High Performance Solvers for Linear Systems in Graph Laplacians

Richard Peng
Georgia Tech

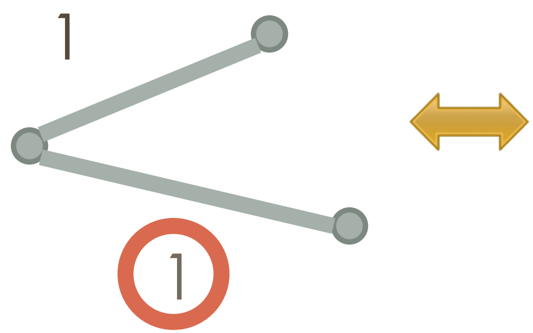
Joint with Kevin Deweese, John Gilbert, Gary Miller, Serban Stan, Haoran Xu, and Shen Chen Xu

OUTLINE

- **Laplacians**
- Tree Based Solvers
- Benchmarks and Evaluations
- Fixes and Modifications

GRAPH LAPLACIANS

Matrices that correspond to undirected graphs



2	-1	-1
-1	1	0
-1	0	1

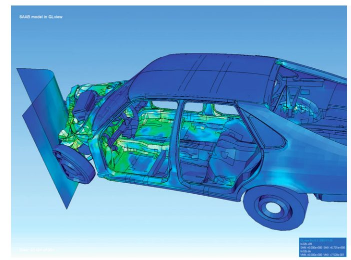
Key idea:

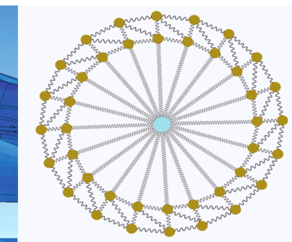
- Variables ⇔ vertices
- Non-zeros ⇔ edges

THE LAPLACIAN PARADIGM

[ST`04]: O(mlogcn) theoretical bounds for solving **Lx = b**

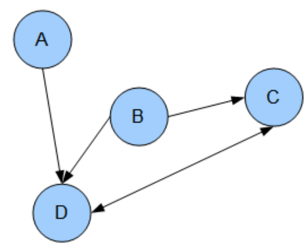
Directly related:
Elliptic systems

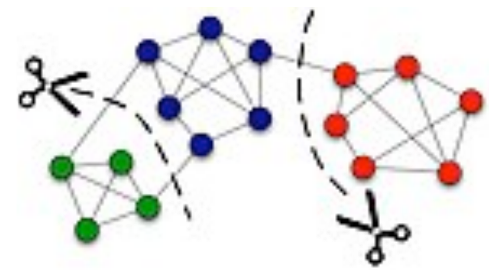




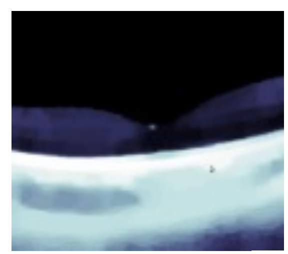
Lx=b

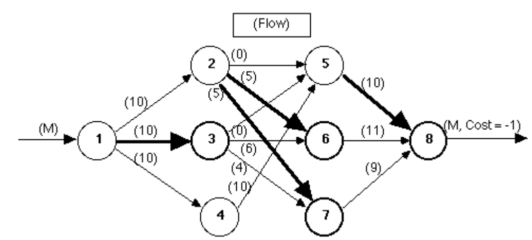
Few iterations:
Eigenvectors,
Heat kernels

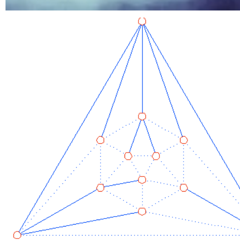


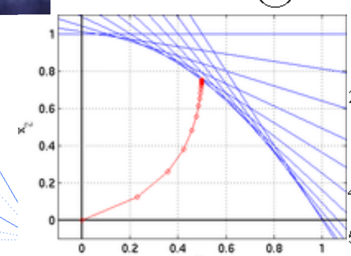


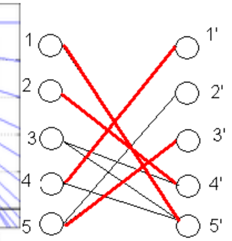
Many iterations / modify algorithm
Graph problems
Image processing





