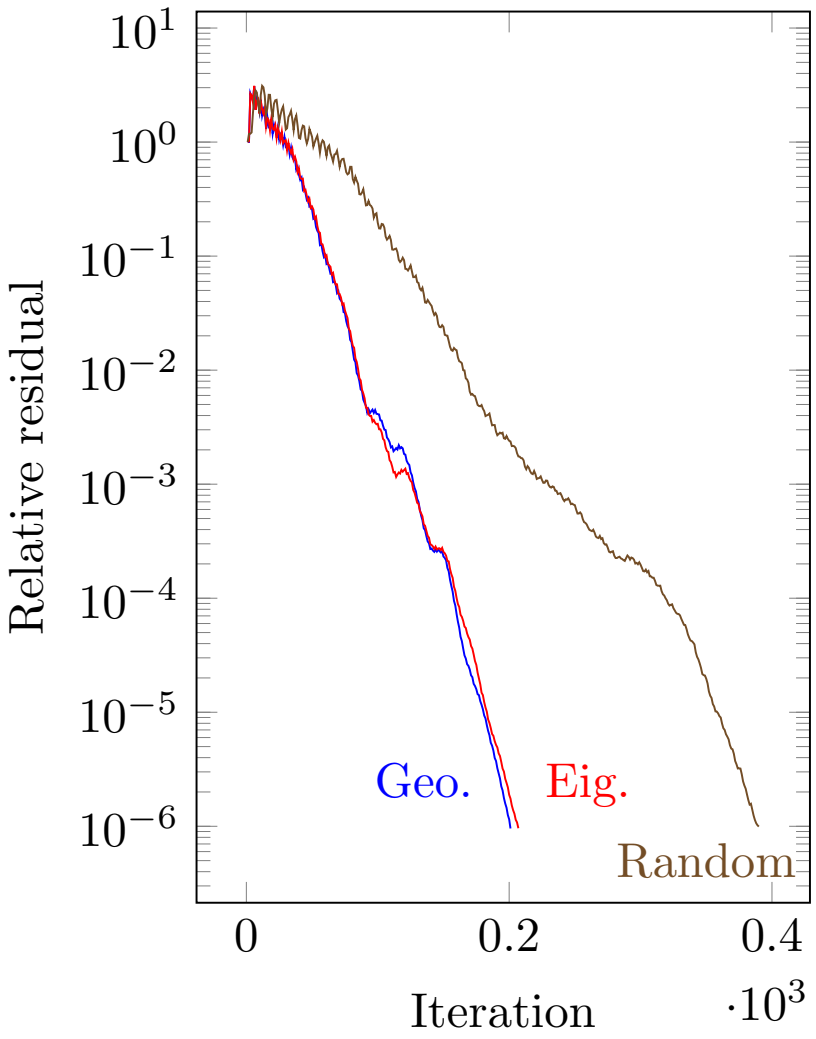


Time and memory comparisons ($p = 1$)





