Extending Algebraic Multigrid Solvers

SciDAC 11

Luke Olson, University of Illinois at Urbana-Champaign

Jacob Schroder	Colorado (LLNL, SNL)	generalized interpolation
Hormozd Gahvari	Illinois (LLNL)	multigrid scalability
James Lai	Illinois (SNL)	high-order conforming
Steven Dalton	Illinois (LLNL, Nvidia)	accelerating multigrid
John Hall	Illinois (LLNL)	automating multigrid
Bill Gropp	Illinois	
Rob Falgout	LLNL	
Ulrike Meier-Yang	LLNL	
Ray Tuminaro	SNL	
Scott MacLachlan	Tufts	

$Ax = b$: old, but persistent

- large, sparse matrix
- represents a major bottleneck in many simulations both small and large
- advanced Krylov solvers + fancy **preconditioning**

the obvious trend...

- complex, non-symmetric, unstructured, not-so-nice applications
- bio, information, high-dimension, stochastic, etc.

- heterogeneous, large-scale architectures
- thinner cores, acceleration units

the obvious trend...

- complex, non-symmetric, unstructured, not-so-nice applications
- bio, information, high-dimension, stochastic, etc.

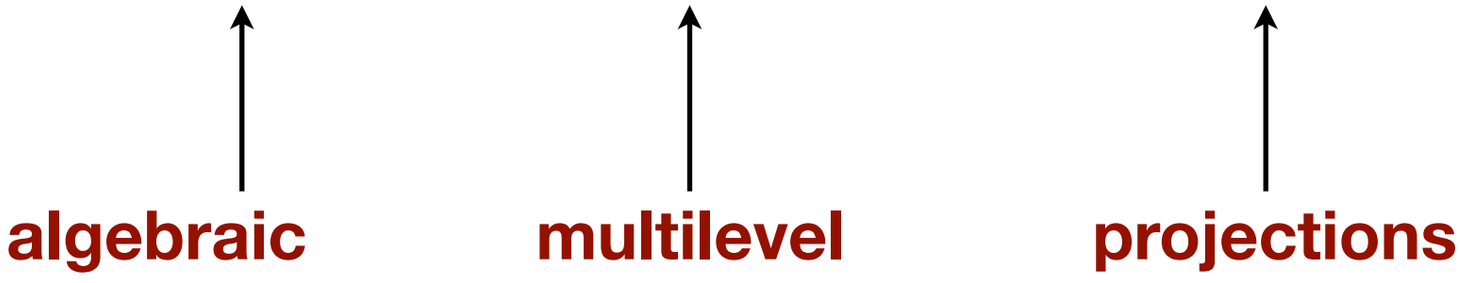
} elliptic not motivating

- heterogeneous, large-scale architectures
- thinner cores, acceleration units

} coarse and fine grained parallelism

wanted: more robust, efficient, and principled solver

wanted: more robust, efficient, and principled solver



wanted: more robust, efficient, and principled solver

algebraic **multilevel** **projections**

solution: move beyond basic problems

solution: consider communication/complexity

solution: optimality into the multilevel hierarchy

show some success in robustness

highlight optimality

progress in high-performance

algebraic multilevel methods are flexible

“...but multigrid is old”

no!	multigrid is only a structure; new approaches are transforming solver development
yes!	history of successful development and continued progress

Multilevel view

attenuate high energy quickly with with relaxation
attenuate low energy error through coarse-grid correction

Multilevel view

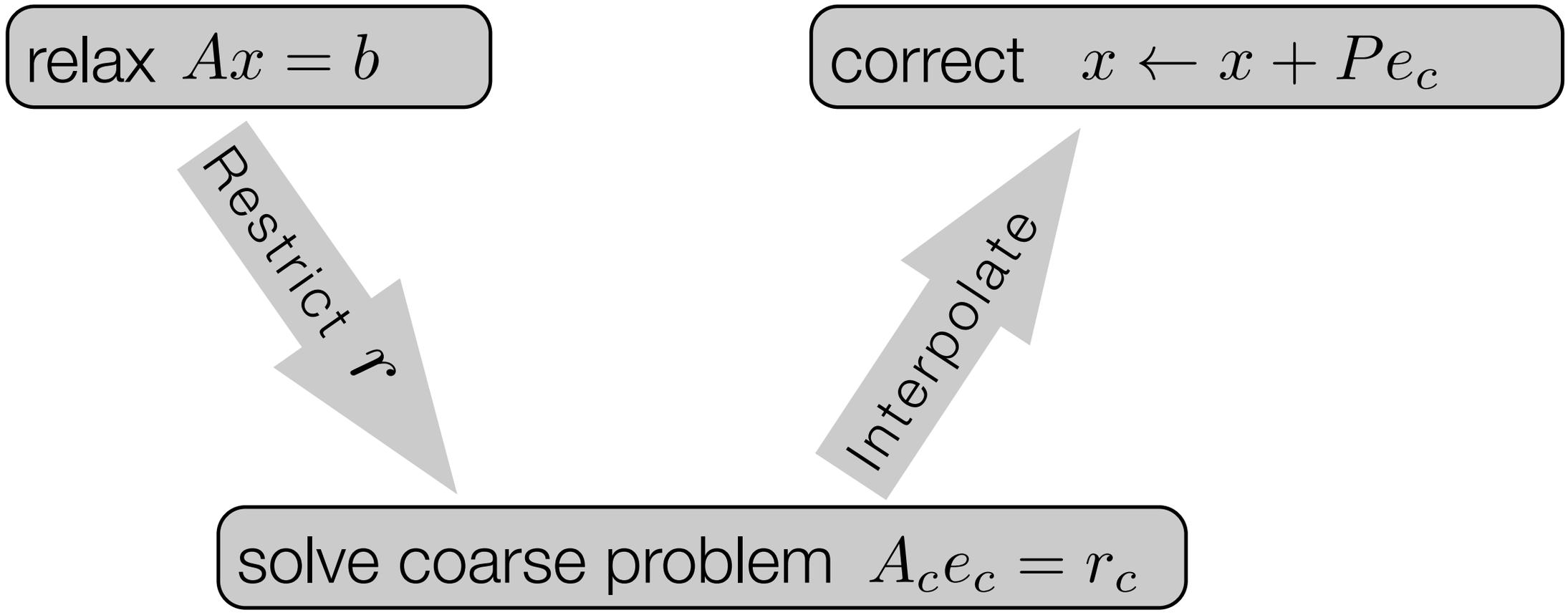
Jacobi, Gauss-Seidel
Chebychev
ILU
Kaczkmarz

attenuate high energy quickly with with relaxation
attenuate low energy error through coarse-grid correction

DD
projection methods
Interpolation

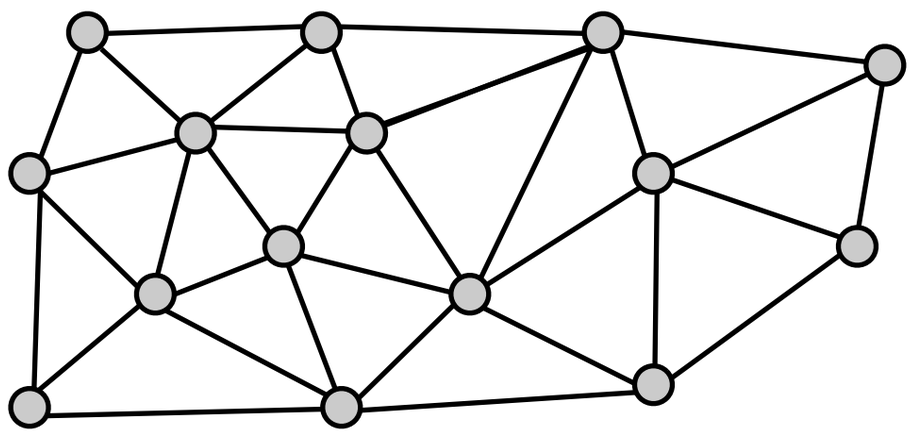
Multilevel view

attenuate high energy quickly with with relaxation
attenuate low energy error through coarse-grid correction



Algebraic Framework

- aggregation: groups of fine nodes form coarse nodes



- an initial interpolation pattern

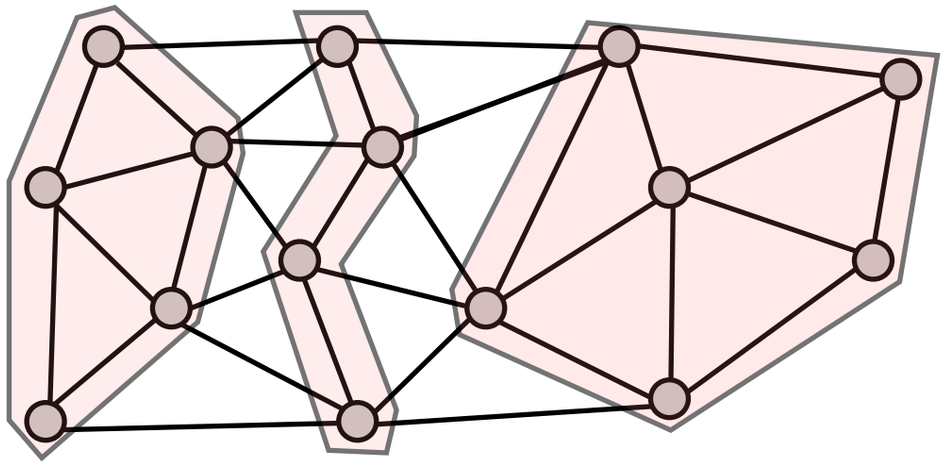
- find an optimal interpolation operator P that contains low energy

- cycle: $x \leftarrow x + P(P^T A P)^{-1} P^T r$

  $x \leftarrow x + P A_{coarse}^{-1} P^T r$

Algebraic Framework

- aggregation: groups of fine nodes form coarse nodes



fine: 15

coarse: 3

- an initial interpolation pattern

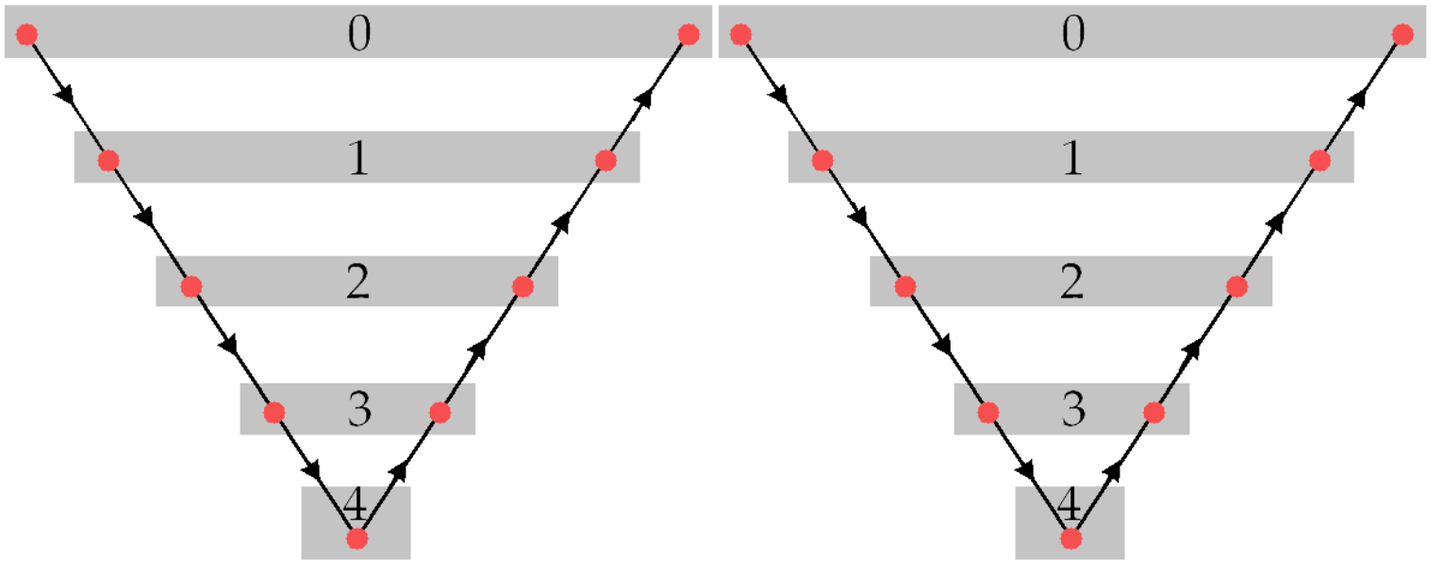
- find an optimal interpolation operator P that contains low energy

