High Performance Linear System Solvers with Focus on Graph Laplacians

Richard Peng
Georgia Tech

Based on work joint with: Serban Stan, Shen Chen Xu, Saurabh Sawlani, John Gilbert, Kevin Deweese, Gary Miller, Hui Han Chin

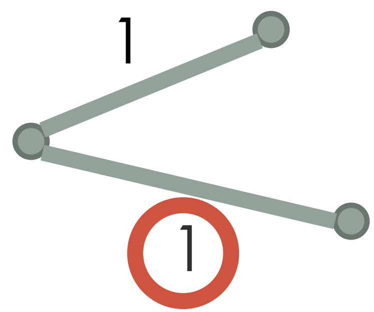
OUTLINE

- **Laplacian solvers and applications**
- Combinatorial preconditioning
- Numerics of tree preconditioners

GRAPH LAPLACIAN MATRIX

- Diagonal: weighted degrees
- Off-diagonal: -edge weights

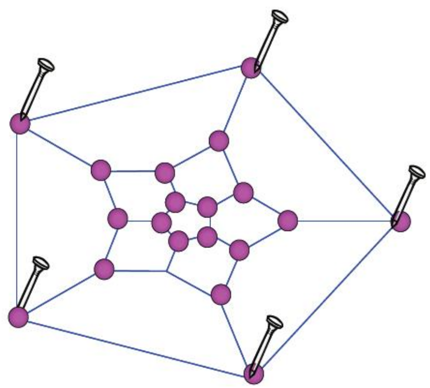
$$\begin{bmatrix} 2 & -1 & -1 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}$$

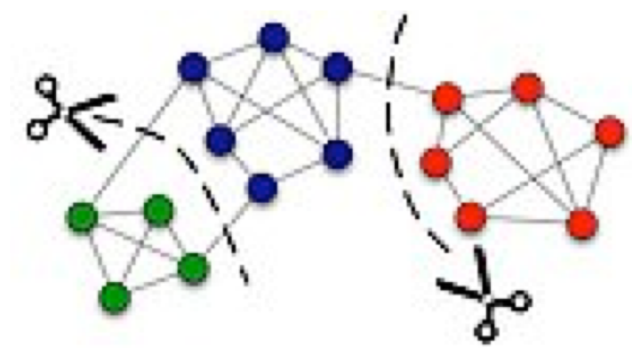


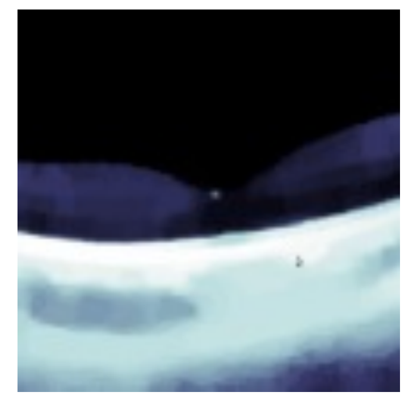
FEW ITERATIONS OF **Lx = b**

Spring-mass system / elliptic problems / M matrices

- [Tutte `61]: planar embedding / graph drawing,
- [ZGL `03], [ZHS `05] [KRS `15]: learning/inference





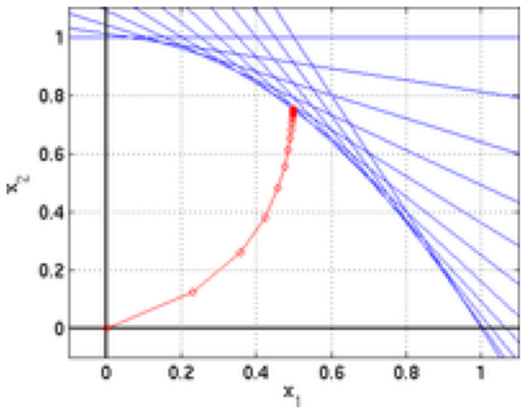


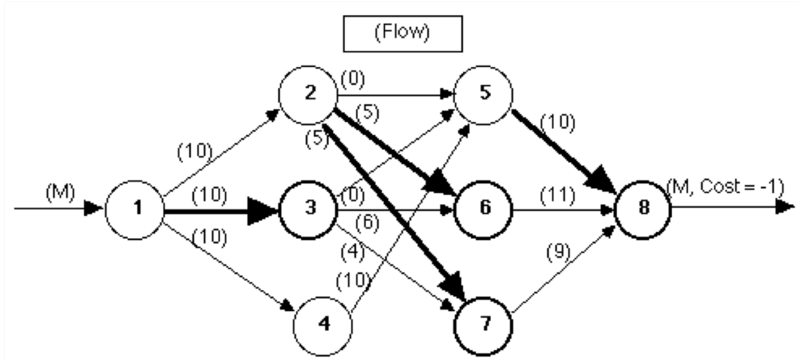
Inverse powering: eigenvectors / heat kernel:

- [AM `85] [OSV `12]: clustering
- [SM `01][KMST `09]: image segmentation

MANY ITERATIONS OF **Lx = b**

[Karmarkar, Ye, Renegar, Nesterov, Nemirovski …]: convex optimization via. solving O($m^{1/2}$) linear systems





[DS `08][CKMST `11][LS `14][AKPS `19][APS`19][AS `20]: graph problems → Laplacian linear systems

LINEAR SYSTEMS SOLVERS

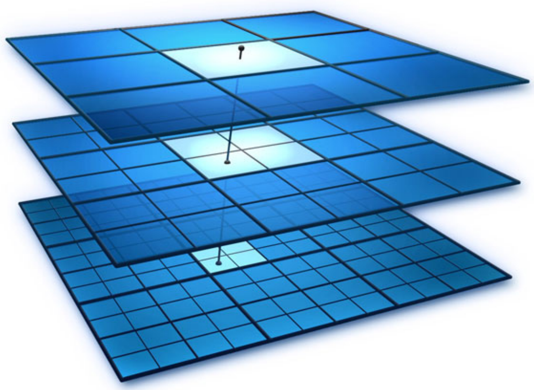
- Matrix multiplication: $O(n^{2.372864\ldots})$
- Conjugate gradient: $O(nnz\ k^{1/2})$ where k is condition number

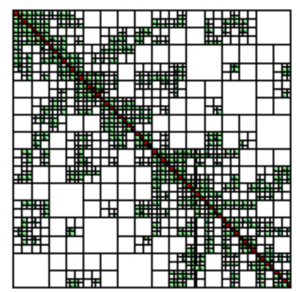
Open: provably faster solver for poly(n) conditioned sparse (O(n) nonzero) systems.