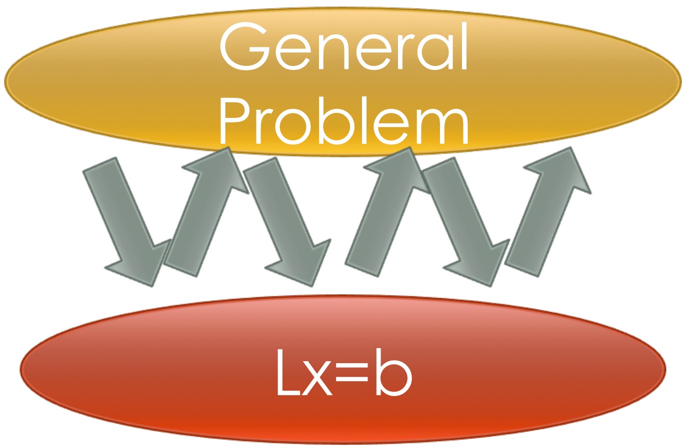






NEED: ROBUST SOLVERS

The Laplacian paradigm:



Sequence of (adaptively) generated linear systems

Main difficulties:

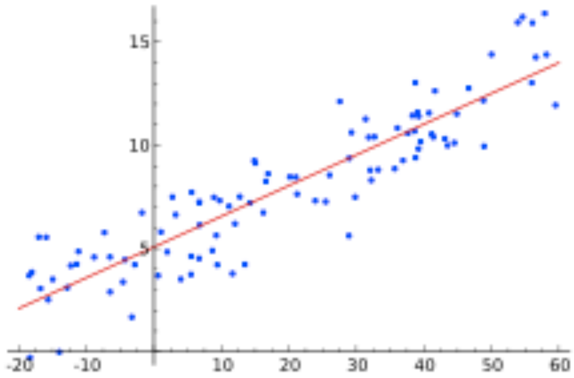
- Widely varying weights
- Multiscale behavior

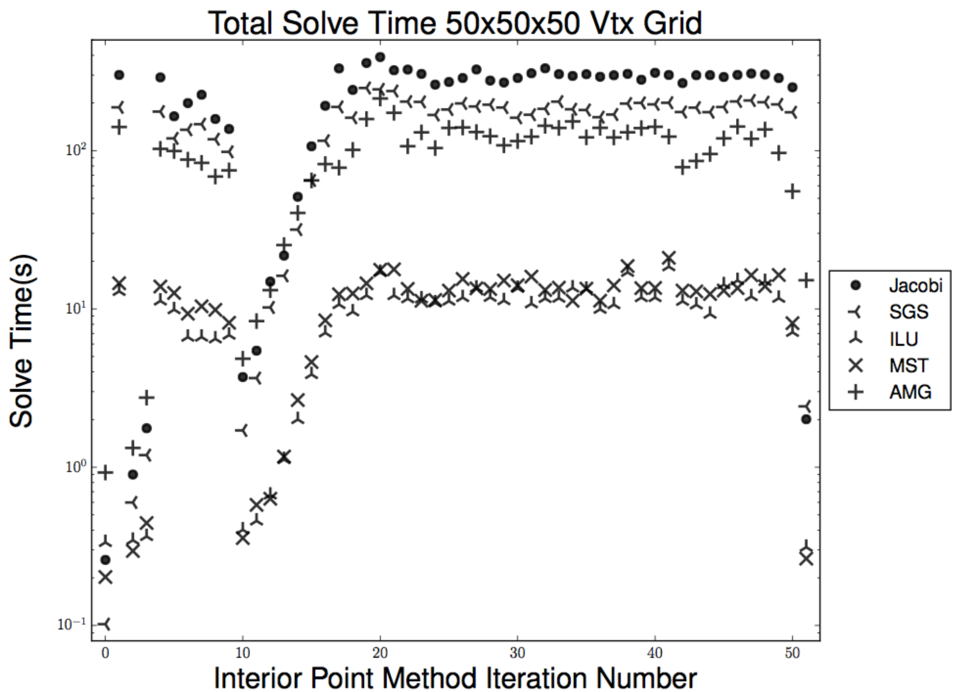
INSTANCE: ISOTONIC REGRESSION

[Kyng-Rao-Sachdeva `15]:

https://github.com/sachdevasushant/Isotonic/blob/master/README.md :

…we suggest rerunning the program a few times and/or using a different solver. An alternate solver based on incomplete Cholesky factorization is provided with the code.



