(Algebraic) Multigrid Methods at the Exascale?

James Brannick

The Pennsylvania State University

$SIMAC_3$, Boston University, November 4, 2013

Joint work with:

Xiaozhe Hu, Jinchao Xu, Ludmil Zikatanov (PSU), and Yao Chen (Microsoft)

Outline:

PENNSTATE

Reliable: iterative methods (fault tolerance)

Optimal: leads *necessarily* to hierarchical methods (multigrid)

User friendly: algebraic multigrid methods (aggregation methods)

Tunable: geometric-algebraic methods (grid-based coarsening)

Conclusions

PENNSTATE

Reliable: iterative methods (fault tolerance)

Optimal: leads *necessarily* to hierarchical methods (multigrid)

User friendly: algebraic multigrid methods (aggregation methods)

Tunable: geometric-algebraic methods (grid-based coarsening)

Conclusions

Variational problem

PENNSTATE

Find $u \in V$ such that $a(u, v) = f(v),\ \forall v \in V$

Model problem: $-\Delta u = f$ in $\Omega \quad \Longrightarrow \quad Ax = b$

A fundamental problem in scientific computing:

How to solve $Ax = b$ efficiently?

Iterative Methods: $x^0, x^1, \cdots, x^{k-1} \longrightarrow x^k$

Basic ideas:

1. Form the residual: $r = b - Ax^{k-1}$
2. Solve the error eqn $Ae = r$ approximately $\hat{e} = Br$ with $B \approx A^{-1}$
3. Update $x^k = x^{k-1} + \hat{e}$

Linear iterative method: $x^k = x^{k-1} + B(b - Ax^{k-1})$. Choice of B:
$B = \mathrm{DIAG}(A)^{-1}$(Jacobi), $B = \mathrm{TRIL}(A)^{-1}$(Gauss-Seidel), $\cdots$

Convergence

PENNSTATE

Linear iterative method ($B \approx A^{-1}$)

$$x^k = x^{k-1} + B(b - Ax^{k-1})$$

with error iteration

$$e^k = (I - BA)e^{k-1}$$

Convergence: $\|I - BA\|_A^2 < 1$ assuming A SPD

PCG for the preconditioned system ($BAu = Bf$)

$$\frac{\|x - x^k\|_A}{\|x - x^0\|_A} \leq 2\left(\frac{\sqrt{\kappa(BA)} - 1}{\sqrt{\kappa(BA)} + 1}\right)^k (k \geq 1), \quad \left(\kappa(BA) = \frac{\lambda_{\max}(BA)}{\lambda_{\min}(BA)}\right)$$

Non-SPD problems: MINRes, GMRes

Theoretical framework: Method of Subspace Corrections

(Bramble-Pasciak-Wang-Xu 1991, Xu, SIAM Review 1992)

- Find $u \in V$ such that $a(u, v) = f(v),\ \forall v \in V$
- Space decomposition: $V = \sum_i V_i$
- Subspace correction: $e_i \approx A_i^{-1} Q_i (f - Au)$

$$u \leftarrow u + \sum e_i \qquad \text{(PSC, BPX)}$$

$$u \leftarrow u + e_i \qquad \text{(SSC)}$$

- Jacobi & ($R^n = \sum_i \{e^i\}$), DD ($V = \sum_i V_i$)
- Gauss-Seidel ($R^n = \sum_i \{e^i\}$), MG $V_1 \subset V_2 \subset \ldots V_J = V$

An identity

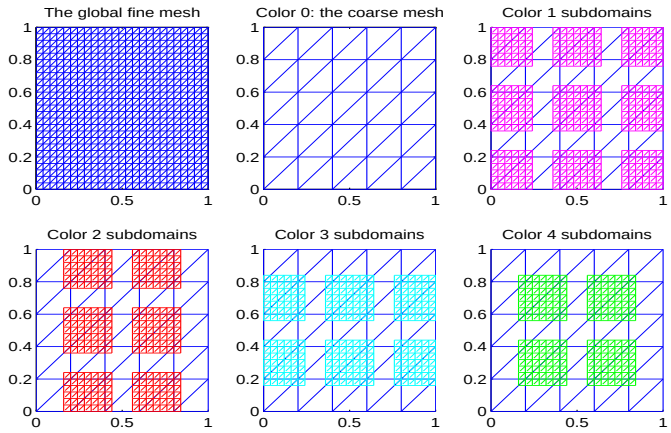
Theorem (Xu-Zikatanov (2002, J. AMS))

The SSC is convergent if each subspace solver is convergent:

$$\|I - BA\|_A^2 = 1 - \frac{1}{c_1}, \quad c_1 = \sup_{\|v\|_A=1} \inf_{\sum_i v_i = v} \sum_{i=1}^{J} (\overline{R}_i^{-1} w_i, w_i)$$

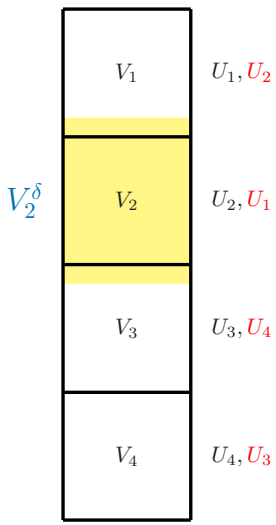
where $\overline{R}_i = R_i^T + R_i - R_i^T A_i R_i$ *and* $w_i = v_i + R_i^T A_i Q_i \sum_{j>i} v_j$.

Example: overlapping Schwarz



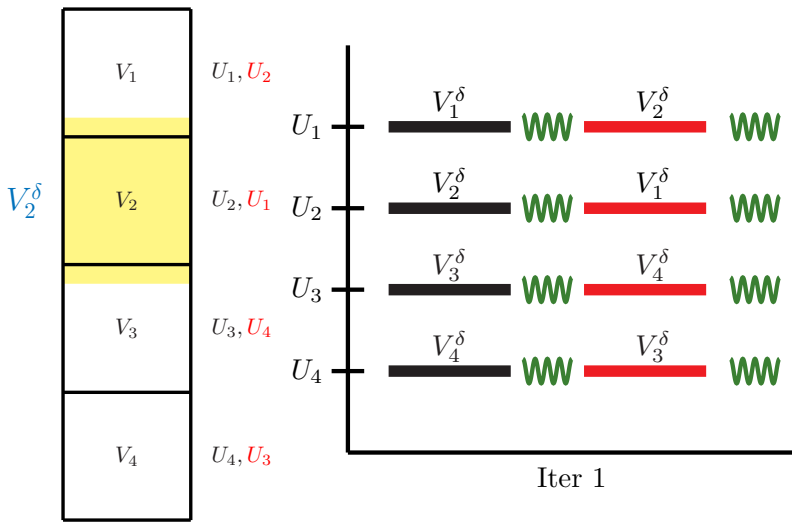
Redundant Subspace Correction: $V = \sum_i V_i + \sum_j V_j$

- Each subspace stored in more than one process ($\underline{V_i \to U_k, V_i \to U_\ell}$)
- Two different subspaces in one process ($\underline{\{V_i, V_j\} \to U_k}$)



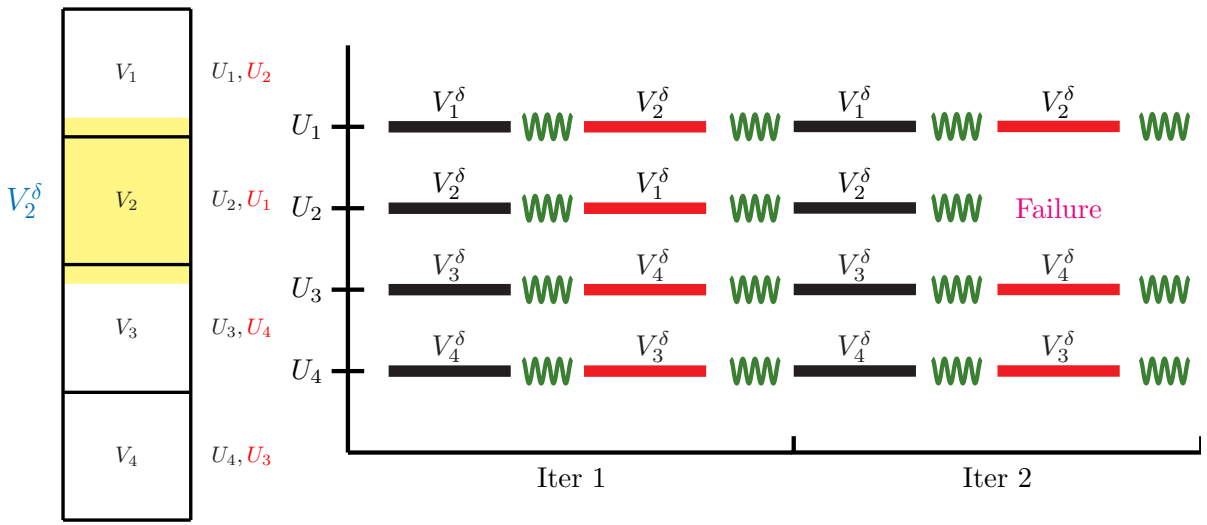
Redundant Subspace Correction: $V = \sum_i V_i + \sum_j V_j$

- Each subspace stored in more than one process ($V_i \to U_k, V_i \to U_\ell$)
- Two different subspaces in one process ($\{V_i, V_j\} \to U_k$)



