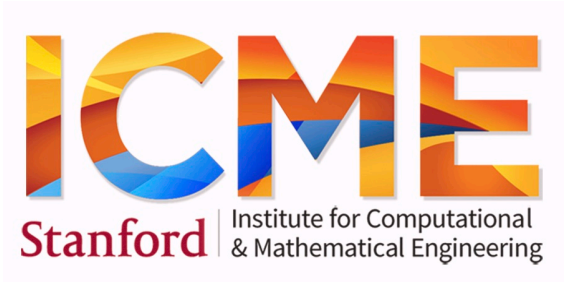


Fast Linear Solvers for Exascale Computing

Léopold Cambier

with B. Klockiewicz, E. Darve, C. Chen,
E. Boman, S. Rajamanickam, R. Tuminaro

June 11, 2019

What & Why?

Fast Linear Solvers

Lack of generic robust solver

- Ill-conditioning (large or stiff problems)
- Non-elliptic PDEs
- No geometry

Solution: Sparsified Algorithms

- Algebraic
- Robust
- Fast on a large class of problems

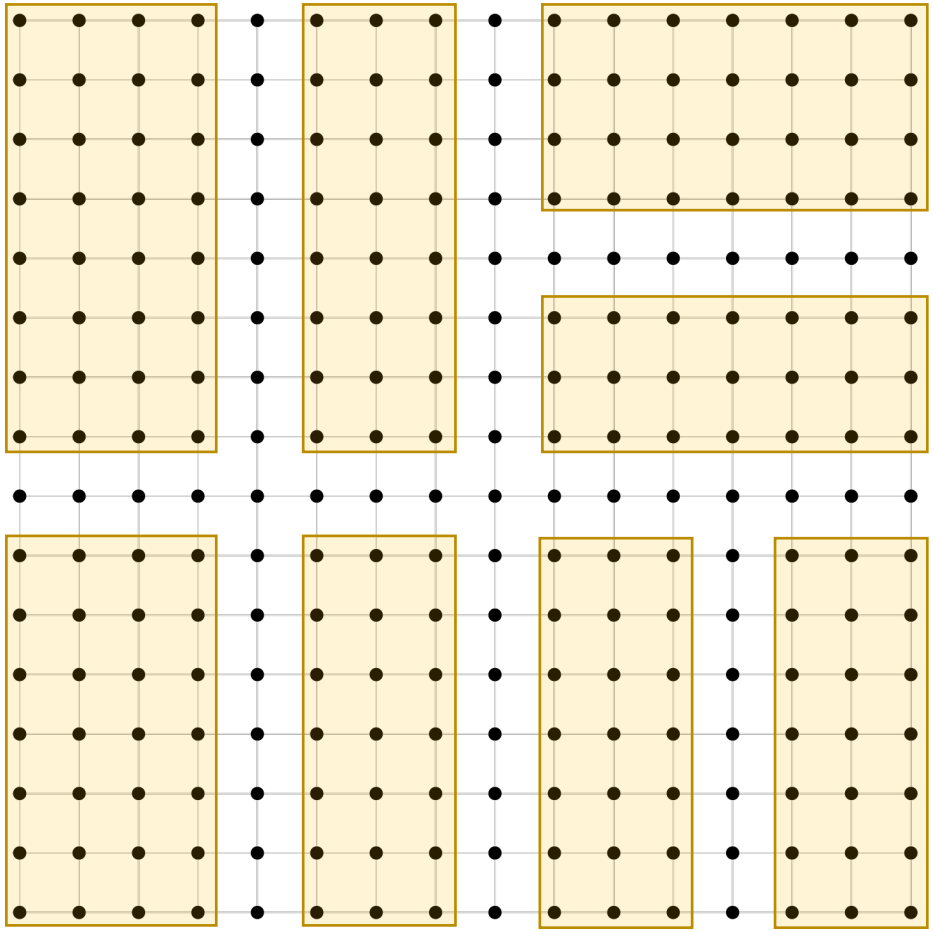
Parallel Computing

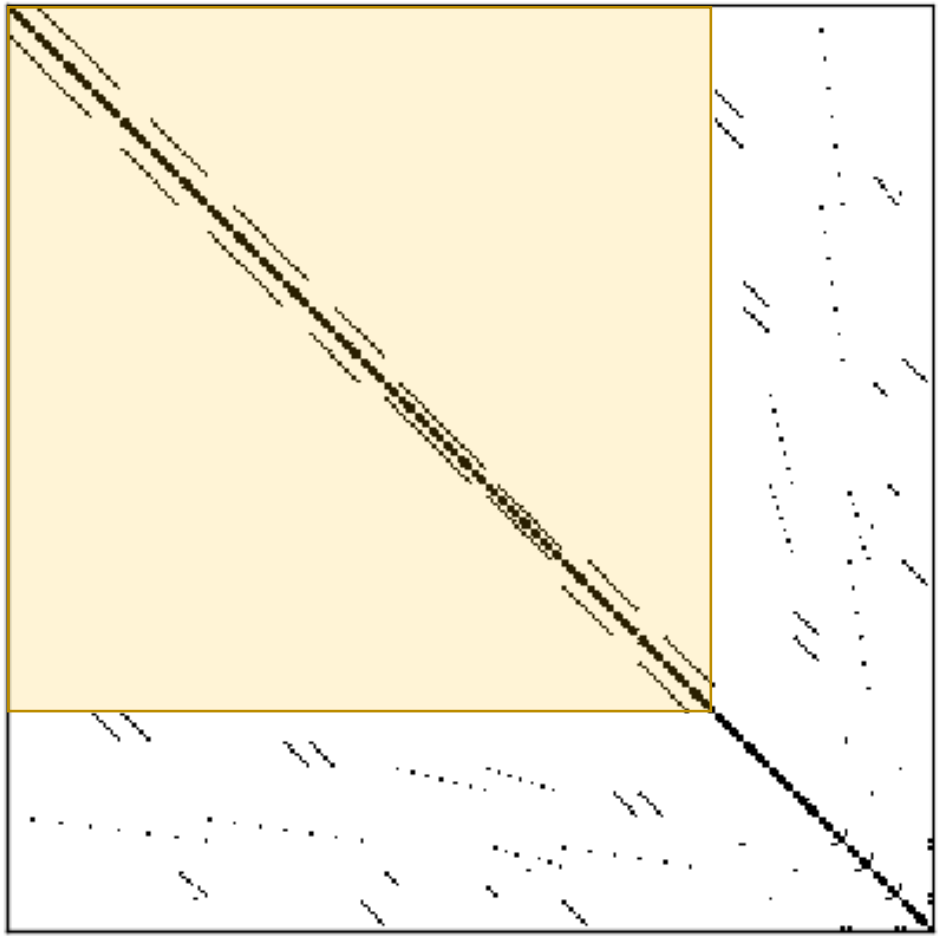
- Compute is cheap (1 flop = few ns)
- Comms are expensive (IB latency = few us)

Solution: TMI, Task and Message Interface
100% asynchronous, using

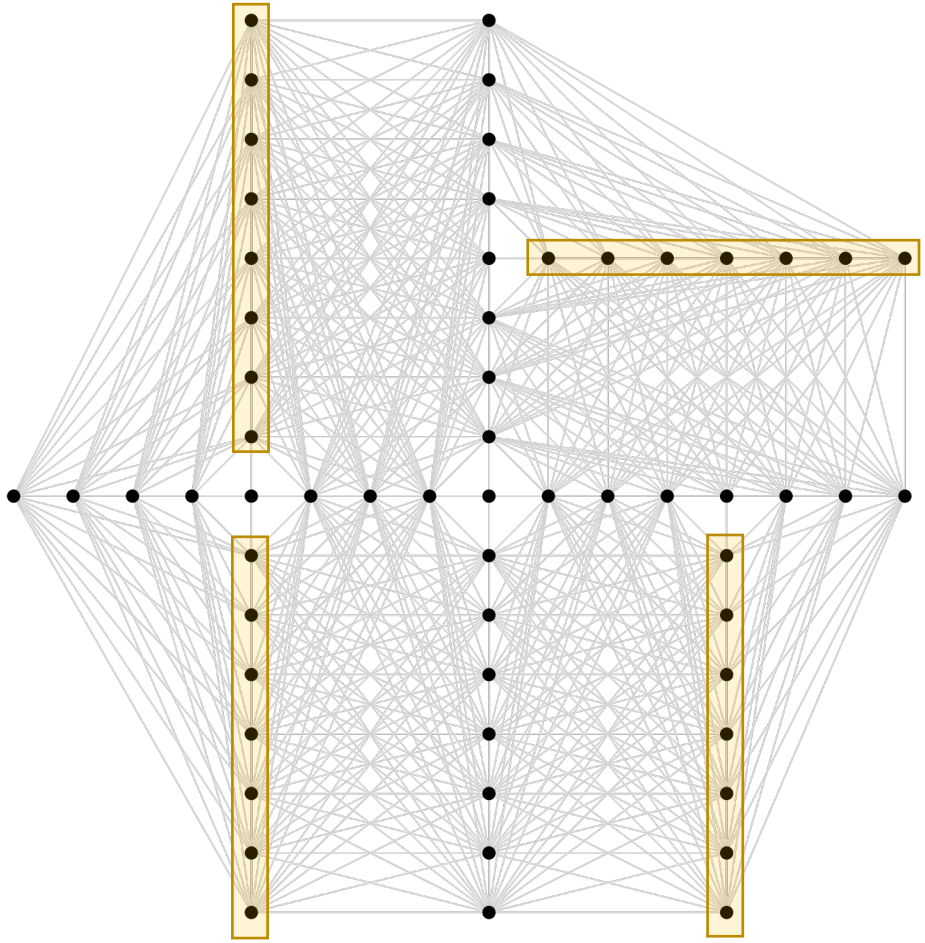
- Tasks-based parallelism
- One-sided asynchronous message passing

