SAND2019-10650C

Photos placed in horizontal position with even amount of white space between photos and header

Hierarchical Low-rank Preconditioners and Solvers for Linear Systems from PDEs

Erik Boman, Sandia National Labs

CIRM, France, Sept. 18, 2019

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2011-XXXXP

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

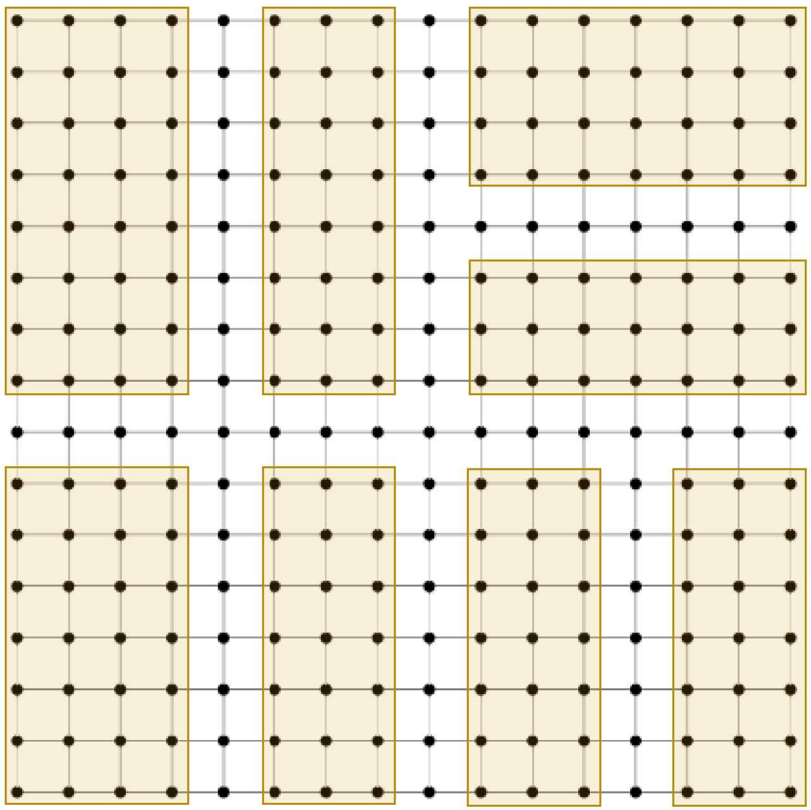
Collaborators

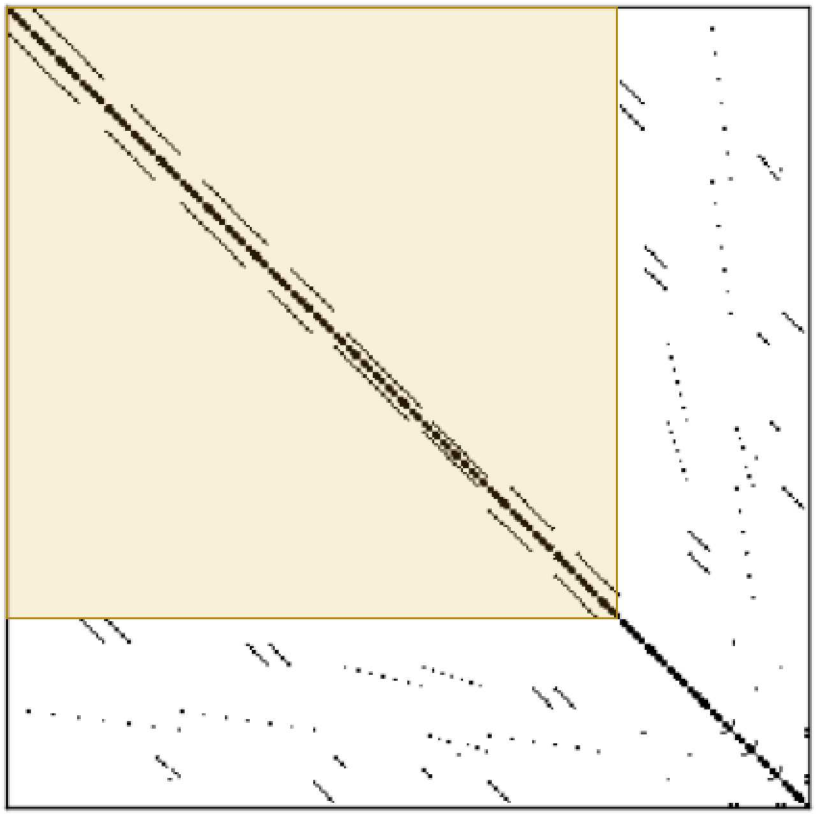
- Sandia:
  - Siva Rajamanickam, Ray Tuminaro, Ichi Yamazaki
- Stanford:
  - Eric Darve, Leopold Cambier
- UT Austin
  - Chao Chen, Will Ruys

Motivation

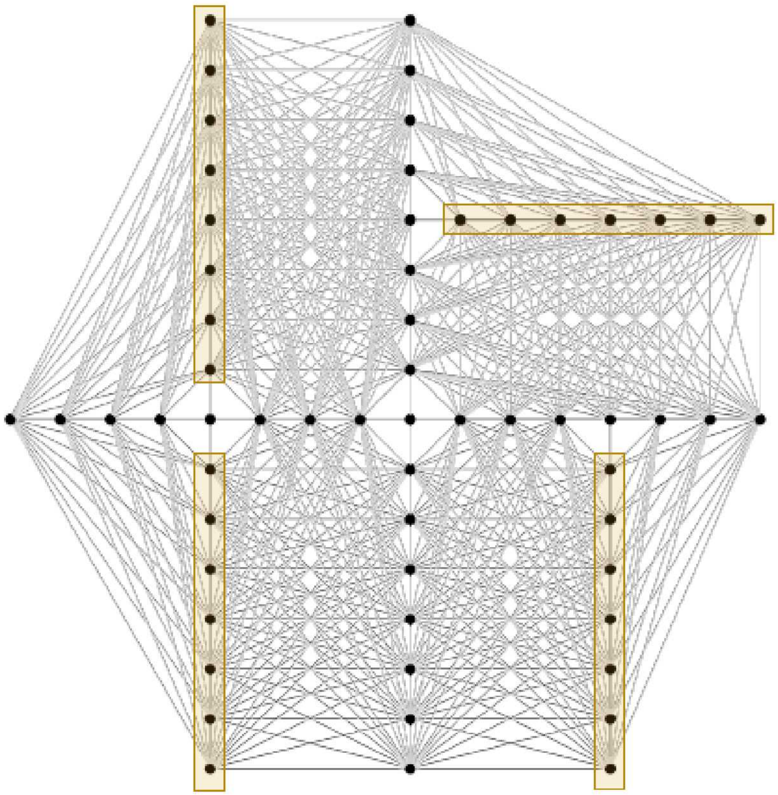
- Sparse direct factorization is robust but too expensive in 3D
- Want robust "black-box" approximate factorization
  - Use as preconditioner
  - Allow trade-off fill versus quality
- Current methods have limitations
  - AMG and DD are scalable but often not robust (e.g., indefinite, nonsym.)
  - Others (Incomplete factorizations, Sparse approximate inverses etc) are algebraic (black-box) but not scalable
- Hierarchical matrix methods could fill this gap
  - Many different algorithms
- Our focus: SpaND (Sparsified Nested Dissection)
  - Similar to HIF method (Ho & Ying)

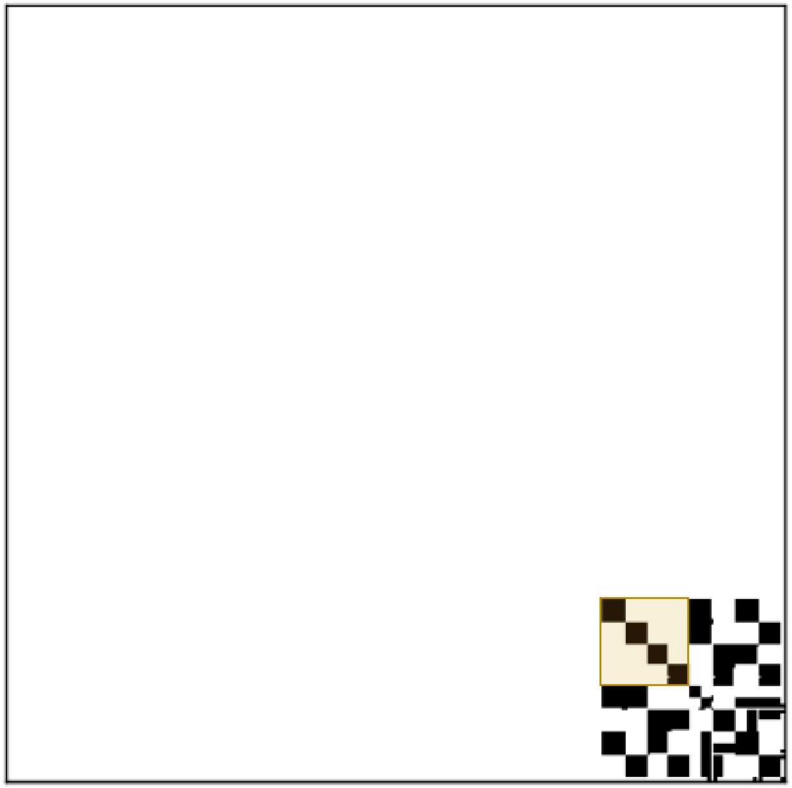
Sparse Factorization (1)