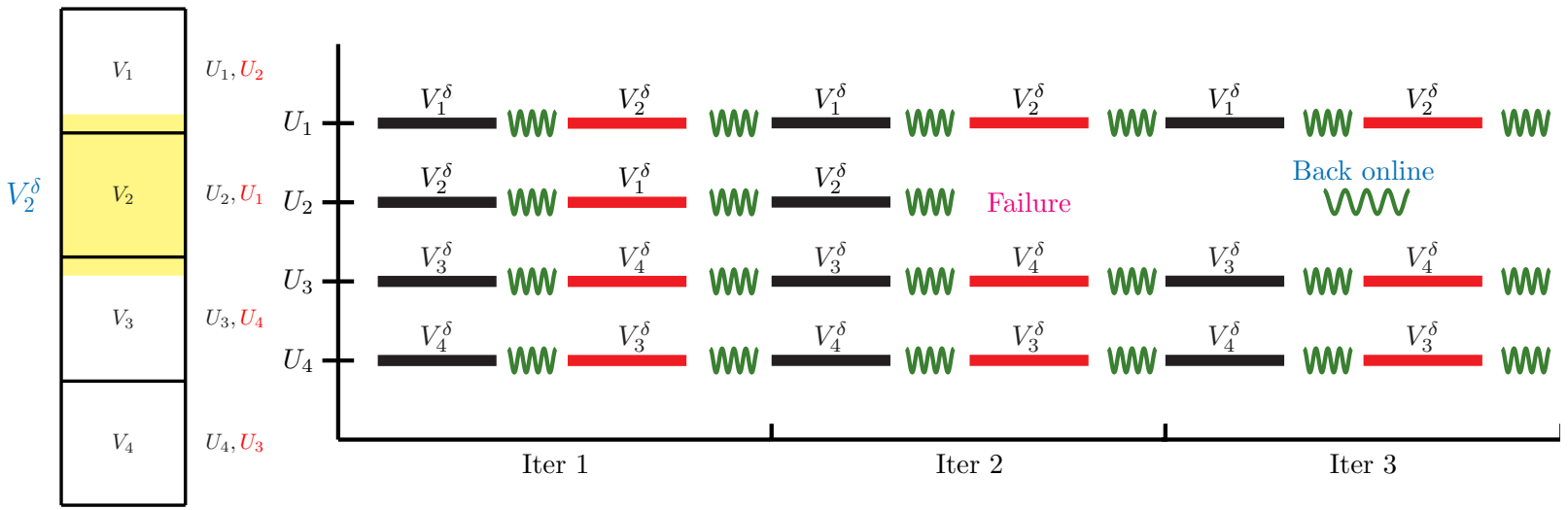
Redundant Subspace Correction: $V = \sum_i V_i + \sum_j V_j$

- Each subspace stored in more than one process ($V_i \to U_k, V_i \to U_\ell$)
- Two different subspaces in one process ($\{V_i, V_j\} \to U_k$)



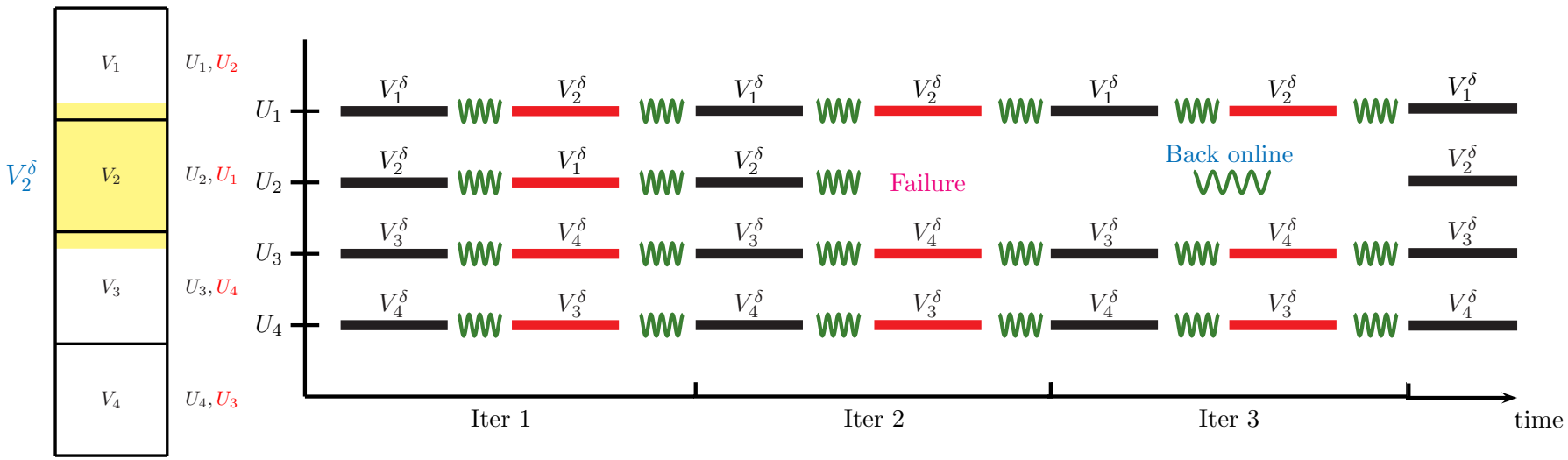
Redundant Subspace Correction: $V = \sum_i V_i + \sum_j V_j$

- Each subspace stored in more than one process ($V_i \to U_k, V_i \to U_\ell$)
- Two different subspaces in one process ($\{V_i, V_j\} \to U_k$)



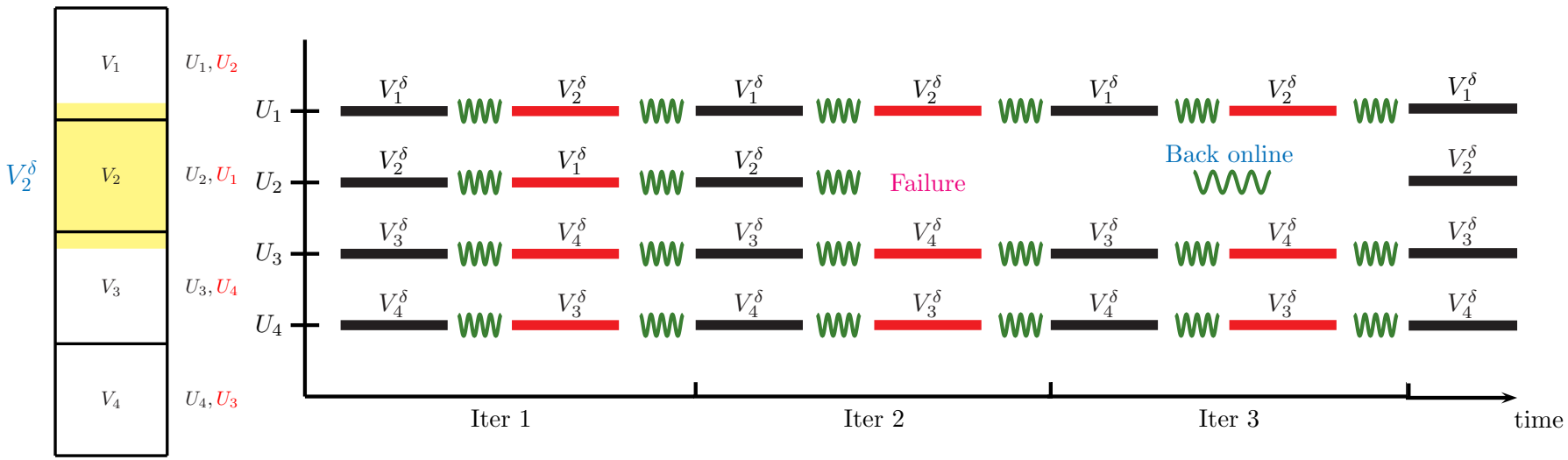
Redundant Subspace Correction: $V = \sum_i V_i + \sum_j V_j$

- Each subspace stored in more than one process ($V_i \to U_k, V_i \to U_\ell$)
- Two different subspaces in one process ($\{V_i, V_j\} \to U_k$)



Redundant Subspace Correction: $V = \sum_i V_i + \sum_j V_j$

- Each subspace stored in more than one process ($\underline{V_i \to U_k, V_i \to U_\ell}$)
- Two different subspaces in one process ($\underline{\{V_i, V_j\} \to U_k}$)



Theorem (Convergence Rate of RSC (Xu & Zhang 2013)

$$\|I - B_{SC}A\|_A^2 \leq \|I - B_{RSC}A\|_A \leq \|I - B_{SC}A\|_A$$

- The solver will not fail as long as one process in each pair works and the convergence is optimal if no error, and degrades slightly in case of error

Numerical Experiments: preconditioned flexible GMRES: (16 processors, 1 fails)

Convergence of multip. Schwarz (with $\delta = 2$ and coloring):

Poisson problem with 2.1 M dofs, Maxwell problem with 1.6 M dofs and Elasticity problem with .8 M dofs. LSSC-III cluster with 282 computing nodes, each with two Intel Quad Core Xeon X5550 2.66GHz processors and 24GB shared memory

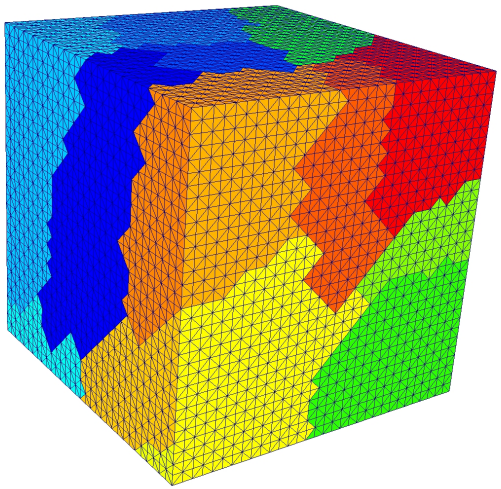
Error	Poisson		Maxwell		Elasticity	
	#Iter	Time	#Iter	Time	#Iter	Time
No	44	70.73	63	68.76	73	223.14
Yes	48	81.01	67	74.28	74	228.21