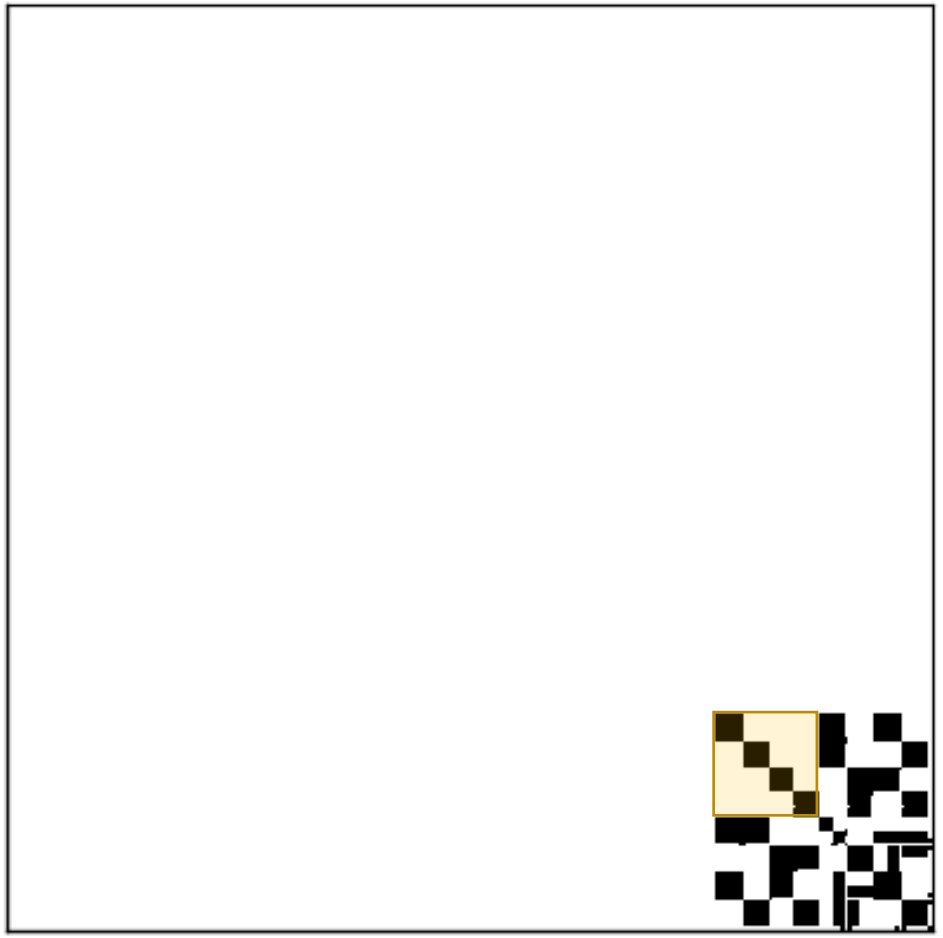
Sparse Linear Systems (1)



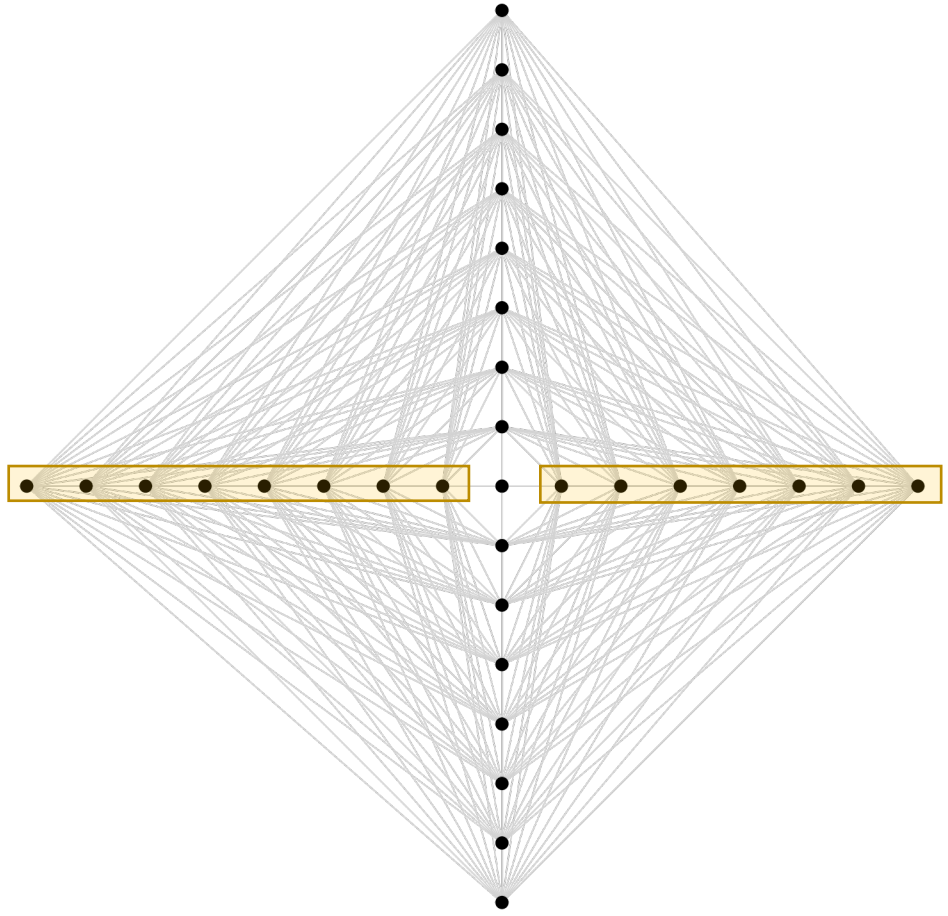


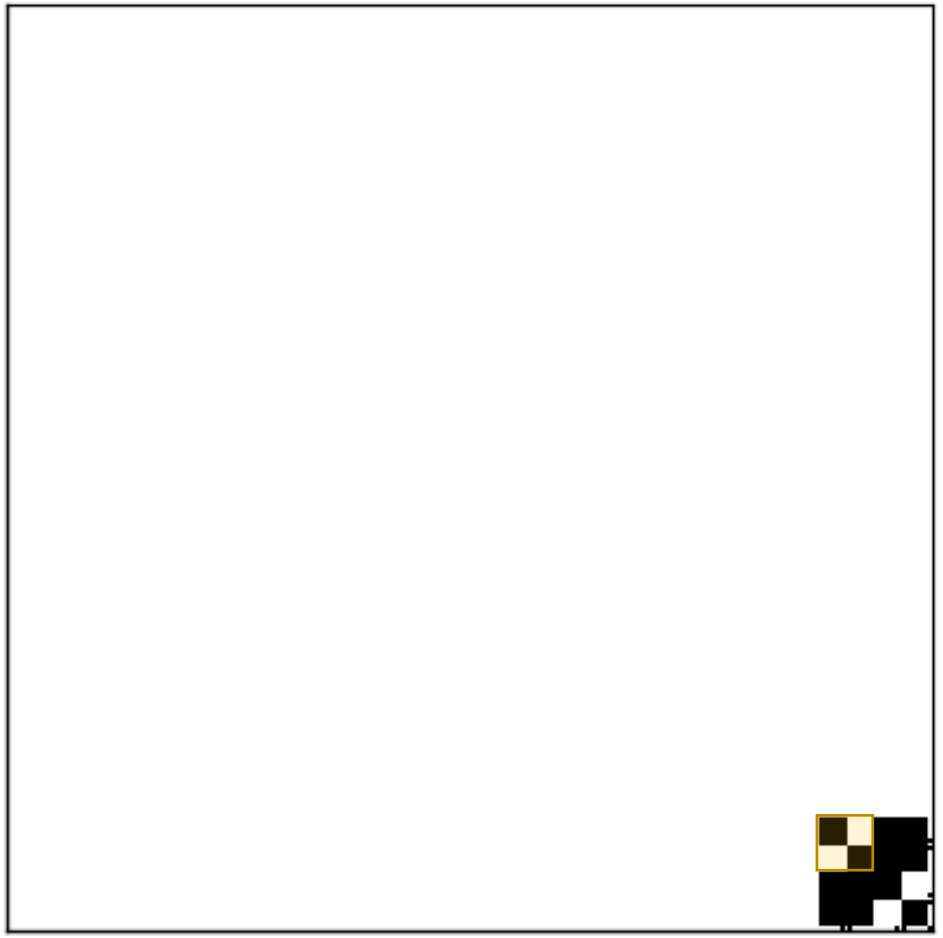
Sparse Linear Systems (2)