- cycle: $x \leftarrow x + P(P^T A P)^{-1} P^T r$

$x \leftarrow x + P A_{coarse}^{-1} P^T r$

what we need

- an idea of the low energy: physics, training, intuition $AB \approx 0$
- a strength measure to determine strong node couplings S_{ij}
- a parallel aggregation method Agg
- low complexity, optimal interpolation P
- better cycling
  - richer coarse grids
  - “parallel” cycling



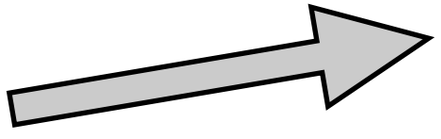
Setup

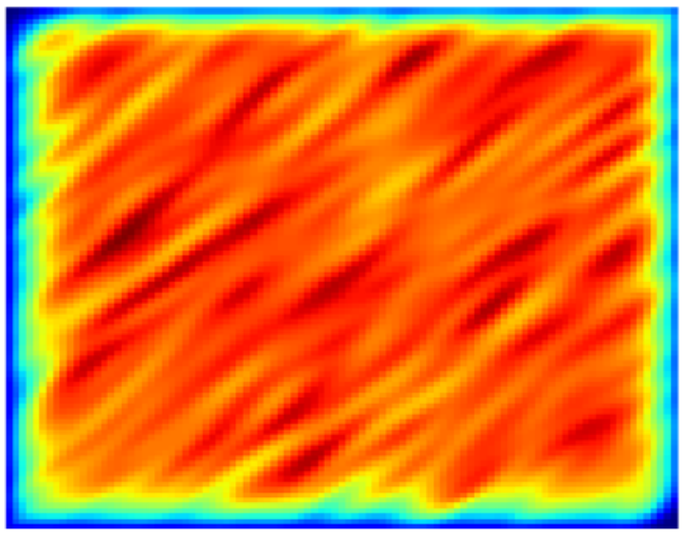
Solve

Idea #1

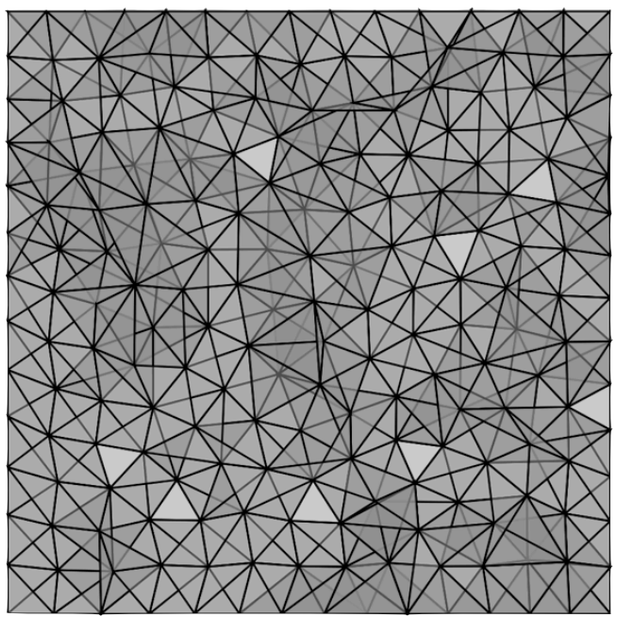
Obtaining Low-energy

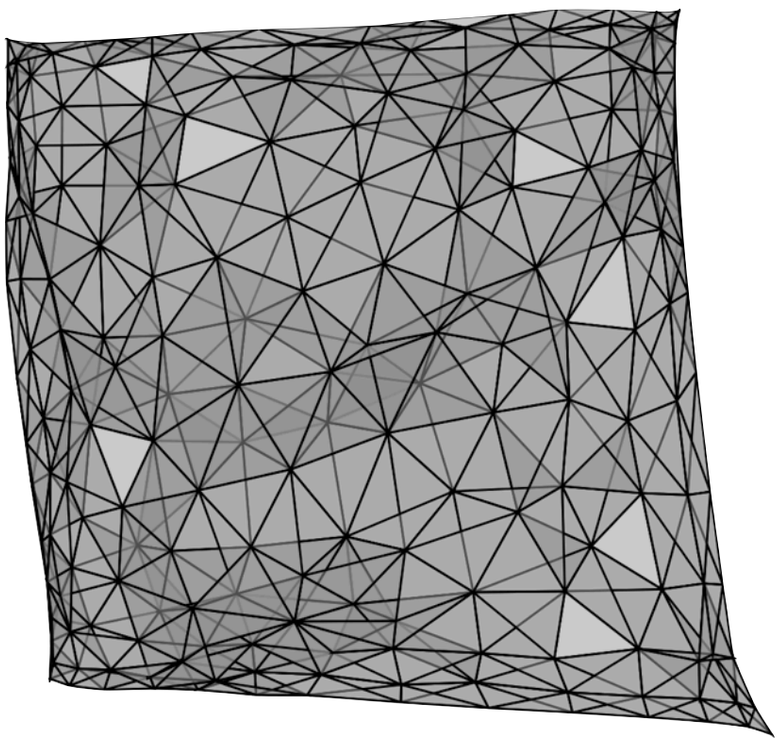
- *a priori* knowledge
- adapting a cycle*
- **pre-smooth guesses**





- topological inference



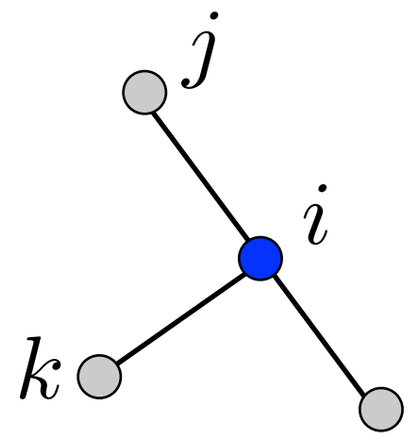


Idea #2

Classic Strength: the first mistake

- i strongly depends on j if

$$\frac{|A_{ij}|}{\sqrt{A_{ii}A_{jj}}} \geq tol$$

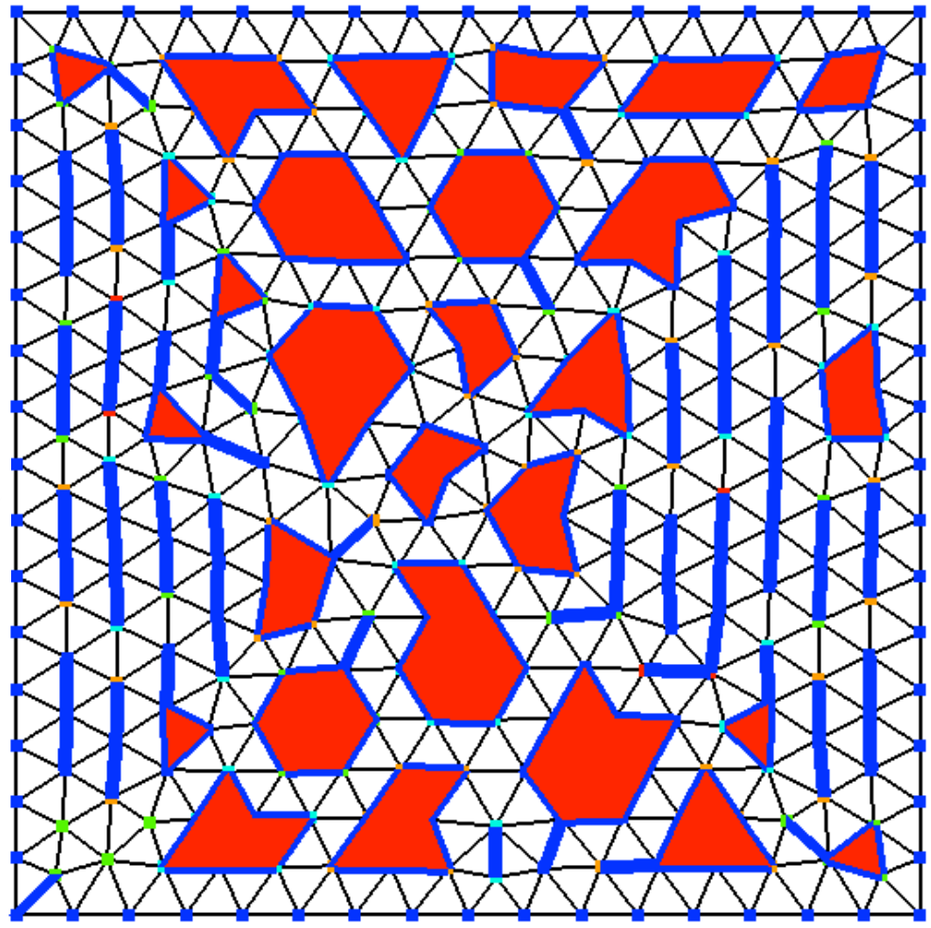


- i strongly depends on j if

$$-A_{ij} \geq tol * \max_{k \neq i} -A_{ik}$$

both “think” elliptic

anisotropy

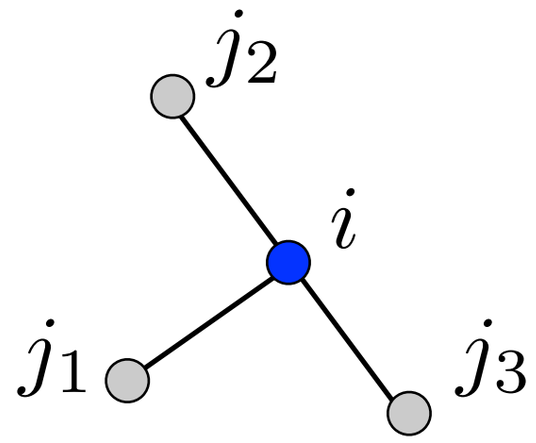


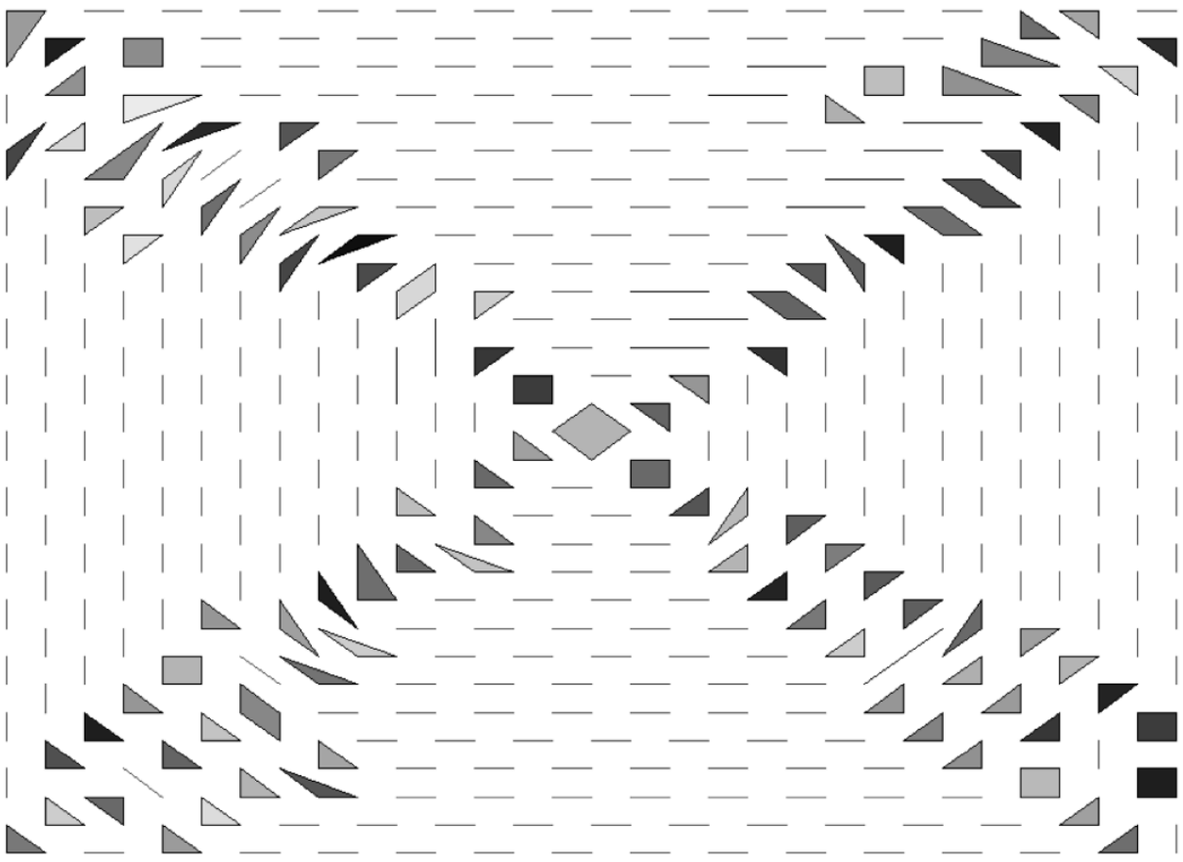
*** Olson, Schroder, Tuminaro, *A new perspective on strength measures in algebraic multigrid*, 2009