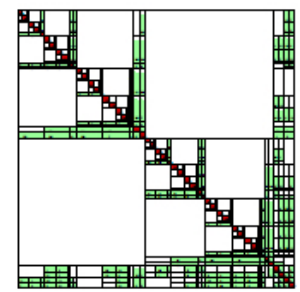
[Zhang `18]: structure often don't help

$\mathbf{Lx} = \mathbf{b}$ in Practice

- Scientific computing: multigrid 'works' for 10^9 nonzeros
- MATLAB: pcg($\mathbf{L}$, ichol($\mathbf{L}$), $\mathbf{b}$, ε) 'works' for 10^6 nonzeros

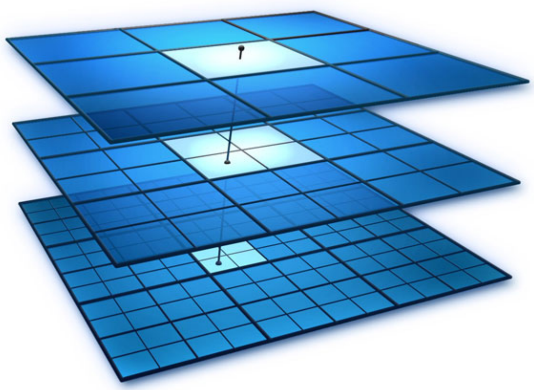


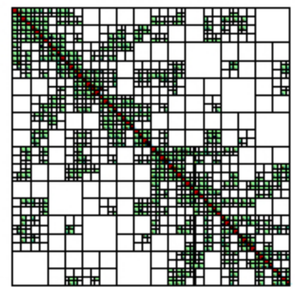


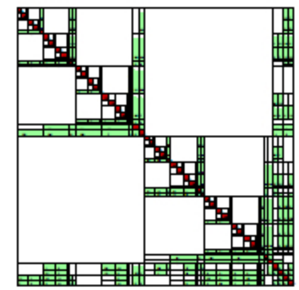


$\mathbf{Lx} = \mathbf{b}$ in Practice

- Scientific computing: multigrid 'works' for 10^9 nonzeros
- MATLAB: pcg(**L**, ichol(**L**), **b**, ε) 'works' for 10^6 nonzeros







'works': gradual numerical convergence

NUMERICAL METHODS

Gradual convergence to solutions

Simplest: $\mathbf{x} \leftarrow \mathbf{x} + \theta(\mathbf{A}\mathbf{x} - \mathbf{b})$

Fixed point: $\mathbf{A}\mathbf{x} = \mathbf{b}$

Better schemes: conjugate gradient, accelerated methods

NUMERICAL METHODS

Gradual convergence to solutions

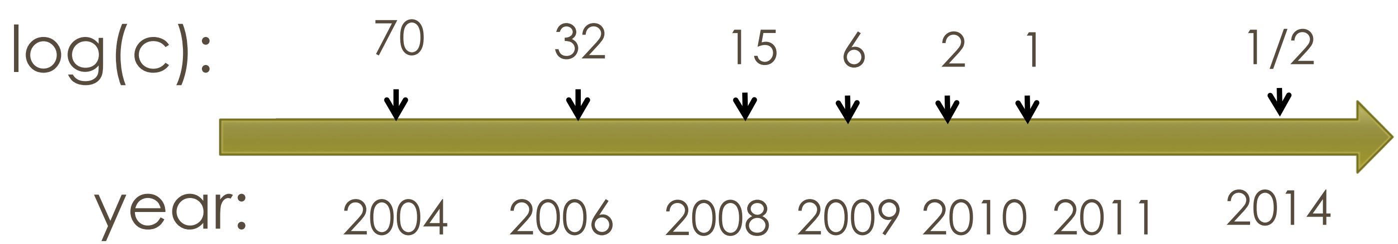
Preconditioned: $\mathbf{x} \leftarrow \mathbf{x} + \theta \mathbf{B}^{-1}(\mathbf{Ax} - \mathbf{b})$

Want: **B** that's close to **A**, but computationally less expensive:

- Jacobi: **B** = Diag(**A**)
- Gauss Siedel: B = TriU(**A**)
- Incomplete Cholesky: drop small entries during sparse elimination

COMBINATORIAL PRECONDITOINING

- [Vaidya `89]: use graph theory to build preconditioners for **L**
- [ST`04]: $O(m\log^c n \log(1/\varepsilon))$ time
- 2004 – 2014: c halved every 2 years

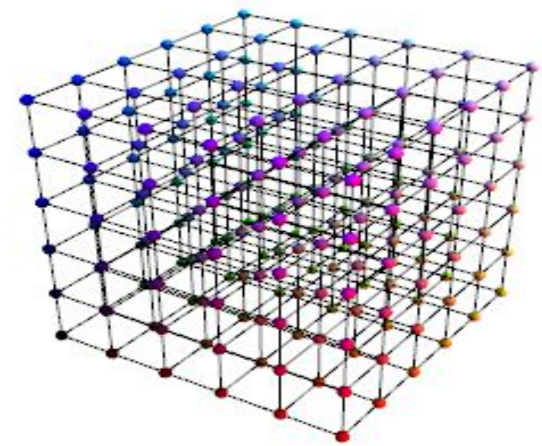


Most recent: [KS `16] showed randomized incomplete Cholesky provably works for **L**

COMPARE? NEW BENCHMARKS:

Structured graphs

- Grids / cubes
- Cayley graphs
- Graph products

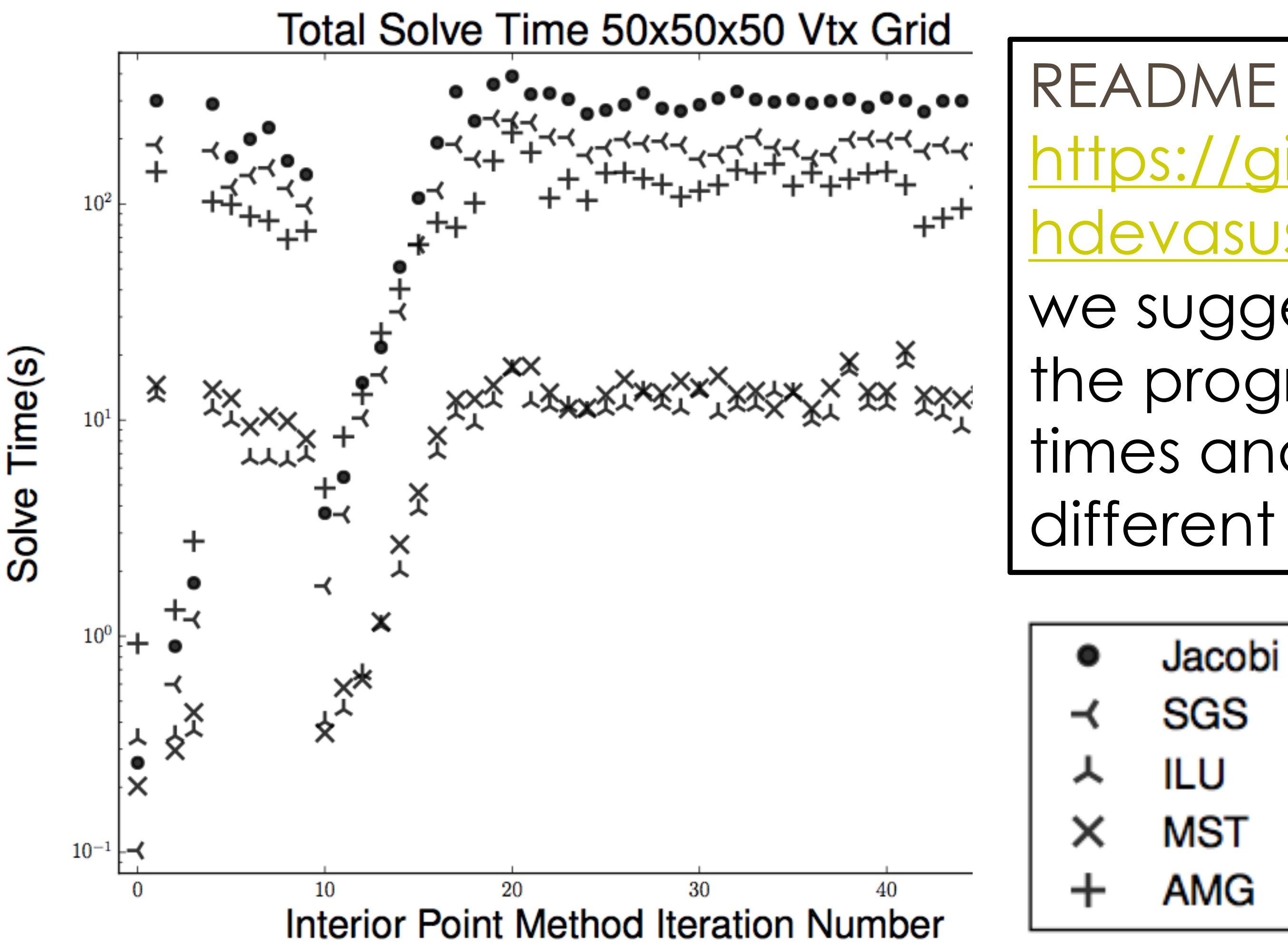


Hard graph problems

- Maxflow problems from DIMACS implementation challenges
- Linear systems arising from second-order optimization (IPM)

[KRS`15] + DIFFERENT SOLVERS

Disclaimer: this behavior depend heavily on IPM implementation / numerics / termination conditions



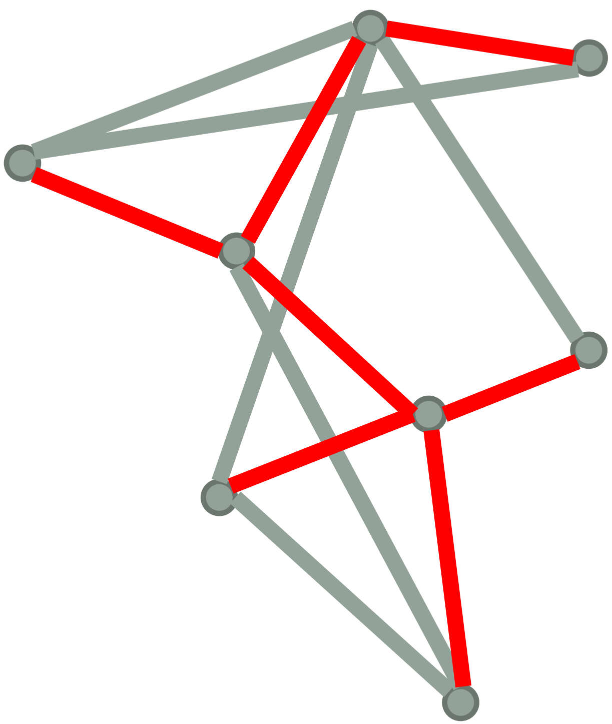
README file at https://github.com/sachdevasushant/Isotonic we suggest rerunning the program a few times and / or using a different solver.