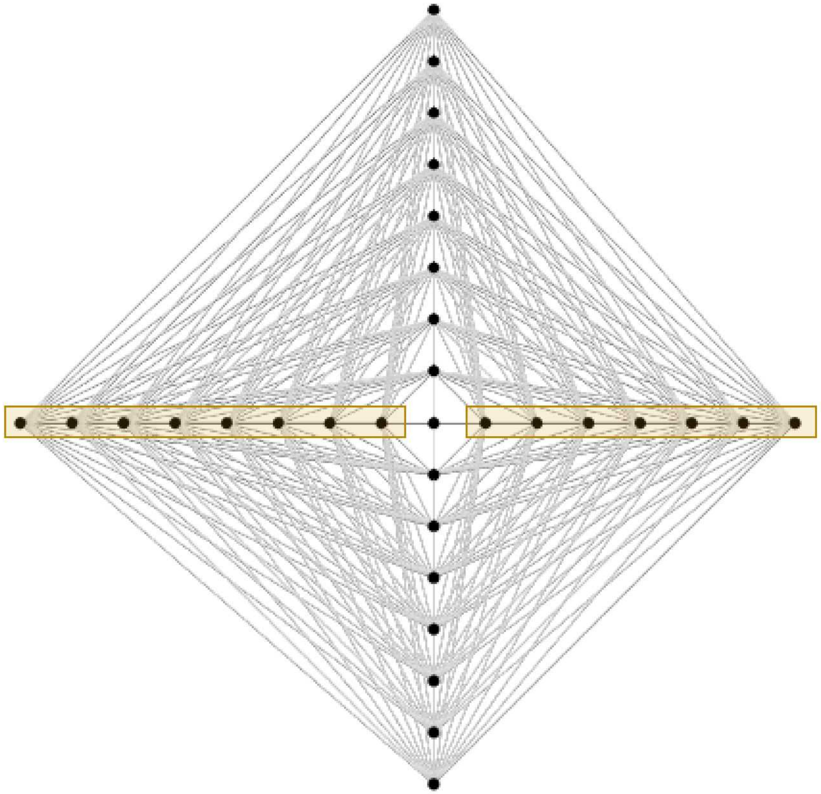


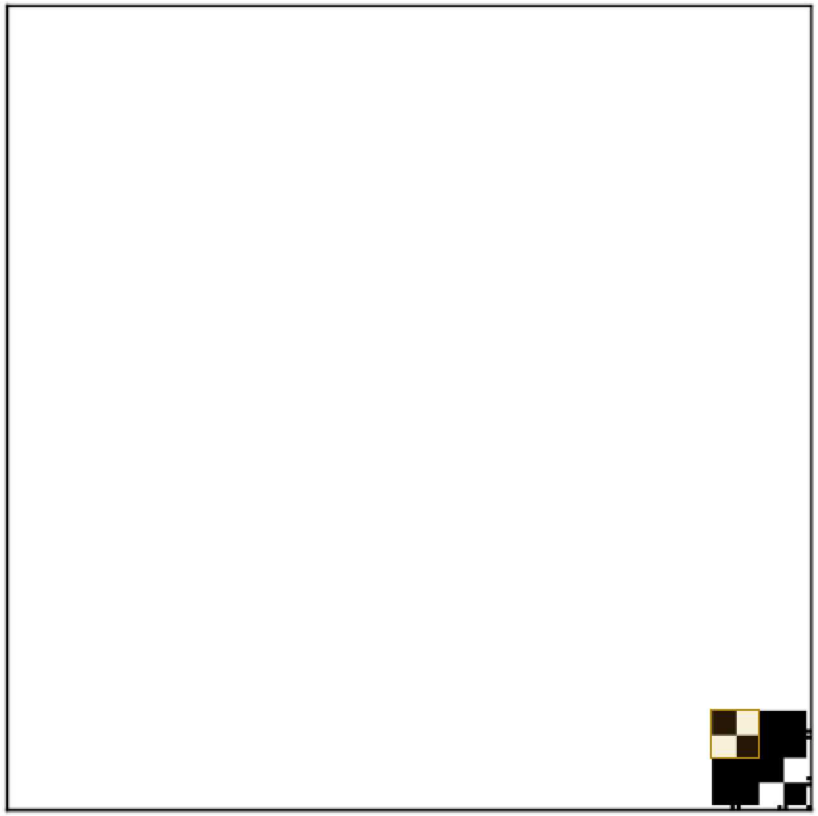
Sparse Factorization (2)



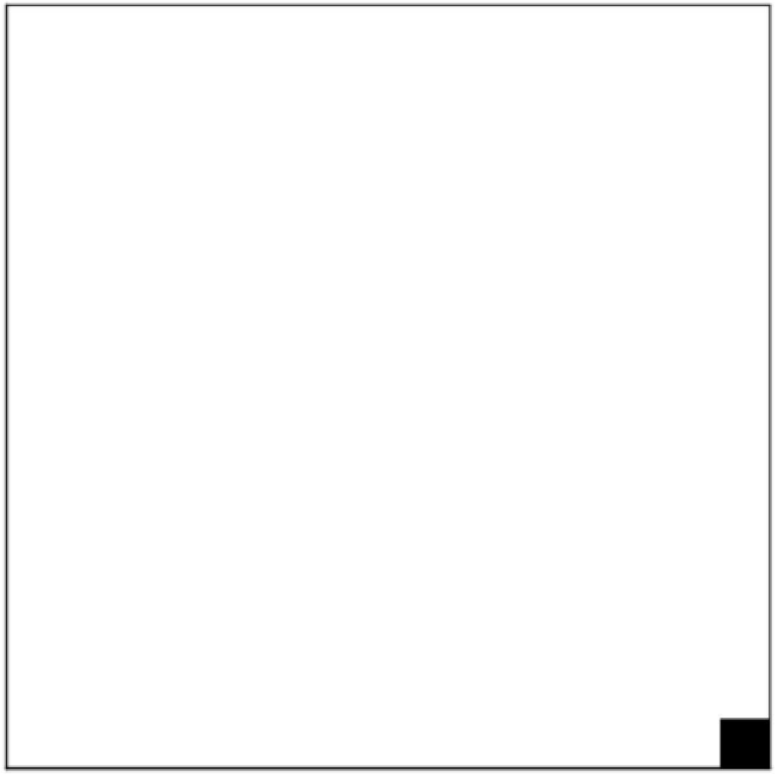


Sparse Factorization (3)