Idea #2

Evolution Measure

1. drop point source at a node
2. evolve point source with A
3. evaluate diffusivity at neighbors in comparison to known low energy



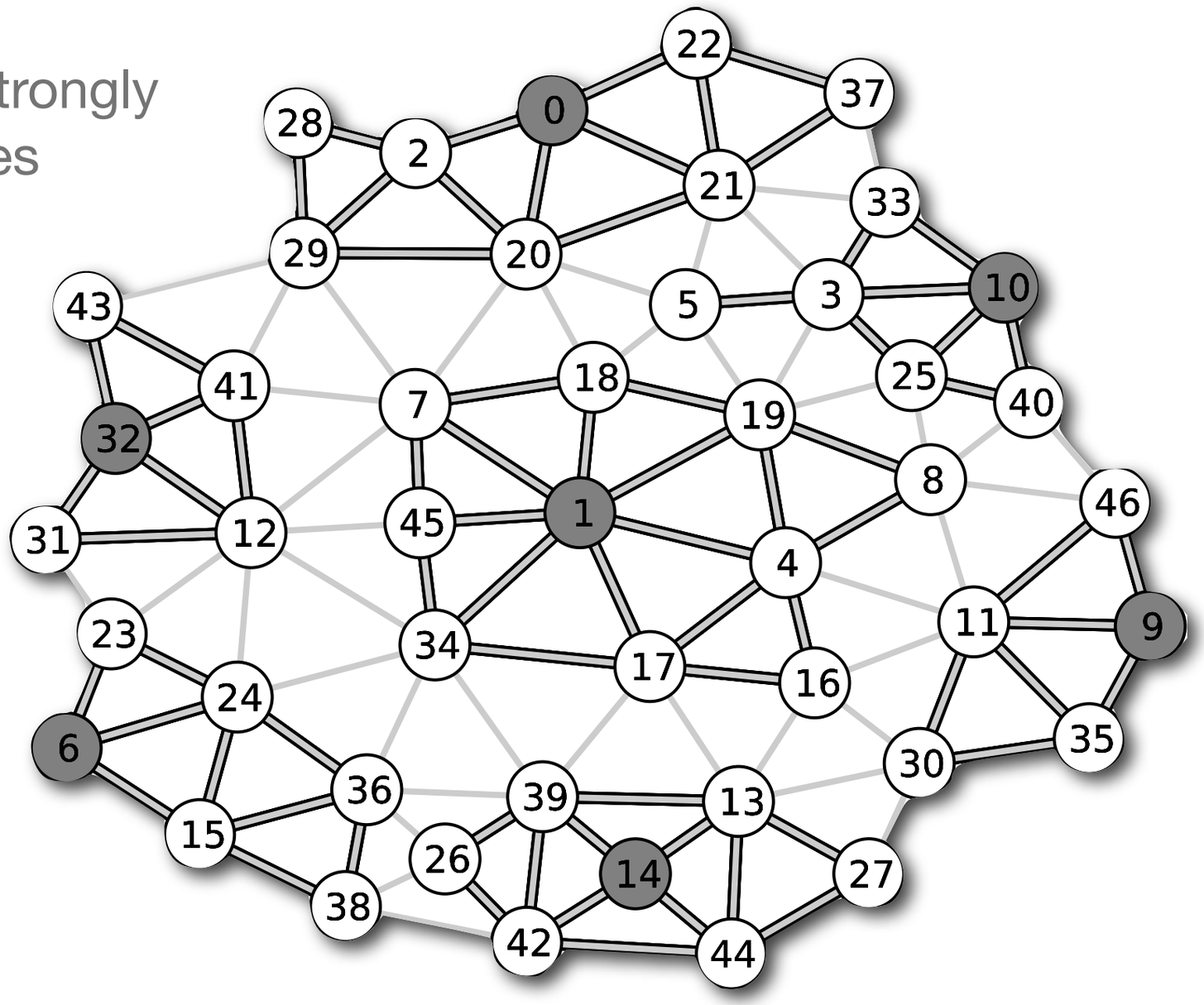


- efficient
- parameter insensitive
- Euler flow
- wave problems
- high-order
- discontinuous elements

Idea #3

Basic Graph Aggregation

- Metis
- Greedy: group collections of strongly connected unaggregated nodes
- Problems:
  1. size fixed
  2. sequential (greedy)

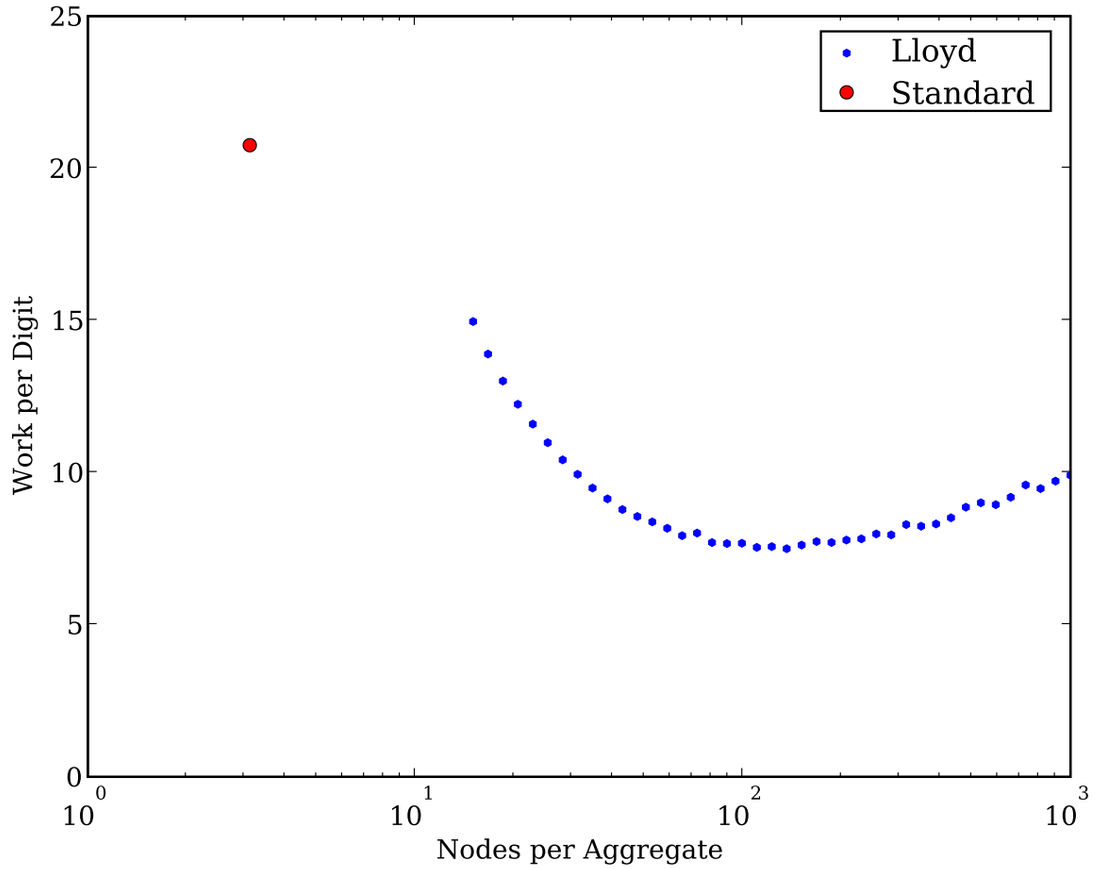


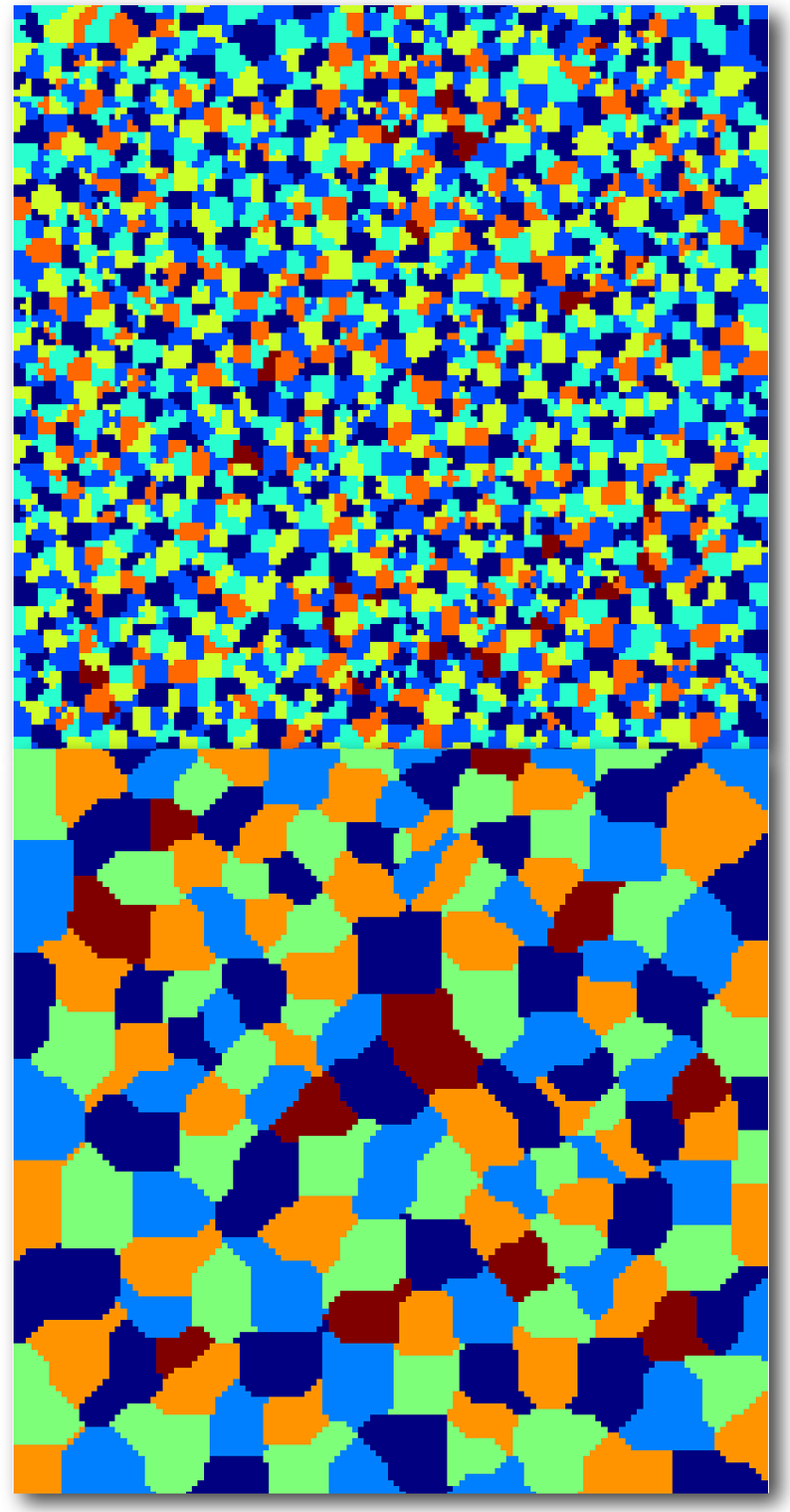
*** Bell, Olson, *Lloyd Aggregation*, 2008
*** Bell, Dalton, Olson, *AMG on the GPU, 2011*

