
Adaptive Algebraic Multigrid

Robert D. Falgout

Lawrence Livermore National Laboratory

Seventh European Multigrid Conference

Hohenwart Forum, Germany

October, 2002

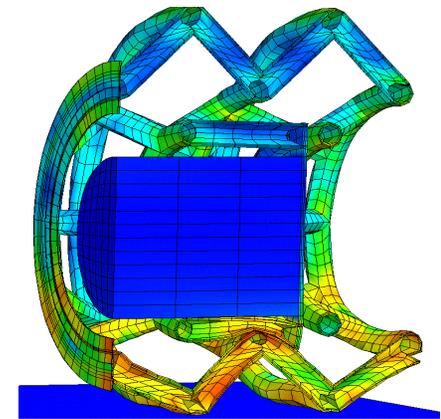


Outline

- **AMG/AMGe Background**
- **The basic Adaptive AMG (*AdAMG*) method**
- **The “wrong approach” (and *PCG*)**
- **Based on Ruge-Stüben (*AdAMG-RS*)**
- **Based on Smoothed Aggregation (*AdAMG-SA*)**
- **Future work / compatible relaxation**

AMG Background

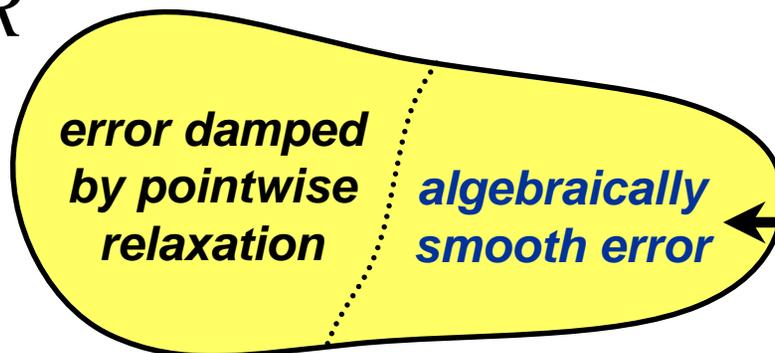
- AMG assumes only information about the underlying matrix structure
- There is no single AMG algorithm
- AMG automatically coarsens “grids”



DYNA3D

AMG Framework

R^n



Choose coarse grids,
transfer operators, etc.
to eliminate

Accurate characterization of smooth error is crucial

Good local characterizations of smooth error is key to robust algebraic multigrid

- **Traditional AMG uses the following heuristic, based on properties of M-matrices:** smooth error varies slowest in the direction of “large” coefficients

$$A = \begin{bmatrix} -1 & -4 & -1 \\ 2 & 8 & 2 \\ -1 & -4 & -1 \end{bmatrix}$$

Stretched quad example ($\Delta x \rightarrow \infty$):
Direction of smoothest error is not readily apparent

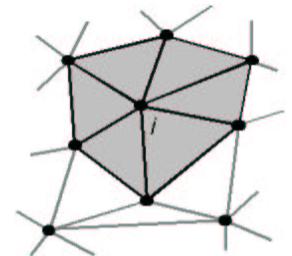
- **New heuristic based on multigrid theory:** interpolation must be able to reproduce a mode up to the same accuracy as the size of the associated eigenvalue

AMGe uses local finite element stiffness matrices to characterize smooth error

- **AMGe heuristic is based on multigrid theory:**
interpolation must be able to reproduce a mode up to the same accuracy as the size of the associated eigenvalue
- **Use local measure to construct AMGe components:**

$$M_i = \max_{e \neq 0} \frac{\langle \varepsilon_i^T (I - Q) e, \varepsilon_i^T (I - Q) e \rangle}{\langle A_i e, e \rangle}; \quad Q = P \begin{bmatrix} 0 & I \end{bmatrix}$$

- **Basic method & theory in *SISC paper*, 2000**
- **Agglomeration approach to appear in *SISC***
- **Element-free variant to appear in *SISC***



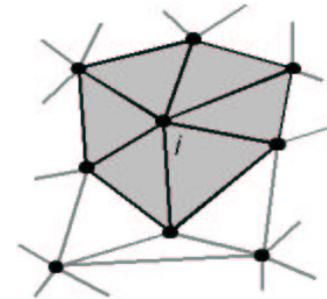
AMGe uses finite element stiffness matrices to localize new heuristic

- Global measure:

$$M(Q) = \max_{e \neq 0} \frac{\langle (I-Q)e, (I-Q)e \rangle}{\langle Ae, e \rangle} ; \quad Q = \begin{bmatrix} W \\ I \end{bmatrix} \begin{bmatrix} 0 & I \end{bmatrix}$$

- Local measure:

$$M_i(q_i) = \max_{e \neq 0} \frac{\langle (\varepsilon_i - q_i)^T e, (\varepsilon_i - q_i)^T e \rangle}{\langle A_i e, e \rangle}$$



where ε_i is a canonical basis vector, q_i^T is a row of interpolation, and A_i is a sum of local stiffness matrices

Using local measure to define interpolation

- Interpolation is defined by the *arg min* of

$$\min_{q_i \in Z_i} M_i(q_i)$$

where we restrict the structure of interpolation to “nearest neighbors” by

$$Z_i = \{v \in R^n : v_j = 0 \text{ for } j \in \Omega \setminus C_i\}$$

- This is easily computed in practice

Computing interpolation in practice

- Partition local matrix by *F* and *C*-pts:

$$A_i = \begin{bmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{bmatrix}$$

- Interpolation to point *i* is defined by

$$q_i^T = \begin{bmatrix} 0 & -\varepsilon_i^T A_{ff}^{-1} A_{fc} \end{bmatrix}$$