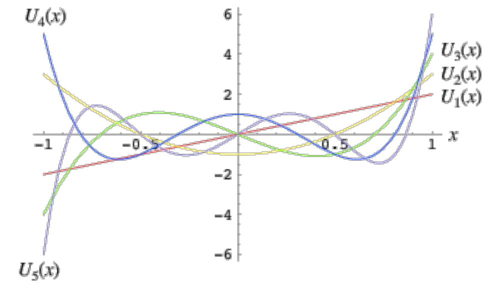


OUTLINE

- Laplacians
- **Tree Based Solvers**
- Benchmarks and Evaluations
- Fixes and Modifications

ITERATIVE METHODS

Division with multiplication: solve **Ax**=**b** using a linear combination of **b**,**Ab**, $\mathbf{A}^2\mathbf{b}$, $\mathbf{A}^3\mathbf{b}$, …



Preconditioned iterative methods: solve $\mathbf{B}^{-1}\mathbf{A}x = \mathbf{B}^{-1}\mathbf{b}$ instead

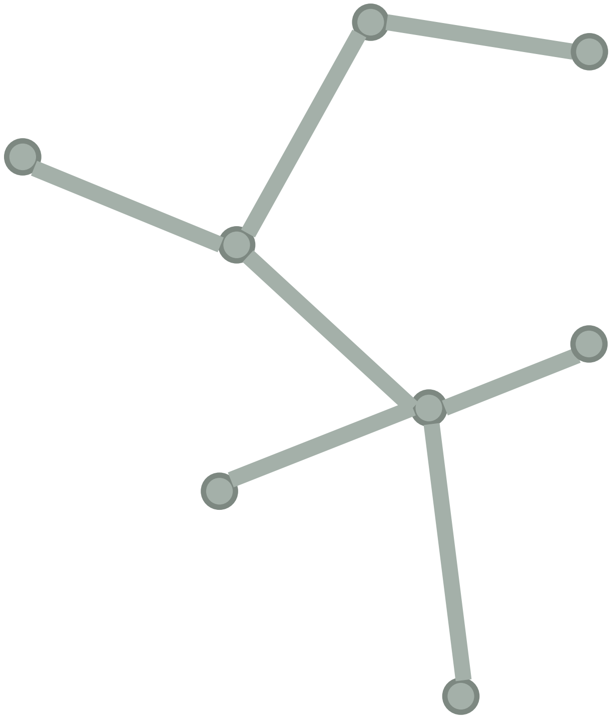
Computational cost:

- # iterations: ≤ (condition number of $\mathbf{B}^{-1}\mathbf{A}$)$^{1/2}$
- Each iteration: solve linear system in **B**

TREE PCG

Preconditioner **B** needs:

- **By** = **r** is easy to solve
- good approximation to **A**



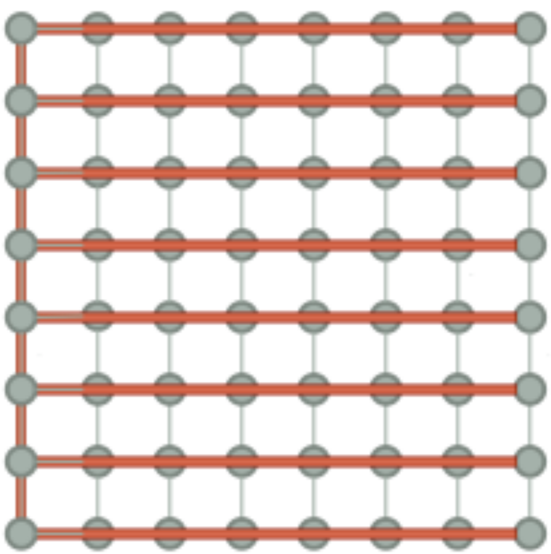
Trees:

- finite approximation
- linear time solve

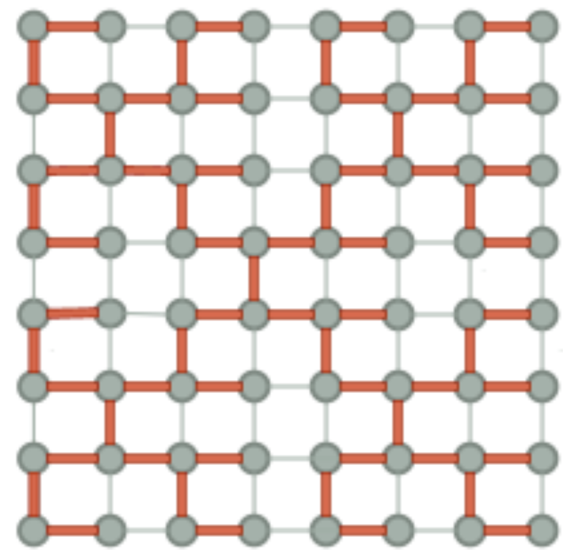
[Vaidya `91]Augmented Trees: precondition with MST plus a few edges at a time