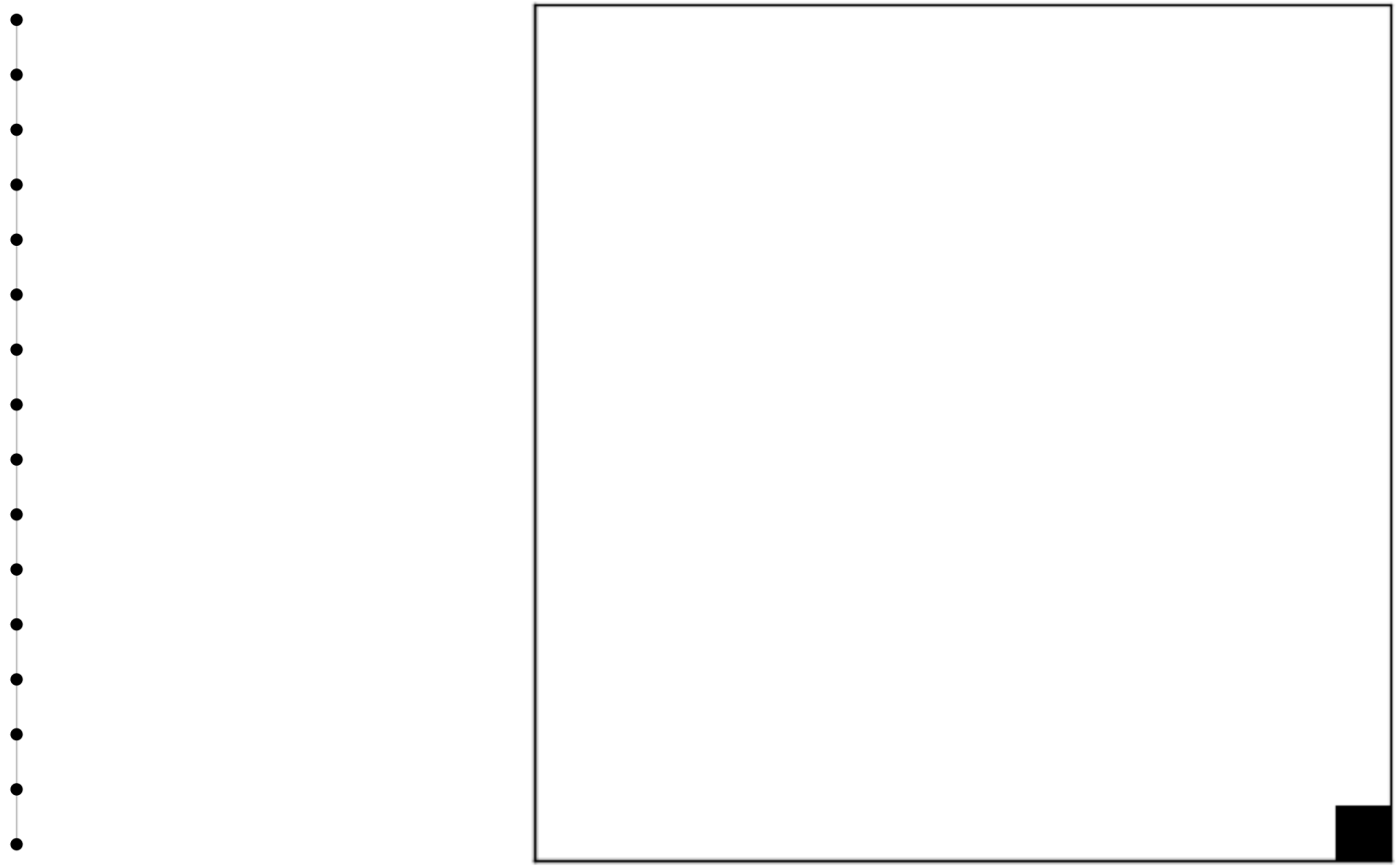


Sparse Linear Systems (3)

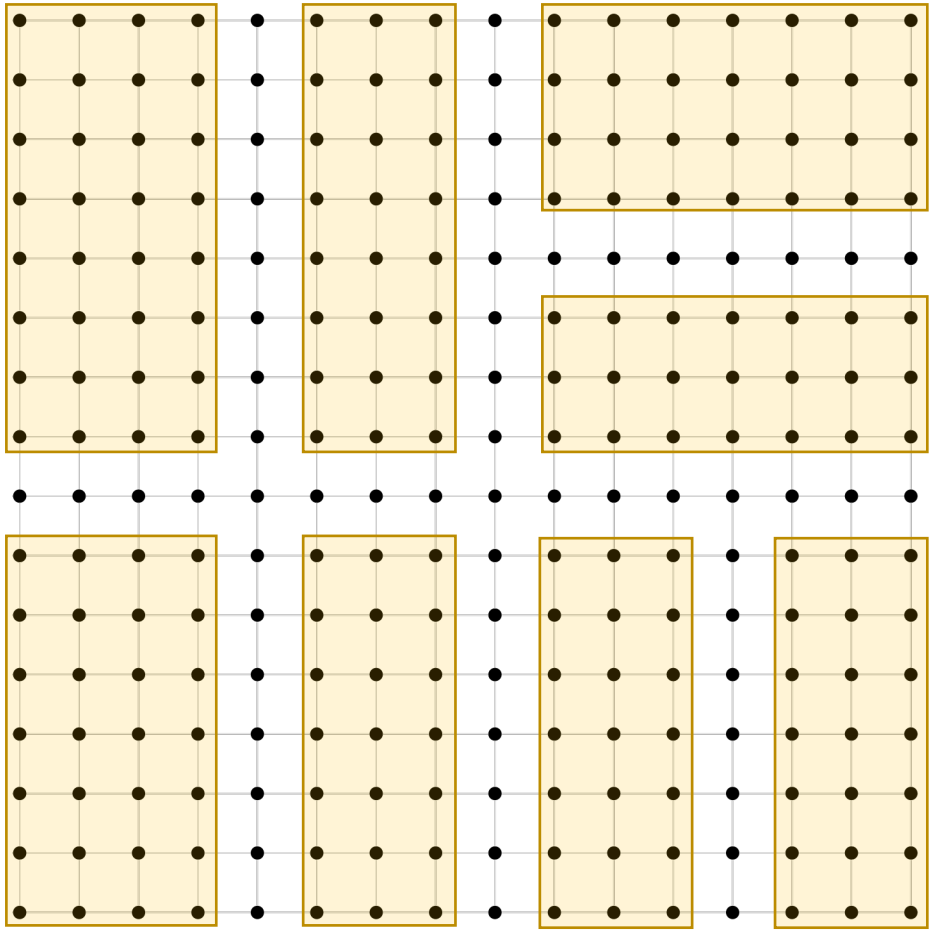


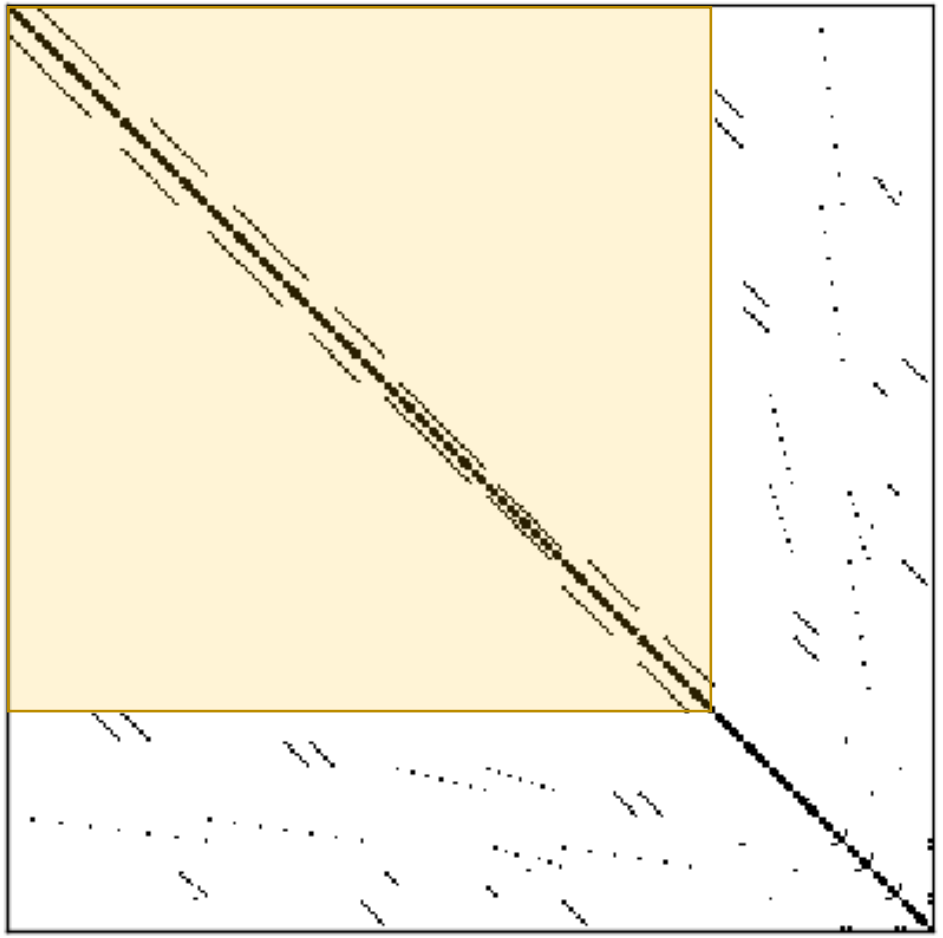


Sparse Linear Systems (4)