Using local measure to define interp. is equivalent to fitting local eigenmodes

- Assume the eigen-decomposition:

$$A_i V_i = V_i \Lambda_i; \quad V_i = \begin{bmatrix} V_{i0} & V_{i+} \end{bmatrix}; \quad \Lambda_i = \begin{bmatrix} 0 & 0 \\ 0 & \Lambda_{i+} \end{bmatrix};$$

- Finding the *arg min* is equivalent to solving the following constrained least-squares problem

$$\min_{q_i} \left\| \Lambda_{i+}^{-1/2} V_{i+}^T (\varepsilon_i - q_i) \right\|^2, \quad \text{subject to} \quad V_{i0}^T (\varepsilon_i - q_i) = 0$$

Adaptive AMG (AdAMG)

Adaptive AMG employs the idea of: *using the method to improve the method*

- **Main idea:** to uncover the slowly-converging error components (the so-called *bad guys*) by applying the “current method” to the system

$$Ax = 0$$

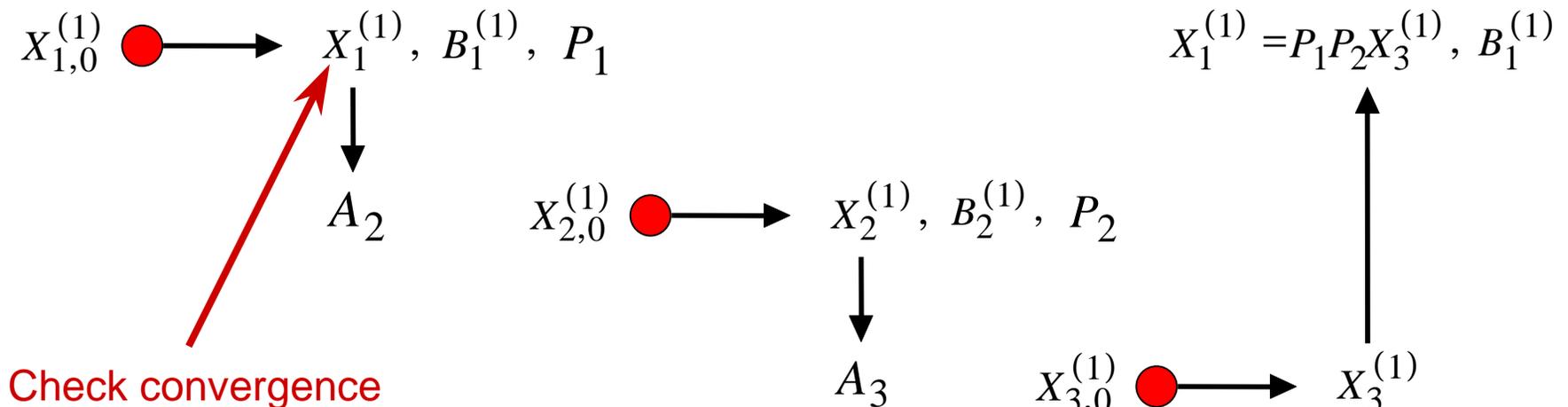
and using them to adapt (improve) the method

- Achi Brandt’s *Bootstrap AMG* is an example
- Notation: Consider solving the system

$$Au = f$$

Basic *AdAMG* cycle (relaxation pass)

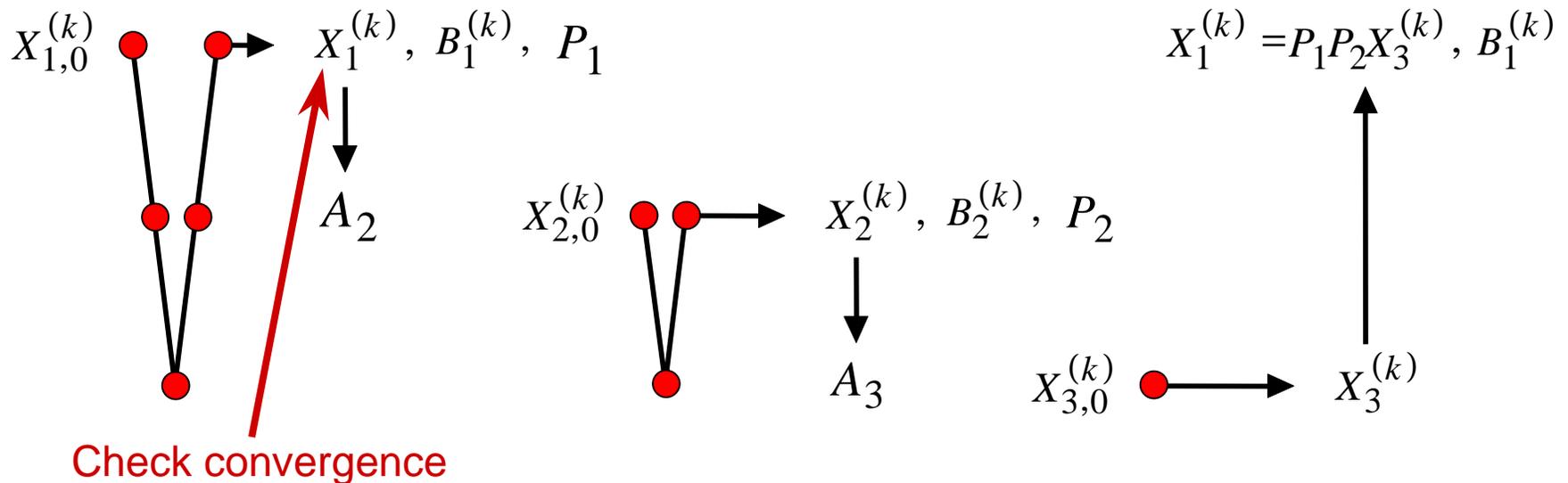
- The columns of $X_l^{(k)}$ are the iteration vectors
- The columns of $B_l^{(k)}$ are the bad guys