WHAT'S THE RIGHT TREE

[BH` 01]: Key quantity: total stretch



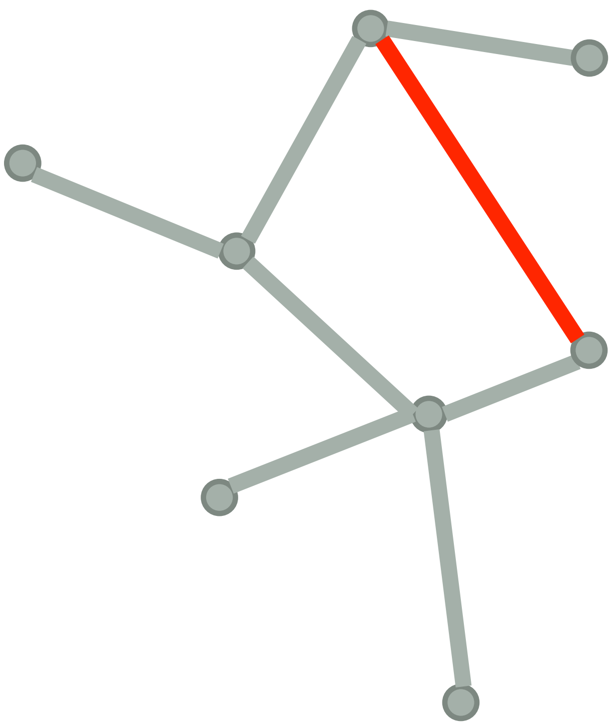
VS



- Graphs have trees with total stretch about mlogn
- [SW `09]: treePCG runs in $(\text{totalStretch})^{1/3}$) iterations

CYCLE TOGGLING

[DS`84] Dual of vertex labels solutions for **Lx**=**b** is an electrical flow

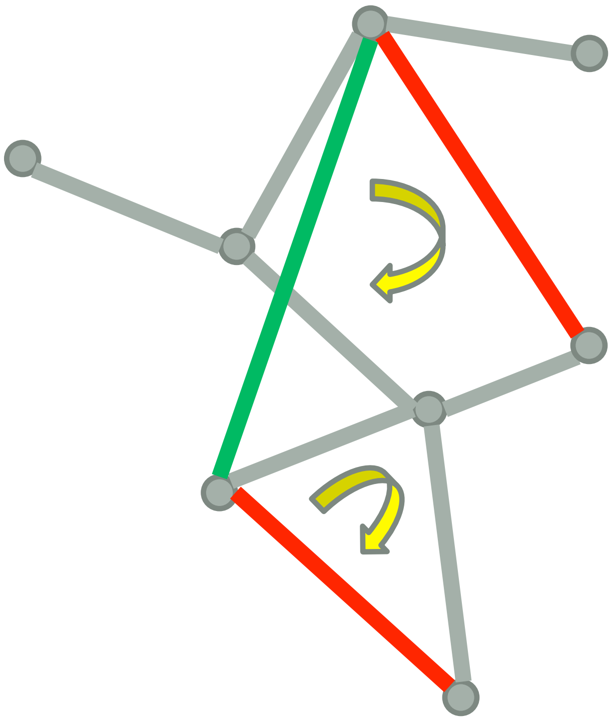


[KOSZ `13]:

- Fundamental cycles: edge + path to tree
- OPT is also optimal on each fundamental cycle

CYCLE TOGGLING

- Fix cycles (sampled via stretch) one at a time
- Speed up calculations using data structures



[KOSZ `13]: O(totalStretch) cycle toggles, each costing O(logn), Kaczmarz method on a tree basis

[LS `13]: Can accelerate to O((totalStretch×m)$^{1/2}$) toggles

OUTLINE

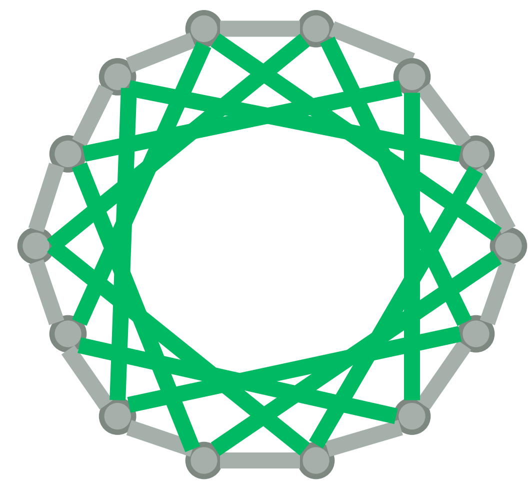
- Laplacians
- Tree Based Solvers
- **Benchmarks and Evaluations**
- Fixes and Modifications