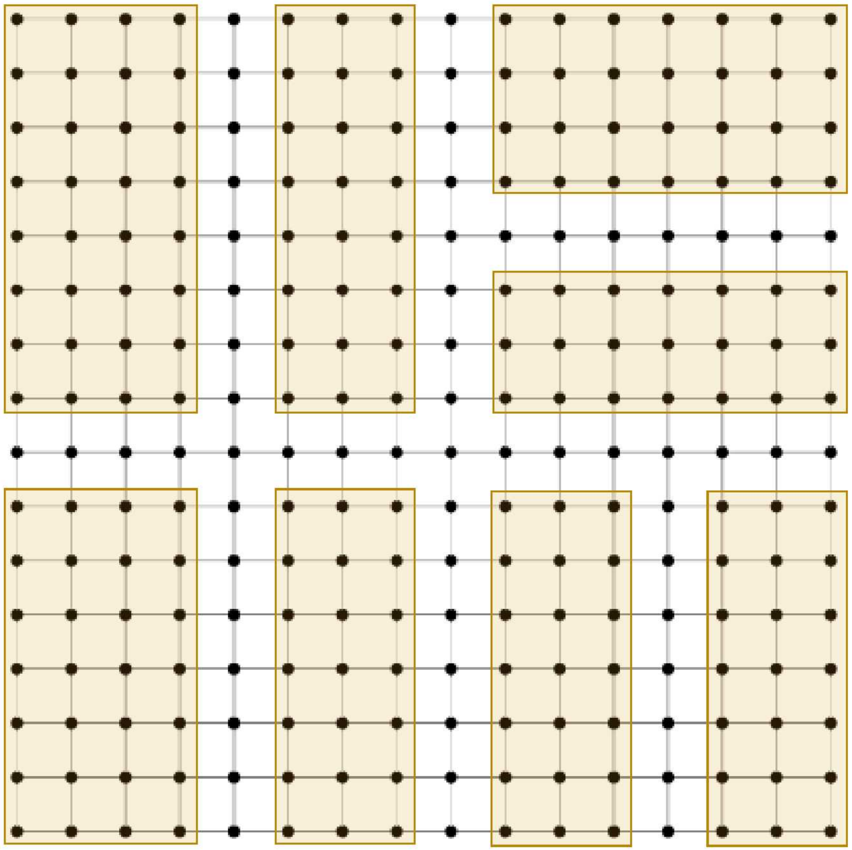


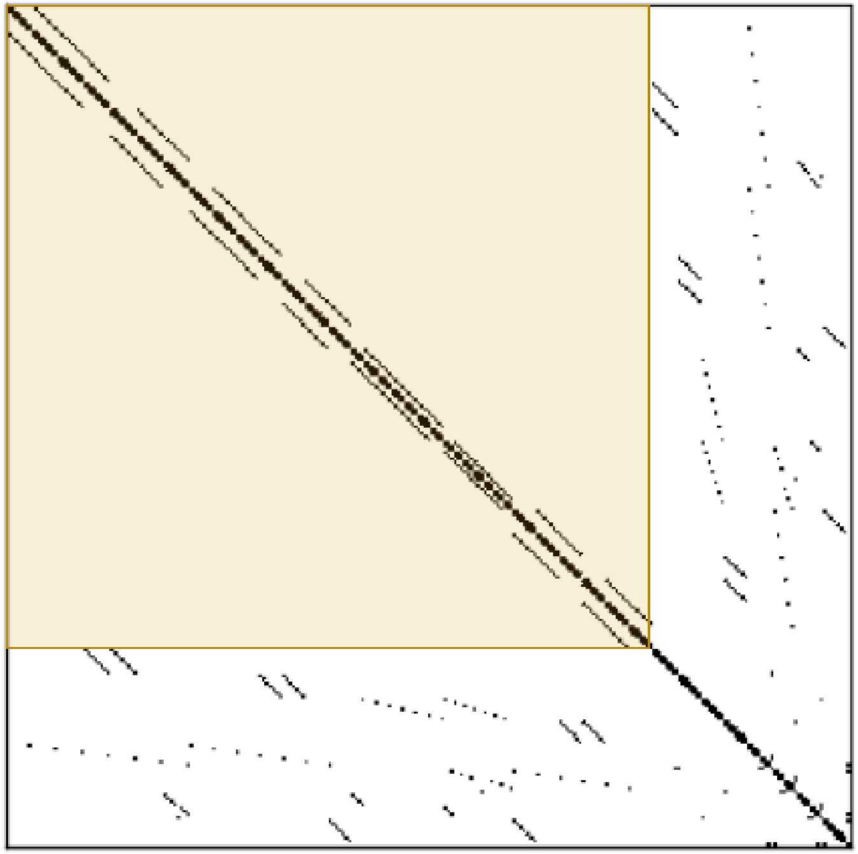


Sparse Factorization (4)

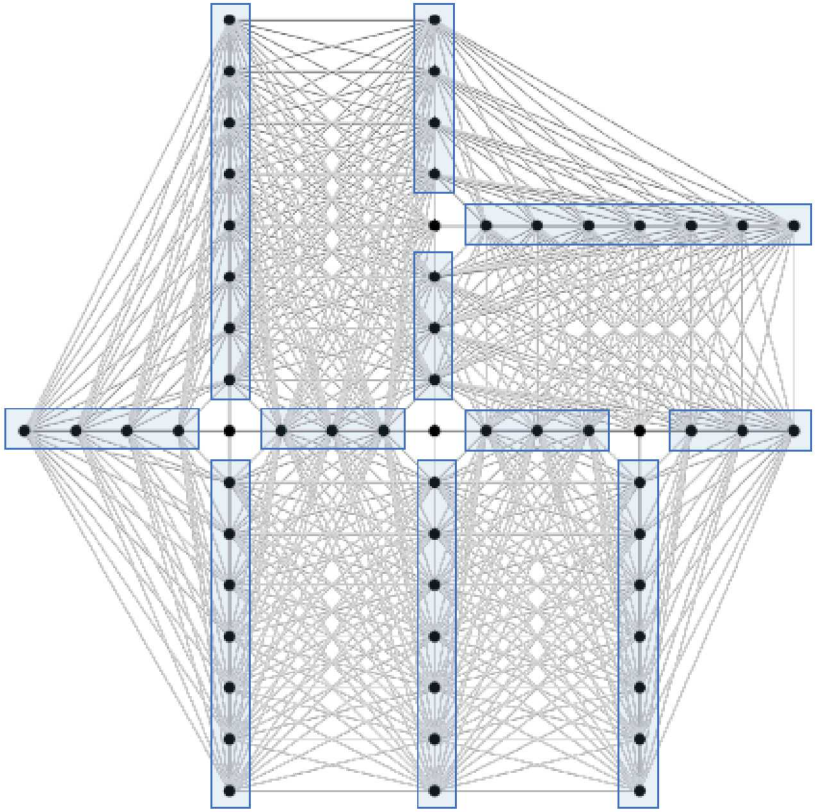


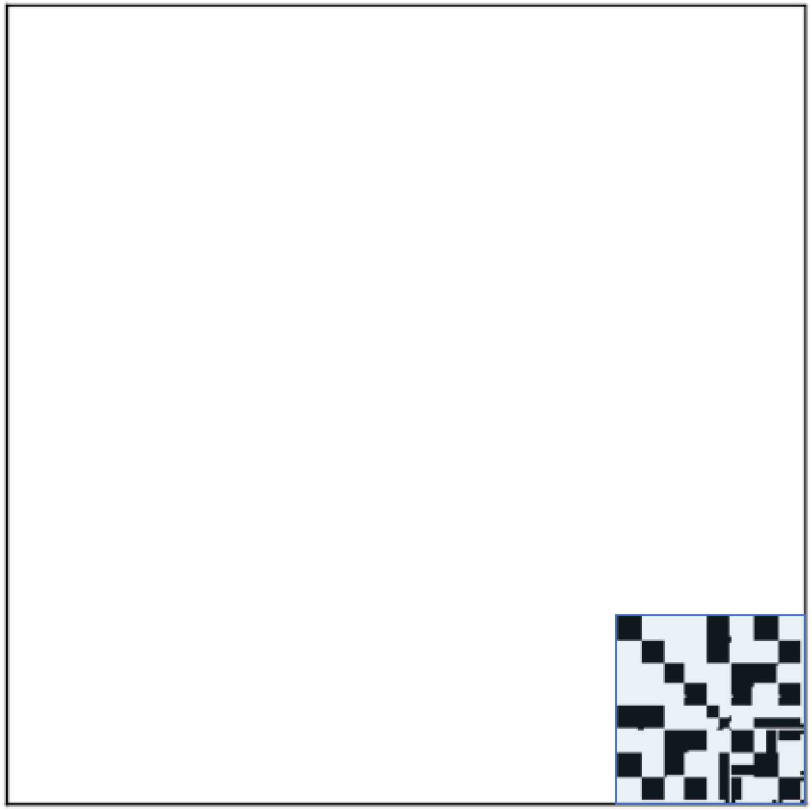
Sparsified Approx. Factorization (1)





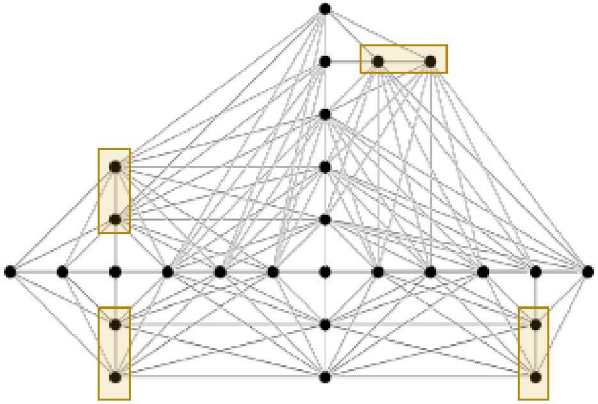
Sparsified Approx. Factorization (2)