Convergence of additive Schwarz (with $\delta = 2$ overlap):

Example	dofs	Error-free B_{PSC}		Error-free B_{PRSC}		B_{PRSC} with Error	
		#Iter	Time	#Iter	Time	#Iter	Time
Poisson	1.3 M	23	7.92	12	8.09	13	8.13

Numerical Experiments: Scalability

(Additive Schwarz with $\delta = 2$ overlap, Poisson, 1 processor failed)



Partition with METIS

dofs	#Cores	Without Error		With Error	
		#Iter	Time	#Iter	Time
1,335,489	16	12	8.09	13	8.13
2,146,689	32	13	8.64	15	8.99
4,243,841	64	14	8.91	16	9.37
10,584,449	128	19	12.87	20	13.95
16,974,593	256	23	18.01	25	19.13
33,751,809	512	25	20.90	27	26.11

Table : PRSC for the Poisson equation

Fault-Tolerant Methods by Randomization

SPD model problem: $Au = f$

Subspace decomposition: $V = \sum_{i=1}^{J} V_i + V_{\text{fault}}$

Randomized SSC method (RSSC)

1. Randomly choose $i \in \{1, 2, \cdots, J\}$ with probability $\frac{1}{J}$
2. If processor i fails, $u^{k+1} = u^k$. Otherwise, $u^{k+1} = u^k + R_i Q_i (f - Au^k)$

Theorem (Convergence of RSSC, Hu et. al. 2013)

Assume that each processor fails with probability $\theta \in [0, 1)$ *(independent of k)*

$$E(||u - u^{k+1}||_A^2) = (1 - \frac{(1-\theta)\delta}{J}) E(||u - u^k||_A^2),$$

where $\delta = \frac{(\sum_{i=1}^{J} \bar{T}_i e^k, e^k)_A}{(e^k, e^k)_A}$.

- Always converges in expectation
- Exponential decay rate after one large sweep: $(1 - \frac{(1-\theta)\delta}{J})^J \approx e^{-(1-\theta)\delta}$

[REF Strohmer & Vershynin 2009, Needell 2010, Eldar & Needell 2011, Griebel & Oswald 2012]

Reliable: iterative methods (fault tolerance)

Optimal: leads *necessarily* to hierarchical methods (multigrid)

User friendly: algebraic multigrid methods (aggregation methods)

Tunable: geometric-algebraic methods (grid-based coarsening)

Conclusions

A convergence result for nearly singular problems:

Theorem (Lee, Wu, Xu and Z 2006)

Consider the system $(A_0 + \epsilon I)x = b$, *where* $A = A_0 + \epsilon I$ *and* A_0 *SPSD. Then, SSC method converges uniformly with respect to* ϵ *as long as the decomposition* $V = \sum_{j=1}^{J} V_j$ *satisfies*

$$N(A_0) = \sum_{j=1}^{J} [V_j \cap N(A_0)],$$

where $N(A_0)$ *denotes the nullspace of* A_0

- One possible decomp.: $V = \sum_{i=1}^{J} V_i + \sum_{j=1}^{m} \text{span}\{\chi_j\}$ where $\{\chi_i\}$ denote the eigenvectors corresponding to nearly zero eigenvalues
- Is splitting condition $N(A_0) = \sum_{j=1}^{J} [V_j \cap N(A_0)]$ necessary for uniform convergence?

Domain decomposition $\Rightarrow$ space decomposition

$$\Omega = \sum_i \Omega_i \Rightarrow V = \sum_i V_i$$

where V_i are "local" subspaces:

$$V_i = \{v \in V : v(x) = 0,\ \forall x \in \Omega \setminus \Omega_i\} \subset V \equiv H^1(\Omega)$$

Theorem (A lower bound, Br., Hu and Zikatanov 2013)

Assume E is error prop. matrix of mult. Schwarz method with $H = kh$ and $\delta \leq \tilde{k}H$ with no coarse space. Then, for the Neumann problem *$A = A_0 + h^2 I$ and $\exists$ constants c and $\tilde{c}$:*

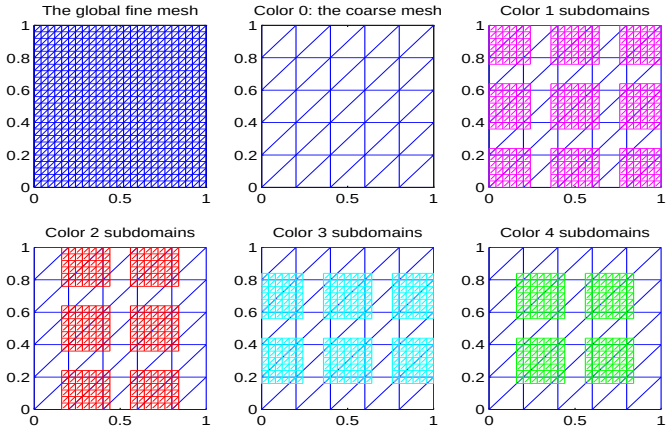
$$1 - \tilde{c}H^2 \leq ||E||_A^2 \leq 1 - cH^2$$

- The convergence of the method approaches 1 as $h \to 0$, implying hierarchical method necessary for optimality
- Result applies to large number of elliptic b.v.p., including linear elasticity, Maxwell's equations, ...

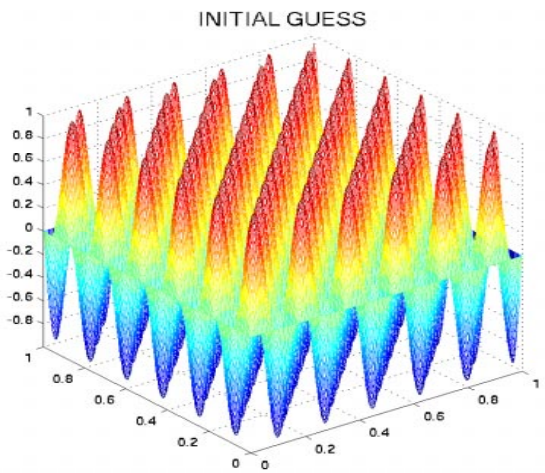
Multigrid: local high frequencies – smoothing

PENNSTATE

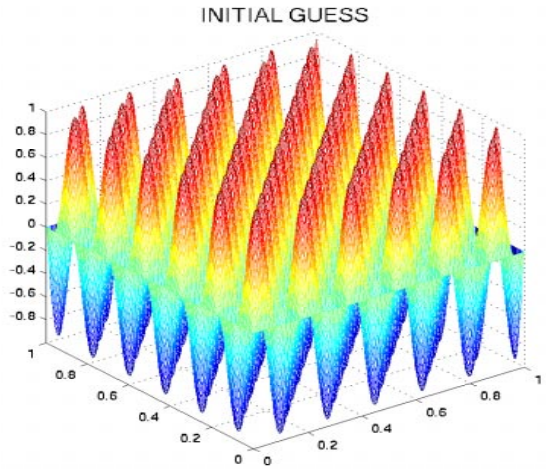
GMG with block smoother

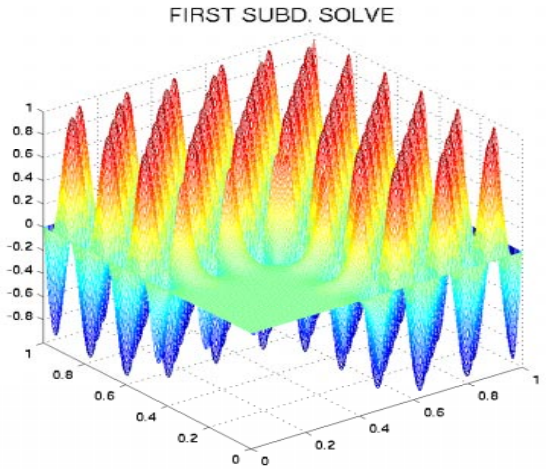


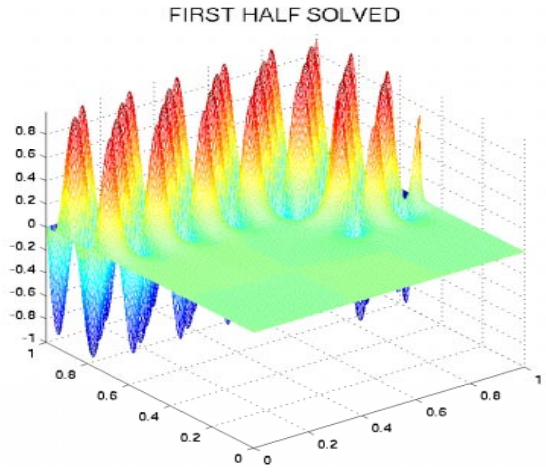
Initial error:

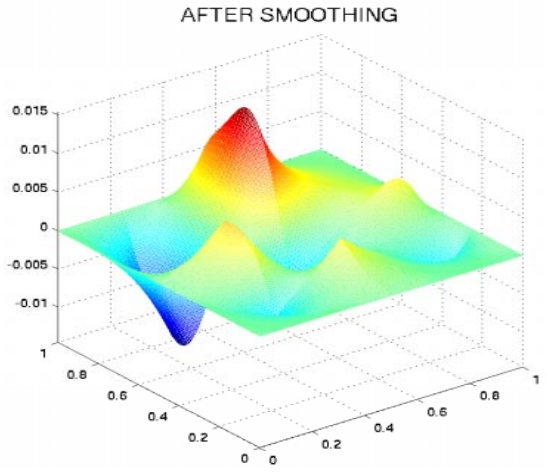


Multigrid: local high frequencies – smoothing

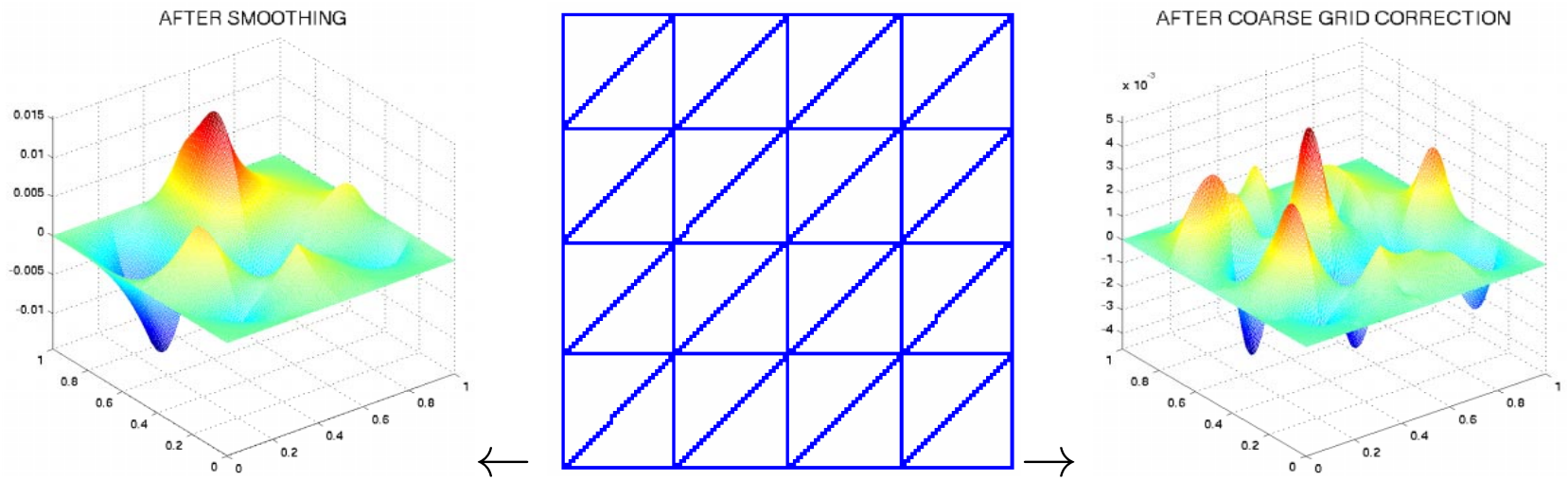






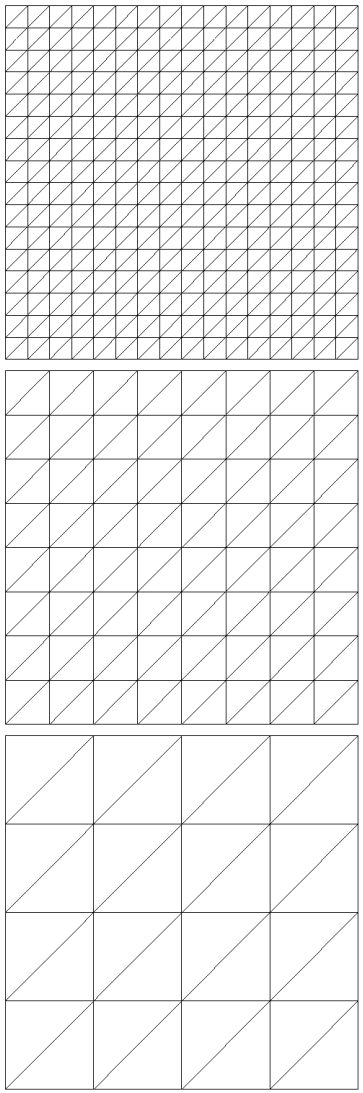


Global low frequencies - coarse grid correction



Geometric Multigrid (GMG) Methods

Recursively Apply Smoothing and Coarse Grid Correction



$$
\begin{array}{ccccccc}
A_h & \Rightarrow & (GS)_h & & & & \\
 & & + & \searrow & & & \\
 & & A_{2h} & \Rightarrow & (GS)_{2h} & & \\
 & & & & + & \searrow & \\
 & & & & A_{4h} & \Rightarrow & (GS)_{4h} \\
 & & & & & & + \quad \searrow \\
 & & & & & & A_{8h} \quad \Rightarrow \\
\mathcal{O}(N_h) & + & \mathcal{O}(N_{2h}) & + & \mathcal{O}(N_{4h}) & \ldots & = \mathcal{O}(N_h)
\end{array}
$$

Pros and Cons of GMG

PENNSTATE

Pros:

- optimal computational complexity: $O(N)$ or $O(N|\log N|^{\delta})$ ($N = 10^8$, Titan takes 50 seconds!)
- suitable for parallel or high performance computers
- applicable to a large class of elliptic PDE systems

Cons:

- mainly restricted to isotropic problems
- problem-dependent in more general settings
- significant effort to integrate with existing codes

User-friendly multigrid methods

Algebraic Multigrid (AMG) methods

Reliable: iterative methods (fault tolerance)

Optimal: leads *necessarily* to hierarchical methods (multigrid)

User friendly: algebraic multigrid methods (aggregation methods)

Tunable: geometric-algebraic methods (grid-based coarsening)

Conclusions

Algebraic Multigrid (AMG) Methods

PENNSTATE

Setup phase: construct the hierarchical structure

- Coarsening: select coarse and fine nodes / form aggregates
- Construct prolongation P and restriction R: standard interpolation / smoothed aggregation / energy minimization / $\cdots$ many others ($R = P^T$)
- Construct coarse grid matrix: $A_c = RAP = P^T AP$

Solve phase: similar to GMG

- Presmoothing: Jacobi / Gauss-Seidel / $\cdots$
- Coarse grid correction: recursively call MG on the coarse level
- Postsmoothing: Jacobi / Gauss-Seidel / $\cdots$

Pros and Cons of AMG

PENNSTATE

Pros:

- Make use of only algebraic information of the system, i.e., are user-friendly – black-box
- Efficient for certain types of problems, mainly discretizations of scalar elliptic PDE, but generalize to anisotropic diffusion problems

Cons:

- Setup phase, which can be costly
- Applicability and efficiency difficult to control in practice and verify theoretically
- Parallelization is difficult

General strategy

(1) *Use auxiliary space preconditioners* to reduce solving a coupled PDE system to solving series of auxiliary scalar anisotropic diffusion problems
   - Auxiliary space problems defined via discrete deRham complexes and finite element spaces used in forming stable discretization of the PDE system
   - Important PDE subsystems: (i.) Linear elasticity; (ii.) Maxwell's equaitons; (iii.) (Navier) Stokes

(2) Solve scalar problems using optimal multigrid methods, integrating algebraic techniques as needed

(3) Develop efficient parallel toolbox for geometric-algebraic solvers using aggregation

Difficulties in parallelizing AMG

- Setup phase:
  - Coarsening: select coarse nodes / form aggregates
  - Prolongation and restriction
  - Coarse grid matrix: triple matrix products
- Solve phase:
  - Parallel smoother
  - Cycling: coarse grids lead to idle processors
- CPU:
  - BoomerAMG (HYPRE): classical AMG method (LLNL)
  - ML: smoothed aggregation AMG method (Sandia)
  - AGMG: unsmoothed aggregation AMG method (Y. Notay)
- GPUs:
  - Solve phase (Góddeke et al, 2008; Haase et al, 2010)
  - Parallel Toolbox: classical AMG (Liebmann, 2008)
  - CUSP: smoothed aggregation AMG (Bell, Dalton and Olson, 2011)

Unsmoothed Aggregation AMG (UA-AMG)

PENNSTATE

Setup Phase

- Aggregation (Coarsening): find a non-overlapping partition $\{G_j\}$ of aggregates
- Prolongation and restriction: boolean matrices

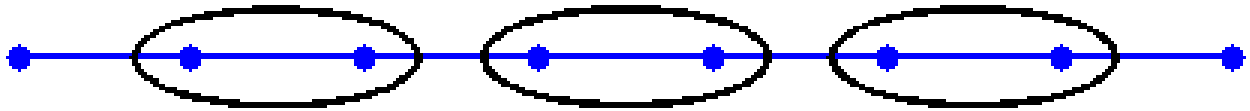
$$(P)_{ij} = \begin{cases} 1 & \text{if } i \in G_j \\ 0 & \text{otherwise} \end{cases}$$

and $R = P^T$

- Coarse grid matrix: $A_c = P^T A P$, can be computed by simple summations

$$(A_c)_{kl} = \sum_{i \in G_k} \sum_{j \in G_l} a_{ij}$$

An Example



Aggregates: $G_1 = \{1, 2\}$, $G_2 = \{3, 4\}$, and $G_3 = \{5, 6\}$

$$A = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & -1 & 2 \end{bmatrix}, \quad P = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\Rightarrow A_c = P^T A P = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

UA-AMG: solve phase

A drawback

Only two-grid method converges uniformly

V-cycle: B_k

Let $B_1 = A_1^{-1}$ and assume that $B_{k-1} : V_{k-1} \to V_{k-1}$ has been defined, then for $f \in V_k$, $B_k : V_k \to V_k$ is defined as follow:

Pre-smoothing: $u_1 = R_k f$

Coarse grid correction: $u_2 = u_1 + P_k B_{k-1} P_k^T (f - A_k u_1)$

Post-smoothing: $B_k f := u_2 + R_k^T (f - A_k u_2)$

$$B_k = \bar{R}_k + (I - R_k^T A_k) P_k B_{k-1} P_k^T (I - A_k R_k)$$

K-cycle

$$B_k^{Vcycle} = \bar{R}_k + (I - R_k^T A_k) P_k B_{k-1} P_k^T (I - A_k R_k)$$

$$\Downarrow \quad \bar{B}_{k-1} \approx A_{k-1}^{-1}$$

$$B_k^{TG} = \bar{R}_k + (I - R_k^T A_k) P_k A_{k-1}^{-1} P_k^t (I - A_k R_k)$$

Choose $\bar{B}_{k-1}$ to be a better approximation of A_{k-1}^{-1} than B_{k-1}

Define $\bar{B}_{k-1}^{(n)}[\cdot]$ by n steps of the Krylov Subspace method using B_{k-1} as the preconditioner

K-cycle: $B_k^{Kcycle}[\cdot]$

Pre-smoothing $u_1 = R_k f$

Coarse grid correction $u_2 = u_1 + P_k \bar{B}_{k-1}^{(n)} [P_k^T (f - A_k u_1)]$

Post-smoothing $B_k^{Kcycle}[f] := u_2 + R_k^T (f - A_k u_2)$

$$B_k^{Kcycle} = \bar{R}_k + (I - R_k^T A_k) P_k \bar{B}_{k-1}^{(n)} [P_k^t (I - A_k R_k)]$$

Optimal Two-grid $\Rightarrow$ Optimal Multigrid:

PENNSTATE

AMLI- and K-cycle: Axelsson and Vassilevski, 1989

Theorem

Assume that the two-grid convergence rate is δ_0. If n is chosen such that $(1-\delta^n)\delta_0 + \delta^n \leq \delta$ has a solution $\delta \in [0,1)$. Then, the K-cycle MG method converges uniformly with convergence rate δ.