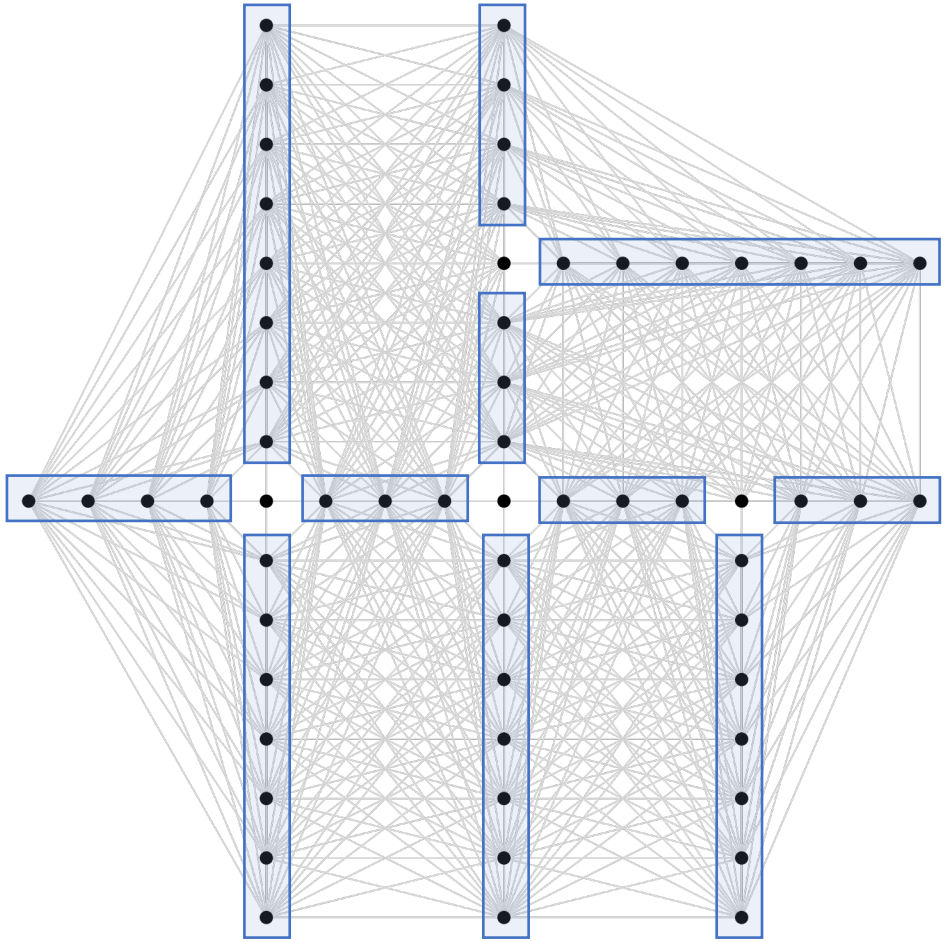


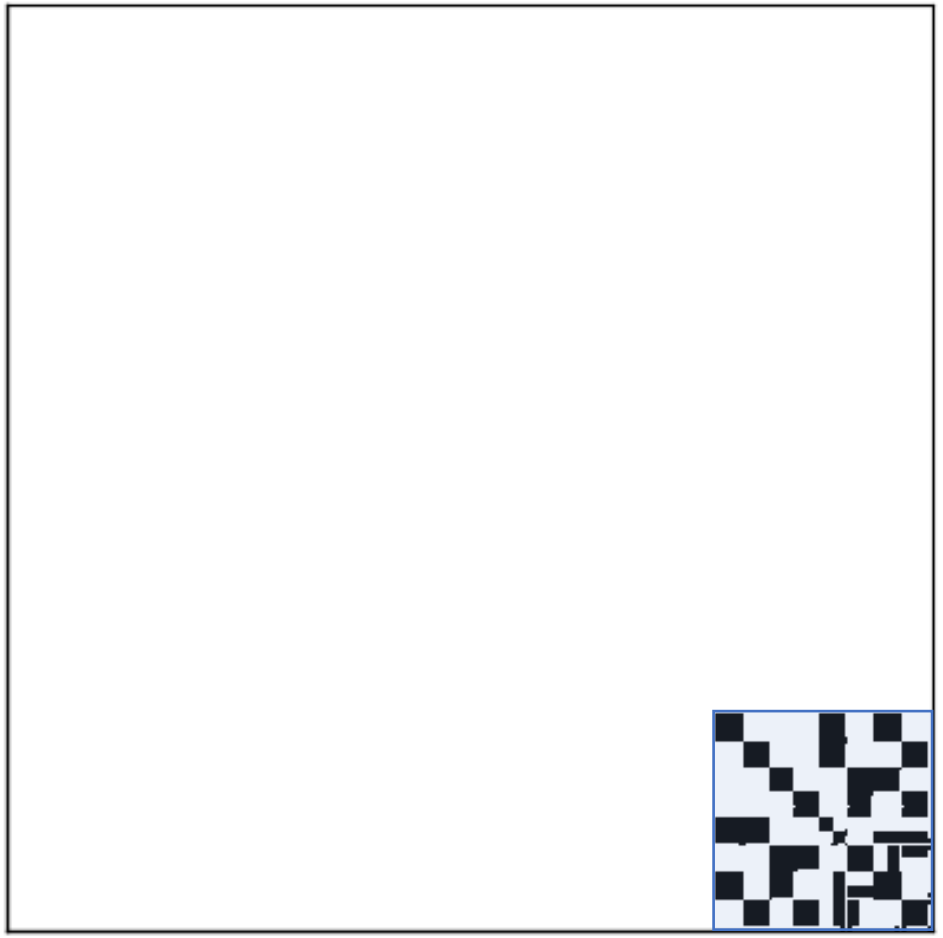
Sparsification (1)





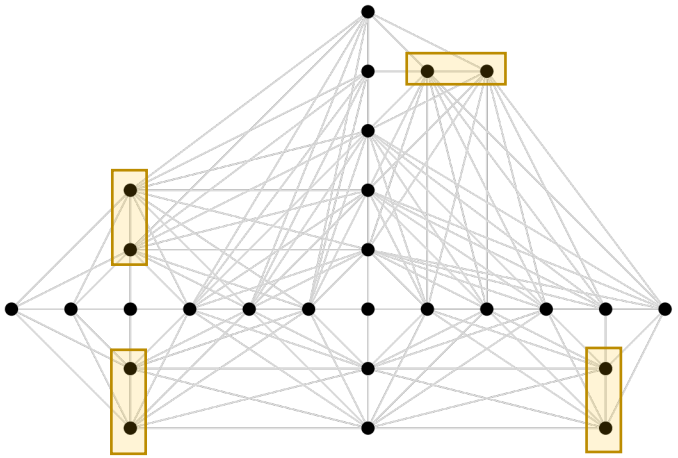
Sparsification (2)

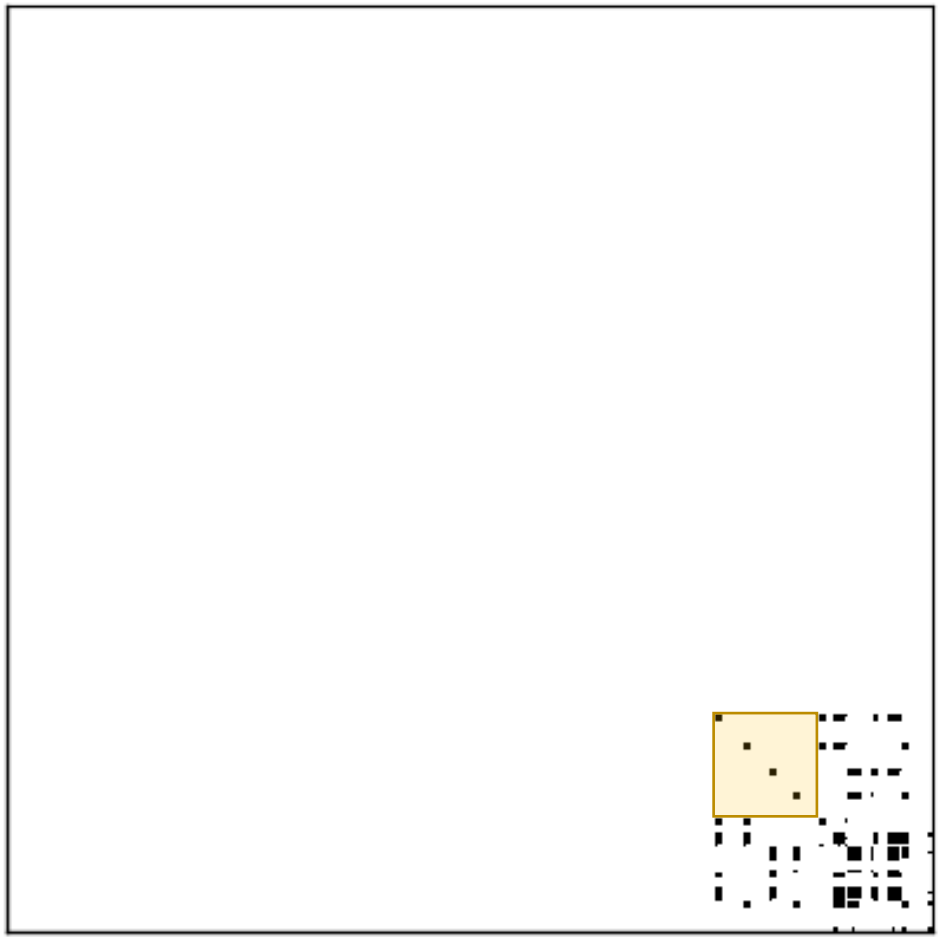




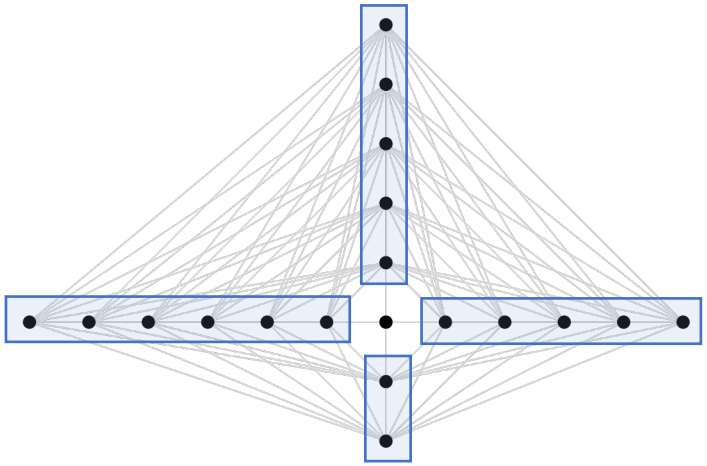
Sparsification (3)

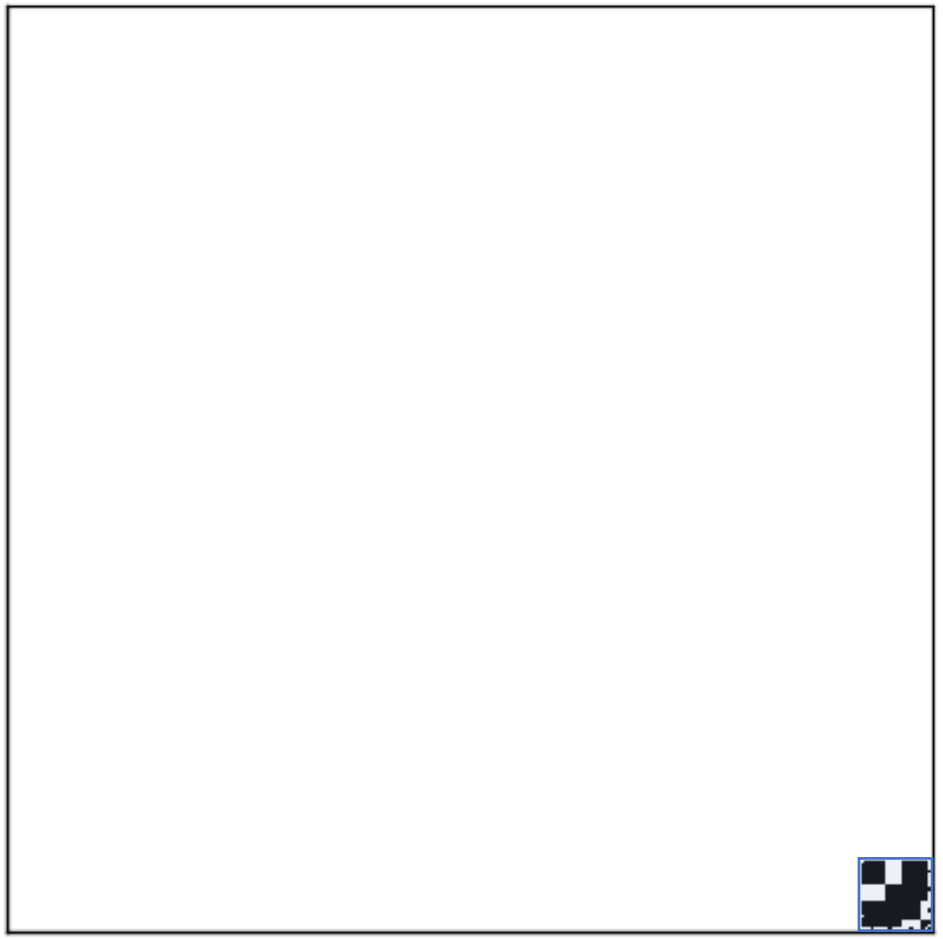
≈





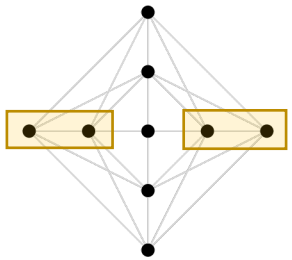
Sparsification (4)