EXPERIMENTAL SETUP

Data sets:

- Grids / cubes / Cayley graphs
- Hard cases for combinatorial graph algorithms from DIMACS
- IPM graphs: run interior point method on these graphs
- Heavy path: next slide

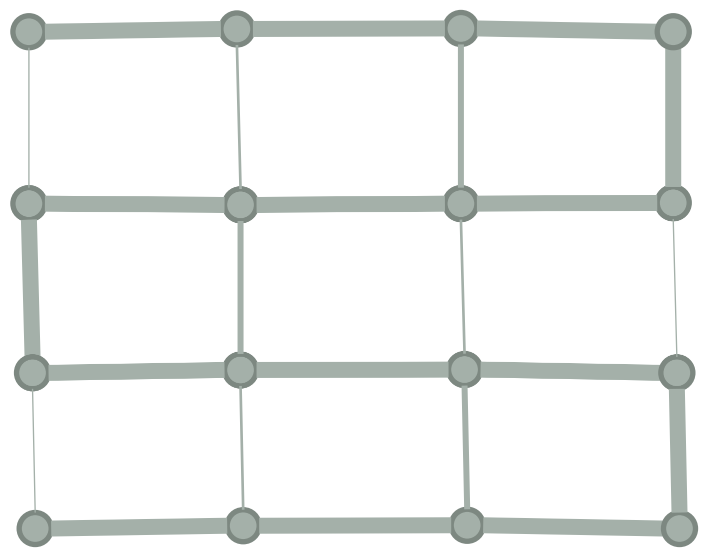
Length n cycle
i~j if |i-j| in S



- Size: 10^6 here, can scale to $10^7/10^8$
- Generate random **x**, then set **b**←**Lx**
- Goal accuracy: $\|\mathbf{Lx} - \mathbf{b}\|_2 < 10^{-6} \|\mathbf{b}\|_2$