Idea #3

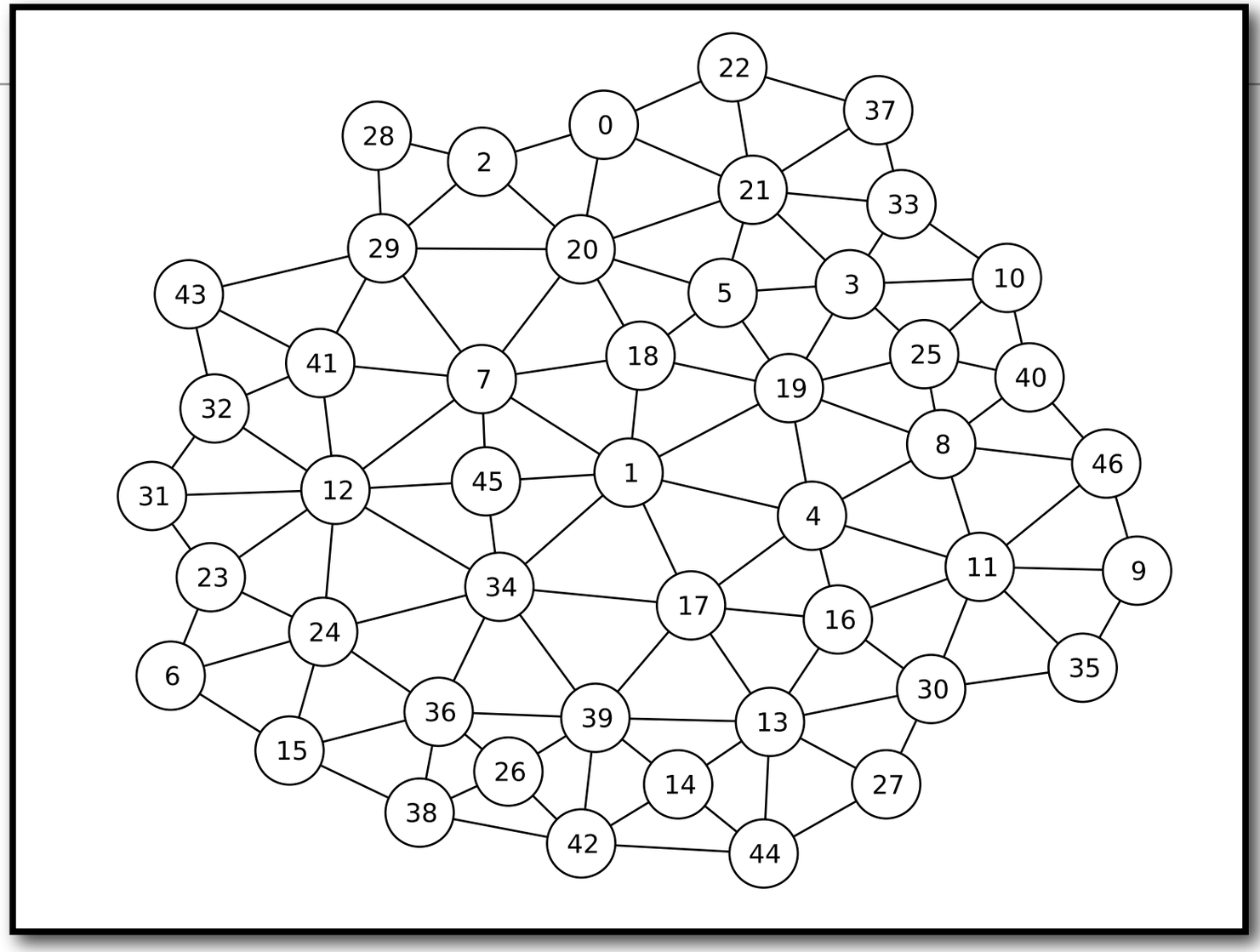
Flexible aggregation

- two approaches:
  1. shortest-path: ability to tune bandwidth
  2. MIS(k): parallel coarse grids == serial coarse grids

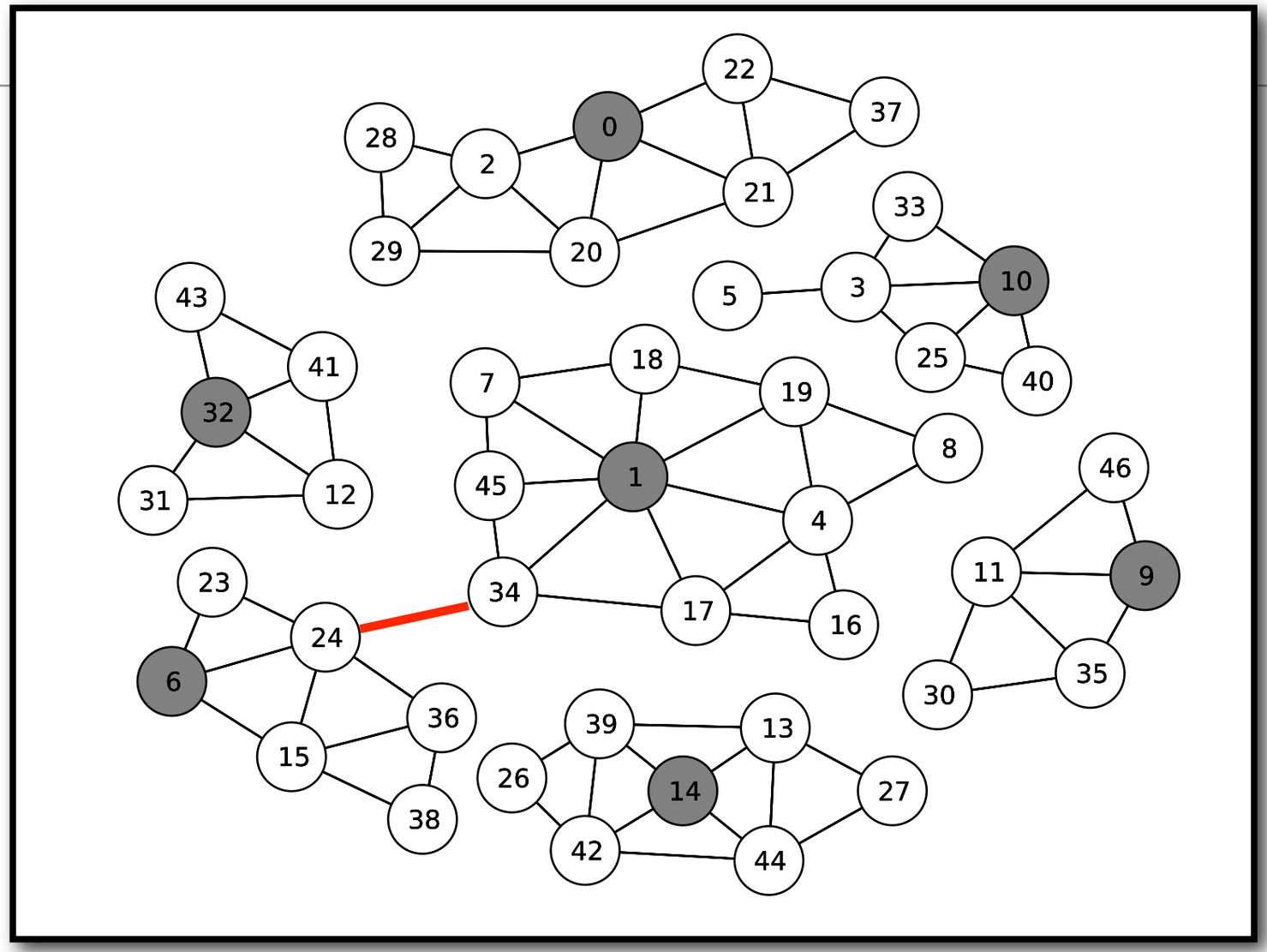




MIS(2)


22
37
28
0
2
21
33
29
20
43
5
3
10
41
18
25
7
40
19
32
8
46
1
31
12
45
4
23
11
9
34
17
16
24
6
35
30
36
39
13
15
26
14
27
38
42
44

MIS(2)



- root nodes more than 2 edges apart (> distance-2)
- an unaggregated node more than 2 edges from a root can become a root

independent

MIS(2)

maximal

Idea #3

MIS(2)

1. Given a MIS(2) of {0,1} (N nodes)

2. prefix scan to enumerate

```
thrust::exclusive_scan(set, set + N, set, init)
```

3. communicate to neighbors
   aggregate index (**SpMV**)

4. communicate index to unaggregated
   neighbors (another **SpMV**)

} ~ std. first pass

} ~ std. second pass

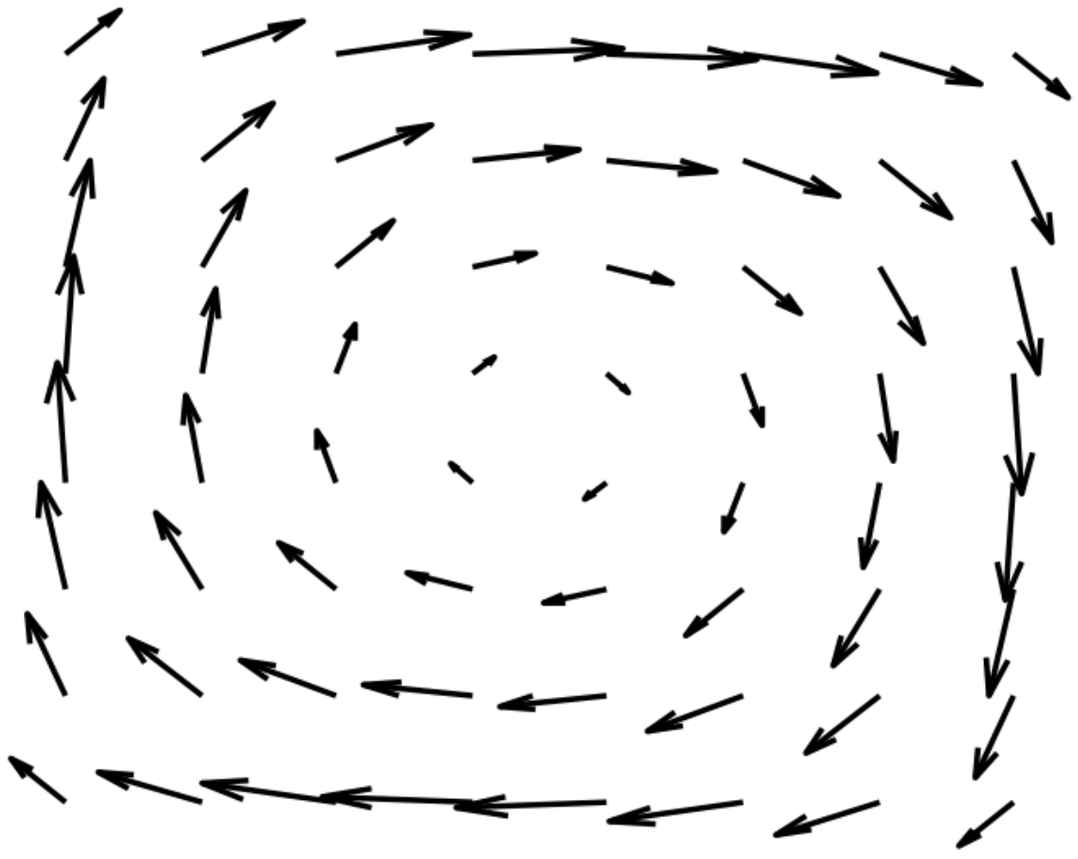
- Generalizes to MIS(*k*)
- allows for variable coarsening

*** Olson, Schroder, Tuminaro, A *general interpolation strategy for algebraic multigrid using energy-minimization,* 2010.