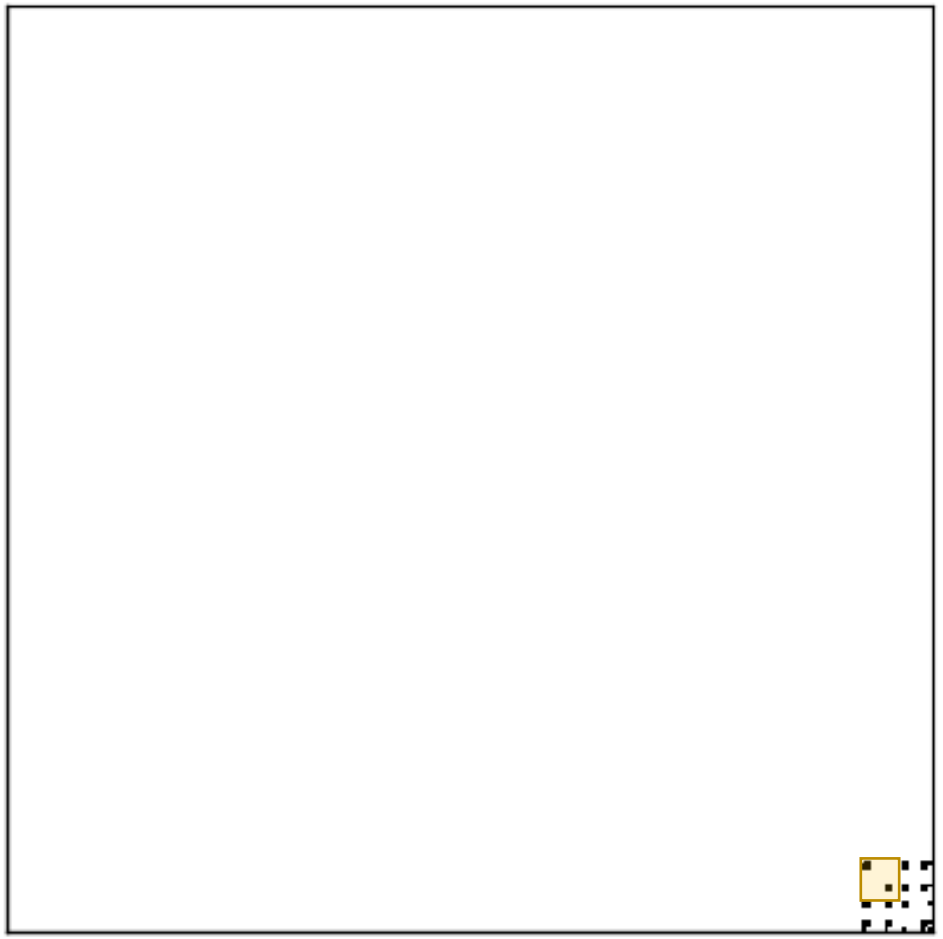




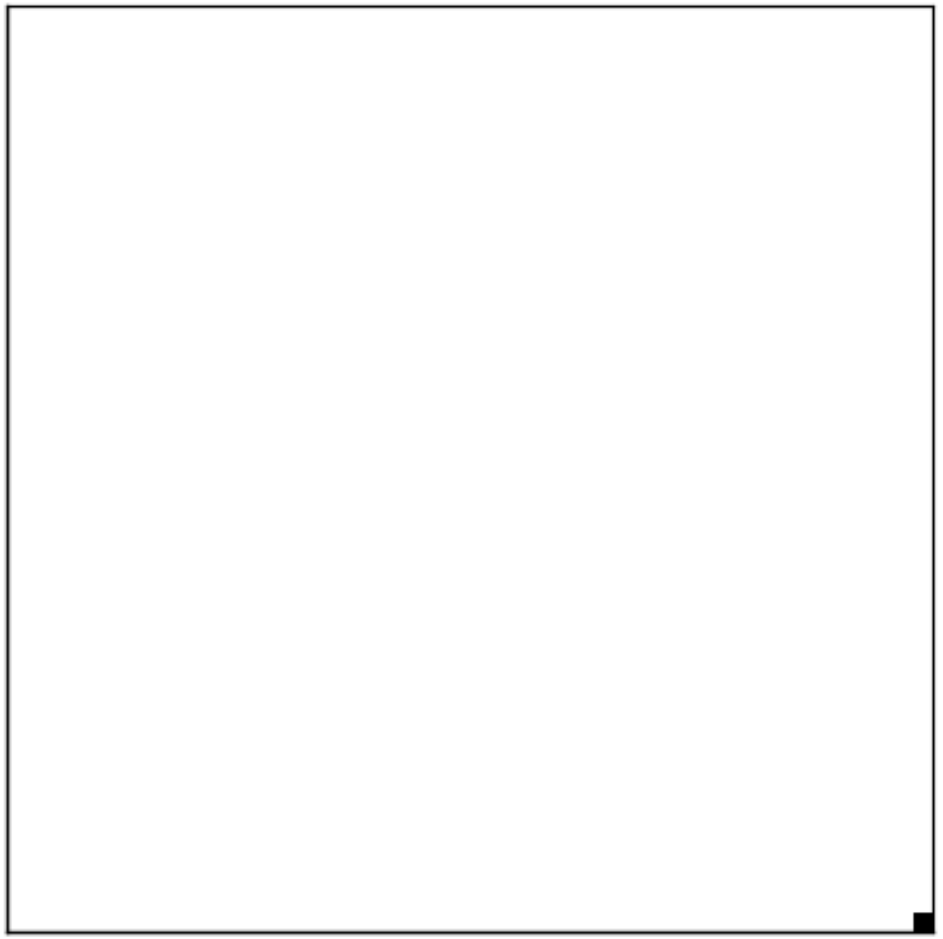
Sparsification (5)

≈



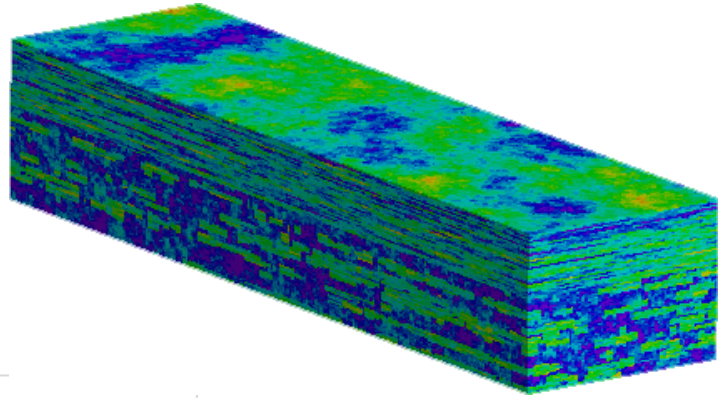


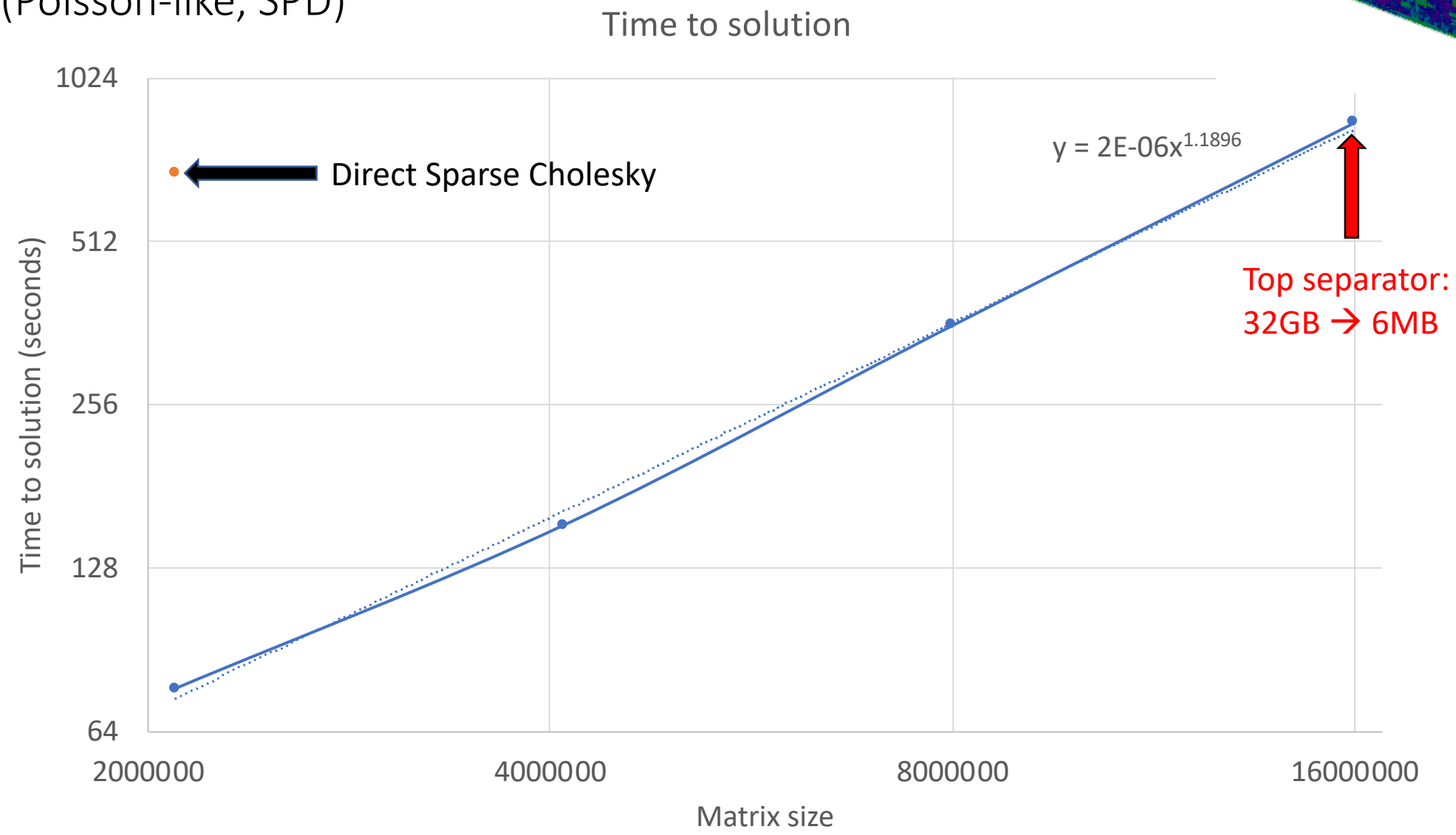
Sparsification (6)