- We now have the components of a multigrid method

Basic *AdAMG* cycle (V-cycle pass)

- We can apply the previously constructed MG method to improve the MG method



- We now have new multigrid components

Can imagine many variations for constructing *AdAMG* methods

- **Start with m bad guys and “refine”**

- $X_{1,0}^{(1)}$ random with m columns
- $X_{1,0}^{(k)} = X_1^{(k-1)}$
- $X_{l,0}^{(k)}$ s.t. $X_{l-1}^{(k)} = P_{l-1} X_{l,0}^{(k)}$
- $B_l^{(k)} = X_l^{(k)}$

- **Start with m bad guys and “build”**

- $X_{1,0}^{(k)}$ random with m columns
- $X_{l,0}^{(k)}$ s.t. $X_{l-1}^{(k)} = P_{l-1} X_{l,0}^{(k)}$
- $B_l^{(k)} = [B_l^{(k-1)}, X_l^{(k)}]$

- **Even the basic cycle could be different**

Several approaches for defining P

- Prolongation must interpolate well all components not eliminated by relaxation
- Two basic approaches:

- Construct bad-guy space rich in all smooth modes, and construct P using only this space
- **BAMG does this**

- Construct bad-guy space rich in only smoothest modes, and construct P using this space plus additional matrix information
- Order number-of-kernel-components bad guys
- **Our methods use this approach**

The “wrong approach” for *AdAMG*

- Consider the following iterative method

$$u_{k+1} = u_k + M^{-1}r_k$$

- With $H = (I - M^{-1}A)$, can rewrite as

$$u_{k+1} = H^{k+1}u_0 + \sum_{i=0}^k H^i M^{-1}f$$

- Error propagation is

$$e_{k+1} = H^{k+1}e_0$$

The “wrong approach” for *AdAMG*

- Consider a 2-level $V(j,0)$ cycle with error propagation

$$e_k = (I - Q_k) H^j e_0; \quad Q_k = P_k (P_k^T A P_k)^{-1} P_k^T A$$

- The Setup Algorithm:

$$P_0 = 0; \quad x_0 = \text{random}$$

for ($k = 0; ; = k + 1$)

$$x_{k+1} = (I - Q_k) H^j x_k$$

if ($\|x_{k+1}\| / \|x_k\| < \varepsilon$), **stop**

$$P_{k+1} = [P_k, x_{k+1}]$$

end for

The “wrong approach” for *AdAMG*

- Easy to show that Q_k is just an A -orthogonal projection onto the space

$$K_k = \{ H^j x_0, H^{2j} x_0, \dots, H^{kj} x_0 \}$$

- This looks like a Krylov space!
- In fact, it is just *PCG* with our baseline iterative method as the preconditioner

AdAMG and PCG

- Error propagation for PCG has the form

$$e_{k+1}^p = (I - Q_k^p) e_0^p$$

where Q_k^p is an **A-orthogonal projection onto**

$$K_k^p = \{ Cr_0, (CA)Cr_0, \dots, (CA)^{k-1}Cr_0 \}$$

- **But** $CA = \sum_{i=0}^j H^i M^{-1} A = I - H^j$ **implies**

$$K_k^p = \{ \hat{r}_0, H^j \hat{r}_0, \dots, H^{(k-1)j} \hat{r}_0 \}$$

- Clear that **AdAMG** and **PCG** have same basic properties
- In fact, the “right choice” of initial guesses will produce exactly the same iterates, analytically

AdAMG / PCG relationship underscores the true role of the bad guy

- The role of the bad guy is to be a **representative** or “**straw man**” for constructing what it means to be locally smooth
- The key word here is “**locally**”
- This “wrong approach” example deals with the bad guys globally, and as such, has sub-optimal convergence properties

AdAMG-RS

AdAMG-RS is an adaptive method based on standard Ruge-Stüben heuristics

- In *AdAMG-RS* we will generate only **one bad guy** to represent the smoothest component (thinking of scalar problems for now)
- Note that more information is needed in order to construct good interpolation operators
 - e.g., piecewise constant interpolation fits the constant kernel function exactly, but not good
 - **use bad guy plus M-matrix assumption**
- We will use an *Element-free AMGe* framework to describe this method
- **More about that next...**

***Element-Free AMGe* builds local element matrices directly from the global matrix**

- Partition local neighborhood about point i into

$$N = F \cup C \cup \chi$$

- Define an extension map of the form

$$E = \begin{bmatrix} I & 0 \\ 0 & I \\ E_{\chi_f} & E_{\chi_c} \end{bmatrix}; \quad E_{\chi_f}: F \rightarrow \chi; \quad E_{\chi_c}: C \rightarrow \chi$$

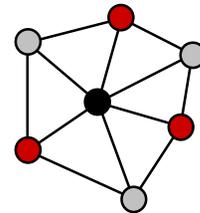
- Then, the needed local operators are defined by

$$\begin{bmatrix} A_{ff} & A_{fc} \end{bmatrix} = \begin{bmatrix} \bar{A}_{ff} & \bar{A}_{fc} & \bar{A}_{f\chi} \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & I \\ E_{\chi_f} & E_{\chi_c} \end{bmatrix}$$

***E-Free AMG* can be used to produce the *R-S AMG* interpolation formulas**

- Let $F = \{ i \}$, the point to which we are interpolating
- Define the extension map as follows (where e is any vector and j is a point in the extension set):

$$(Ee)_j = \begin{cases} e_i & j \text{ weakly connected to } i \\ \sum_{k \in C} \left(\frac{a_{jk}}{\sum_{k' \in C} a_{jk'}} \right) e_k & \text{otherwise} \end{cases}$$