A sufficient condition: $n > \frac{1}{1-\delta_0} > 1$.

Corollary

Under the assumption that $\delta_0 < 0.5$ we have $(1-\delta^n)\delta_0 + \delta^n \leq \delta$ has a solution $\delta \in [0,1)$ when $n = 2$, and the K-cycle MG method converges uniformly with convergence rate δ.

[REF Brannick, Chen, Kraus & Zikatanov, 2013]

[REF Hu, Vassilevski & Xu, 2013]

An Example

$$-\Delta u = f, \quad \text{in } \Omega = [0,1] \times [0,1],$$
$$u = 0, \quad \text{on } \partial\Omega,$$

Size	V-cycle	K-cycle
3,969	100	40
16,129	244	41
65,025	519	41
261,121	713	41
1,046,529	1753	40

Parallel UA-AMG Method

PENNSTATE

- Setup phase:
  - Coarsening: find aggregates, parallel random aggregation
  - P and R: form P and $R = P^T$, unsmoothed P and $R = P^T$
  - Coarse grid matrix: compute $A_c = P^T AP$
- Solve phase:
  - Smoother: Jacobi / Gauss-Seidel / polynomial / ...
  - Cycle: K-Cycle

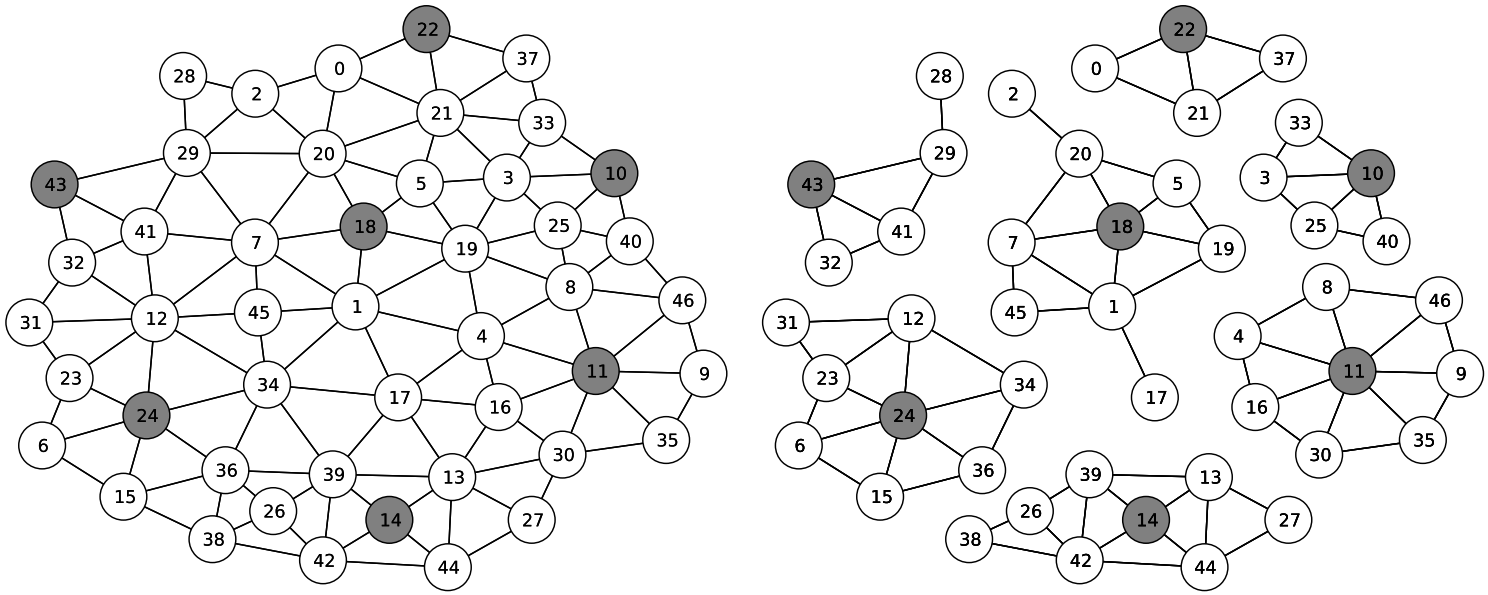
Two approaches

- Purely algebraic approach: do not use any grid information
- Auxiliary grid approach: use the unstructured grid

Coarsening: find Aggregates

Parallel Random Aggregation Algorithm

Based on finding maximal independent set of A^2 in parallel



[REF Luby 1986; Bell, Dalton and Olson, 2011]

Parallel Random Aggregation Algorithm

(1) Generate a quasi-random number and store it in v_i, as

$v_i \longleftarrow$ quasi_random(i); mark vertex i as "unprocessed"; wait until all threads complete these operations.

(2) (2a) Goto (2d) if i is marked "processed", otherwise continue to (2b).

(2b) Determine if the vertex i is a coarse vertex, i.e., check if the following holds true

$$v_i > v_j, \qquad \forall j : (A^2)_{ij} \neq 0 \text{ and } j \text{ is unprocessed}$$

If so, continue to (2c); if not, goto (2d).

(2c) Form an aggregate centered at i. Let S_i be a set of vertices defined as $S_i = \{j \mid v_i \geq v_j, \forall j : (A^2)_{ij} \neq 0 \text{ and } j \text{ is unprocessed}\}$. Define a column vector w such that

$$w_k = \begin{cases} 1, & k \in S_i; \\ 0, & k \notin S_i. \end{cases}$$

Mark vertices $j \in S_i$ "processed" and request an atomic operation to update the prolongator P as $P \longleftarrow [P, w]$.

(2d) Synchronize all threads.

(2e) Stop if i is marked "processed", otherwise goto step (2a).

Prolongation, Restriction, and Smoother

Prolongation: $v = Pv^c$

$$(v)_i = (Pv^c)_i = (v^c)_j, \quad i \in G_j$$

Restriction: $v^c = P^T v$

$$(v^c)_i = (P^T v)_i = \sum_{j \in G_i} (v)_j.$$

Key point: in parallel UA-AMG prolongation and restriction are simple and easy to compute

Smoother: ℓ_1 Jacobi smoother

$$x_i \leftarrow x_i + M_{ii}^{-1} r_i$$

where $r = b - Ax$, and $M_{ii} = a_{ii} + d_{ii}$ with $d_{ii} = \sum_{j \neq i} |a_{ij}|$.

Test: Poisson problem on unit square

	1 million			4 million		
	Setup	#iter./Solve	Total	Setup	#iter./Solve	Total
CPU	0.80	7/0.69	1.49	3.28	7/2.75	6.03
CUSP	0.63	36/0.35	0.98	2.38	41/1.60	3.98
FASP	0.13	19/0.47	0.60	0.62	19/2.01	2.63

- Processor:
  - CPU: Intel i7-2600 3.4GHz
  - GPU: NVIDIA Tesla C2070
- Algorithm:
  - CPU: Classical AMG + V-cycle;
  - GPU(CUSP): Smoothed Aggregation AMG + V-cycle;

Speedup: 2.5× CPU, 1.5× GPU(CUSP)

[REF Br., Chen, Hu & Zikatanov, NLAAA, 2013]

Reliable: iterative methods (fault tolerance)

Optimal: leads *necessarily* to hierarchical methods (multigrid)

User friendly: algebraic multigrid methods (aggregation methods)

Tunable: geometric-algebraic methods (grid-based coarsening)

Conclusions

Auxiliary Grid Approach

PENNSTATE

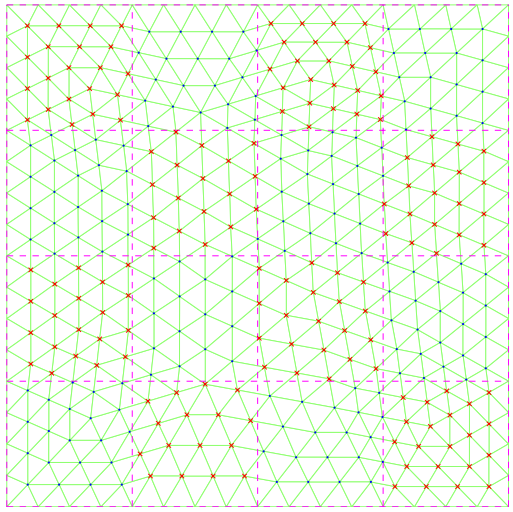
Basic idea: use the unstructured grid

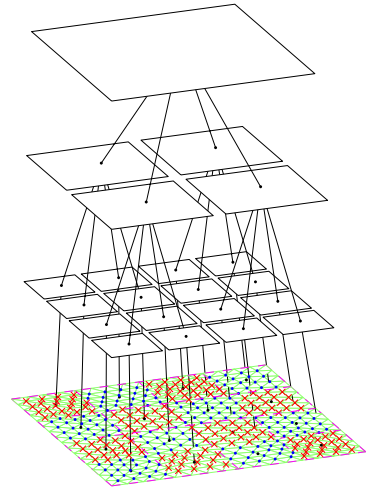
- Construct an auxiliary structured grid
- Use quadtree to manage the structure grid
- Apply UA-AMG + K-cycle
- Parallelize each step with the help from the grid and quadtree