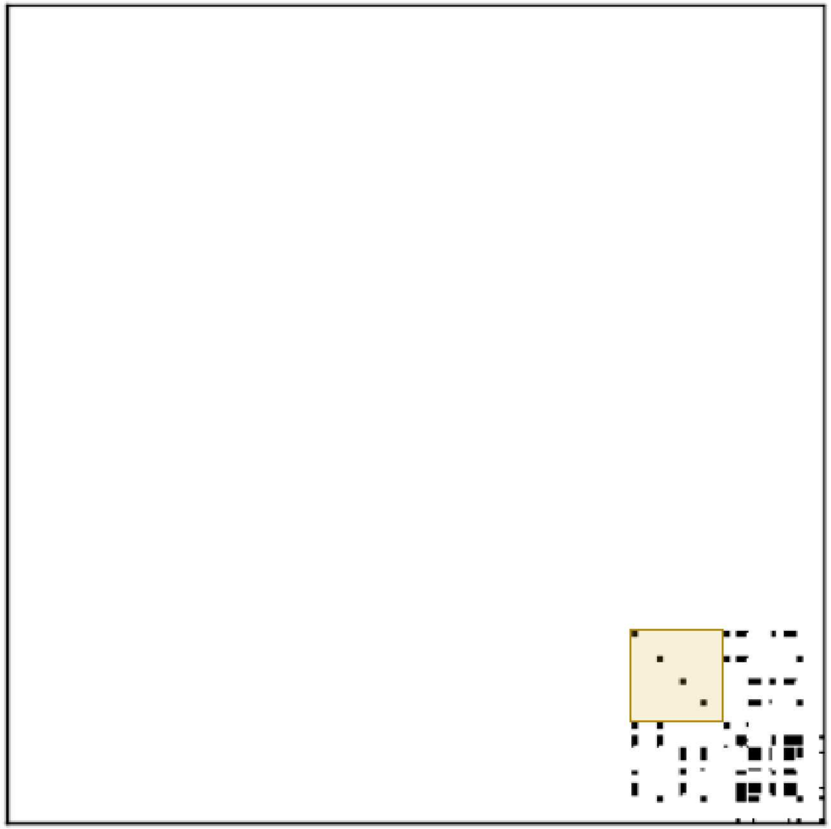




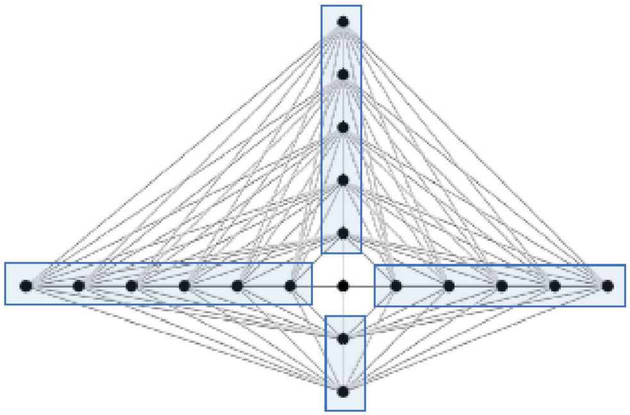
Sparsified Approx. Factorization (3)

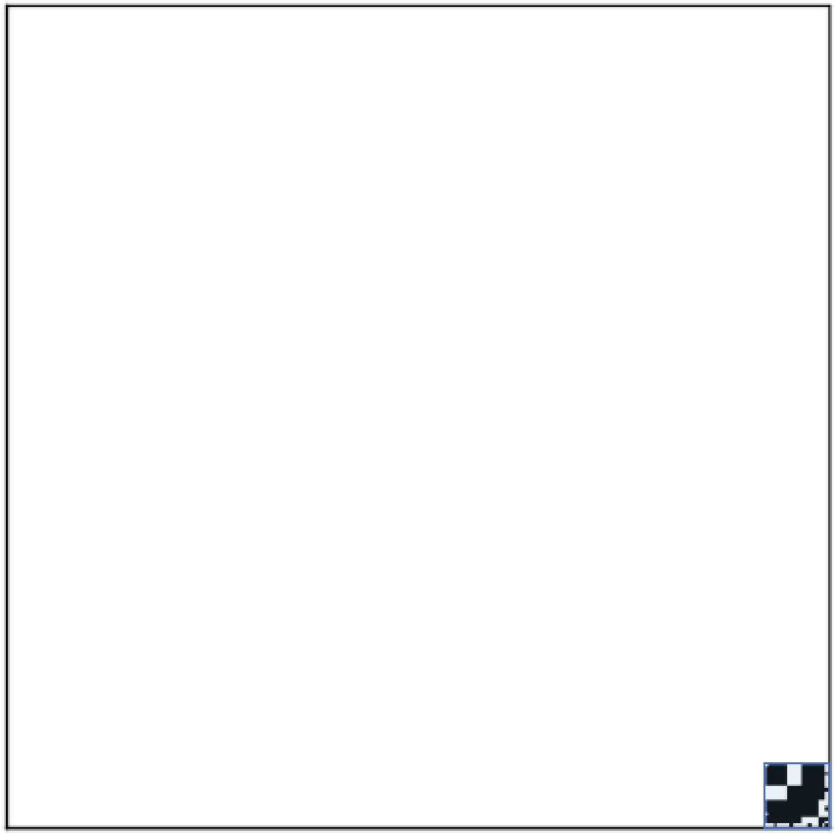
$\approx$





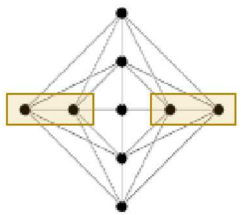
Sparsified Approx. Factorization (4)