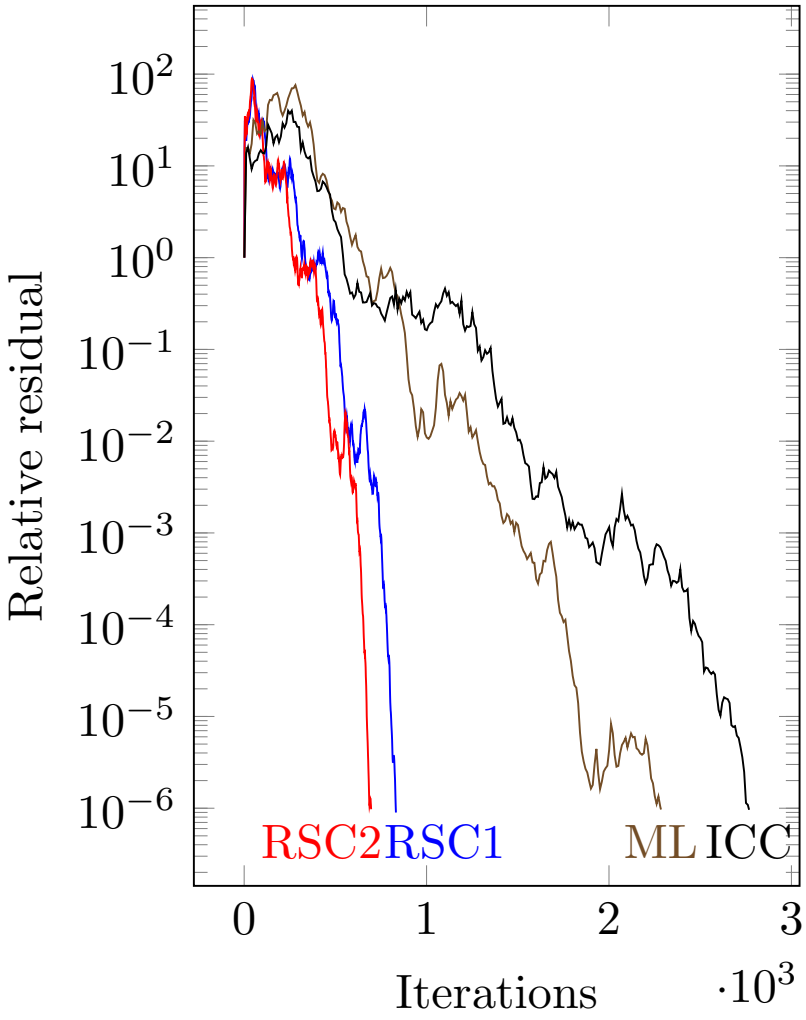
Effect of in-separator ordering

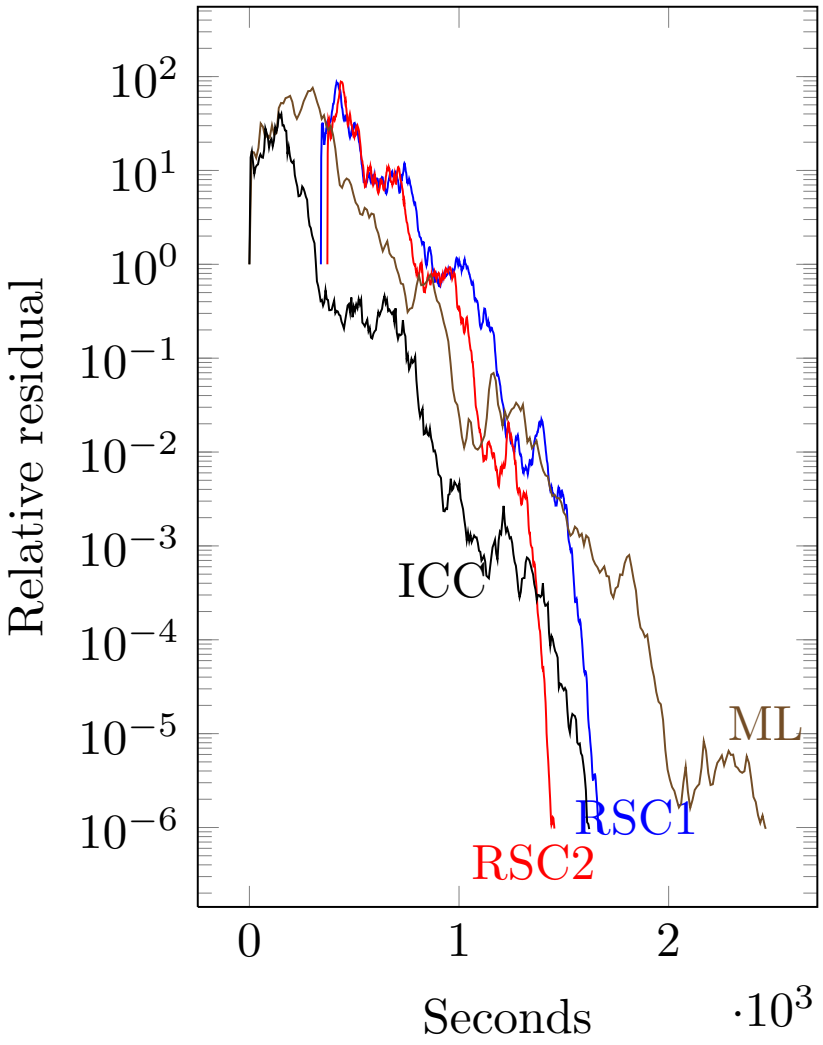


Semi-sep diag relies on variable order – don't want any old order!