OUTLINE

- Laplacian solvers and applications
- **Combinatorial preconditioning**
- Numerics of tree preconditioners

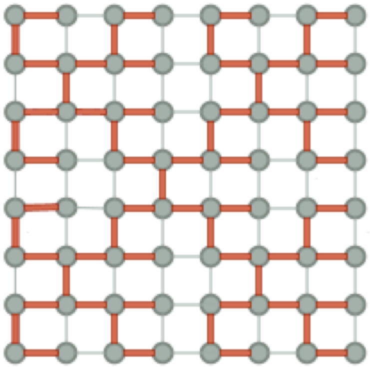
TREE BASED PRECONDITIONERS

Gradually transform a tree-based solution to a solution on the entire graph

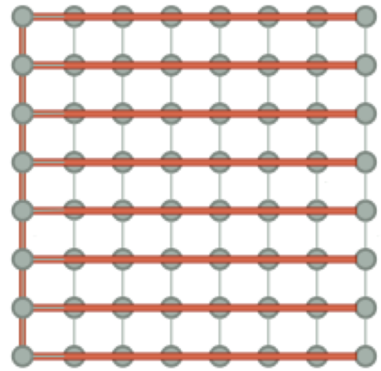


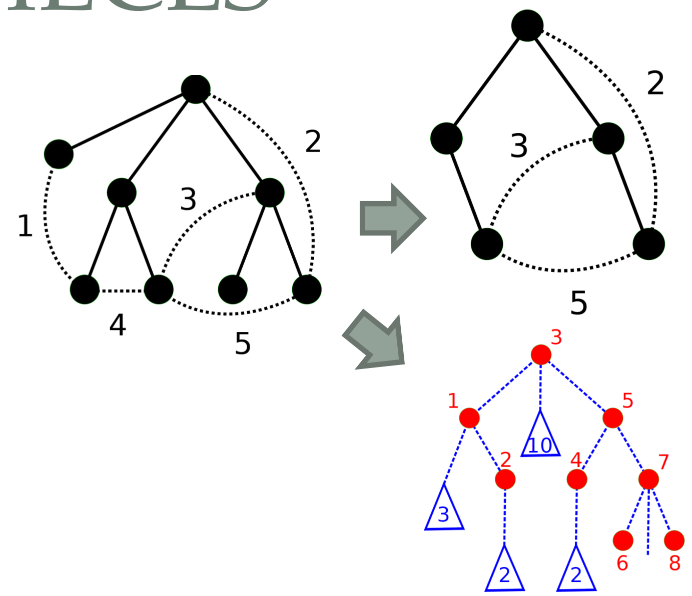
Method	Cycle Toggle	Precondition
Cost / Iter	logn	$m + (m/k)^2$
# Iters	$m\log^{1/2}n\log(1/\varepsilon)$	$k^{1/2}\log(1/\varepsilon)$
Related to	SGD	Grad. descent
Primitives	Data structures	Mat-Vec

MOVING PIECES



vs

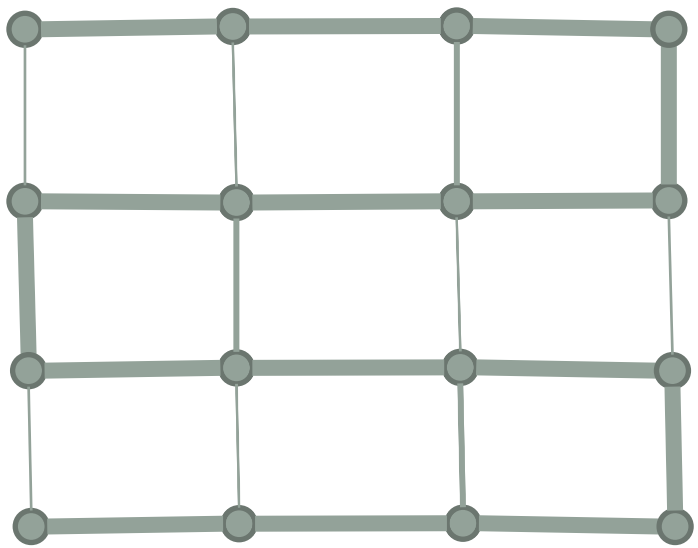


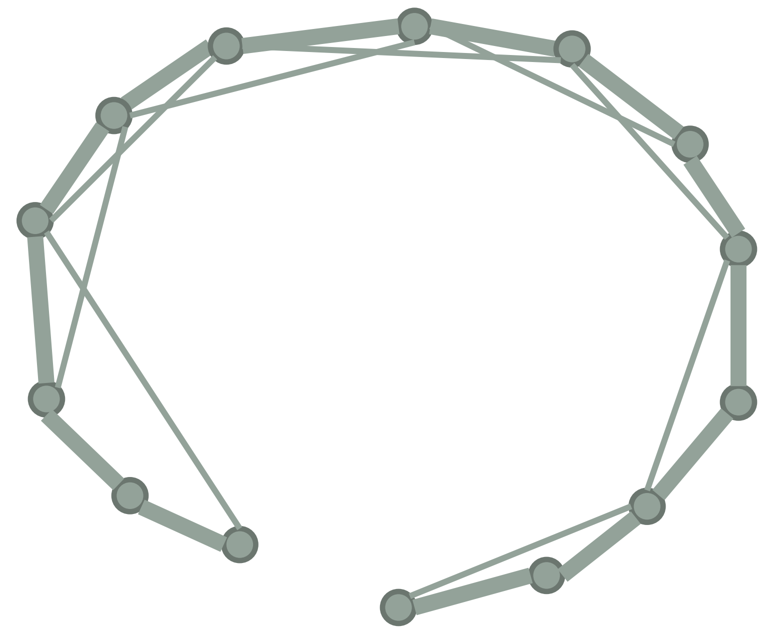


- Trees: MST / bottom-up / top-down / adaptive
- Numerics: batched / local, accelerated / CG
- Memory layout, recursive vs. iterative