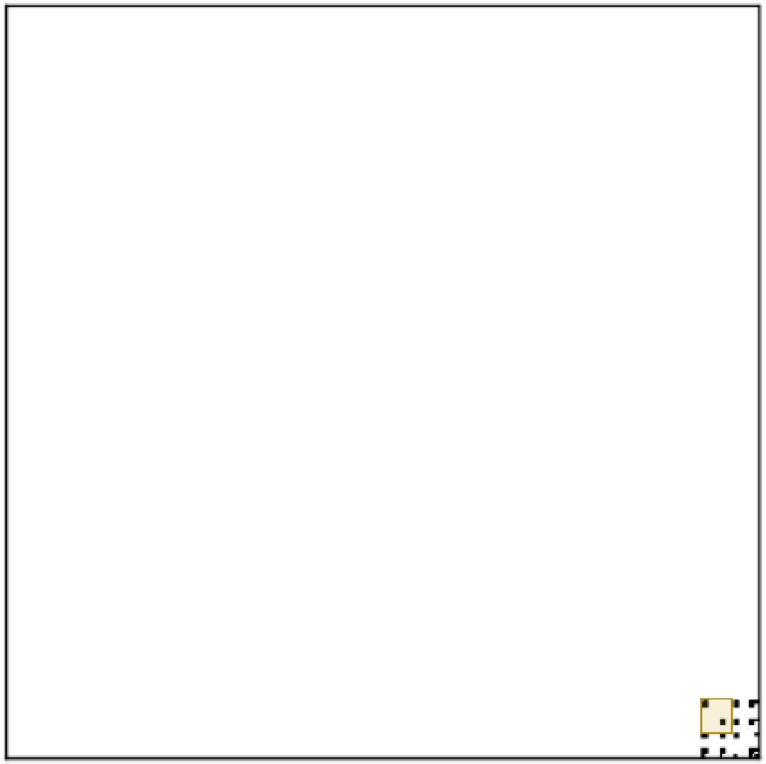




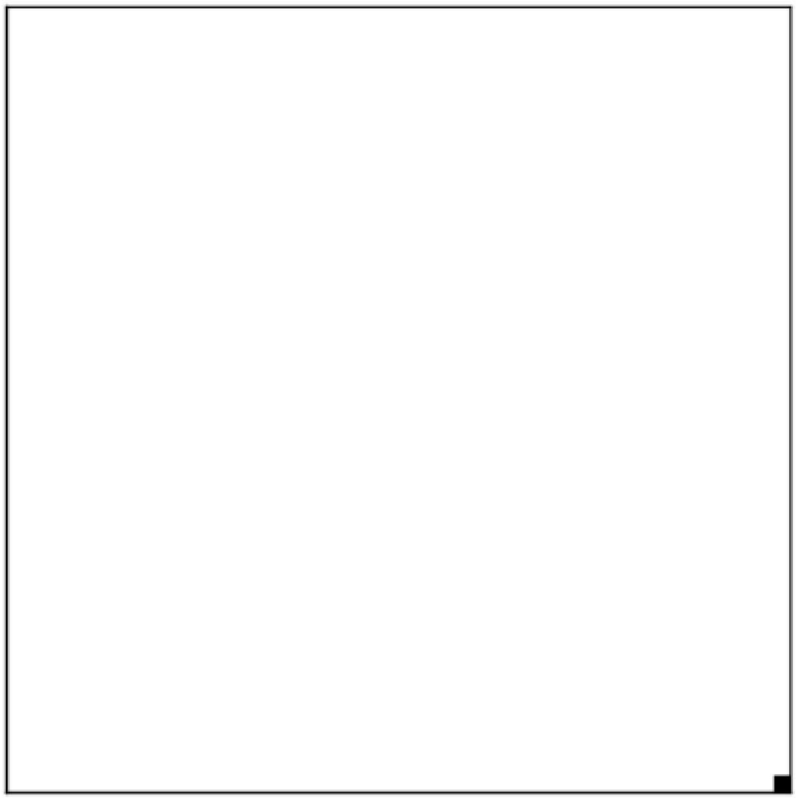
Sparsified Approx. Factorization (5)

≈



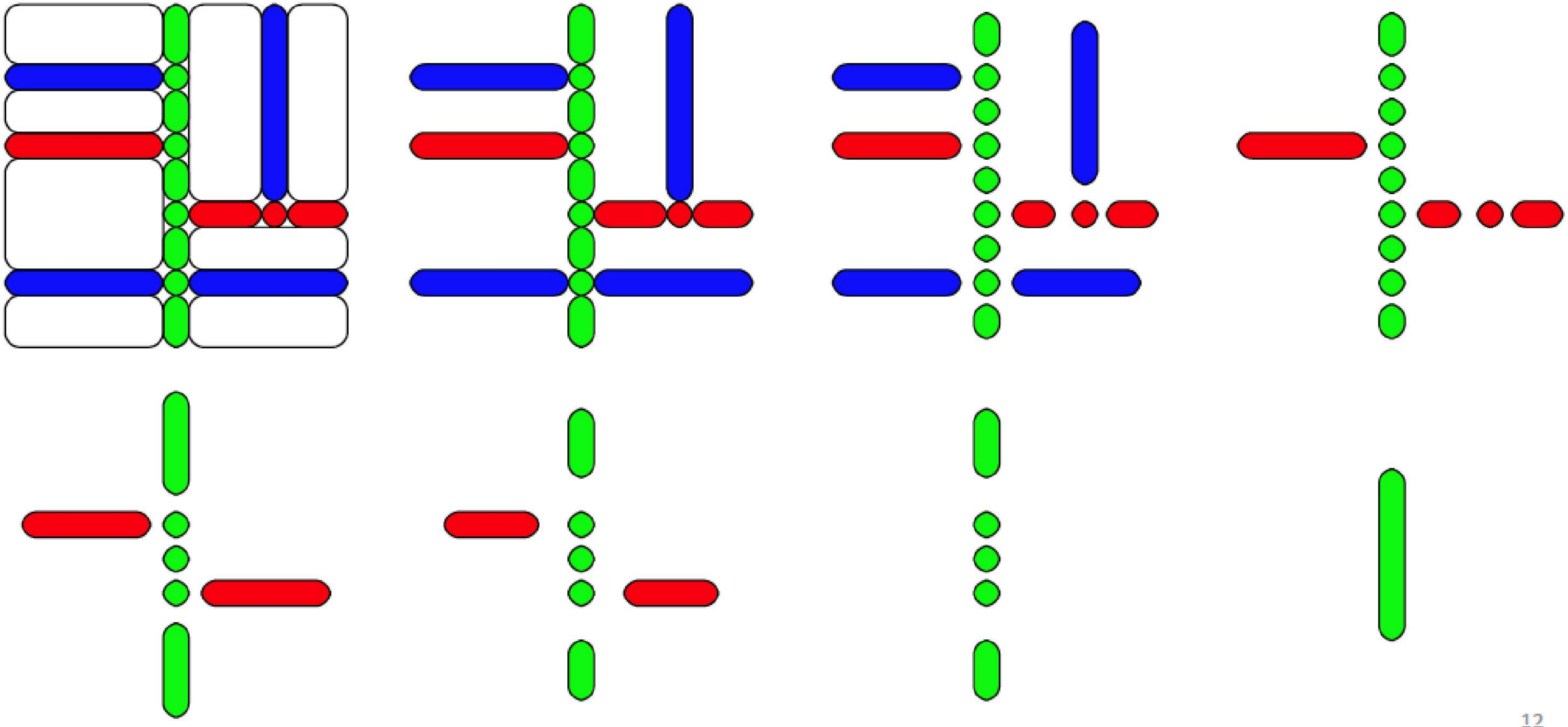


Sparsified Approx. Factorization (6)



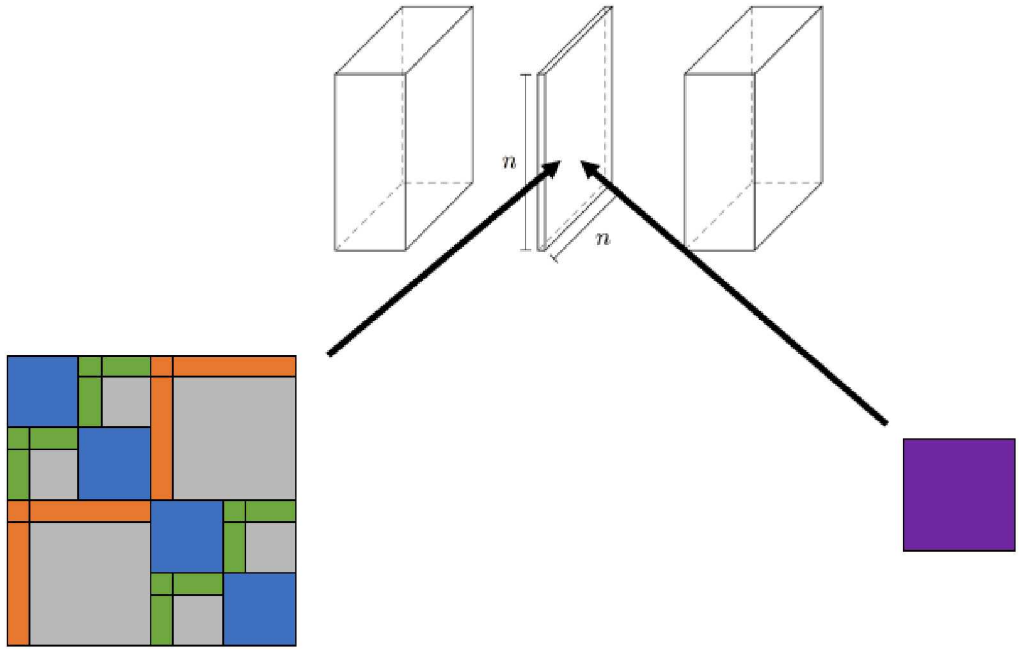
SpaND Summary

- Sparsify separators (low-rank compression) during elimination



Different from fast-algebra on dense

- Common approach: Fast algebra (H/HSS/BLR) on dense blocks
  - Ex: Strumpack, MUMPS, PasTix, etc.
- Instead we reduce the size of the separator blocks!



Sparsification Step

- Block scaling, low-rank elimination, drop negligible blocks

$$
\underbrace{\underbrace{\begin{bmatrix} L_{ss}^{-1} & & \\ & I & \\ & & I \end{bmatrix} \begin{bmatrix} A_{ss} & & A_{sn} \\ & A_{ww} & A_{wn} \\ A_{ns} & A_{nw} & A_{nn} \end{bmatrix} \begin{bmatrix} L_{ss}^{-\top} & & \\ & I & \\ & & I \end{bmatrix}}
\begin{bmatrix} Q^\top & & \\ & I & \\ & & I \end{bmatrix} \begin{bmatrix} I & & A_{sn} \\ & A_{ww} & A_{wn} \\ A_{ns} & A_{nw} & A_{nn} \end{bmatrix} \begin{bmatrix} Q & & \\ & I & \\ & & I \end{bmatrix}}
$$

$$
\begin{bmatrix} I & & & \varepsilon \\ & I & & W_{cn} \\ & & A_{ww} & A_{wn} \\ \varepsilon & W_{cn}^\top & A_{nw} & A_{nn} \end{bmatrix} \qquad \varepsilon \cong 0
$$

Sparsification via Low-rank Approx.