Idea #4

optimizing energy

- P should have low energy
  (low A-norm or A^*A-norm)
  1. determine sparsity pattern
  2. minimize energy column-wise (parallel)



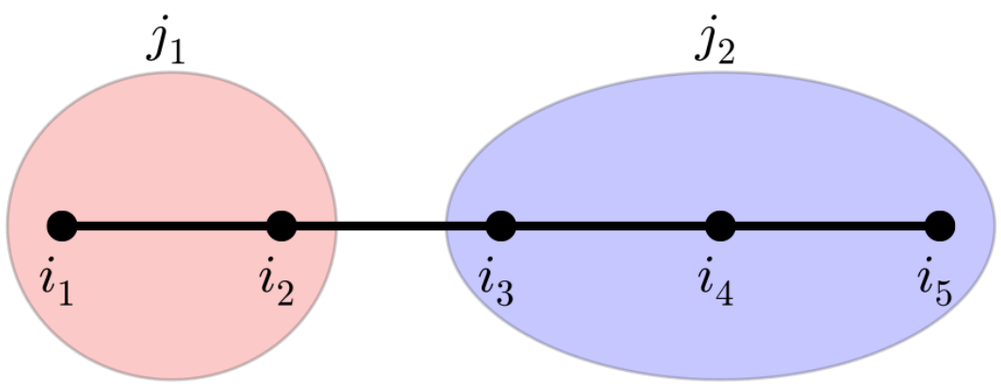
	std.	opt.
1/64	>150	24
1/128	>150	28
1/256	>150	33
1/512	>150	33

h

Idea #4

Interpolation: standard approach

- Set the sparsity pattern from aggregation



$$Agg = \begin{array}{c} \\ i_1 \\ i_2 \\ i_3 \\ i_4 \\ i_5 \end{array} \begin{array}{c} \begin{array}{cc} j_1 & j_2 \end{array} \\ \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \end{array}$$

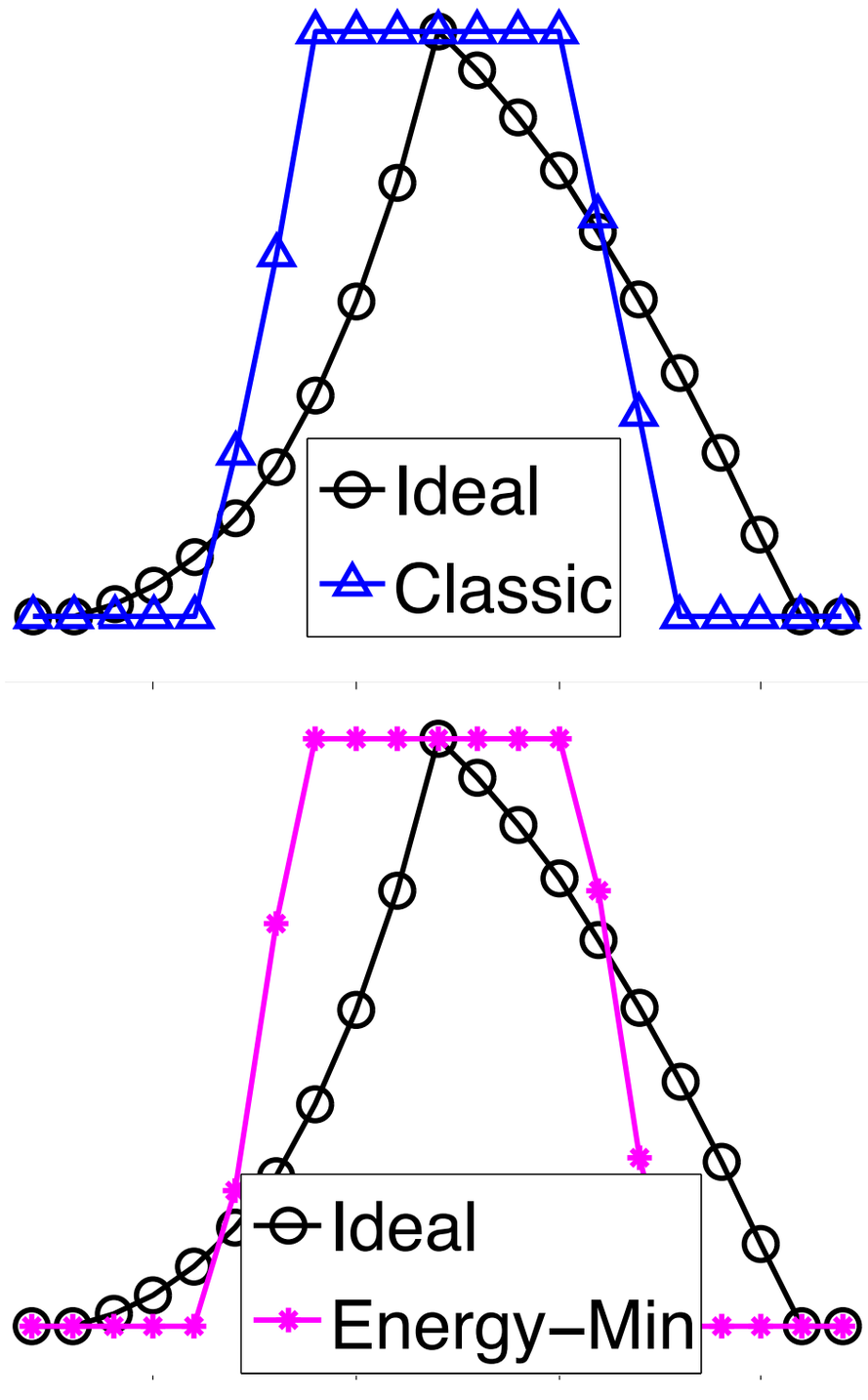
$$P_{tent}$$

$$P \leftarrow (I - \omega D^{-1} A) P_{tent}$$

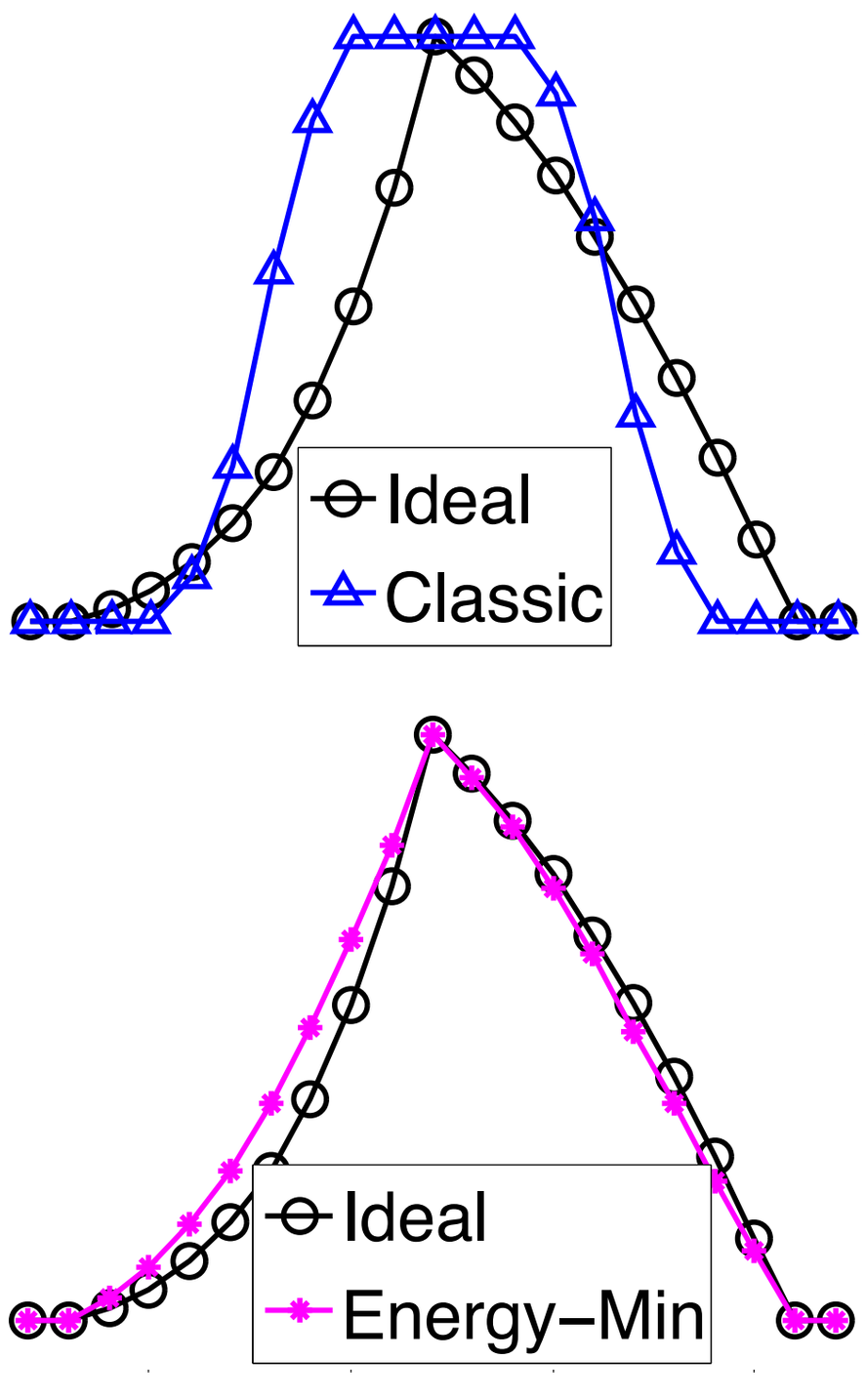
- reduce energy
- improve accuracy
- increase complexity

Idea #4

Range of Interpolation $-u_{xx} + \sin(x)u_x = f$



old



new

Idea #4

Toward General Interpolation

- Want P so that $u_{low} \in \mathcal{R}(P)$

1. Grow and fix sparsity pattern as $S^k P_{tent}$
2. Minimize residual of

$$AP_j = 0 \quad \text{for each column } j$$

3. Constraint the minimization with

$$PB_c = B$$

Idea #4

Toward General Interpolation

- Hermitian (and positive definite): use CG

$$AP_j = 0 \Leftrightarrow \min \|P_j\|_A$$

$$R = P^*$$

- Non-Hermitian: use GMRES

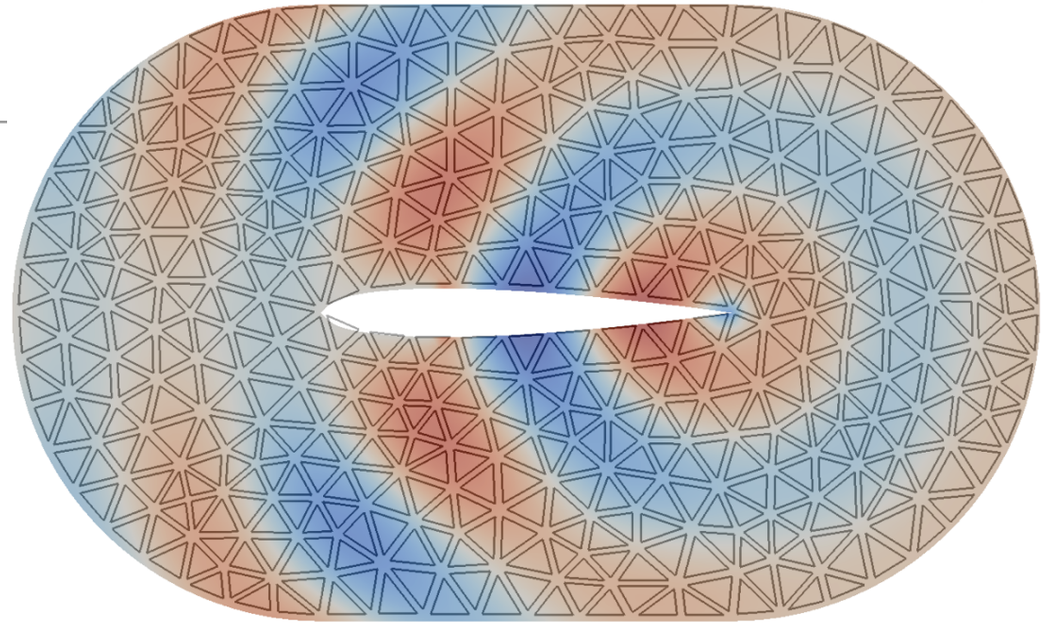
$$AP_j = 0 \Leftrightarrow \min \|P_j\|_{A^*A}$$

$$A^*R_j^* = 0 \Leftrightarrow \min \|R_j^*\|_{AA^*}$$

- Range of interpolation targets “right” low-energy
- Range of restriction* targets “left” low-energy
- Cost is comparable to that of standard smoothing

Ex. #1

Wave problems (Helmholtz)



- AMG problem: standard “low energy” $e^{i\omega x}$ modes (constant) break Nyquist rate