[DGMPXX CSC`16]: HEAVY PATHS

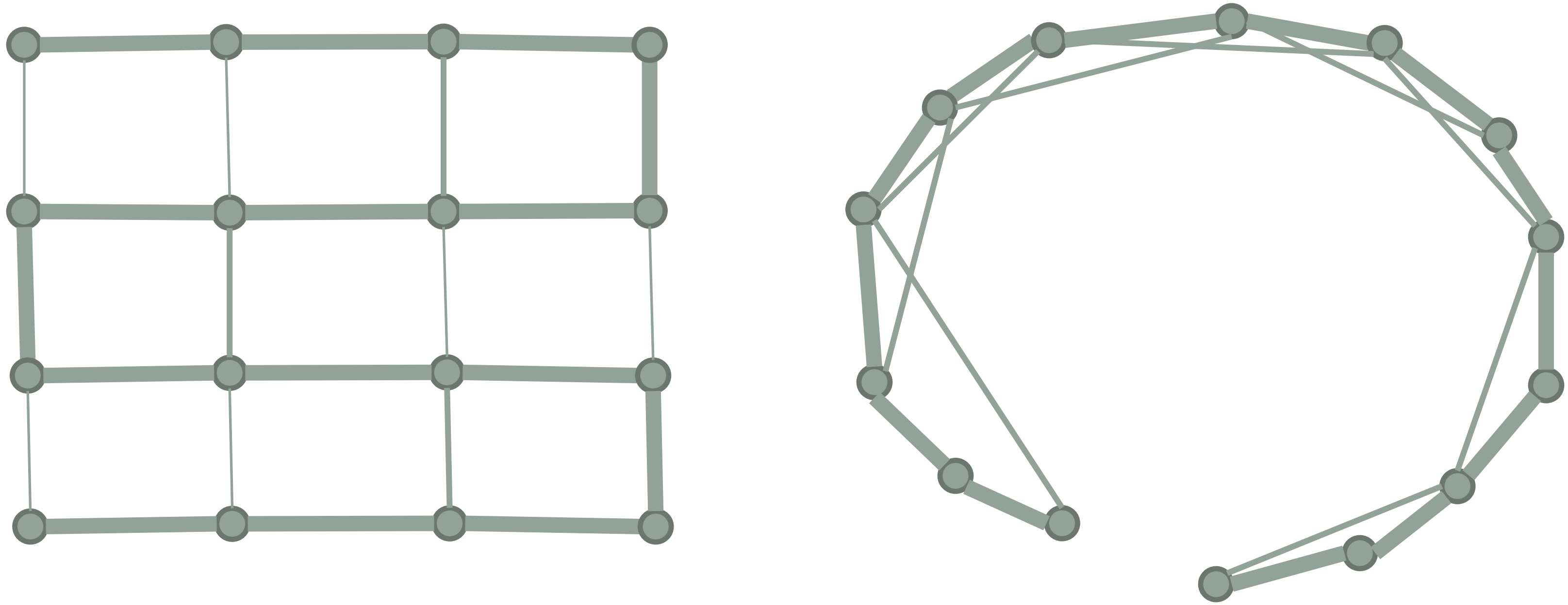
Pick a Hamiltonian path, weight all other edges so each has stretch 1