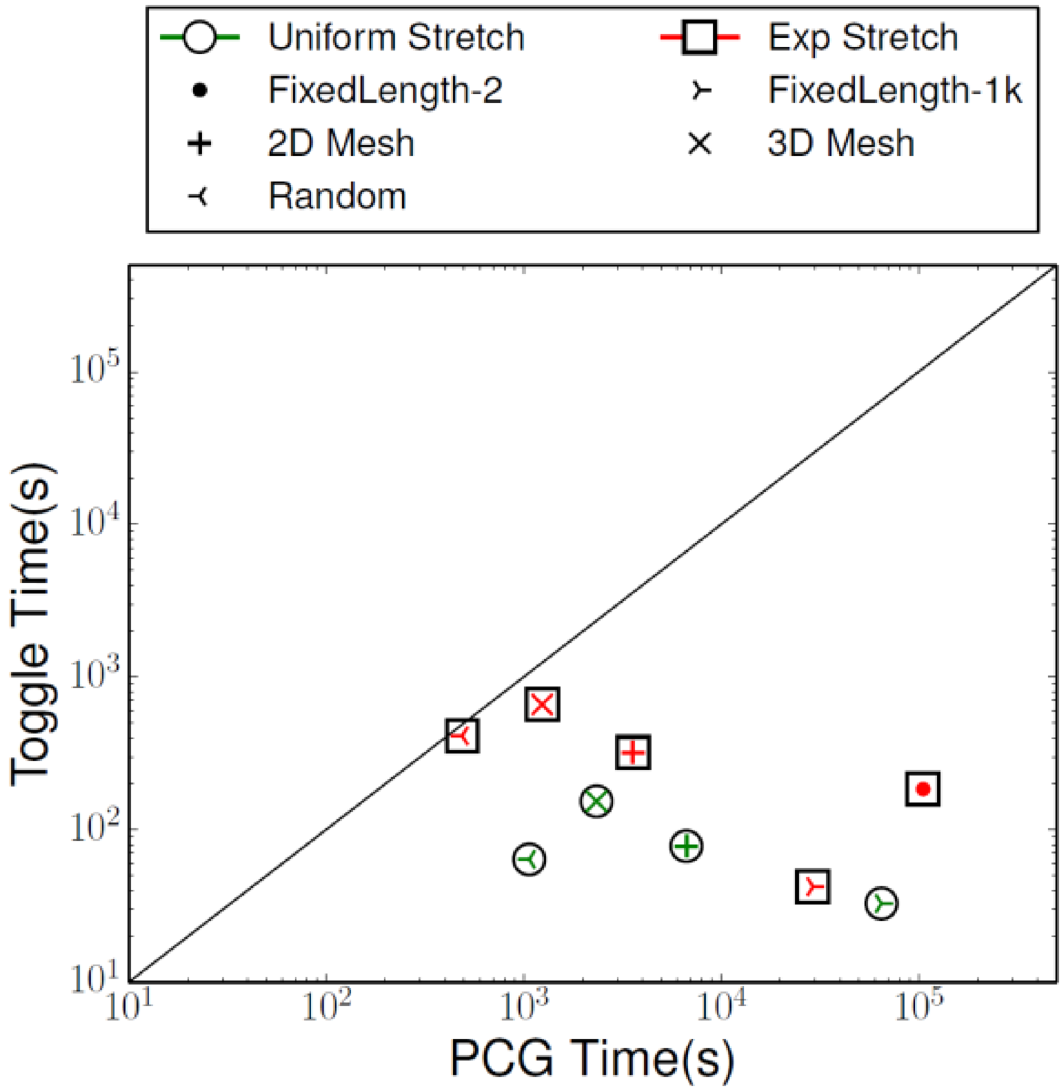
HEAVY PATH GRAPHS

Pick a Hamiltonian path, weight all other edges to get a certain stretch distribution