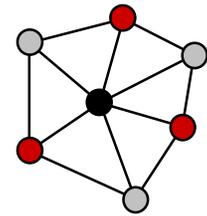
- ***RS-AMG* interpolation is defined by setting the residual at point i to zero:**

$$r_i = \bar{A}_{ff}e_i + \bar{A}_{fc}e_c + \bar{A}_{f\chi}(E_{\chi_f}e_i + E_{\chi_c}e_c) = A_{ff}e_i + A_{fc}e_c = 0$$

AdAMG-RS: Deriving interpolation

- Assume extension map to point j of the form

$$(Ee)_j = -d_j^{-1} \sum_{k \in C} a_{jk} e_k$$



- Want to extend the bad guy x exactly:

$$x_j = -d_j^{-1} \sum_{k \in C} a_{jk} x_k \quad \Rightarrow \quad -d_j^{-1} = \frac{x_j}{\sum_{k \in C} a_{jk} x_k}$$

- This gives the following:

$$(Ee)_j = \begin{cases} \begin{pmatrix} x_j \\ x_i \end{pmatrix} e_i & j \text{ weakly connected to } i \\ \sum_{k \in C} \left(\frac{a_{jk} x_j}{\sum_{k' \in C} a_{jk'} x_{k'}} \right) e_k & \text{otherwise} \end{cases}$$

AdAMG-RS: Some Comments

- Note that the *RS-AMG* heuristic (also notions of “strong” and “weak” connections) changes in this more general setting: smooth error varies **most like the bad guy** in the direction of “large” coefficients
- Note that this new method can solve systems that have been arbitrarily diagonally scaled
- Note also that this is really all the current method extends *RS-AMG* to be able to solve
 - **need to consider alternative approaches not based on M-matrix assumption**

AdAMG-RS: Results

$$-\nabla \cdot (D(x, y) \nabla u(x, y)) = 0 \quad \text{on } [0, 1] \times [0, 1]$$

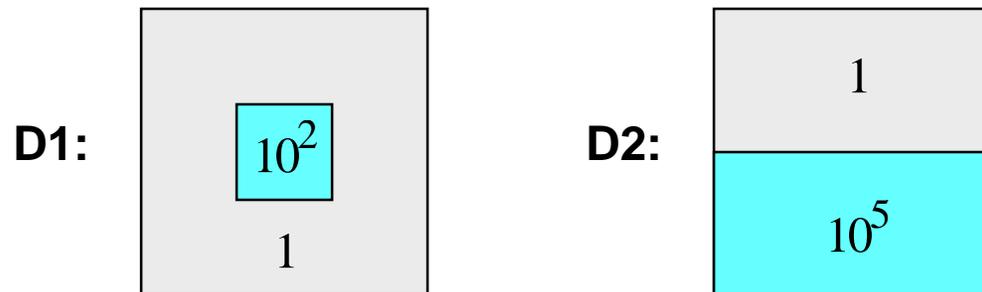
- **Geometric choice of coarse grids**
- **Bad guy generated by 2 cycles of the multilevel setup algorithm**

AdAMG-RS: Results (Laplace)

size	Dirichlet	Neumann
32x32	0.06	0.06
64x64	0.07	0.07
128x128	0.07	0.07
256x256	0.07	0.07
512x512	0.07	0.07
1024x1024	0.07	0.07

- Nice scalability

AdAMG-RS: Results (piecewise constant)



size	D1	D2
32x32	0.08	0.06
64x64	0.09	0.07
128x128	0.08	0.07
256x256	0.11	0.07
512x512	0.17	0.07
1024x1024	0.38	0.07

- Geometric coarsening may be influencing D1 results

AdAMG-RS: Results (scaled)

- Scale by the following function

$$1 + \sin (547\pi x) \sin (496\pi y) + 10^{-7}$$

Scaled Laplacian

size	Dirichlet	Neumann
32x32	0.06	0.06
64x64	0.07	0.07
128x128	0.07	0.07
256x256	0.07	0.07
512x512	0.07	0.07
1024x1024	0.07	0.07

Scaled P-W Constant

size	D1	D2
32x32	0.08	0.06
64x64	0.09	0.07
128x128	0.09	0.07
256x256	0.35	0.07
512x512	0.19	0.07
1024x1024	0.92	0.07

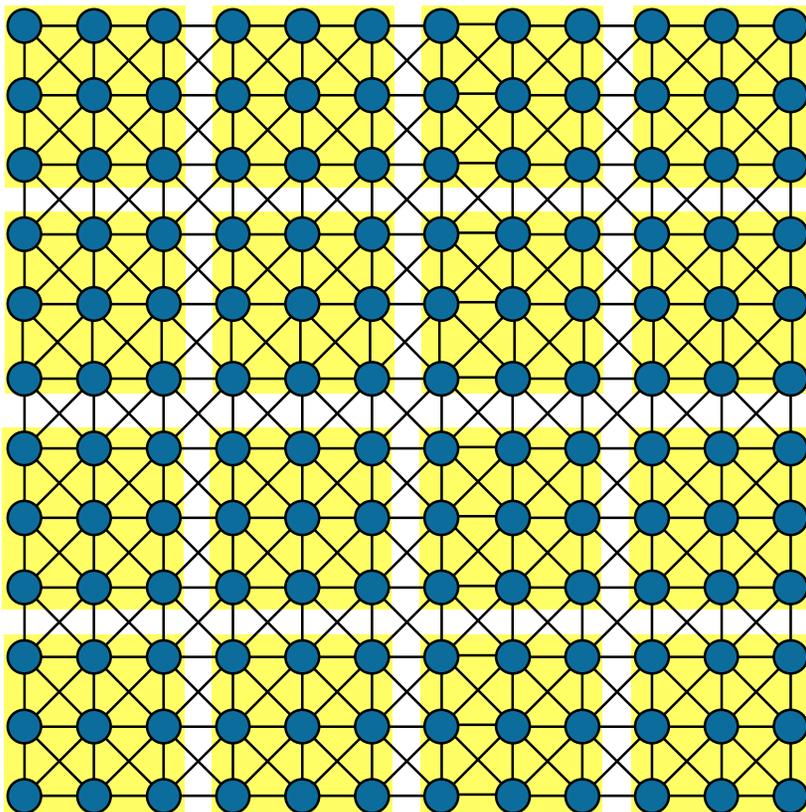
- Scaled results are similar to the unscaled results