We need low-rank approximation of off-diagonal (rectangular) block.

1. Interpolative decomposition (ID)
   - Computed via RRQR (QRCP)
   - A.k.a. skeletonization
2. Orthogonal transform
   - Use RRQR or SVD
   - More stable, but may be more expensive

- For both methods there is a user parameter ε
  - Trade-off accuracy vs cost

Sparsification 1: ID

(1) We start with

$$\begin{bmatrix} A_{ss} & & A_{sn} \\ & A_{ww} & A_{wn} \\ A_{ns} & A_{nw} & A_{nn} \end{bmatrix}$$

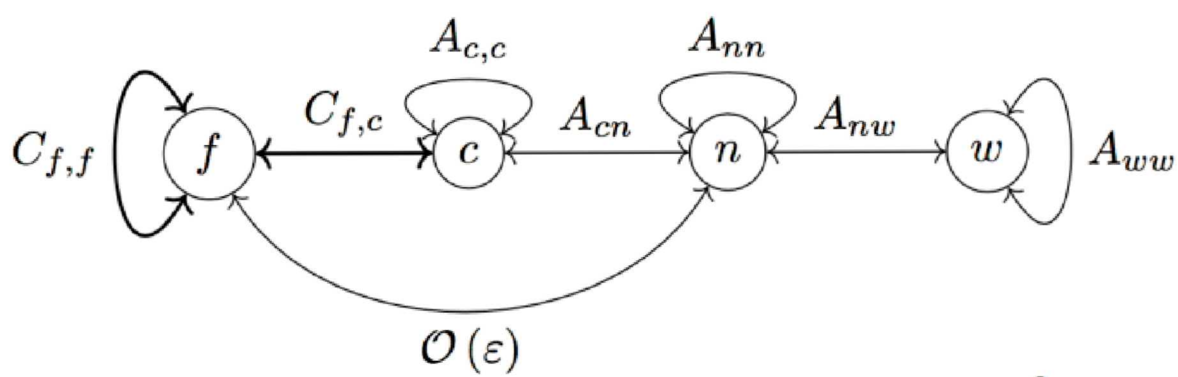
(2) We then approximate

$$A_{sn} = \begin{pmatrix} T_{fc} \\ I \end{pmatrix} A_{cn} + \varepsilon$$

$$s = f \cup c$$

(3) We end up with

$$\begin{bmatrix} C_{ff} & C_{fc} & & \varepsilon \\ C_{cf} & A_{cc} & & A_{cn} \\ & & A_{ww} & A_{wn} \\ \varepsilon & A_{nc} & A_{nw} & A_{nn} \end{bmatrix}$$



8

Sparsification 2: Orthogonal

(1) We start with

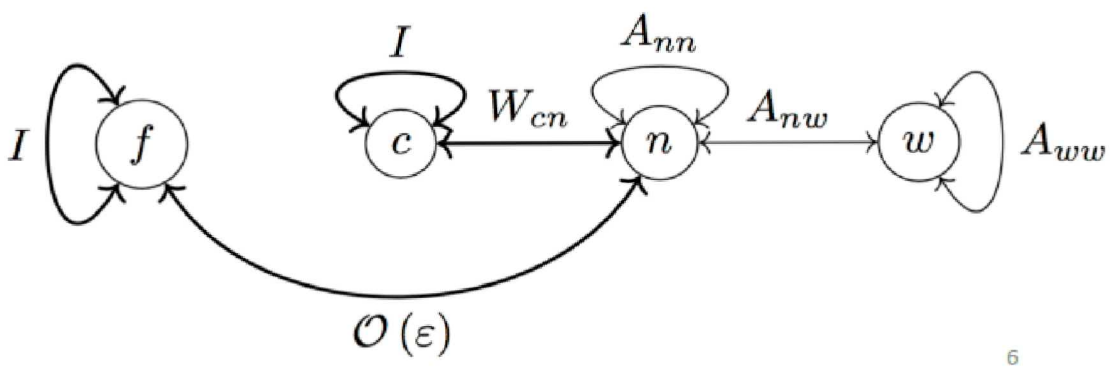
$$\begin{bmatrix} I & & A_{sn} \\ & A_{ww} & A_{wn} \\ A_{ns} & A_{nw} & A_{nn} \end{bmatrix}$$

(2) We then approximate

$$A_{sn} = Q_{sc} W_{cn} + \varepsilon$$
$$Q^\top s = f \cup c$$

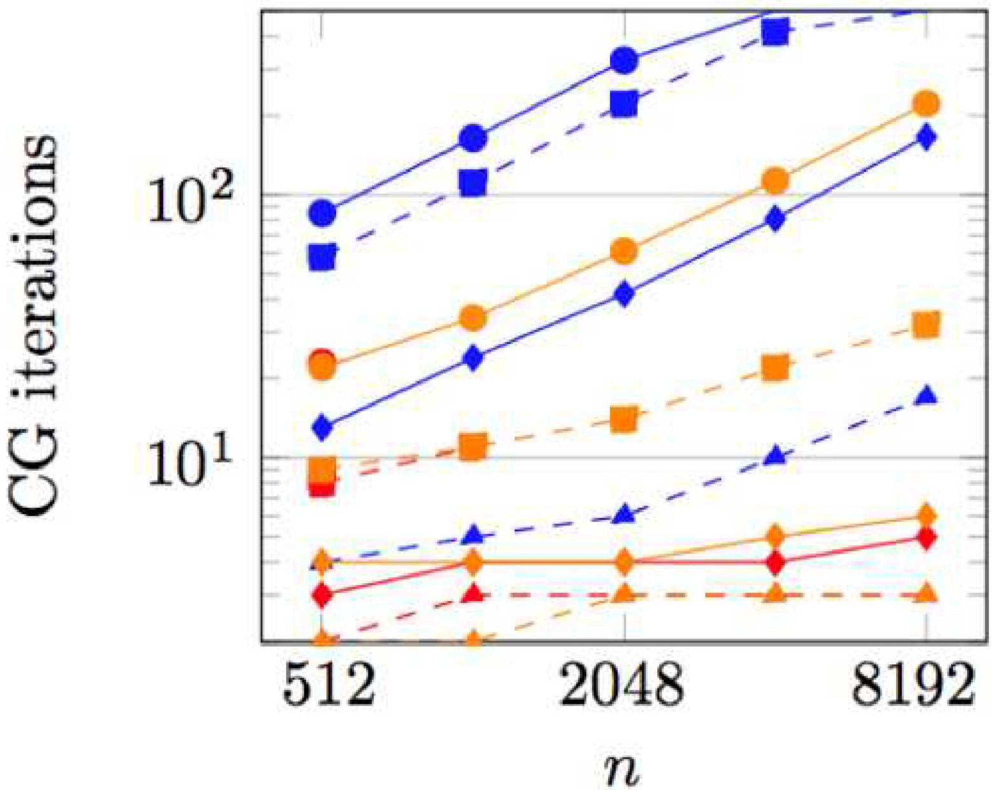
(3) We end up with

$$\begin{bmatrix} I & & & \varepsilon \\ & I & & W_{cn} \\ & & A_{ww} & A_{wn} \\ \varepsilon & W_{cn}^\top & A_{nw} & A_{nn} \end{bmatrix}$$