Construct auxiliary structured grid and use MG on the structured grid to form preconditioner for unstructured grid problem (Xu 1996, Kolev & Vassilevski, 2006)

Construct auxiliary structured grid and use it to build hierarchical structure and use AMG method to form preconditioner for unstructured grid problem (Brezina, Vanek, & Vassilevski, 2011, Wang, Hu, Cohen, & Xu, 2013)

Auxiliary Structured Grid and Quadtree





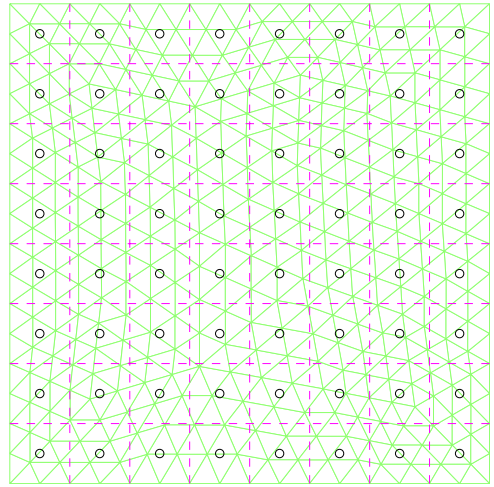
- Start with a simple square (rectangular) $[x_{min}, x_{max}] \times [y_{min}, y_{max}]$
- Generate a quadtree by recursively dividing a node into 4 leaf nodes
- Set the depth of the quadtree: e.g., $\mathtt{L} = \mathtt{int}\left[\frac{\log(N)}{\log 4}\right]$
- Construct the auxiliary structured grid using the deepest leaf nodes

[REF Wang, Hu, Cohen, & Xu, SISC, to appear]

Parallel Aggregation: Finest Level

Finest Level

- All the dofs in the same deepest leaf node form an aggregate
- One thread takes care one dog and simultaneously determines which aggregate it belongs to



Coarse levels

- 4 leaf nodes from the same root form an aggregate
- One thread takes care of one leaf node and simultaneously determines which aggregate it belongs to

Coarse Grid Matrix: $A_c = P^T AP$

PENNSTATE

Triple matrix product:

- In general, triple matrix multiplication consists of two steps:
  1. Step 1: determine the sparsity pattern of A_c (dry run)
  2. Step 2: compute the entries of A_c
- In CUSP, this is done in COO format (1.65×)

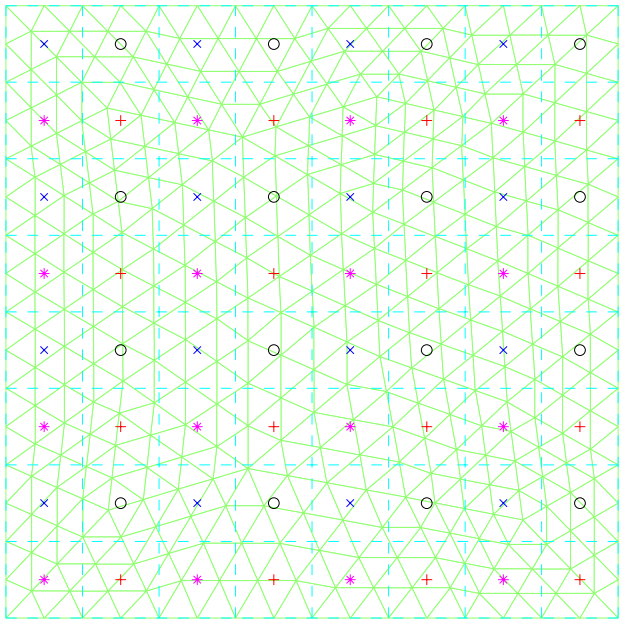
Compute A_c using auxiliary structured grid:

- Sparsity pattern is fixed for the coarse matrices: e.g. 9-point stencil
- Compute entries: $(A_c)_{kl} = \sum_{i \in G_k} \sum_{j \in G_l} a_{ij}$, $i \in G_k$ and $j \in G_l$ just means finding leafs of a root, which can be done in parallel
- Improved parallelism over standard matrix multiplication

Parallel Smoother: 9-point Stencil $\Rightarrow$ 4 Colors

PENNSTATE

Auxiliary structured grid used to color GS



- 4-color blockwise GS smoother on the finest level
- 4-color pointwise GS smoother on the coarse levels

Model Problem &Test Platforms

Poisson equation

$$-\Delta u = f, \quad \text{in } \Omega \in \mathbb{R}^2,$$
$$u = 0, \quad \text{on } \partial\Omega,$$

Platforms

GPU	NVIDIA Tesla C2070
CPU	Intel Xeon E5640 2.66 GHz
OS	Red Hat Linux Server 5.7
Host Compiler	GCC 4.4.4
Device Compiler	NVCC 4.0
CUSP version	0.2.0
Precision	double

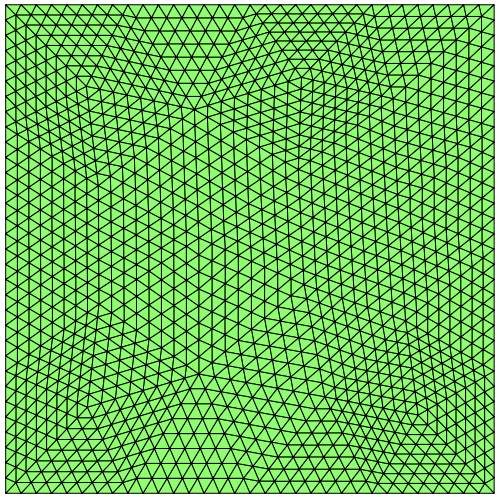
Uniform Grid

	1024×1024				2048×2048			
	Setup	Solve	Total	Iter.	Setup	Solve	Total	Iter.
CPU	0.80	0.69	1.49	7	3.28	2.75	6.03	7
CUSP	0.63	0.35	0.98	36	2.38	1.60	3.98	41
FASP	0.03	0.13	0.16	10	0.11	0.43	0.54	11

Speedup on uniform gird

- Setup: 28.2× CPU, 21.3× CUSP
- Solve: 5.9× CPU, 3.2× CUSP
- Total: 10.2× CPU, 6.7× CUSP
- Basically GMG with p.w. constant coarse space (i.e., have uniform quad-tree and A_c given by stencil)

Quasi-uniform Grid



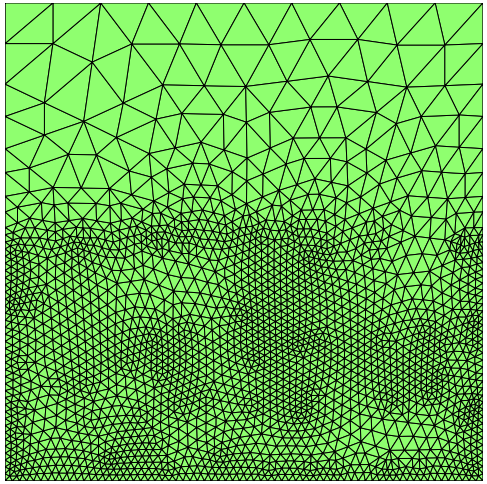
dof	1.4 million			
	Setup	Solve	Total	Iter.
CUSP	1.09	0.82	1.91	40
FASP	0.16	0.49	0.65	15

dof	5.8 million			
	Setup	Solve	Total	Iter.
CUSP	3.65	4.11	7.76	50
FASP	0.53	1.52	2.05	13

Speedup on quasi-uniform gird

- Setup: 6.8$\times$ CUSP
- Solve: 2.2$\times$ CUSP
- Total: 3.4$\times$ CUSP

Shape Regular Grid



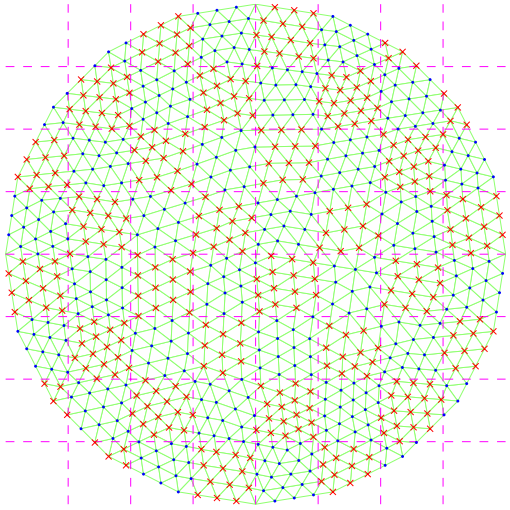
dof	3.4 million			
	Setup	Solve	Total	Iter.
CUSP	2.23	2.51	4.74	52
FASP	0.35	2.12	2.47	25

dof	13.6 million			
	Setup	Solve	Total	Iter.
CUSP	–	–	–	–
FASP	1.62	8.67	10.29	27