AdAMG-SA

AdAMG-SA is an adaptive method based on ***Smoothed Aggregation***

- In *AdAMG-SA* we will generate **several bad guys** to represent the smoothest components
- As with *AdAMG-RS* more information is needed in order to construct good interpolation operators
- We will get this information from the matrix indirectly through the prolongator smoother
- **More about that next...**

Smoothed Aggregation basics



- Points are aggregated
- Tentative prolongator is constructed from aggregates and local basis functions
- Prolongation is built by “smoothing” the tentative prolongator

$$P_l = S_l \hat{P}_l$$

AdAMG-SA: Results (Poisson)

- Scaled version is scaled by diagonal with entries given by 10^β , $\beta \in [-6, 6]$ random

size	scaled	kernel	convergence	complexity
41x41x41		1 (exact)	0.1	1.04
		1	0.1	1.04
	yes	1	0.1	1.04
101x101x101		1 (exact)	0.1	1.04
		1	0.1	1.04
	yes	1	0.1	1.04

AdAMG-SA: Results (2D Elasticity)

- Same scaling as before

size	scaled	kernel	convergence	complexity
80,400		3 (exact)	0.2	1.29
		4	0.1	1.51
		5	0.1	1.79
	yes	4	0.1	1.51
	yes	5	0.1	1.79
180,600		3 (exact)	0.2	1.29
		4	0.2	1.51
		5	0.1	1.79
	yes	4	0.1	1.51
	yes	5	0.1	1.79

Future Directions for *AdAMG*

- **Come up with a better name!**
- ***AdAMG-RS*:**
 - other approaches for using the matrix entries
 - systems version
 - compatible relaxation to choose coarse grids
- **Improve efficiency of setup phase**
- **Apply to harder problems (e.g., Maxwell, Helmholtz)**
- **Theory**

Compatible Relaxation (CR)

CR and AMGe: Preliminaries

- Consider solving the SPD system

$$Au = f$$

- Reorder the equations

$$A = \begin{bmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{bmatrix}$$

- Define the global measure

$$M(Q, e) = \frac{\langle (I-Q)e, (I-Q)e \rangle}{\langle Ae, e \rangle}; \quad Q = \begin{bmatrix} W \\ I \end{bmatrix} \begin{bmatrix} 0 & I \end{bmatrix}$$

CR and AMGe: Measuring the quality of the coarse grid

- Assume we are given a coarse grid. **Then, the following measures the ability of the coarse grid to represent algebraically smooth error.**

$$M_c = \min_Q \max_{e \neq 0} M(Q, e)$$

- Note that we have not restricted interpolation to be local here, but for PDE problems, we should be able to approximate Q with local interpolation.
- We have that

$$W = -A_{ff}^{-1} A_{fc}; \quad M_c = \frac{1}{\lambda_{\min}(A_{ff})}$$

CR and AMGe: Convergence of CR

- Compatible relaxation (Brandt, ETNA, 2000) is in general a modified relaxation scheme that keeps the coarse-level variables invariant
- In general, the coarse-level variables are defined to be some linear combination of fine-level variables:

$$u_i^c = \sum_j \mu_{ij} u_j$$

- **A general measure for the quality of the set of coarse variables is the convergence rate of CR**
- **General idea:** *If CR is slow to converge, either increase the size of the coarse grid or do more relaxation*
- **F-relaxation is a specific instance of CR**

CR and AMGe: Convergence of CR

- Consider weighted pointwise Jacobi *F*-relaxation

$$\begin{aligned}\rho (I - \omega M_{ff}^{-1} A_{ff}) &= \max_{\lambda} \left| \lambda (I - \omega M_{ff}^{-1} A_{ff}) \right| \\ &= \max_{\lambda} \left| 1 - \omega \lambda (M_{ff}^{-1} A_{ff}) \right| \\ &\geq 1 - \omega \lambda_{\min} (M_{ff}^{-1} A_{ff}) \\ &= 1 - \omega \lambda_{\min} (M_{ff}^{-1/2} A_{ff} M_{ff}^{-1/2})\end{aligned}$$

then $M_c \leq \frac{\omega}{1 - \rho}$

- For appropriate choice of weight: *Jacobi F-relaxation is fast to converge iff the AMGe measure is small*

Using *CR* to Define Coarse Variables

- To check convergence of *CR*, relax on the equation

$$A_{ff} x = 0$$

and monitor pointwise convergence to 0

- *CR* coarsening algorithm:

Initialize $U = \Omega$; $C = \emptyset$; $F = \Omega - C$

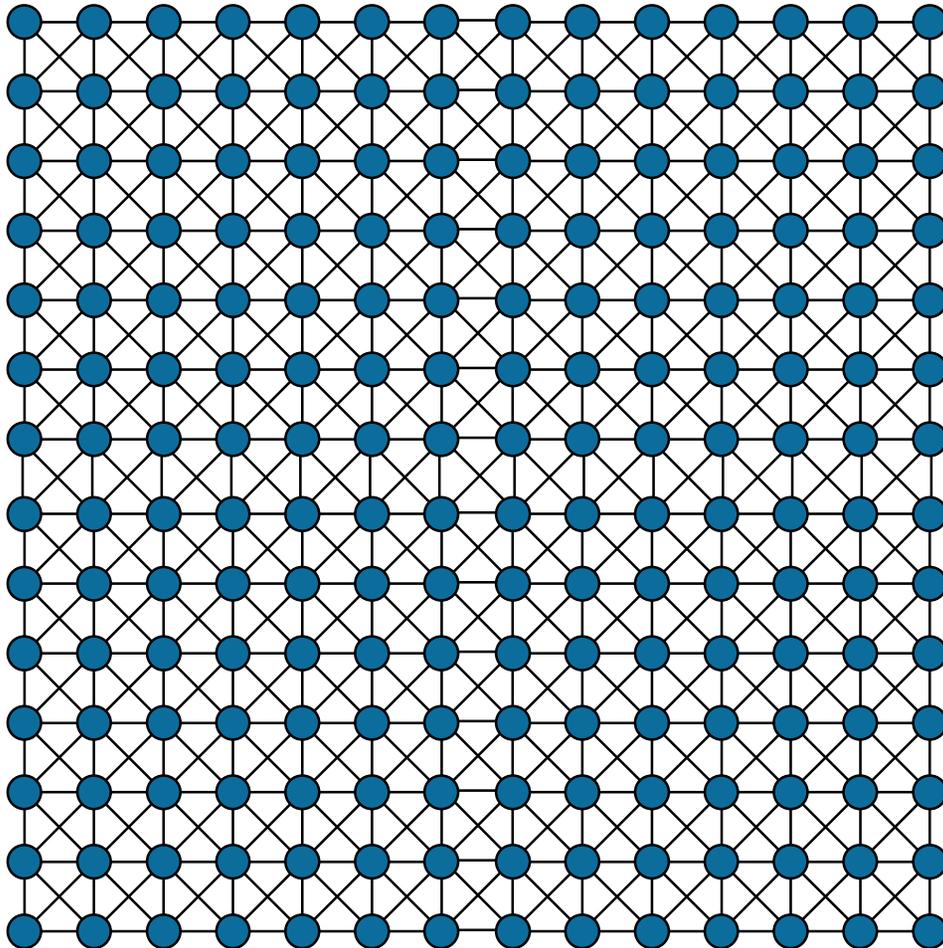
While $U \neq \emptyset$

Do ν compatible relaxation sweeps

$$U = \{i : x_i^\nu / x_i^{\nu-1} > \theta\}$$

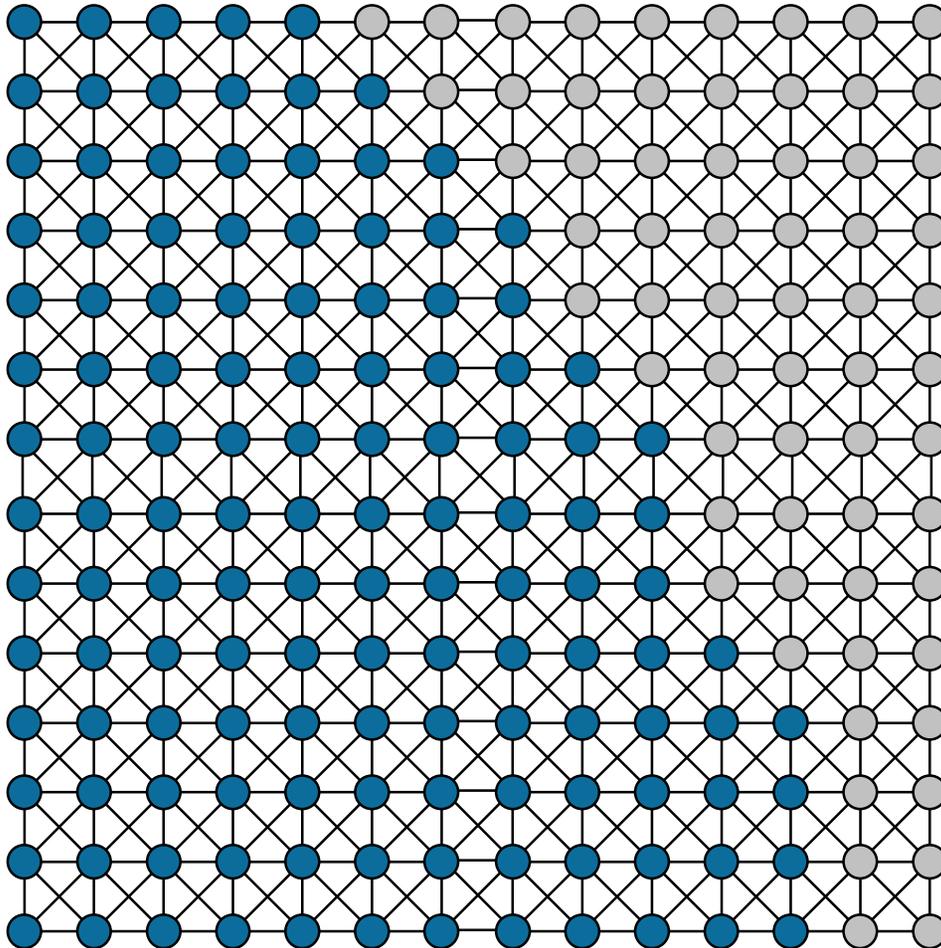
$$C = C \cup \{\text{independent set of } U\}; F = \Omega - C$$

Using *CR* to Define Coarse Variables



- ➔ **Initialize U-pts**
- ➔ Do *CR* and redefine U-pts as points slow to converge
- ➔ Select new C-pts as indep. set over U