6

Results: 2D Laplacians



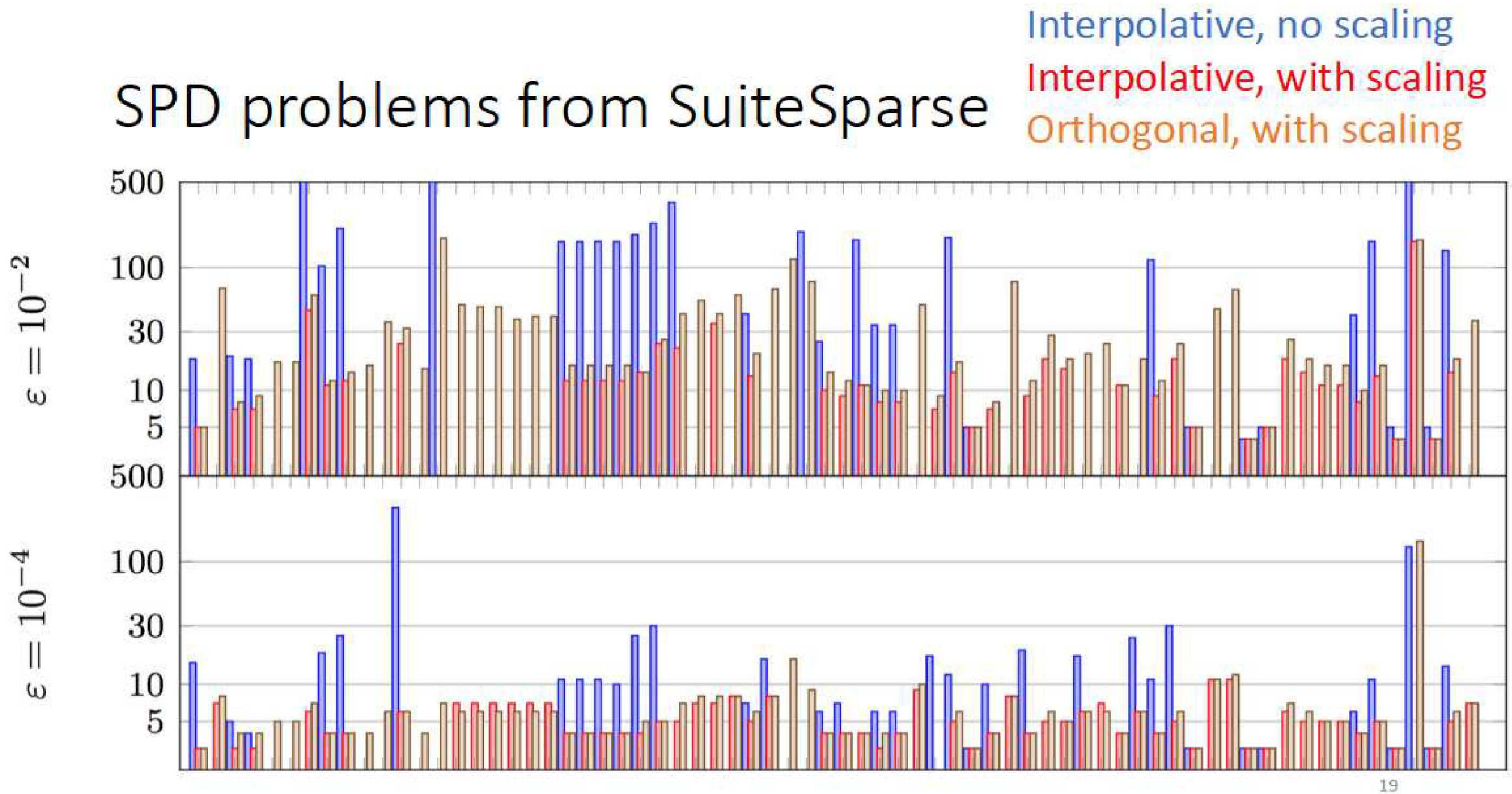
Interpolative, no scaling

Interpolative, with scaling

Orthogonal, with scaling

$$\varepsilon = 10^{-1} \to 10^{-6}$$

Results: SuiteSparse Collection



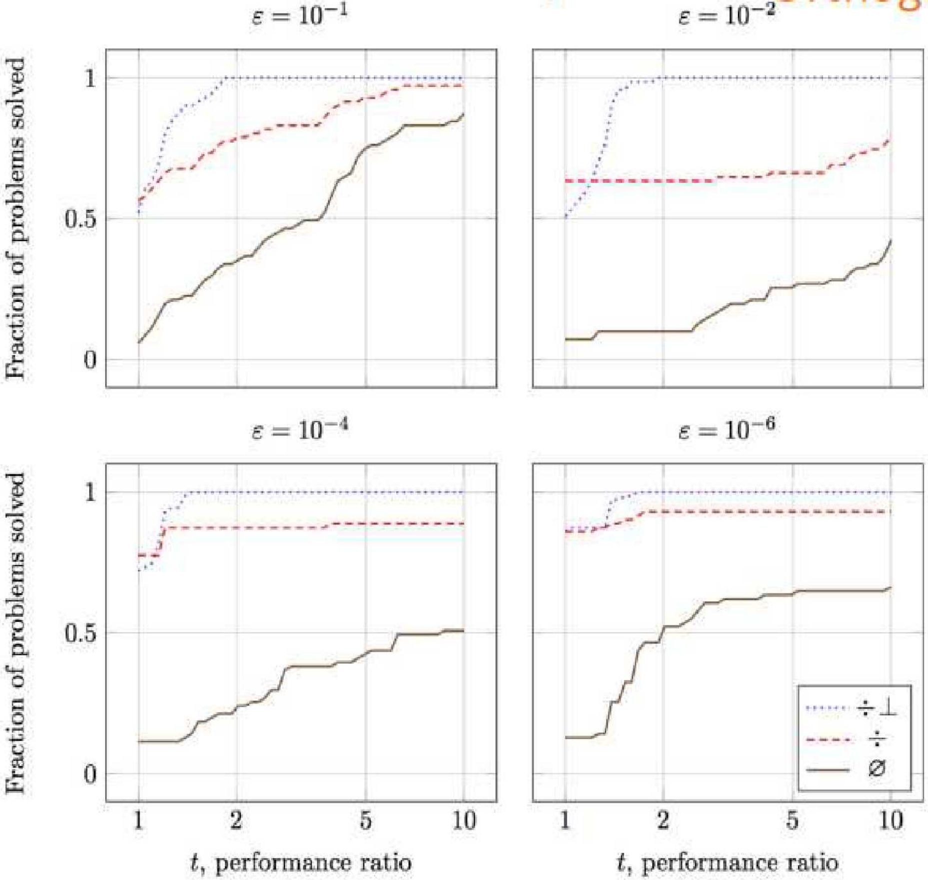
Results: Performance Profile

Interpolative, no scaling
Interpolative, with scaling
Orthogonal, with scaling

SPD problems from SuiteSparse

Perf ratio(t) =

$$\frac{\#\left\{p \in P \mid \frac{CG_{pv}}{CG_p^*} \le t\right\}}{\#P}$$

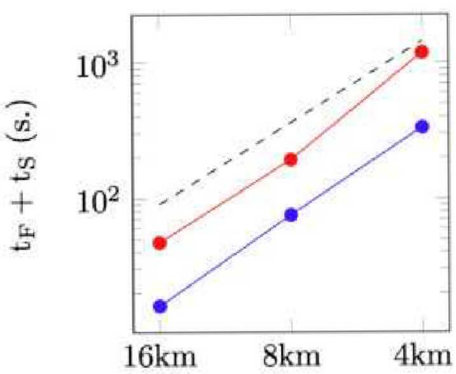


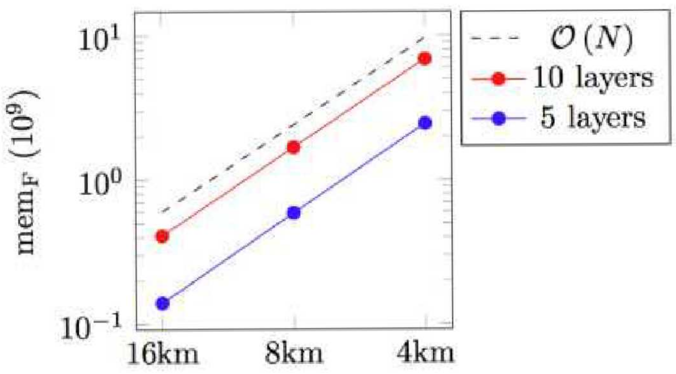
20

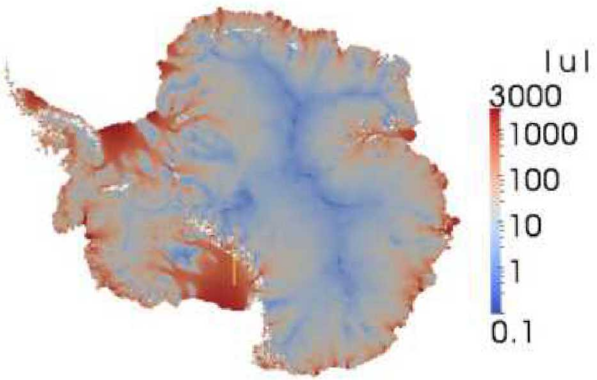
Results:

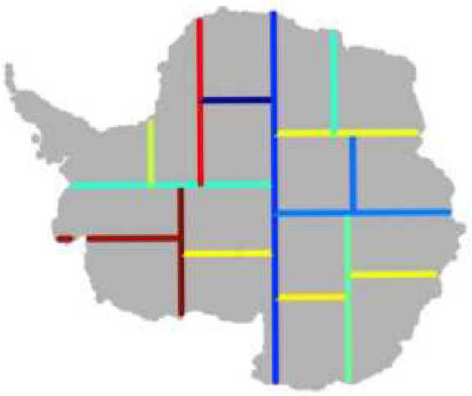
Ice-Sheet modeling $\kappa(A) > 10^{11}$