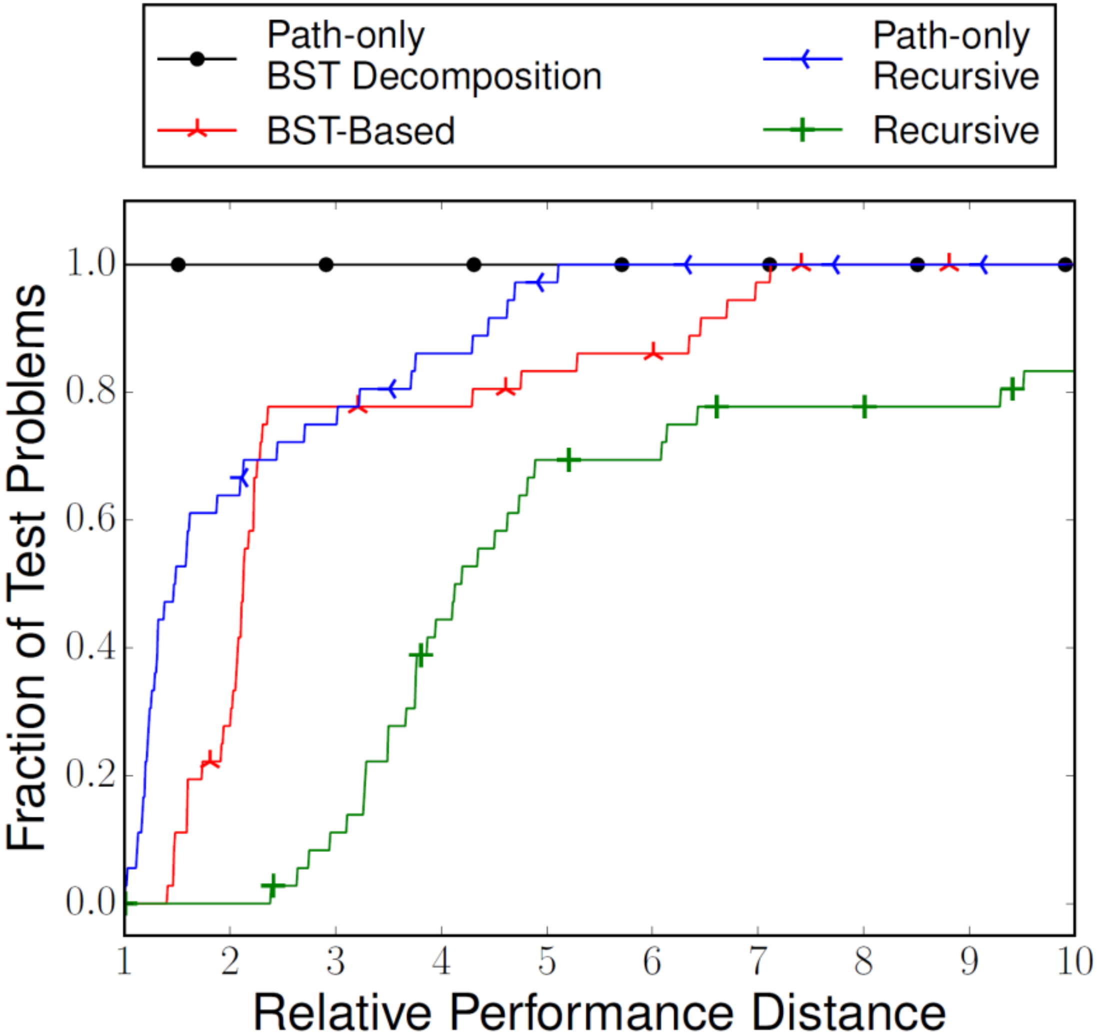


- Bad case for unpreconditioned CG
- Easy for data structures: no forks in tree

PERFORMANCES VS PCG



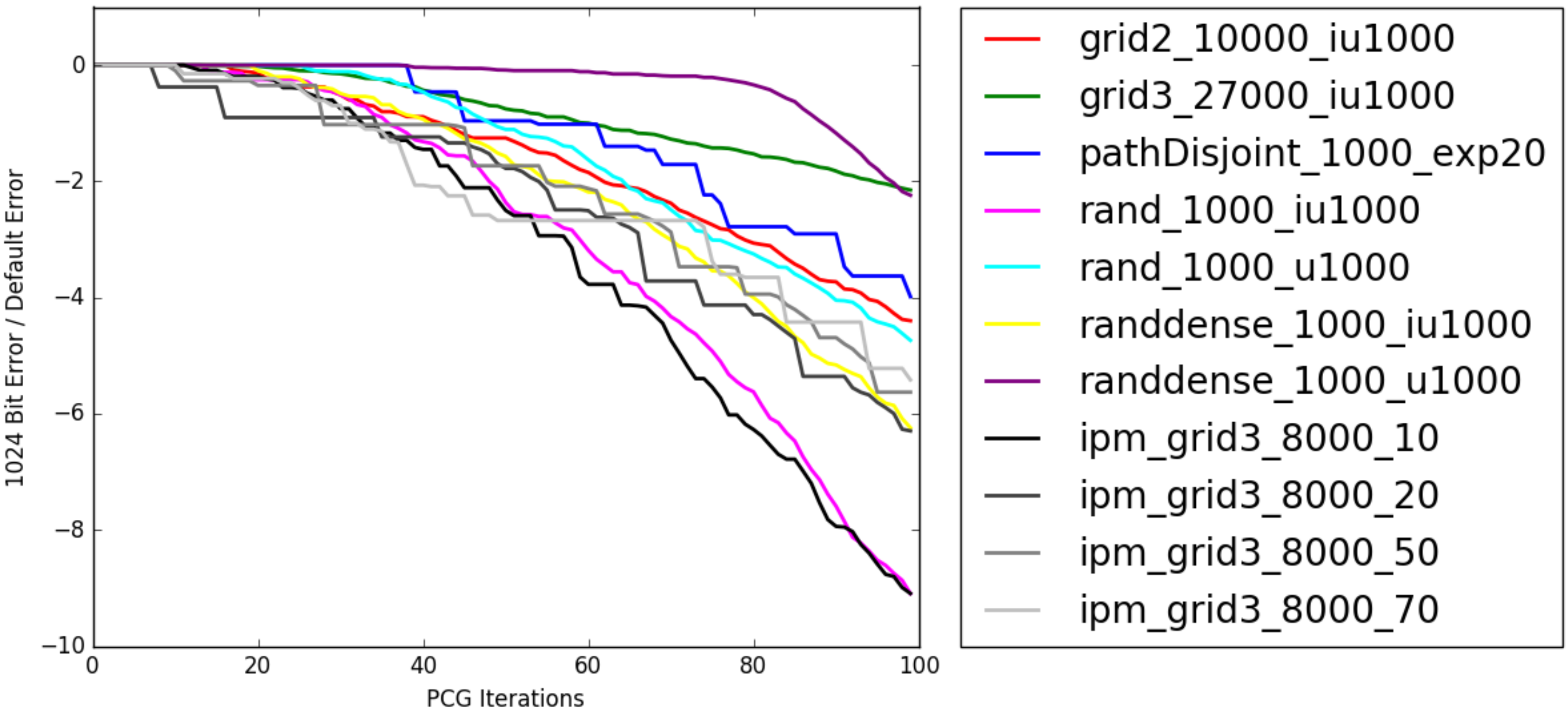
DIFFERENT DATA STRUCTURES



Heavy path graphs allow testing for:

- Overhead of trees
- Online vs. offline data structures

(IN)STABILITY OF TREE PCG

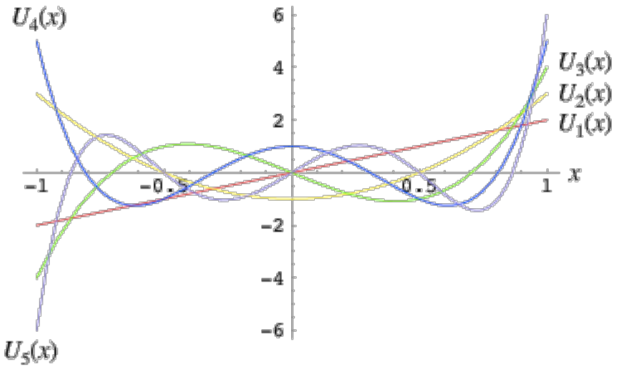


Errors at iterations decrease as one switch to 1024-bit precision via MPFS

OUTLINE

- Laplacians
- Tree Based Solvers
- Benchmarks and Evaluations
- **Fixes and Modifications**

THEORETICAL EXPLANATION



Core idea in the analysis of [SW`09]:

- Relative eigenvalues $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$
- Total stretch = $\lambda_1 + \lambda_2 + \ldots + \lambda_n$

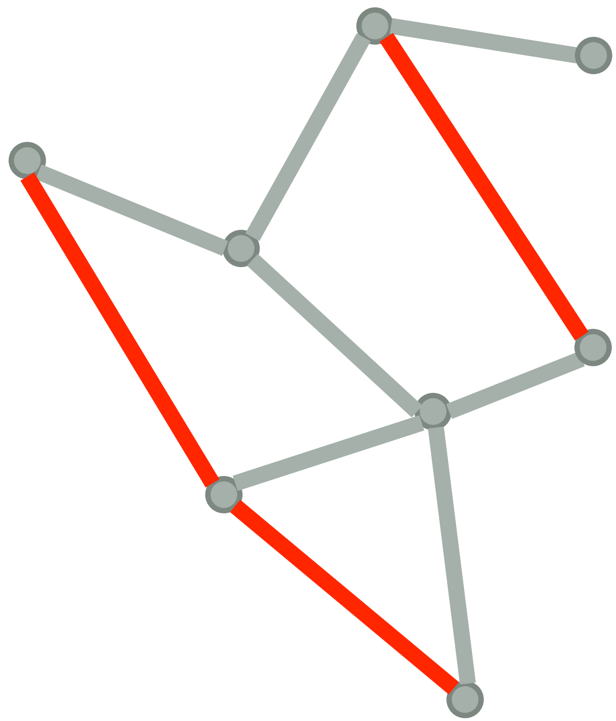
Krylov space view of CG: k steps finds the best fit degree-k polynomial for $(\lambda_i, 1/\lambda_i)$

- [SW`09]: existence of such a polynomial of degree ≤ totalStretch$^{1/3}$.

Issue [CKPPSV`16]: interpolating $m^{1/3}$ values exactly requires high precision coefficients.

ULTRA-SPARSIFIERS

[KMP `10, `11]: adding O(klogn) edges to tree gives preconditioner with condition # O(totalStretch / k)



Greedy elimination → O(klogn):

- Theoretically: $m^{5/4}$ total
- Practically: exact methods run well on about 10^4 edges
- Optimization interpretation: block Kaczmarz with tree basis

Essentially back to Vaidya's MST + edges

AUGTREE VS TREEPCG

	Tree PCG		Augmented Tree PCG	
	# iter	time	# iter	time
2MeshUnweightedUniformStretch	168	14.5s	38	4.57s
2MeshUnweightedUniformStretch	1382	116s	84	9.24s
2MeshUniformStretch	139	11.9s	30	3.57s
2MeshExpStretch	1482	123s	72	8.11s
3MeshUnweightedUniformStretch	461	30.6s	55	5.70s
3MeshUnweightedExpStretch	2745	186s	137	12.8s
3MeshWeightedUniformStretch	407	26.5s	44	4.91s
3MeshWeightedExpStretch	2212	147s	120	11.2s
ChainUniformStretch	11	0.502s	11	0.772s
ChainExpStretch	589	26.6s	72	4.93s
FixedLengthUniformStretch	115	5.11s	23	1.81s
FixedLengthExpStretch	1366	62.1s	42	3.04s

- Size = 10^6
- k optimized for runtime

What we need: a better C++ Cholesky package

ONGOING WORK

- Repos:
  - https://github.com/sxu/cycleToggling
  - https://github.com/serbanstan/TreePCG
  - Also see: Laplacians.jl (by group at Yale)
- Stability of tree preconditioned CG?
- How fast, or how parallel can augmented tree solver become?
- Combine with Sparsified Gaussian Elimination? (Kyng-Sachdeva?)
- Compare vs. other solvers
- Why is Julia's MPFR slower?