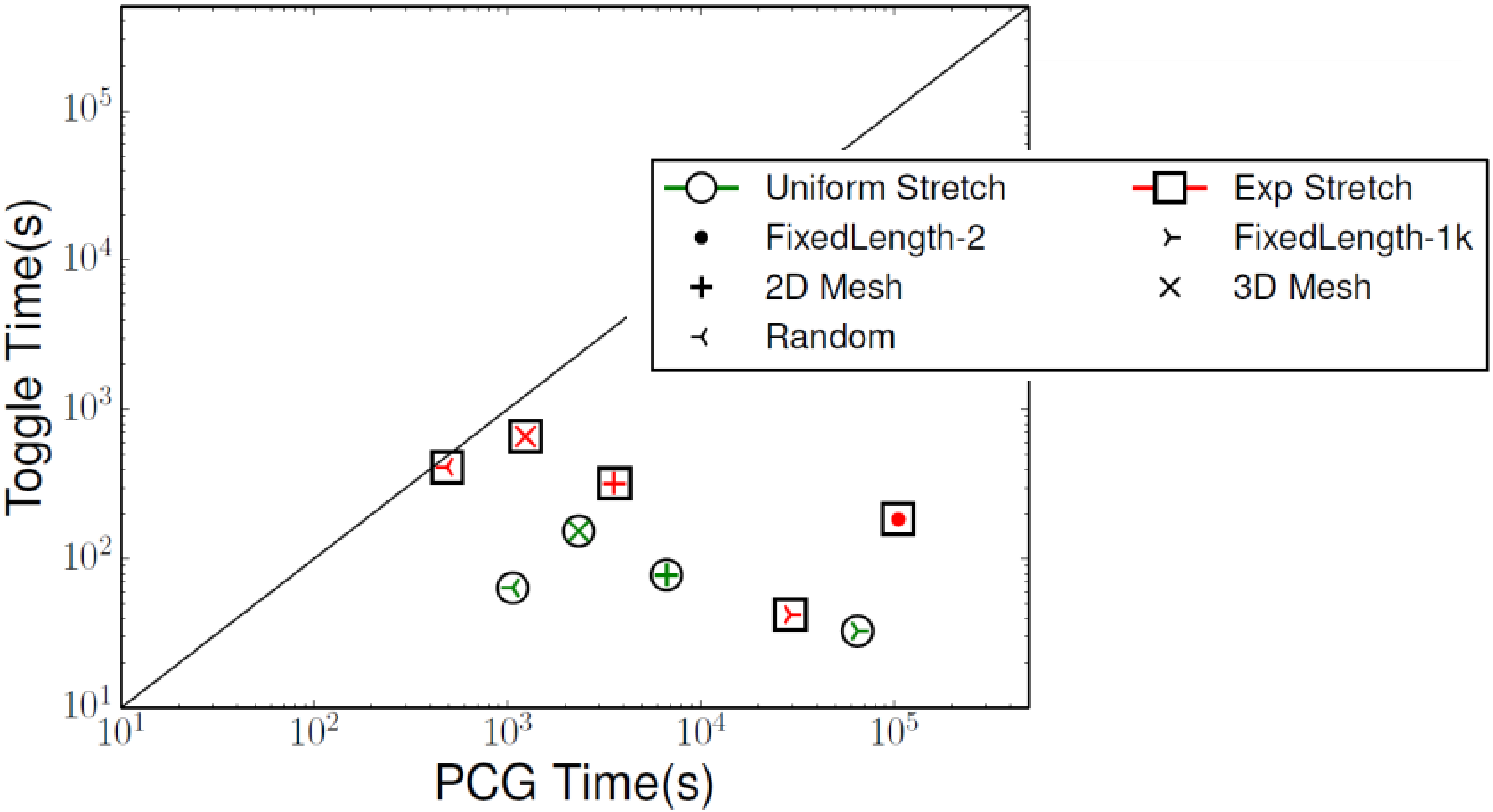
[DGMPXX CSC`16]: HEAVY PATHS

Pick a Hamiltonian path, weight all other edges so each has stretch 1



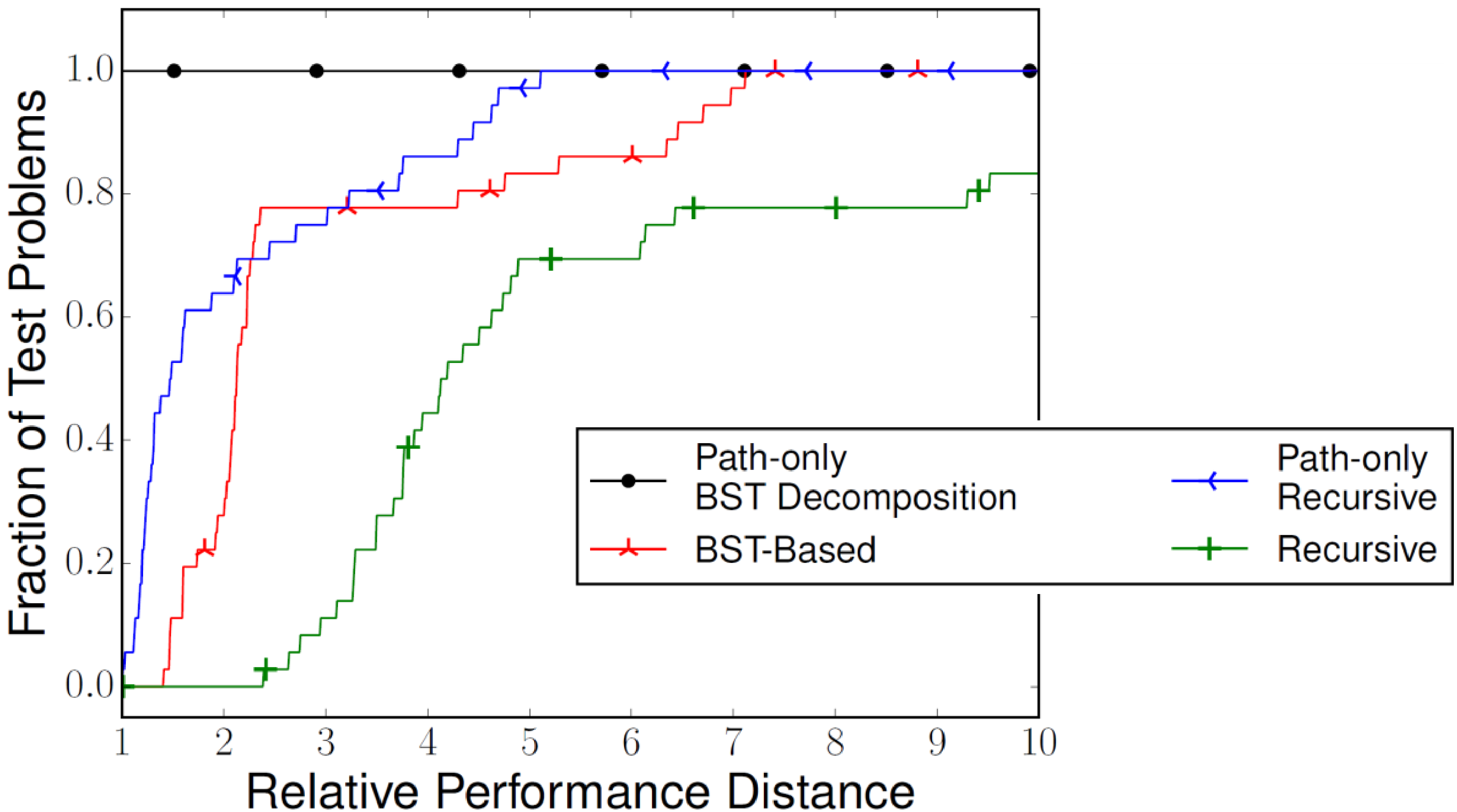
- Bad case for PCG,
- `easy' for tree data structures