Speedup on shape regular gird

- Setup: 6.4$\times$ CUSP
- Solve: 1.2$\times$ CUSP
- Total: 2.0$\times$ CUSP

Circle Domain



dof	2.8 million			
	Setup	Solve	Total	# Iter.
CUSP	2.19	1.68	3.87	50
FASP	0.30	1.65	1.85	21

dof	11.5 million			
	Setup	Solve	Total	# Iter.
CUSP	–	–	–	–
FASP	1.37	7.21	8.58	23

Speedup on circle domain

- Setup: 7.3$\times$ CUSP
- Solve: 1.0$\times$ CUSP
- Total: 2.1$\times$ CUSP
- And roughly $1.2 - 1.5\times$ UA-AMG for general meshes

Isotropic PDE problems – GMG methods

2D FE Poisson problem: GMG with block smoother

- Block Gauss Seidel with minimal overlap and coloring
- Use 4 blocks on coarsest level and double number of blocks in each dimension on each successive finer level
- As size of blocks increased on coarsest level the coarsening becomes more aggressive
- Each column corresponds to recursive application of two-level multiplicative Schwarz method w/ fixed block size

levels / blksz	4^2	8^2	16^2	32^2	64^2	128^2	256^2
2	.01	.06	.18	.44	.66	.81	.92
3	.06	.14	.23	.46	.69	*	*
4	.11	.15	.25	*	*	*	*
5	.11	.15	*	*	*	*	*

Table : Convergence rates of the GMG V(1,1) solver with block GS smoother (using 4 colors) applied to the 5-pt FE Poisson problem on unit square

GMG methods with block smoothers ...

levels / blksz	4^2	8^2	16^2	32^2	64^2	128^2	256^2
2	.003	.005	.09	.29	.46	.70	.85
3	.01	.02	.11	.31	.51	*	*
4	.04	.06	.13	*	*	*	*
5	.04	.06	*	*	*	*	*

Table : Convergence rates of the GMG V(2,2) solver with block GS smoother (using 4 colors) applied to the 5-pt FE Poisson problem on unit square

Note: can be optimized using preconditioned polynomial smoothers (on semi-structured grids sharp LFA predictions possible)

[REF Br., 2013, Br., Hu, Rodrigo, Zikatanov, 2013]

Aggressive coarsening and polynomial smoothing

Let $\rho(A) \leq \lambda_1$ and set $M^{-1} = q_m(A)$, where $q_m(t) \in \mathcal{P}_m$ ia a polynomial of degree m. Consider the smoother with error propagation operator $1 - q_m(x)x$ and let

$$E_m := \max_{x \in [\lambda_0, \lambda_1]} |1 - q_m(x)x| = \max_{x \in [\lambda_0, \lambda_1]} \left| \frac{1}{x} - q_m(x) \right| \cdot x$$

where $q_m(t)$ defined as the unique solution to the minimization problem

$$q_m(x) = \arg\min \{ \| \frac{1}{x} - p \|_{\infty, [\lambda_0, \lambda_1]}, \quad p \in \mathcal{P}_m \}$$

Note that $q_m(t)$ given by three-term recurrence of (scaled and trans.) Chebychev poly. and the smoother given by

$$R = q_\nu(R_0 A) R_0 \quad \text{or equivalently} \quad R = R_0^{\frac{1}{2}} q_\nu(R_0^{\frac{1}{2}} A R_0^{\frac{1}{2}}) R_0^{\frac{1}{2}}. \quad (1)$$

[REF Kraus, Pillwein and Zikatanov, 2010]

Guaranteed smoothing rate on $[\lambda_0, \lambda_1]$

Since λ_1 is a point of Chebyshev alternance

$$E_m = \left| \frac{1}{\lambda_1} - q_m(\lambda_1) \right| \lambda_1 = \frac{\delta^m(\kappa - 1)}{2}$$

where $\kappa = \frac{\lambda_1}{\lambda_0}$, $\delta = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$. Sufficient condition for $q_m(\lambda_1) > 0$ is

$$\frac{1}{\lambda_1} - E_m > 0 \quad \text{or} \quad \delta^m \leq \frac{2}{\lambda_1(\kappa - 1)}$$

giving convergent smoother in A-norm. Given smoothing rate ρ consider

$$\frac{\delta^m(\kappa - 1)}{2} \leq \rho \quad \Rightarrow \quad \delta^m \leq \frac{2\rho}{\kappa - 1}$$

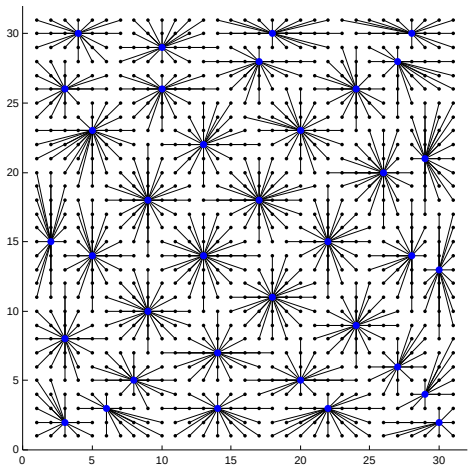
Thus, choose minimal m such that

$$m \geq \frac{1}{|\log \delta|} \max \left\{ \left| \log \frac{2\rho}{\kappa - 1} \right|, \left| \log \frac{2}{\lambda_1(\kappa - 1)} \right| \right\}$$

Aggregation-based AMLI preconditioner

n	AMLI r_a	NAMLI r_a
512^2	0.38	0.38
1024^2	0.40	0.40
2048^2	0.42	0.41
4096^2	0.45	0.40

Table : Convergence rates, r_a for UA-AMG AMLI and NAMLI methods on uniform mesh, grid comp. = 1.03, and oper. comp. = 1.04



- Agg. algorithm for A^4, $n/n_H \approx 30$
- Poly. smoother with $m = 4$
- $W(1,1)$ AMLI and nonlinear AMLI (K-cycle) preconditioners

Figure : Ex. of an aggregation
[Br., DD22 Proc., 2013]

Jump coefficients: finite volume discretization

Consider elliptic equation with jumps in diffusion coefficient:

$$\begin{cases} \nabla \cdot (a(x)\nabla u) = f, & \text{in } \Omega, \\ u = 0, & \text{on } \partial\Omega, \end{cases}$$

where $\Omega \in \mathbb{R}^d$

- For simplicity, consider $d = 2$ and $\Omega = [0,1] \times [0,1]$
- Coefficient $a(x)$ has jump discontinuity of several orders in magnitude but aligns with mesh

The discrete system is

$$a_e u_{i+1,j} + a_w u_{i-1,j} + a_n u_{i,j+1} + a_s u_{i,j-1} - (a_e + a_w + a_n + a_s) u_{i,j} = f_{i,j},$$

where $a_e = \frac{2a^+a^-}{a^+ + a^-}$, with a_w, a_n, and a_s defined similarly

Distribution of jump coefficients

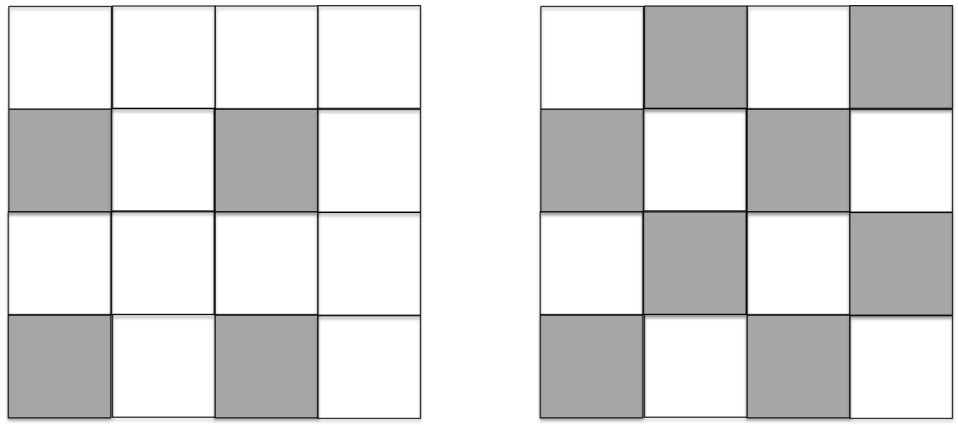


Figure : Distribution of the jump coefficient $a(x)$. Left: Distribution of P1 and P3; Right: Distribution of P2 and P4

$$a(x) = \begin{cases} 1 & x \in \Omega_1 \ , \\ 10^{-k_{ij}} & x \in \Omega \setminus \Omega_1 \ , \end{cases}$$

where Ω_1 corresponds to white regions in the figure

1. P1, P2: $k_{ij} = k \in \mathbb{Z}^+$ for all i, j;
2. P3, P4: $k_{ij} \in \{0, 1, \ldots, k\}$, where the values are selected randomly with uniform distribution (Matlab function `randi`).