SPE10

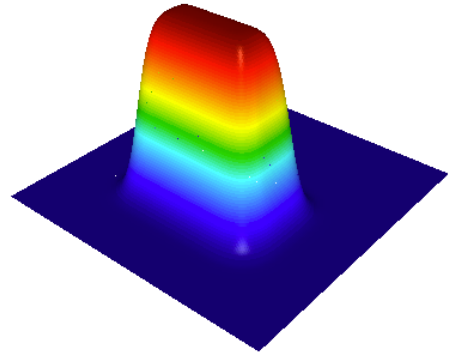
(Poisson-like, SPD)

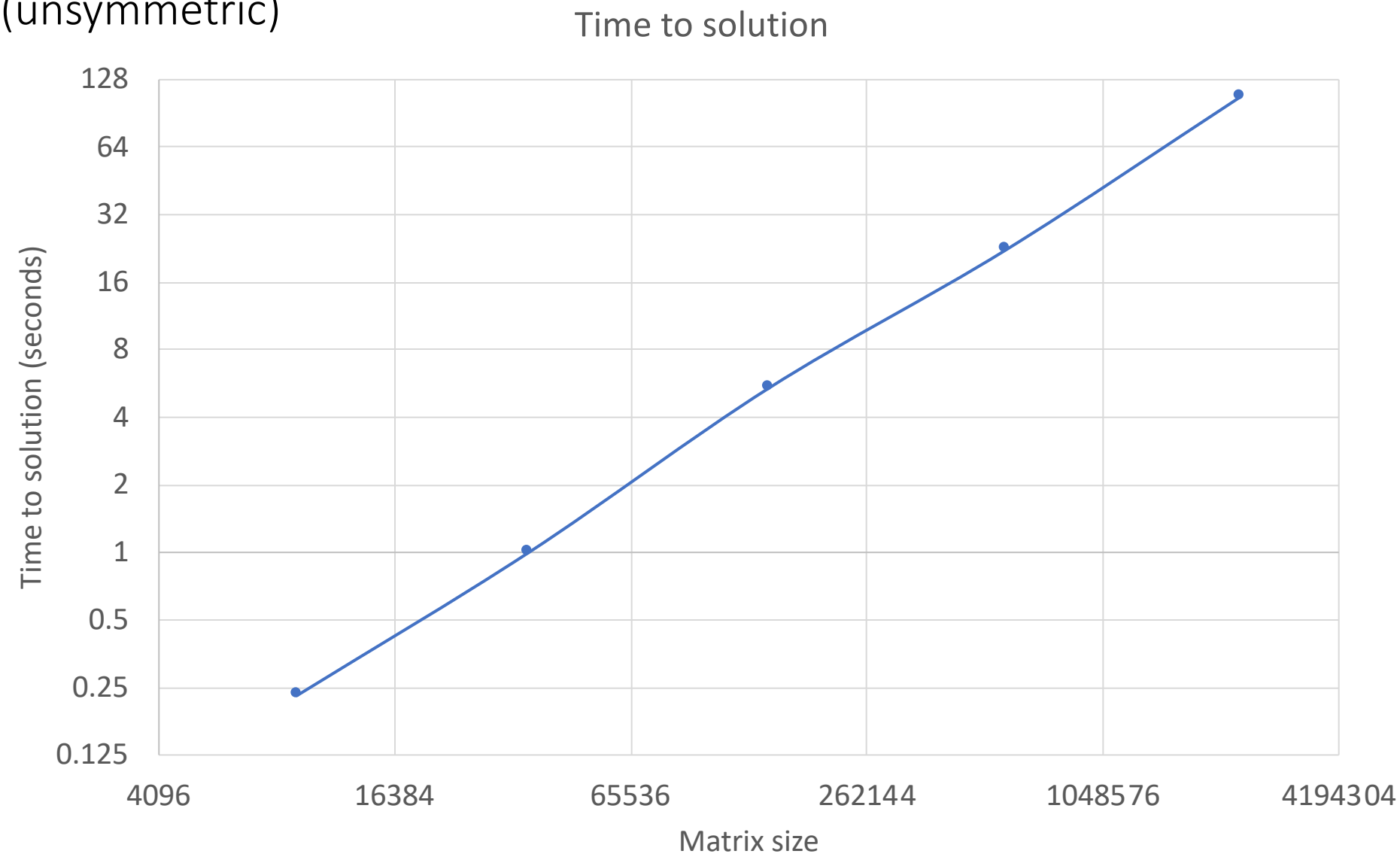




Advection

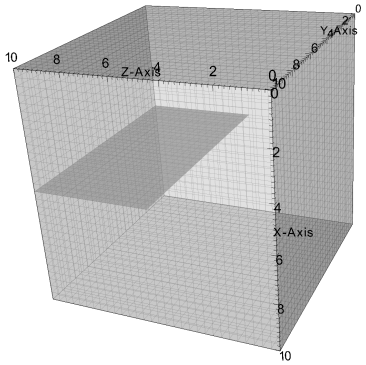
(unsymmetric)

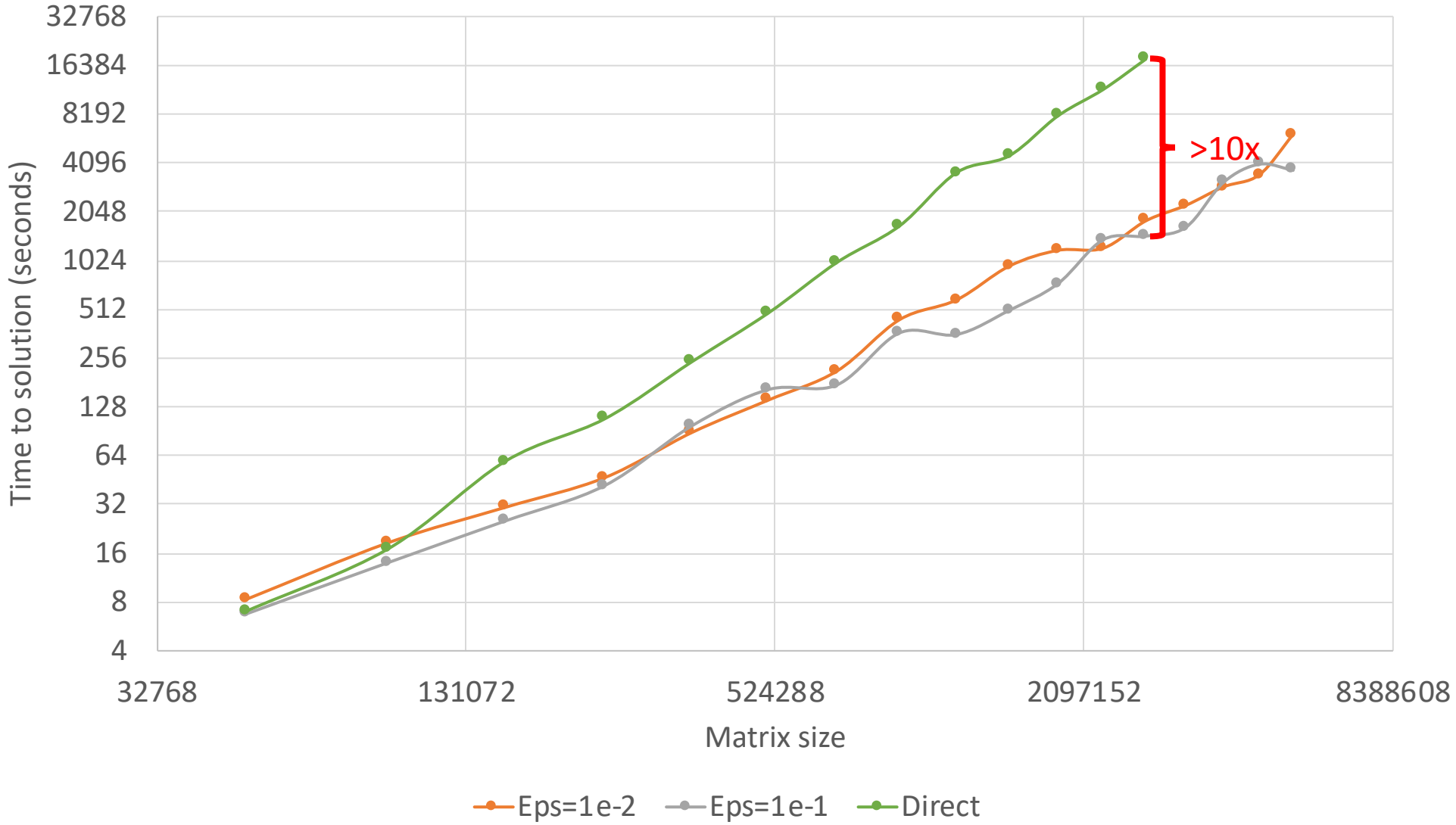




Fracture Problems

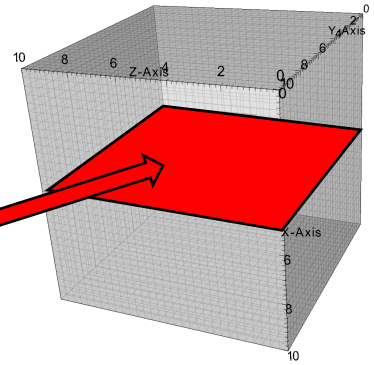
(symmetric, not SPD, many singular blocs)