Using *CR* to Define Coarse Variables

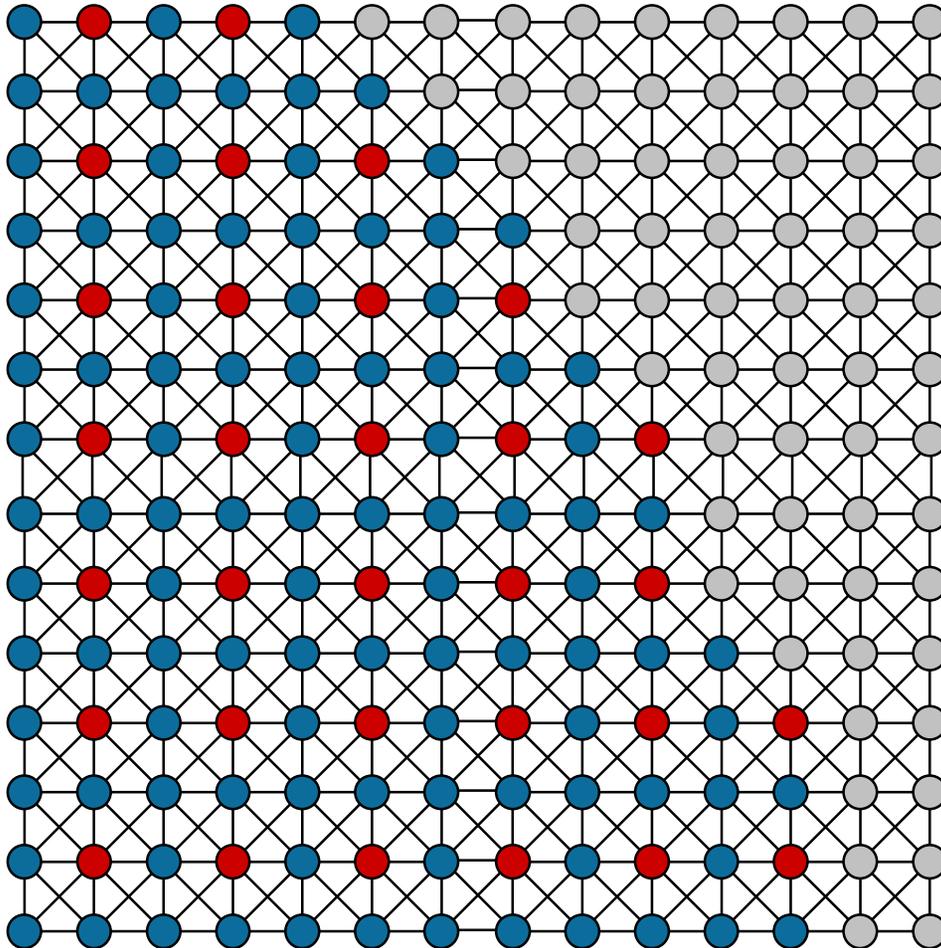


➔ Initialize U-pts

➔ Do *CR* and redefine U-pts as points slow to converge

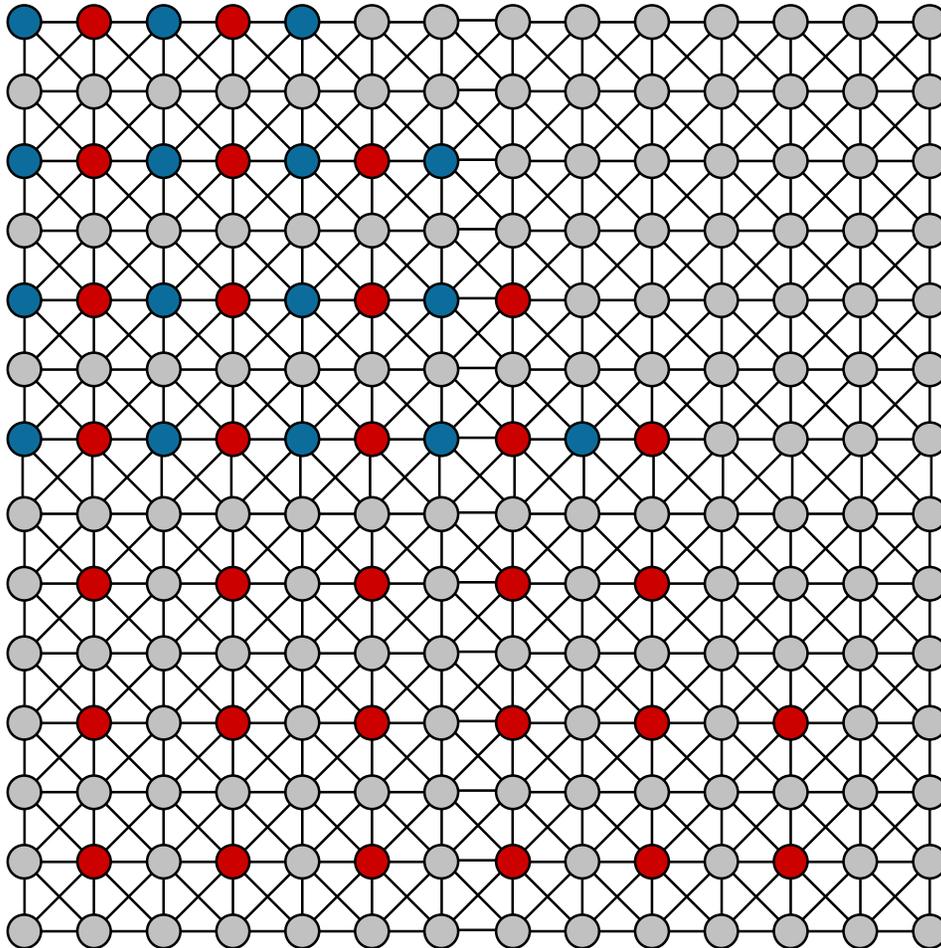
➔ Select new C-pts as indep. set over U

Using *CR* to Define Coarse Variables



- ➔ Initialize U-pts
- ➔ Do *CR* and redefine U-pts as points slow to converge
- ➔ **Select new C-pts as indep. set over U**