- answer: introduce wave modes at all levels

multiple wave modes: $\Rightarrow$ $e^{i\omega\,\cos(\theta)x+\sin(\theta)y}$

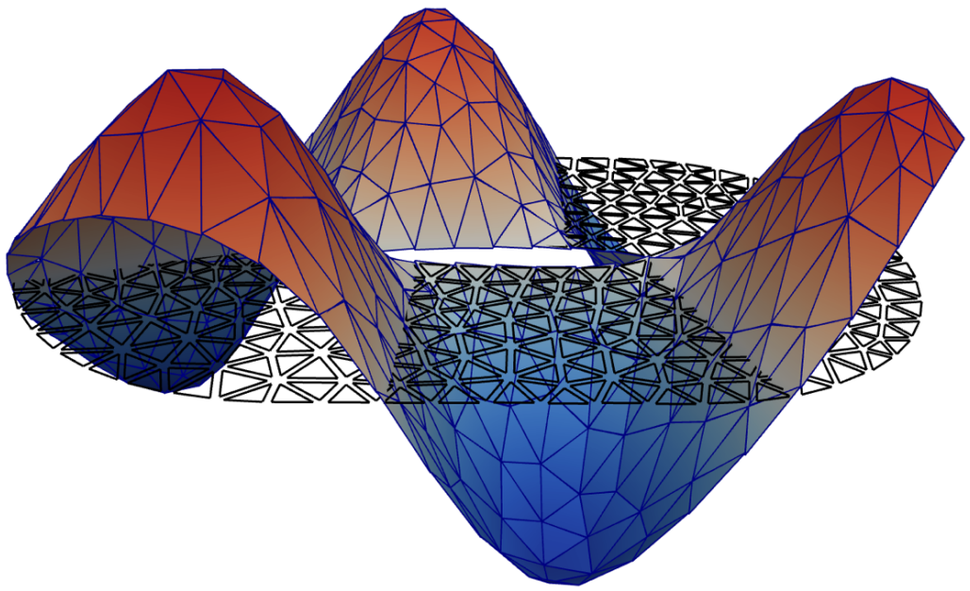
infuse wave modes: $\Rightarrow$ level 1: $\theta = 0, \frac{\pi}{2}$

level 2: $\theta = 0, \frac{\pi}{8}, \frac{3\pi}{8}, \frac{5\pi}{8}, \frac{7\pi}{8}$

adapt to grid through relaxation: $\Rightarrow$ $Ae^{i\omega\mathbf{x}} = 0$

optimize interpolation: $\Rightarrow$ $\|P\|_{A^*A}$

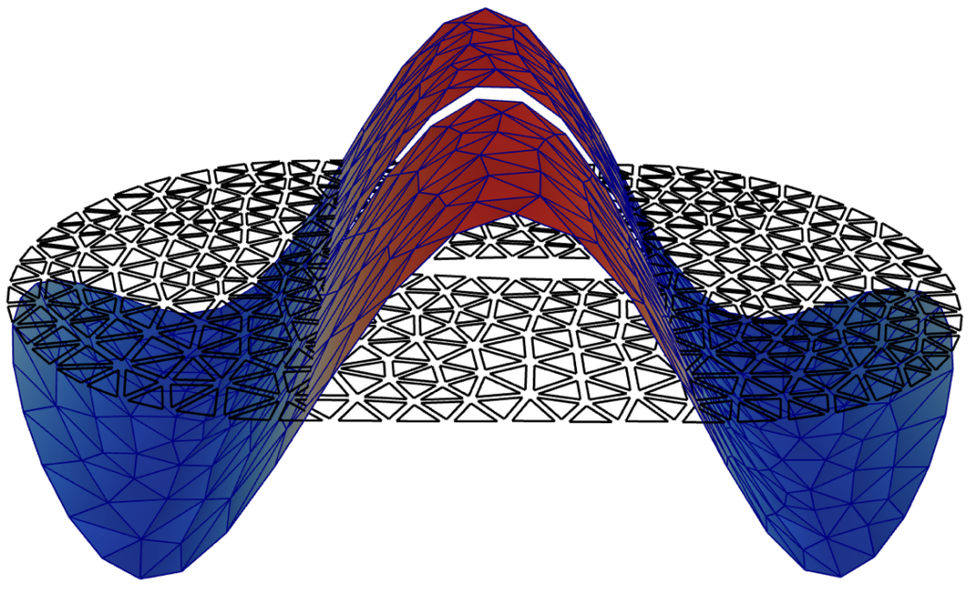
Ex. #1



left singular vector

*** Olson, Schroder, *A smoothed aggregation multigrid method for Helmholtz problems,* 2010.

injected mode



ppw	h	$h/2$	$h/4$	$h/8$	$h/12$
5.0	9	11	15	25	54
10.0	9	11	11	11	14
30.0	9	10	10	11	11
90.0	9	10	11	11	11
All	4	15	17	21	22

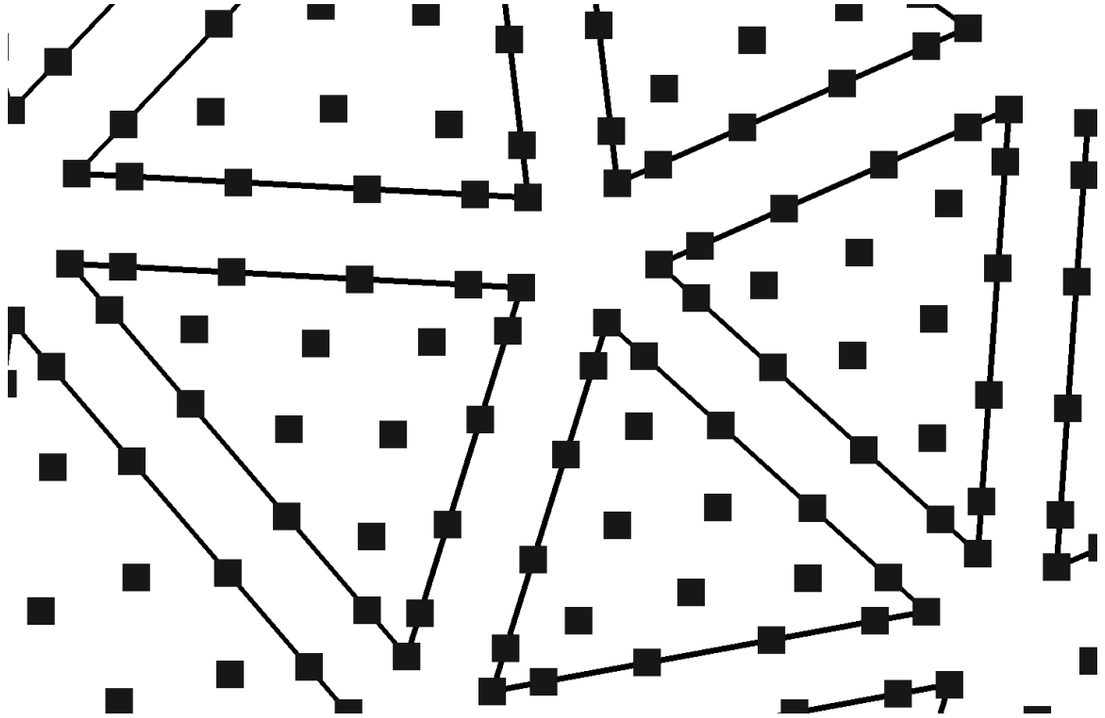
pgmres iterations, complexity

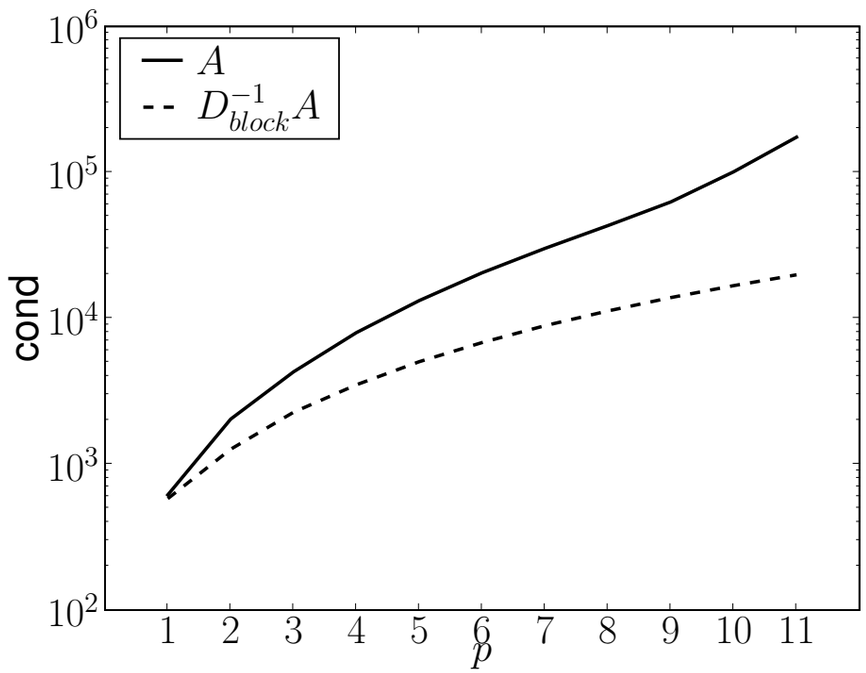
*** Olson, Schroder, *Smoothed aggregation multigrid solvers for high-order discontinuous galerkin methods,* 2011.

Ex. #2

High-order discontinuous Galerkin

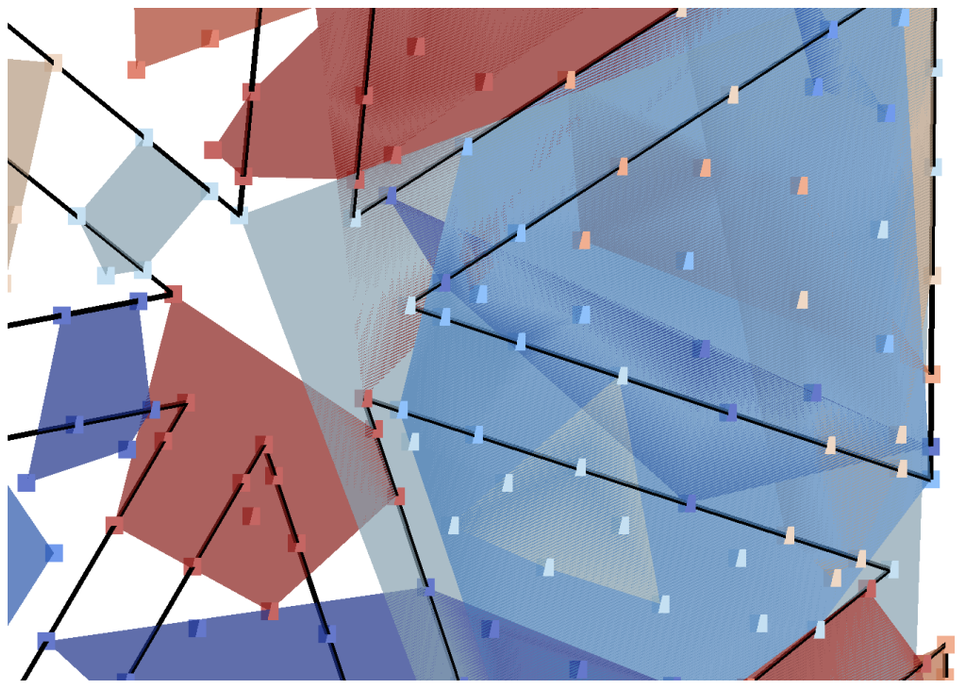
- different types of d.o.f.
- loss of locality
- increase in condition number
- most heuristics in AMG break down