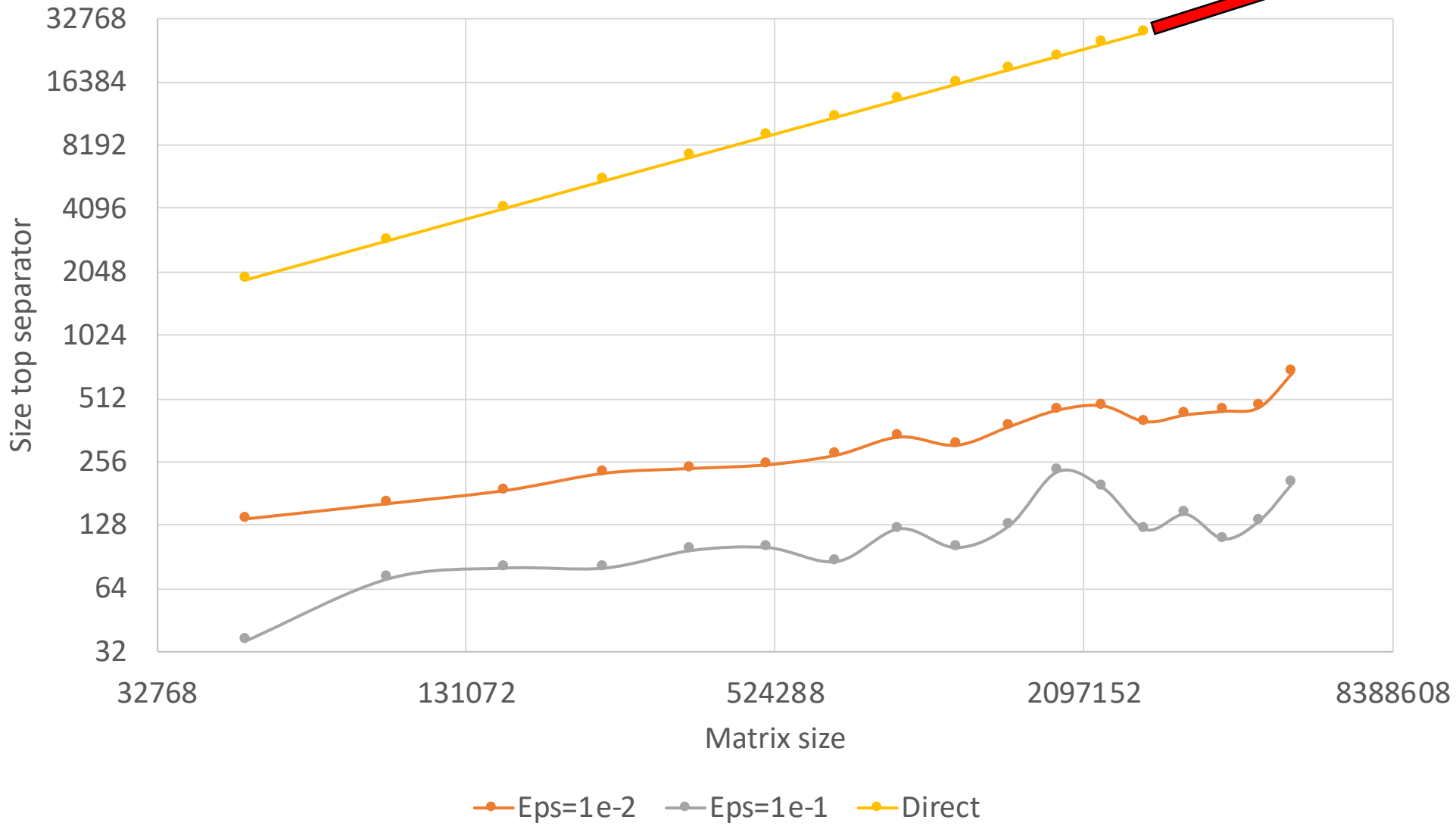




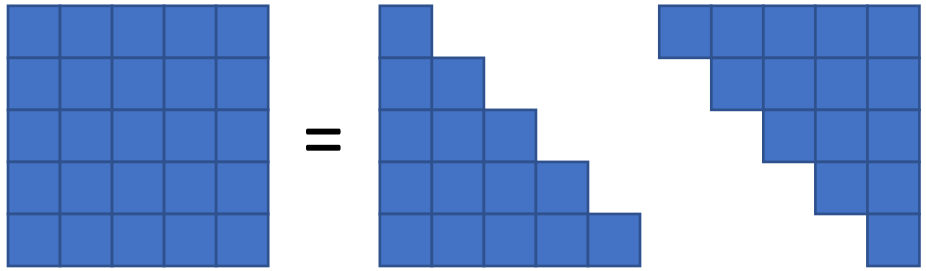
Fracture Problems

(symmetric, not SPD, many singular blocs)