DOING BETTER THAN CG



https://arxiv.org/abs/1609.02957
https://github.com/sxu/cycleToggling

COMPARISONS OF GRAPHS → TREES

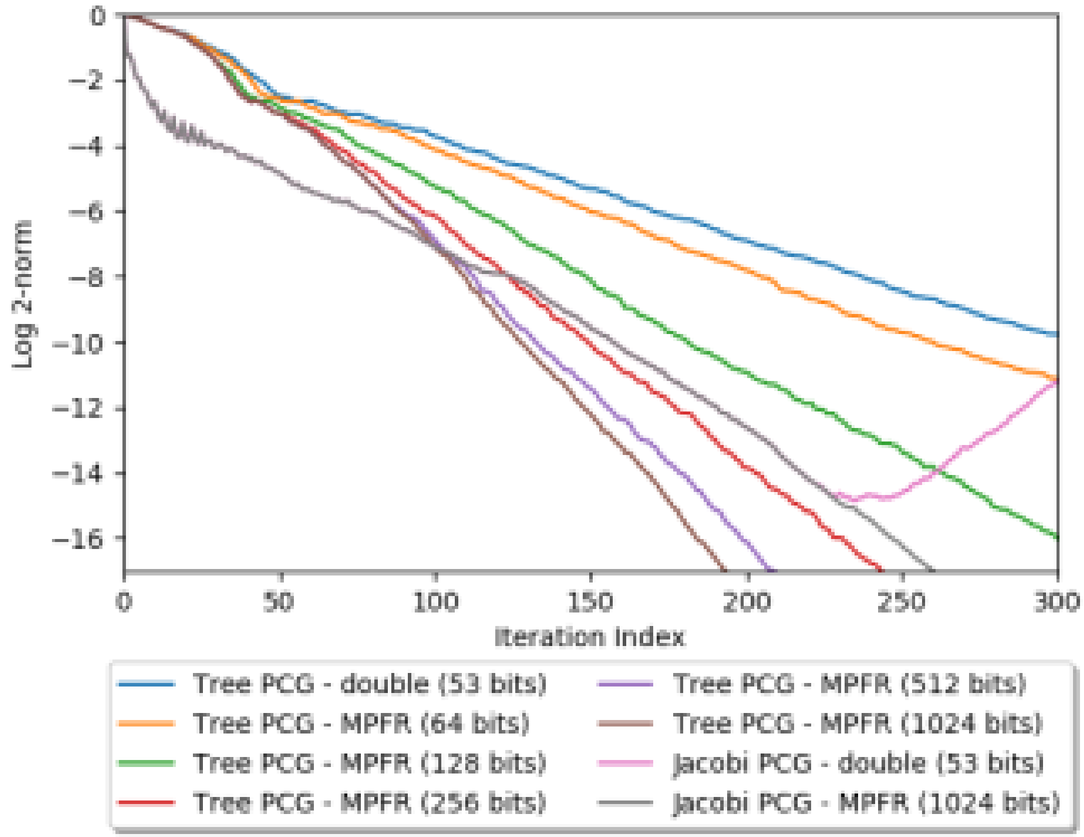


https://arxiv.org/abs/1609.02957
https://github.com/sxu/cycleToggling

OUTLINE

- Laplacian solvers and applications
- Combinatorial preconditioning
- **Numerics of tree preconditioners**

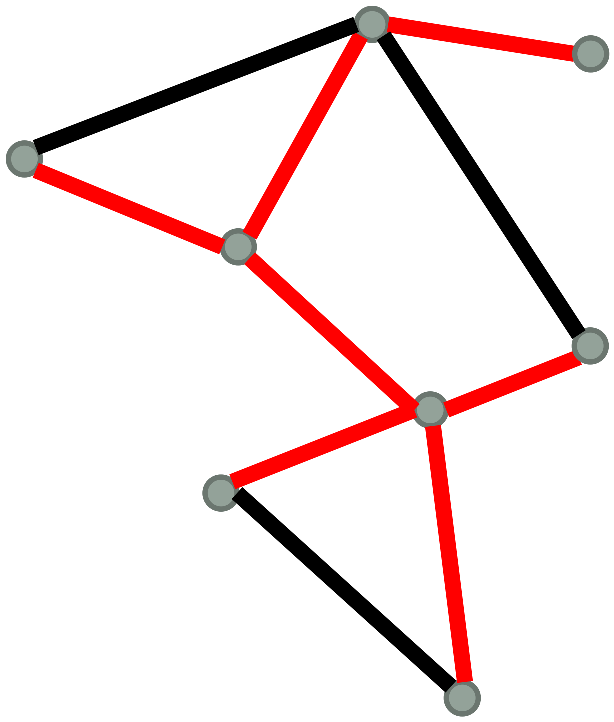
[DPSX CSC'20] NUMERICAL ISSUES



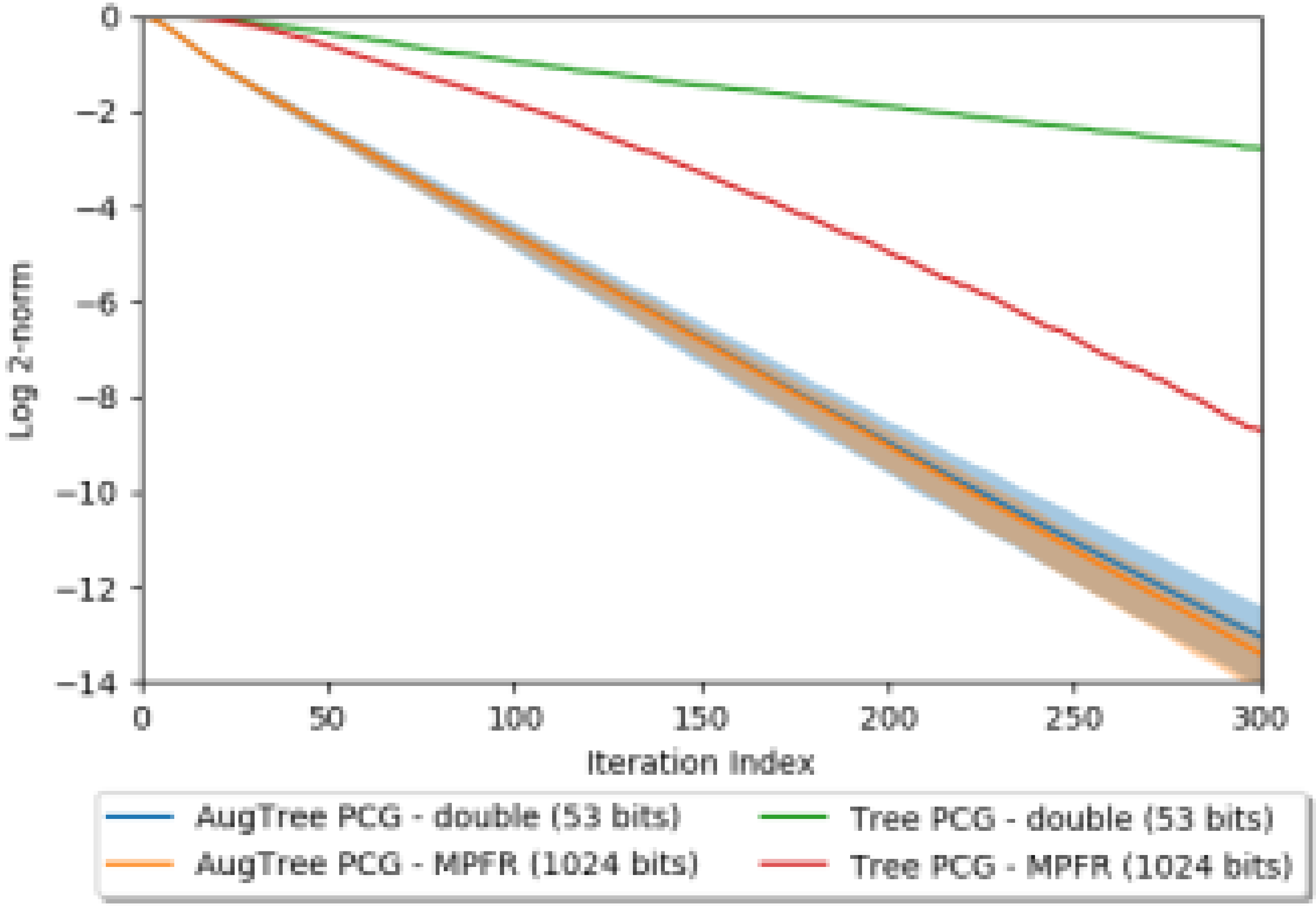
CG with a low-stretch tree as preconditioner