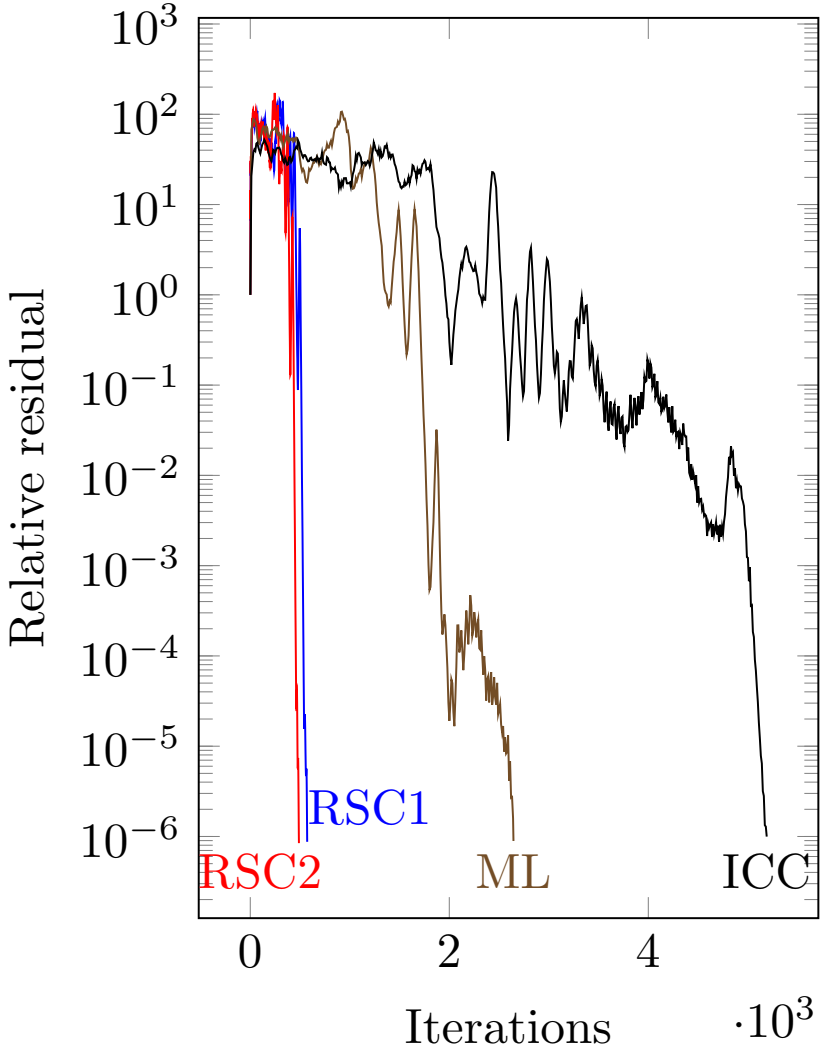
- Apply recursive bisection based on spatial coords
- Use coordinates if known
- Else assign spectrally