N	spaND t_F (s.)	t_S (s.)	n_{CG}	size$_{\text{Top}}$	mem$_F$ (10^9)	Direct $t_F + t_S$ (s.)
5 layers						
629 544 (16 km)	13	3	7	76	0.14	22
2 521 872 (8 km)	55	20	8	89	0.59	206
10 096 080 (4 km)	217	115	10	100	2.45	1578
10 layers						
1 154 164 (16 km)	39	8	7	136	0.41	90
4 623 432 (8 km)	148	44	8	148	1.68	710
18 509 480 (4 km)	798	384	10	159	6.86	—







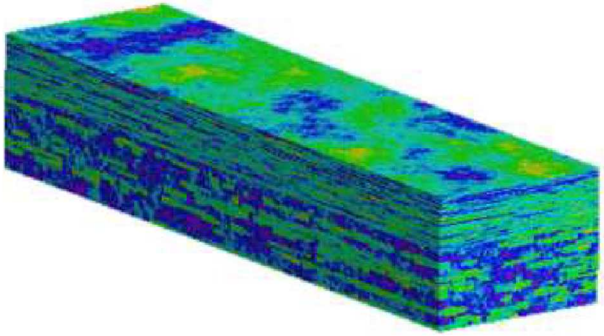


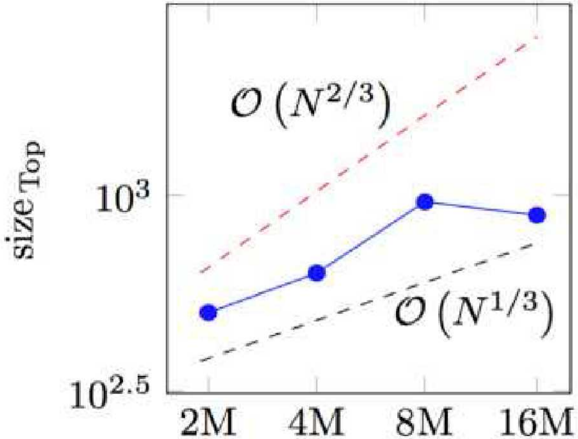
21

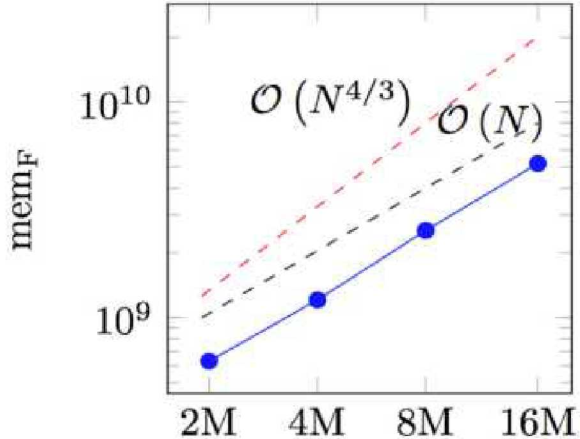
Results: SPE

The SPE problem

n	$N = n^3$	spaND t_F (s.)	t_S (s.)	n_{CG}	size_{Top}	mem_F (10^9)	Direct. $t_F + t_S$ (s.)
128	2 097 152	61	23	12	502	0.63	686
160	4 096 000	175	46	13	634	1.21	—
200	8 000 000	287	158	16	962	2.54	—
252	16 003 008	963	369	16	890	5.19	—





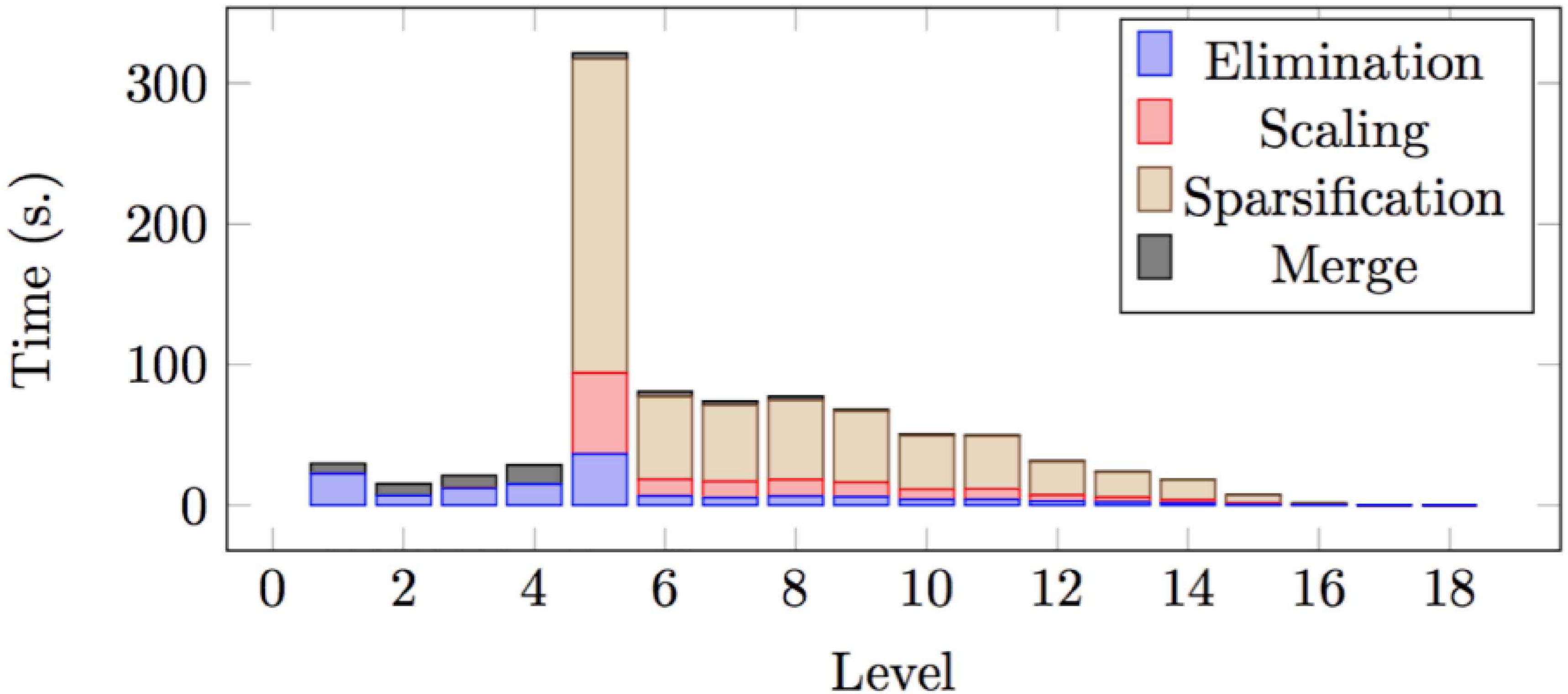


Top separator block would be 32 GB without the sparsification!

22

Profiling

- Most expensive part is sparsification (RRQR)
- Skip sparsification on bottom levels (no benefit)



Parallel Approaches

We are exploring two approaches for parallel SpaND:

- Task-based
  - Dynamic scheduling works well on shared-memory systems
- Level-based
  - Process level-by-level, going up the tree
  - Need batched BLAS/LAPACK, many small operations in parallel
  - Use Kokkos library to run on both CPU qnd GPU

- This is work in progress.

Conclusions

- SpaND is a clever approximate factorization
  - combines features from sparse direct and hierarchical matrices
- Tunable trade-off factorization cost and preconditioner quality
  - Observed near-linear scaling (total time) on many problems
- Based on HIF but several improvements
  - Unstructured, block scaling, orthogonal compression, etc.
- We focused on SPD case (Cholesky) but
  - Method can be generalized to nonsymmetric (LU)
  - Work in progress

References

- SpaND

- *SpaND: An Algebraic Sparsified Nested Dissection Algorithm Using Low-Rank Approximations*, L. Cambier, C. Chen, E.G. Boman, S. Rajamanickam, R.S. Tuminaro, E. Darve, 2019, under review (on arXiv)

- HIF

  - *Hierarchical interpolative factorization for elliptic operators: differential equations*, K. Ho and L. Ying, Comm. On Pure and Applied Math., v.69, 2016