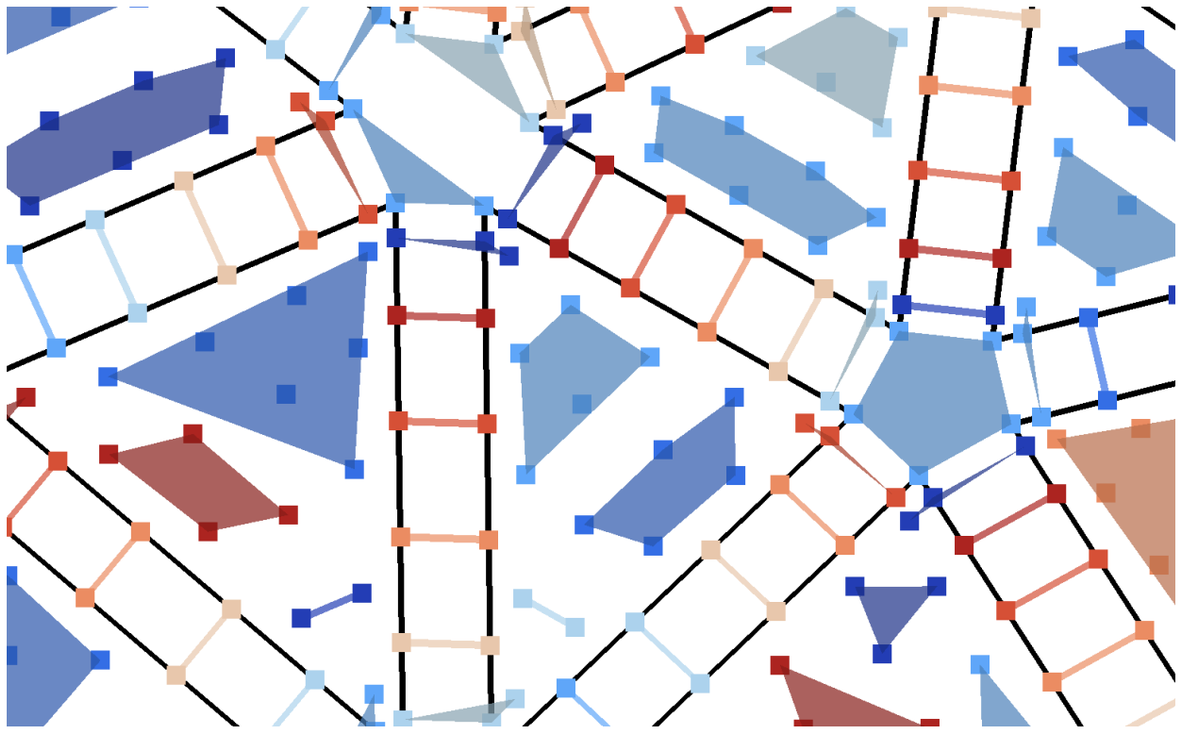




Ex. #2

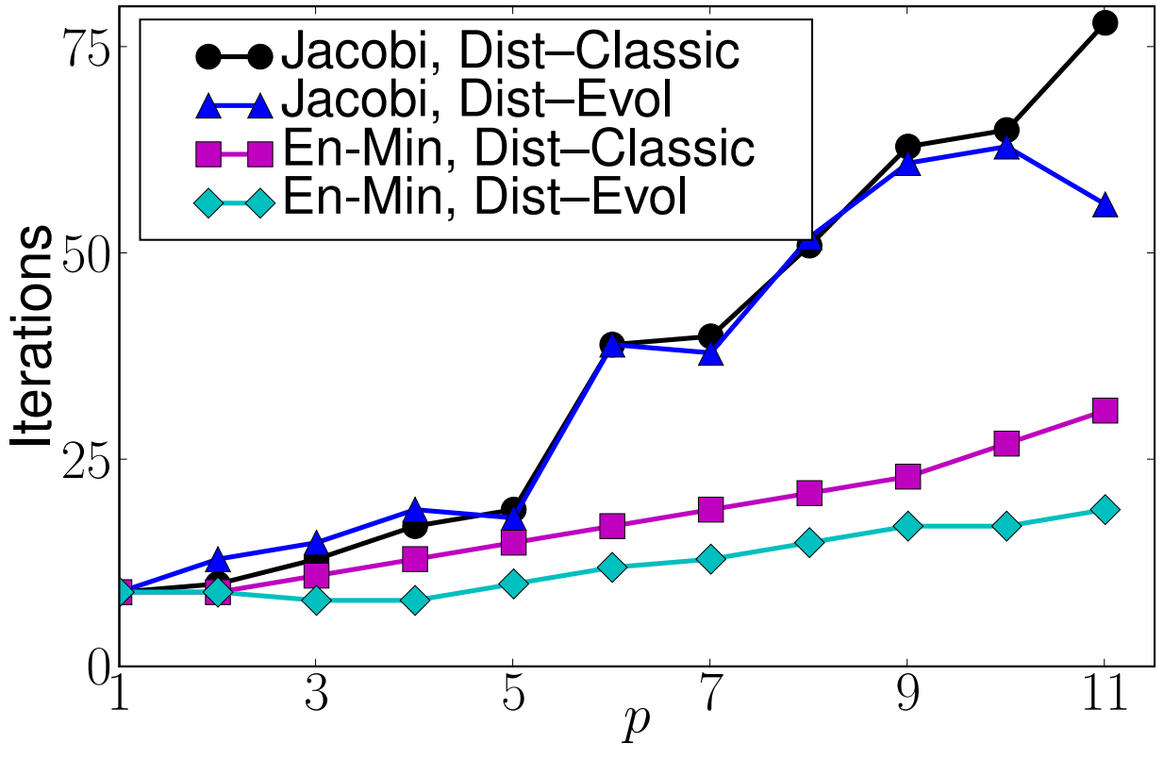
High-order discontinuous Galerkin: Poisson





key ingredients:

- conforming aggregations step
- adapt the near null space
- optimal interpolation



GPU

Acceleration Motivation 1: potential

1M d.o.f.
10 nonzero neighbors
1.5 cycle complexity
15M computations

2 for a V(1,1)-cycle
20 iterations
600M for solution

16 bytes for (data,row,col) (8,4,4)
10 Gbytes of data to process

125 Gbyte/sec SpMV on the GPU?
~ 0.08 sec

25 Gbyte/sec SpMV on host?
~ **0.4 sec**

GPU

Acceleration Motivation 2: software + hardware

real acceleration units: dedicated computation, double precision, high speeds

useable software: CUDA + Thrust + Cusp

AMG “asks” for acceleration:

✓adaptive

✓multiple candidates, cycles

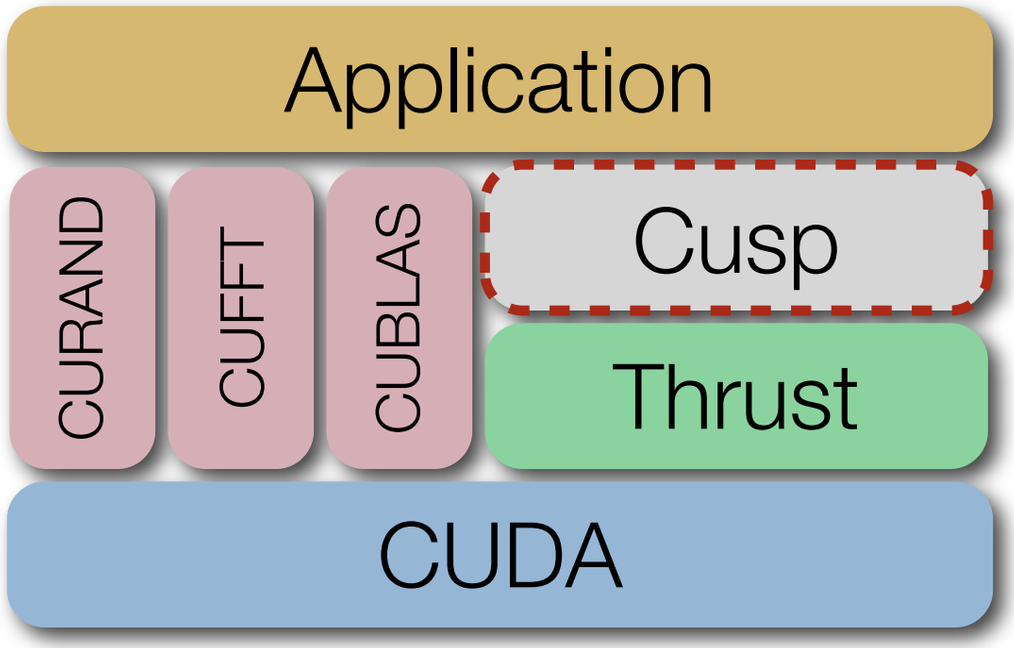
✓less/more aggressive coarsening

✓multiple RHS

GPU

- fast development
- low overhead
- open source

*** Bell, Dalton, Olson, *Exposing fine-grained parallelism in algebraic multigrid, 2011*.



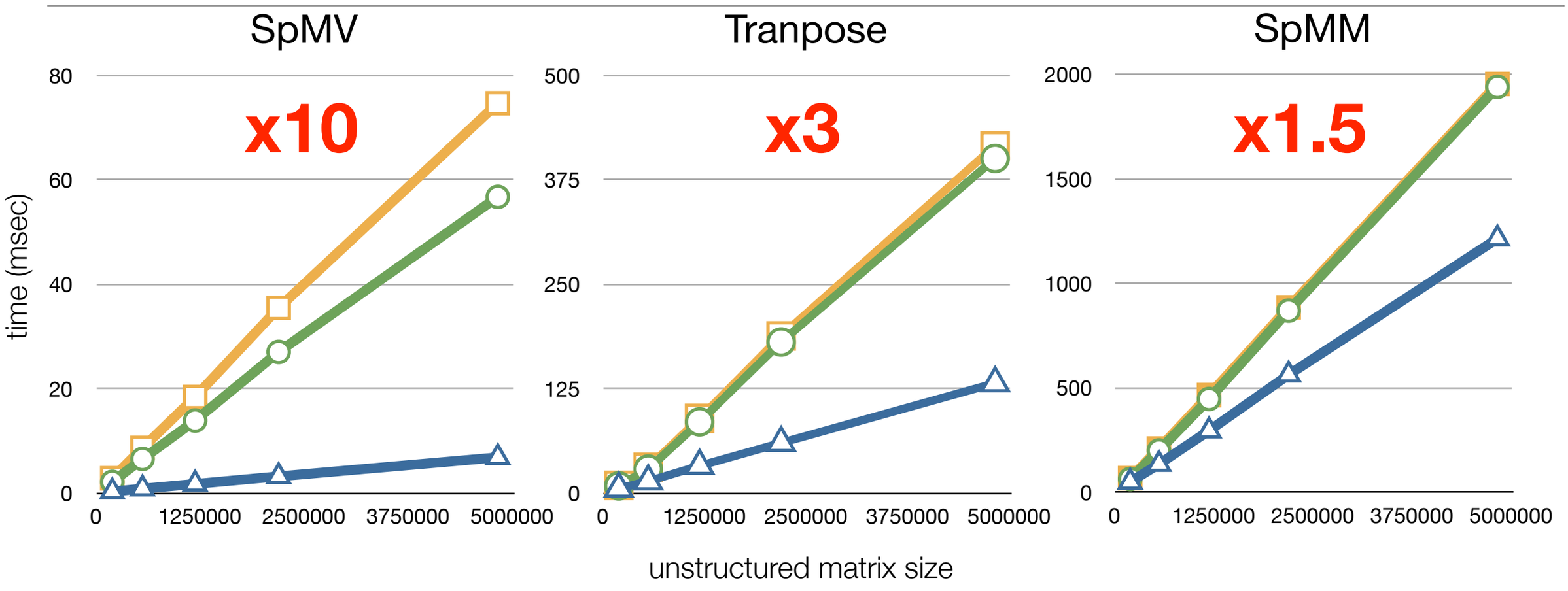
Method:

•expose fine-grained parallelism

•utilize fast kernels (gather, scatter, scans, sort, etc)