ONE FIX: BATCHED PROCESSING

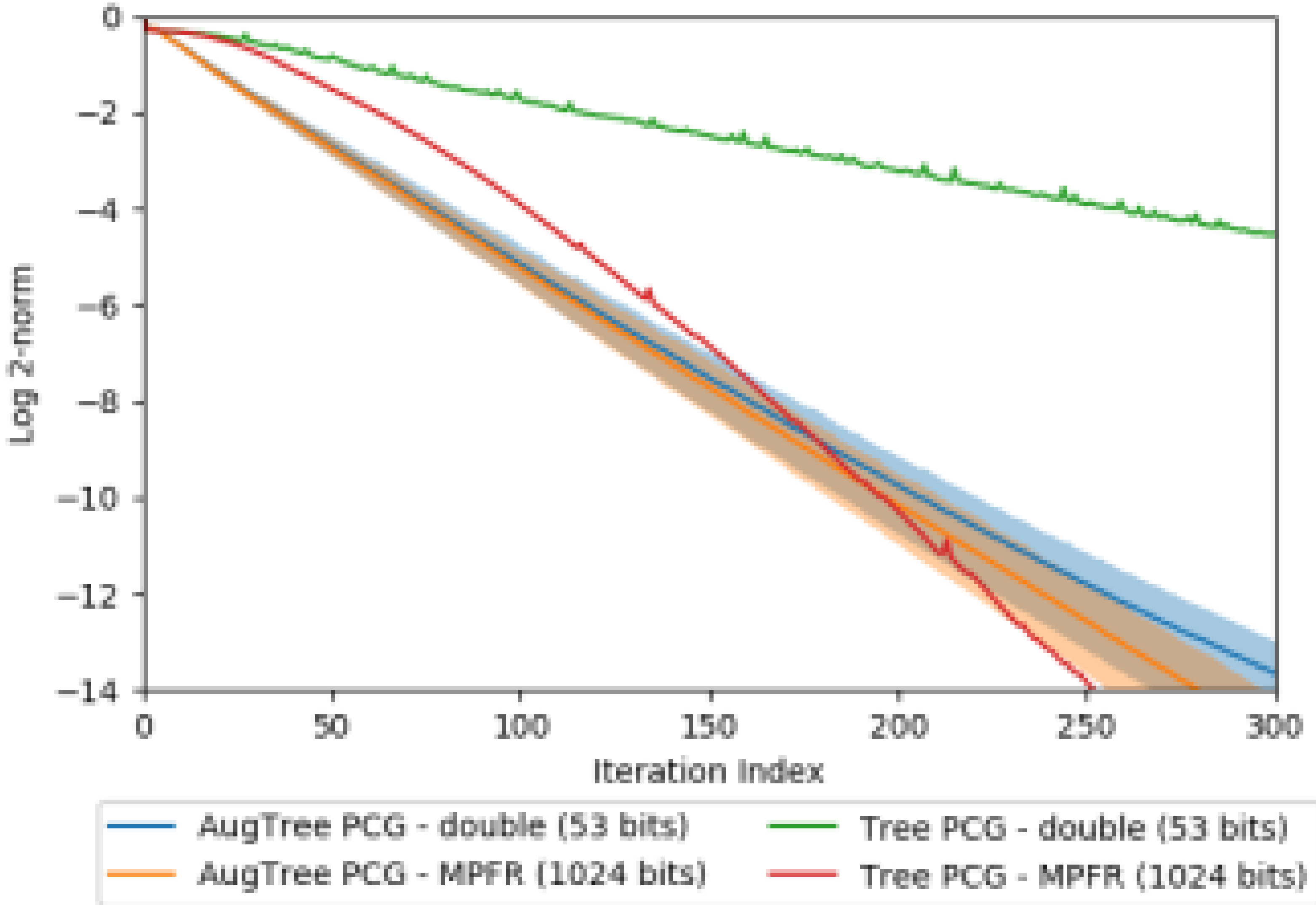
- Add some edges to a tree to form a `batched' preconditioner
- Use direct methods to factorize preconditioner explicitly



COMPARISON ON 3D CUBE



COMPARISON ON IPM MATRIX



EVALUATING LAPLACIAN SOLVERS

- We have a much better idea of what are the instances to test on now:
  - Instances that are numerically close to paths
  - Weighted grid graphs
  - Inner loops of optimization algorithms
  - [Deweese-Gilbert `18]: evolve instances
- Benchmark incomplete Cholesky?
- Measuring numerical behaviors?
- Benchmark w.r.t. applications: ground-truth instead of residual error?