Two-level aggregation with quad-tree coarsening

	$k=1$				$k=2$				$k=8$			
Size	ρ	#It	n_H	nnz(A_H)	ρ	#It	n_H	nnz(A_H)	ρ	#It	n_H	nnz(A_H)
16^2	0.45	34	64	576	0.56	46	64	576	0.69	54	64	576
32^2	0.53	43	256	2,304	0.67	57	256	2,304	0.77	63	256	2,304
64^2	0.64	55	1,024	9,216	0.75	69	1,024	9,216	0.91	77	1,024	9,216
128^2	0.75	63	4,096	36,864	0.88	81	4,096	36,864	0.94	91	4,096	36,864

Table : Results for Problem P1, $tol = 1.0e-08$

UA-AMG: aggregates need to align with interfaces at jump discontinuities when using p.w. constant coarse vector space, or extra near kernel functions can be used

Energy Minimization

Assume $\{G_j\}$ given. Consider the following affine subspace of $\mathbb{R}^{n \times n_H}$:

$$\mathcal{X}_H \quad = \quad \{Q \ : \ Q_{ji} = 0 \ \text{ for all } j \notin V_i, \ Q1_H = e\}$$

Let $I_i \in \mathbb{R}^{n \times n_i}$ be the characteristic function over V_i and define $A_i = I_i^T A I_i$. Then P defined as

$$P = \arg\min J(Q) := \arg\min \operatorname{trace}(Q^T A Q), \quad Q \in \mathcal{X}_H$$

The i-th column of the **unique** solution to the minimization problem is given by

$$[P]_i = I_i A_i^{-1} I_i^T M_a e, \quad M_a^{-1} = \sum_{i=1}^{n_H} I_i A_i^{-1} I_i^T$$

[REF Xu & Zikatanov, 2004, Br. & Zikatanov, 2006]

Two-level results for P1

	$k=1$				$k=2$				$k=8$			
Size	ρ	#It	n_H	nnz(A_H)	ρ	#It	n_H	nnz(A_H)	ρ	#It	n_H	nnz(A_H)
16^2	0.18	34	128	1,026	0.19	5	128	1,026	0.19	5	128	1,026
32^2	0.18	5	512	4,354	0.19	5	512	4,354	0.19	5	512	4,354
64^2	0.19	5	2,048	17,922	0.19	5	2,048	17,922	0.19	5	2,048	17,922
128^2	0.19	5	8,192	72,706	0.19	5	8,192	72,706	0.19	5	8,192	72,706

Table : Classical AMG

	$k=1$				$k=2$				$k=8$			
Size	ρ	#It	n_H	nnz(A_H)	ρ	#It	n_H	nnz(A_H)	ρ	#It	n_H	nnz(A_H)
16^2	0.12	5	64	712	0.11	5	64	712	0.12	5	64	712
32^2	0.12	5	256	3,012	0.12	5	256	3,012	0.14	4	256	3,012
64^2	0.15	5	1,024	12,676	0.15	5	1,024	12,676	0.16	5	1,024	12,676
128^2	0.15	5	4,096	40,626	0.16	5	4,096	40,626	0.16	5	4,096	40,626

Table : Quad-tree aggreg. and one smoothing step to form sparsity structure of ($Q_{ji} = 0$ if $(AP)_{ji} = 0$), with coeff. computed using Energy min.

Two-level results for P2

	$k=1$				$k=2$				$k=8$			
Size	ρ	#It	n_H	nnz(A_H)	ρ	#It	n_H	nnz(A_H)	ρ	#It	n_H	nnz(A_H)
16^2	0.18	5	128	998	0.18	5	128	998	0.19	2	128	998
32^2	0.19	5	512	4,354	0.19	4	512	4,354	0.19	2	512	4,294
64^2	0.20	5	2,048	17,922	0.19	4	2,048	17,798	0.19	2	2,048	17,798
128^2	0.20	5	8,192	72,706	0.19	4	8,192	72,454	0.19	2	8,192	72,454

Table : Classical AMG

	$k=1$				$k=2$				$k=8$			
Size	ρ	#It	n_H	nnz(A_H)	ρ	#It	n_H	nnz(A_H)	ρ	#It	n_H	nnz(A_H)
16^2	0.11	5	64	712	0.11	5	64	712	0.12	5	64	712
32^2	0.11	5	256	3,012	0.11	5	256	3,012	0.13	4	256	3,012
64^2	0.12	5	1,024	12,676	0.12	5	1,024	12,676	0.14	5	1,024	12,676
128^2	0.12	5	4,096	40,626	0.12	5	4,096	40,626	0.14	5	4,096	40,626

Table : Quad-tree aggreg. and one smoothing step to form sparsity structure of P, with coeff. computed using Energy min.

Two-level results for P3

	$k=1$				$k=2$				$k=8$			
Size	ρ	#It	n_H	nnz(A_H)	ρ	#It	n_H	nnz(A_H)	ρ	#It	n_H	nnz(A_H)
16^2	0.20	5	128	1,026	0.20	5	128	1,026	0.20	5	128	1,026
32^2	0.20	5	512	4,354	0.21	5	512	4,354	0.20	5	512	4,354
64^2	0.20	5	2,048	17,922	0.20	5	2,048	17,922	0.20	5	2,048	17,922
128^2	0.20	5	8,192	72,706	0.21	5	8,192	72,706	0.20	5	8,192	72,706

Table : Classical AMG

	$k=1$				$k=2$				$k=8$			
Size	ρ	#It	n_H	nnz(A_H)	ρ	#It	n_H	nnz(A_H)	ρ	#It	n_H	nnz(A_H)
16^2	0.08	3	64	712	0.090	3	64	712	0.11	4	64	712
32^2	0.11	4	256	3,012	0.10	4	256	3,012	0.11	4	256	3,012
64^2	0.12	4	1,024	12,676	0.12	4	1,024	12,676	0.12	4	1,024	12,676
128^2	0.12	4	4,096	40,626	0.12	4	4,096	40,626	0.12	4	4,096	40,626

Table : Quad-tree aggreg. and one smoothing step to form sparsity structure of P, with coeff. computed using Energy min.

Two-level results for P4

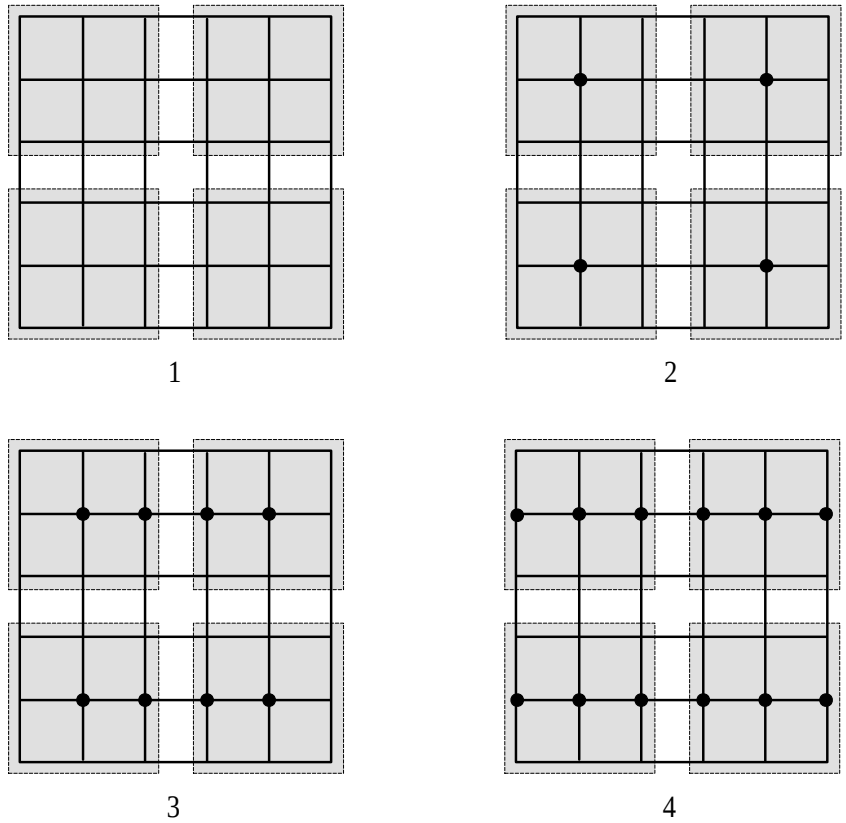
	$k=1$				$k=2$				$k=8$			
Size	ρ	#It	n_H	nnz(A_H)	ρ	#It	n_H	nnz(A_H)	ρ	#It	n_H	nnz(A_H)
16^2	0.20	5	128	998	0.23	5	128	998	0.22	5	128	998
32^2	0.21	5	512	4,354	0.23	5	512	4,354	0.23	5	512	4,354
64^2	0.22	5	2,048	17,922	0.23	5	2,048	17,922	0.24	5	2,048	17,922
128^2	0.22	5	8,192	72,706	0.23	5	8,192	72,706	0.25	5	8,192	72,706

Table : Classical AMG

	$k=1$				$k=2$				$k=8$			
Size	ρ	#It	n_H	nnz(A_H)	ρ	#It	n_H	nnz(A_H)	ρ	#It	n_H	nnz(A_H)
16^2	0.22	6	64	712	0.322	6	64	712	0.54	8	64	712
32^2	0.21	6	256	3,012	0.35	7	256	3,012	0.84	13	256	3,012
64^2	0.22	6	1,024	12,676	0.36	7	1,024	12,676	0.90	18	1,024	12,676
128^2	0.22	6	4,096	40,626	0.36	7	4,096	40,626	0.93	20	4,096	40,626

Table : Quad-tree aggreg. and one smoothing step to form sparsity structure of P, with coeff. computed using Energy min.

Anisotropic problems



- Sequence of steps in coarsening algorithm for bilinear FE discretization of $u_{xx} + \epsilon^{-1} u_{yy}$
- Grey boxes indicate $\{G_j\}$ and black circles denote $\mathcal{C}$ points after successive iterations of algorithm

Reliable: iterative methods (fault tolerance)

Optimal: leads *necessarily* to hierarchical methods (multigrid)

User friendly: algebraic multigrid methods (aggregation methods)

Tunable: geometric-algebraic methods (grid-based coarsening)

Conclusions

Conclusions

- Redundant and randomized iterative methods fault tolerant and convergent in expectation
- Optimal methods are necessarily hierarchical
- Use GMG for isotropic PDE problems
- For PDE systems, use mathematical structures to reduce to solving series of scalar elliptic problems
- Use GAMG for scalar anisotropic diffusion problems to explicitly control complexity
- Algebraic methods needed for general graph problems and aggregation methods provide viable approach
- Can develop specialized algorithms depending on constraints, e.g., aggressive coarsening with polynomials (block) smoothers
- Discretization enhanced AMG methods possible for many important applications, e.g., LQCD