GPU

Setup kernels: SpMM, Transpose, SpMV

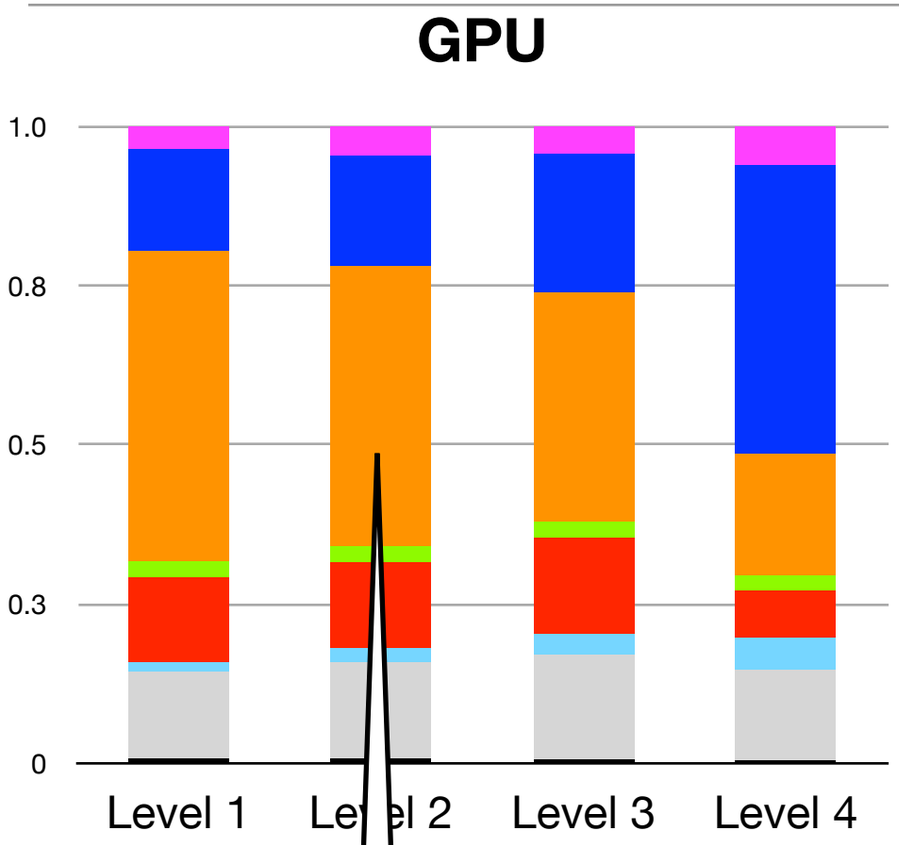


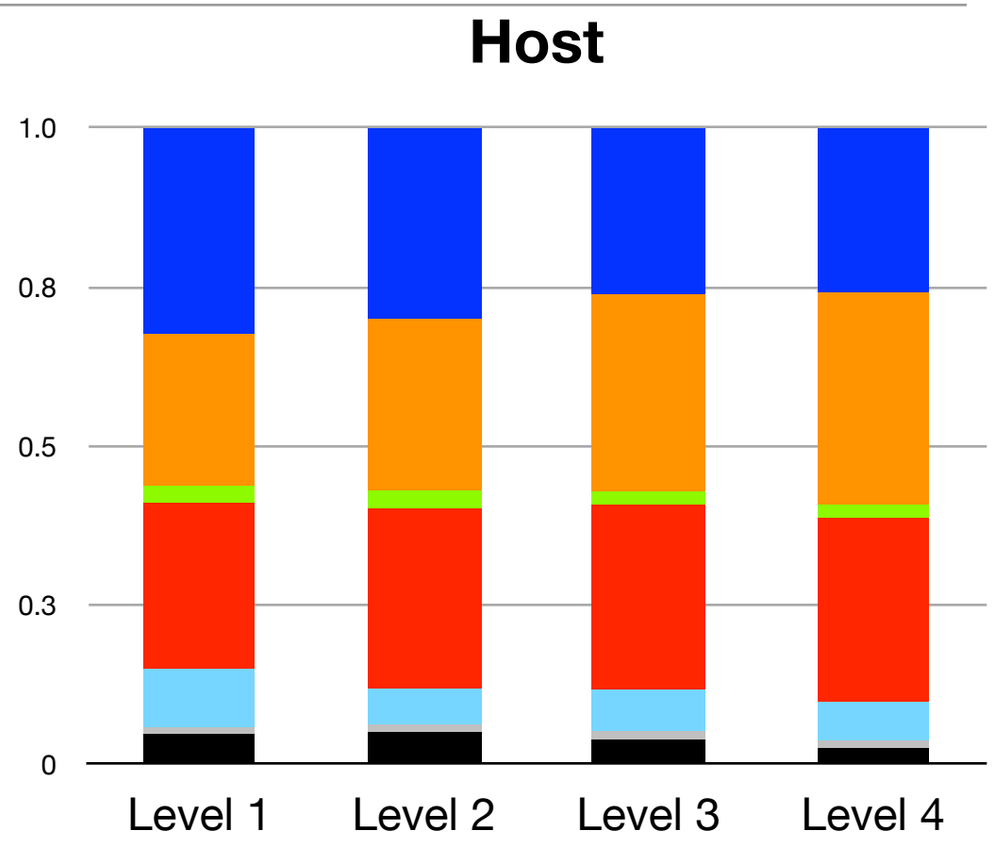
speed up

size	A*B	BT*(A*B)	BT*A*B	Tranpose	SpMV
5E+06	1.8	1.4	1.6	3.2	10.8
2E+06	1.7	1.4	1.6	3.1	10.9
1E+06	1.7	1.3	1.6	2.7	10.1
6E+05	1.7	1.3	1.5	2.1	9.5
2E+05	1.6	1.0	1.4	1.6	7.9

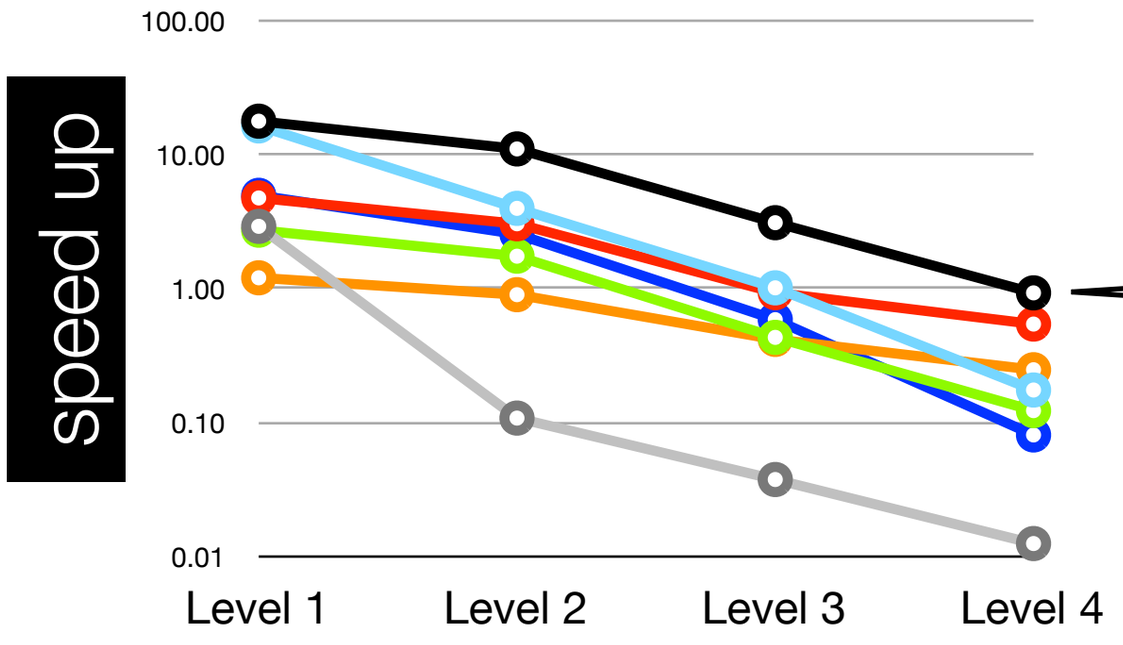
GPU

Setup components: total time, 1M dof, unstructured



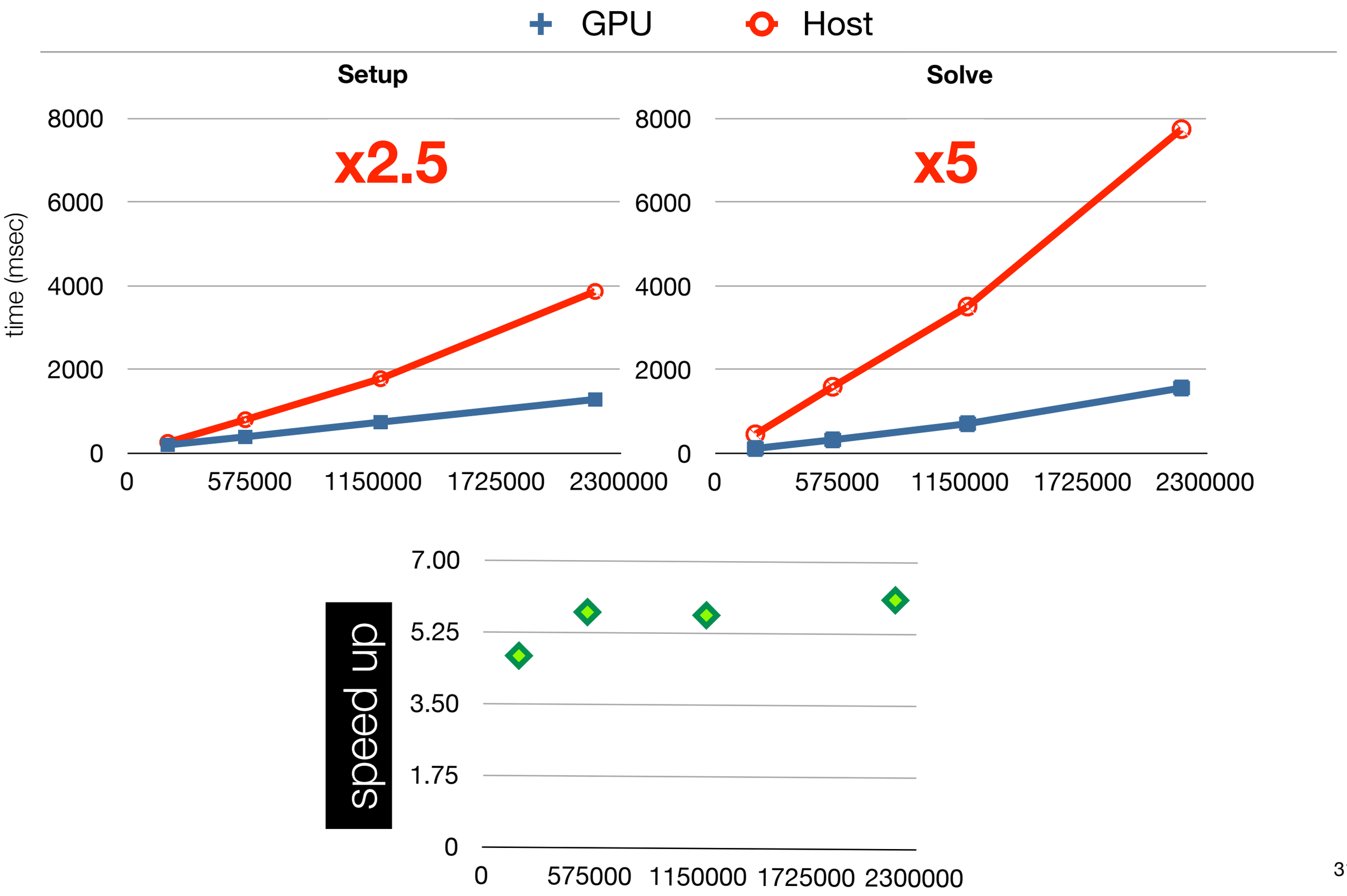


Triple products are expensive



GPU likes fine (sparse) levels

GPU


GPU
Host
Setup
Solve
x2.5
x5
time (msec)
8000
6000
4000
2000
0
0
575000
1150000
1725000
2300000

speed up
7.00
5.25
3.50
1.75
0
0
575000
1150000
1725000
2300000

solvers and applications (discretizations)

- **goal:** a solver that can utilize natural scale in the problem
  - wave problems
  - high-order elements
  - many graph problems

- **goal:** best balance of accuracy and cost
  - LS FEM often lead to more cooperative sparse systems
  - high-order may have higher degrees of fine-grained parallelism
  - conforming elements may lead to a more natural hierarchy/null-space

- **goal:** more useable solver for applications
  - not blackbox
  - more guidance on tuning and a measure of success

solvers and theory

- **goal:** predictive theory to assess quality/efficiency *a priori*
  - sharp upper bounds on convergence
  - useful lower bounds on convergence
  - performance measures that assess the complexity

- **goal:** fully adaptive methods
  - fine-tune the solver based on history

- **goal:** ubiquitous complexity
  - globally aware decisions in the solver setup
  - architecture sensitive theory

solvers and computation

- golden age of multigrid and computation?

- **goal:** scalability
  - parallel modeling (Gahvari, Gropp, et al.) tied to development
  - additive, repetitive, or redundant cycling

- **goal:** learning, automating, tuning
  - use a scaling phase for learning
  - parameter pool

- **goal:** heterogeneous environments
  - fine-grained approach
  - flexible use of resources
- low-power?

Contributions

- NSF-DMS
- Teragrid for allocations
- Nvidia for hardware
- software development: CUSP::MG
- software development: PyAMG
- LLNL, SNL for student support