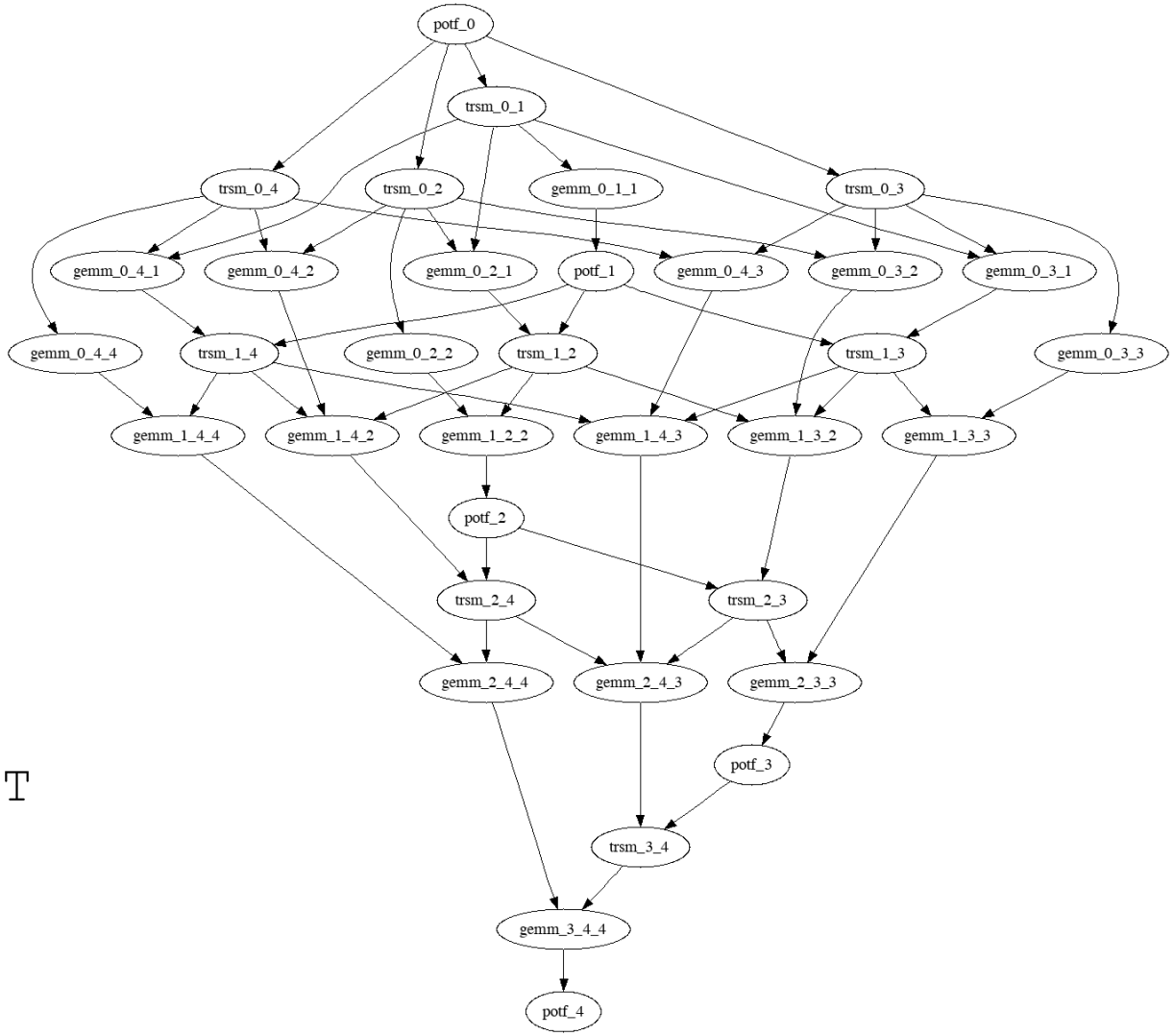




Runtimes for Scientific Computing

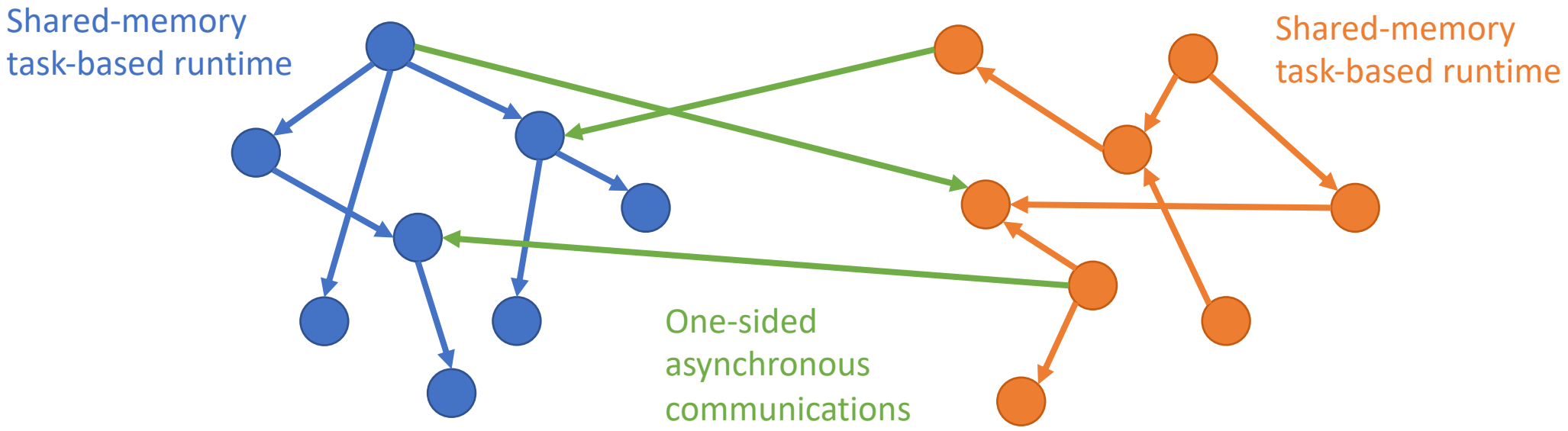


```
for k = 1…n
  A[k] = potf(A[k])
  for i = k…n
    A[i,k] = A[i,k] / A[k]^T
    for j = i…n
      A[i,j] += A[i,k] * A[j,k]^T
```



Runtimes for Scientific Computing

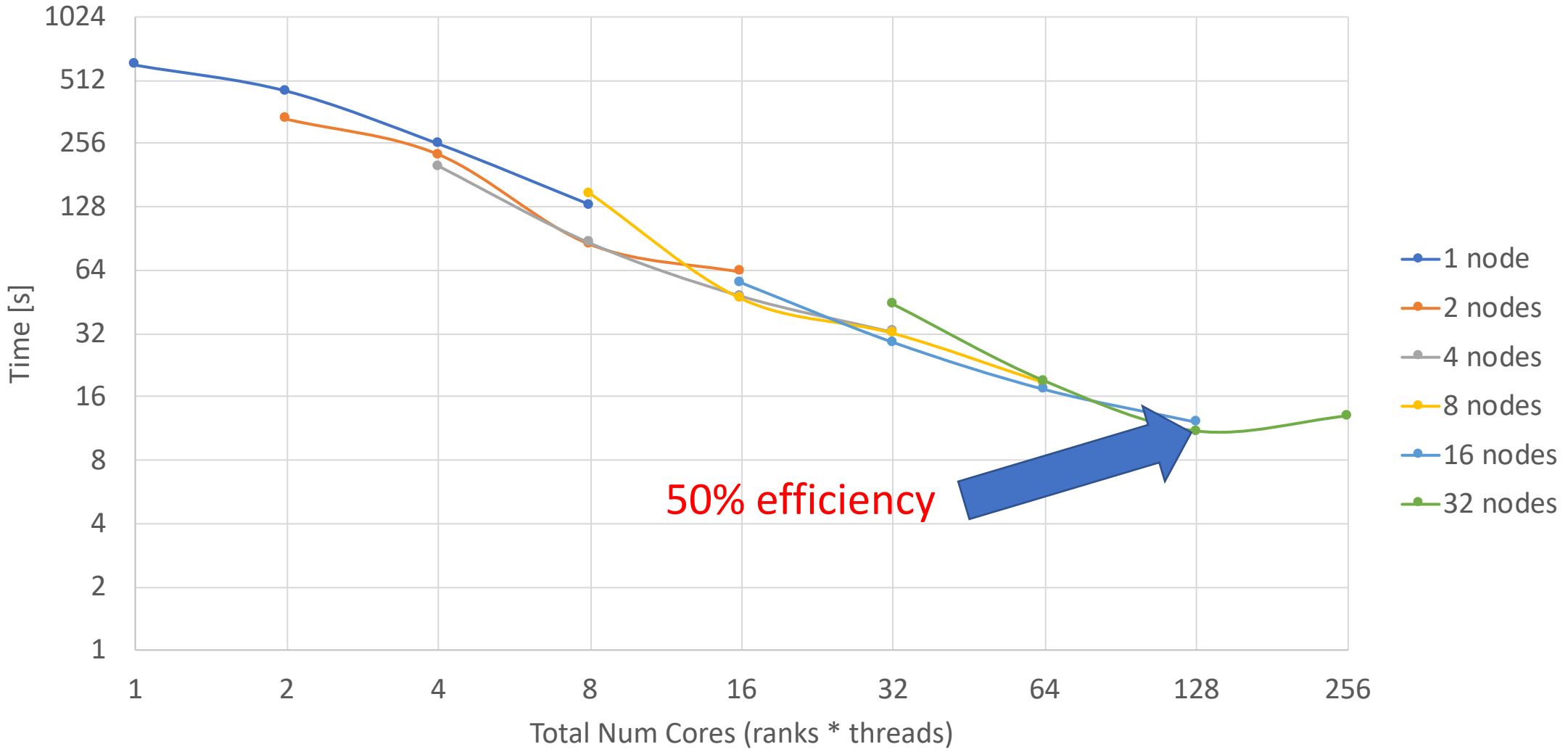
- Using standard technologies (MPI) & programming languages (C++ & threads)
- Minimal overhead, maximal asynchrony thanks to
  - Task-based parallelism
  - One-sided asynchronous & non-blocking communications



Sparse Direct Cholesky

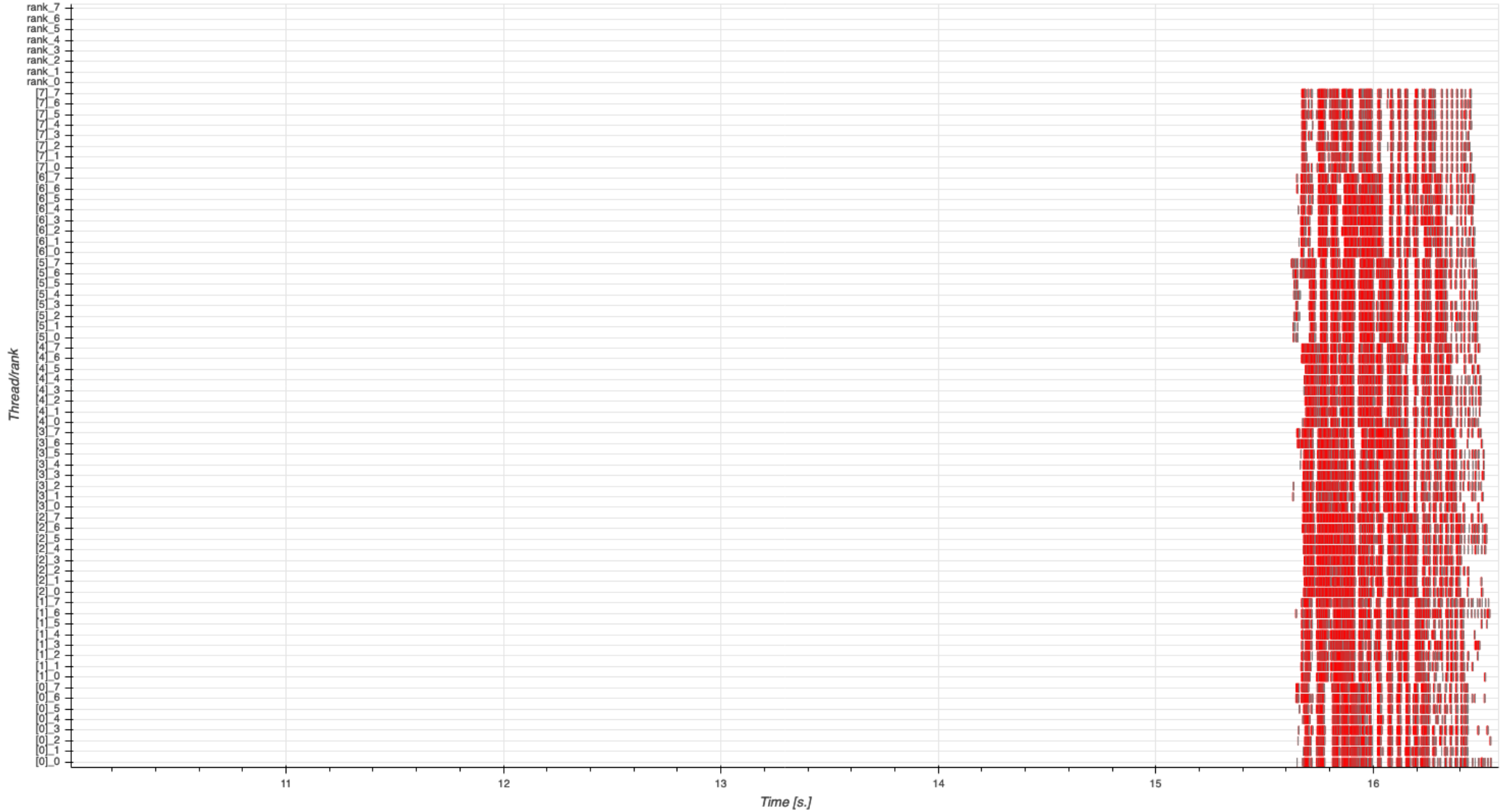
Strong scaling

128^3 7-points stencil
Factorization time
About 10^6 tasks total
On Stanford Sherlock



Sparse Direct Cholesky

Parallel Trace (file:///Users/lcambier/git/ptdag/tools/profile_8_8_2097152_10_end.html)



Future work

Fast Linear Solvers  Runtimes

Scalable Linear Solvers

Acknowledgements

- Initial spaND work with Chao Chen, Eric Darve, Erik Boman, Ray Tuminaro, Siva Rajamanickam: https://arxiv.org/abs/1901.02971
- Fault+spaND work with Bazyli Klockiewicz
- Support from Sandia National Lab & Total