Using *CR* to Define Coarse Variables

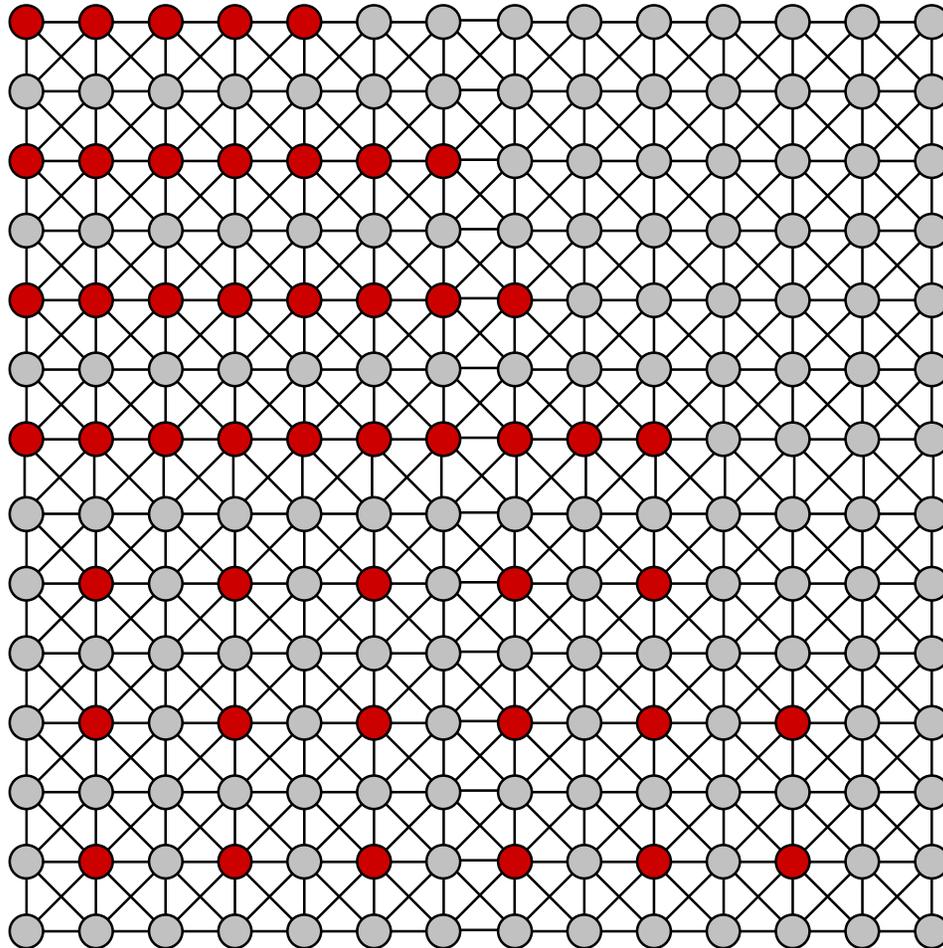


➔ Initialize U-pts

➔ Do *CR* and redefine U-pts as points slow to converge

➔ Select new C-pts as indep. set over U

Using *CR* to Define Coarse Variables



- ➔ Initialize U-pts
- ➔ Do *CR* and redefine U-pts as points slow to converge
- ➔ **Select new C-pts as indep. set over U**

***CR* and *AMGe*: The Future**

- **Have (very) recently generalized the *AMGe* measure and theory to handle the more general *CR* setting**
- **Thinking about how we might apply these ideas to automatically adapt to harder problem settings**

Some Relevant Publications

(see http://www.llnl.gov/casc/linear_solvers)

- M. Brezina, A.J. Cleary, R.D. Falgout, V.E. Henson, J.E. Jones, T.A. Manteuffel, S.F. McCormick, and J.W. Ruge, “**Algebraic Multigrid Based on Element Interpolation (AMGe)**,” *SIAM J. Sci. Comput.* , **22** (2000), pp. 1570–1592.
- A.J. Cleary, R.D. Falgout, V.E. Henson, J.E. Jones, T.A. Manteuffel, S.F. McCormick, G.N. Miranda, and J.W. Ruge, “**Robustness and Scalability of Algebraic Multigrid**,” *SIAM J. Sci. Comput.* , **21** (2000), pp. 1886–1908.
- V.E. Henson and P. S. Vassilevski, “**Element-Free AMGe: General Algorithms for Computing Interpolation Weights in AMG**”, *SIAM J. Sci. Comput.* , **23** (2001), pp. 629–650.
- J. Jones and P. Vassilevski, “**AMGe Based on Element Agglomeration**,” *SIAM J. Sci. Comput.* , **23** (2001), pp. 109–133.
- T. Chartier, R.D. Falgout, J.E. Jones, T.A. Manteuffel, S.F. McCormick, J.W. Ruge, and P.S. Vassilevski, “**Spectral AMGe**”, submitted.
- T. Chartier, R.D. Falgout, J.E. Jones, T.A. Manteuffel, S.F. McCormick, J.W. Ruge, and P.S. Vassilevski, “**Spectral Agglomeration AMGe**”, in preparation.
- M. Brezina, R.D. Falgout, S. MacLachlan, T.A. Manteuffel, S.F. McCormick, and J.W. Ruge, “**Self-Correcting Smoothed Aggregation (scSA)**,” in preparation.



This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract no. W-7405-Eng-48.

UCRL-PRES-149422