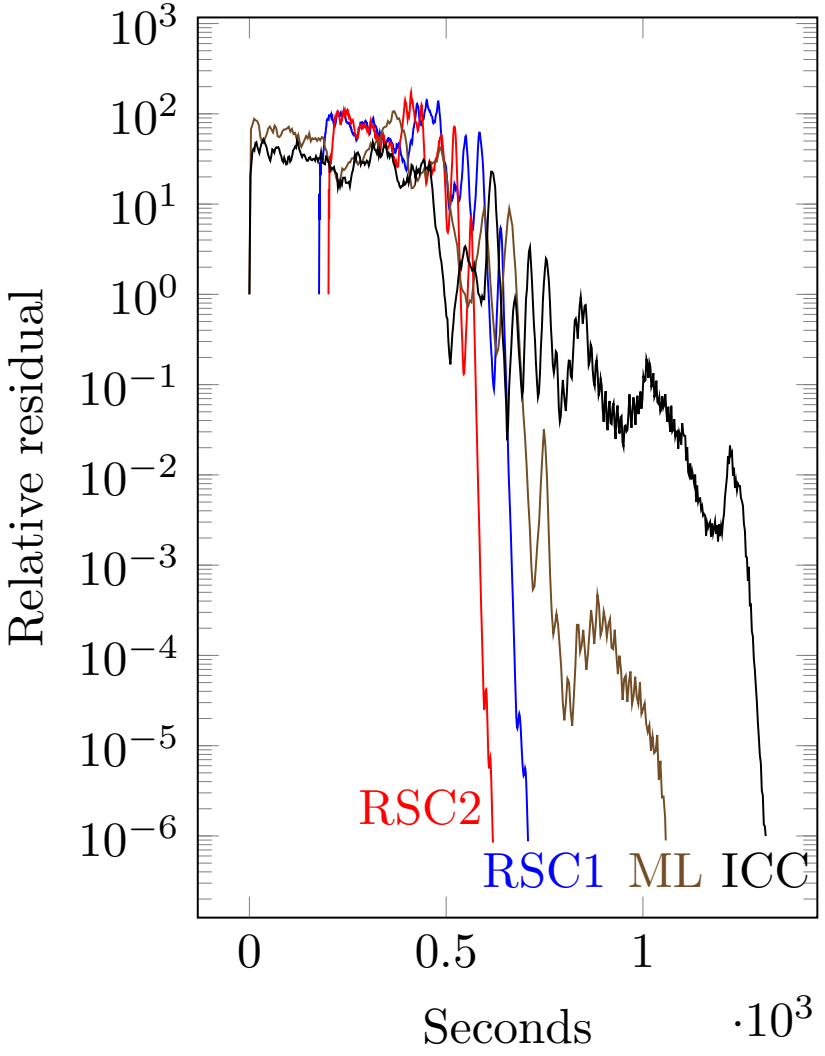
Example: Trabecular bone model ($\approx 1M$ dof)





Example: Steel flange ($\approx 1.5M$ dof)





Competition

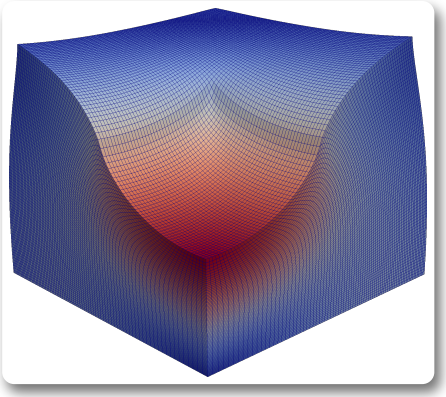
Most direct competitor is STRUMPACK (Li, Ghysels, Rouet):

`http://portal.nersc.gov/project/sparse/strumpack/`

- Multifrontal vs left-looking (extend-add vs “pull” updates)
- LU vs Cholesky
- STRUMPACK is explicitly parallel
- See also Sherry Li’s plenary from SIAM LA 2015

Also work by Xia and Li; HODLR (Aminfar, Ambikasaran, Darve); BLR (Amestoy, Ashcraft, Boiteau, Buttari, L’Excellent, Weisbecker); H-matrix work (Hackbush, Bebendorf, et al); ...

Conclusions



For more:

www.cs.cornell.edu/~bindel
bindel@cs.cornell.edu

J. Chadwick and D. Bindel. An Efficient Solver for Sparse Linear Systems Based on Rank-Structured Cholesky Factorization.
http://arxiv.org/abs/1507.05593

https://github.com/jeffchadwick/rank_structured_cholesky