Recent Advances in System Solvers

Tom Manteuffel

University of Colorado at Boulder

Office of Advanced Scientific Computing Research

2007 Applied Mathematics Principle Investigators Meeting

May 22-24, 2007

Outline

- Introductory Remarks
- Linear Solvers
  - Context
  - MG/AMG
  - Adaptive AMG
- Nonlinear Systems
  - Context
  - Nested Iteration/Newton/Krylov/AMG
- Summary

Outline

- Introductory Remarks
- Linear Solvers
  - Context
  - MG/AMG
  - Adaptive AMG
- Summary

Theme song:

Everything is linear,

Linear Solvers

Theme song:

Everything is linear,

...in its own way

Linear Solvers

Theme song:

Everything is linear,

...in its own way

Take a simple Newton step,

Theme song:

Everything is linear,

...in its own way

Take a simple Newton step,

and iterate from 1 *to* k*.*

Sung to the tune of "Everything is Beautiful" by Ray Stevens

Krylov Methods $\Leftrightarrow$ Polynomial Methods

$$A\underline{x} = \underline{b}$$

$\underline{x}_0$		initial guess
$\underline{x}_i$		iterate
$\underline{e}_i$	$= \underline{x} - \underline{x}_i$	error
$\underline{r}_i$	$= \underline{b} - A\underline{x}_i$	residual

Krylov Methods $\Leftrightarrow$ Polynomial Methods

$$A\underline{x} = \underline{b}$$

$\underline{x}_0$		initial guess
$\underline{x}_i$		iterate
$\underline{e}_i$	$= \underline{x} - \underline{x}_i$	error
$\underline{r}_i$	$= \underline{b} - A\underline{x}_i$	residual

Error Equation: $p_i(0) = 1.0$

$$\underline{e}_i = p_i(A)e_0$$

$$\underline{r}_i = p_i(A)r_0$$

Polynomial Methods: Error Bounds

Jordan Decomposition

$$A = SJS^{-1}$$

Error Bound

$$\|e_i\| \leq \|S\| \|S^{-1}\| \|p_i(J)\|$$

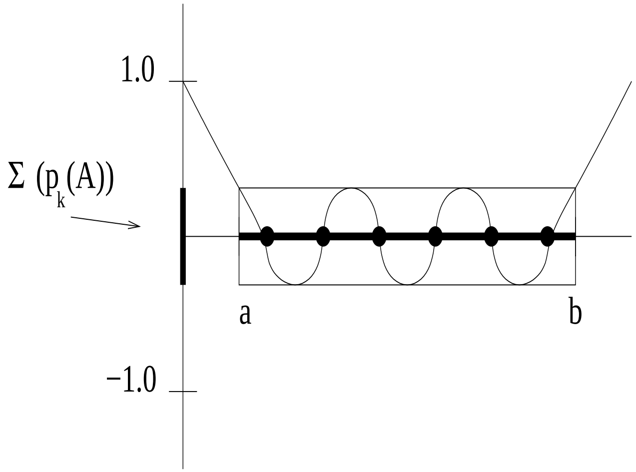
Polynomial Methods: Error Bounds

Jordan Decomposition

$$A = SJS^{-1}$$

Error Bound

$$\|e_i\| \leq \|S\|\|S^{-1}\|\|p_i(J)\|$$



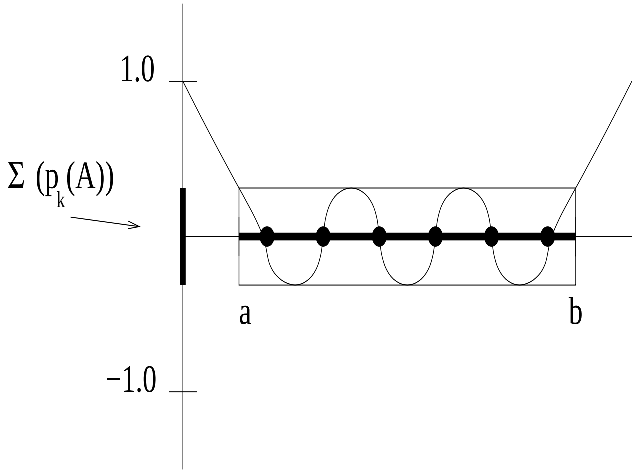
Polynomial Methods: Error Bounds

Jordan Decomposition

$$A = SJS^{-1}$$

Error Bound

$$\|e_i\| \leq \|S\|\|S^{-1}\|\|p_i(J)\|$$



If condition of A is large, it is hard to make a polynomial small on all of the eigenvalues and still have $p_i(0) = 1$.

Preconditioning

$$C A \underline{x} = C \underline{b}$$

C – Any linear process

Choose C so that system with CA is easier to solve in some sense

For example, condition of CA is much smaller than that of A

Preconditioning

$$C A \underline{x} = C \underline{b}$$

C – Any linear process

Examples: $A = L + D + U$

- $C = D^{-1}$, Jacobi Preconditioning

Preconditioning

$$C A \underline{x} = C \underline{b}$$

C – Any linear process

Examples: $A = L + D + U$

- $C = D^{-1}$, Jacobi Preconditioning
- $C = (D + L)^{-1}$ Gauss/Seidel

Preconditioning

$$C A \underline{x} = C \underline{b}$$

C – Any linear process

Examples: $A = L + D + U$

- $C = D^{-1}$, Jacobi Preconditioning
- $C = (D+L)^{-1}$ Gauss/Seidel
- $C = ((D+L)D^{-1}(D+U))^{-1}$ Symmetric Gauss/Seidel

Preconditioning

$$C A \underline{x} = C \underline{b}$$

C – Any linear process

Examples: $A = L + D + U$

- $C = D^{-1}$, Jacobi Preconditioning
- $C = (D + L)^{-1}$ Gauss/Seidel
- $C = ((D + L) D^{-1} (D + U))^{-1}$ Symmetric Gauss/Seidel
- $C = ((\hat{D} + L) \hat{D}^{-1} (\hat{D} + U))^{-1}$ Incomplete Factorization

Preconditioning

$$C A \underline{x} = C \underline{b}$$

C – Any linear process

Examples: $A = L + D + U$

- $C = D^{-1}$, — Jacobi Preconditioning
- $C = (D + L)^{-1}$ — Gauss/Seidel
- $C = ((D + L)D^{-1}(D + U))^{-1}$ — Symmetric Gauss/Seidel
- $C = ((\hat{D} + L)\hat{D}^{-1}(\hat{D} + U))^{-1}$ — Incomplete Factorization
- $C = A^*$ — Normal Equations

Preconditioning

$$C A \underline{x} = C \underline{b}$$

C – Any linear process

Examples: $A = L + D + U$

- $C = D^{-1}$, Jacobi Preconditioning
- $C = (D + L)^{-1}$ Gauss/Seidel
- $C = ((D + L)D^{-1}(D + U))^{-1}$ Symmetric Gauss/Seidel
- $C = ((\hat{D} + L)\hat{D}^{-1}(\hat{D} + U))^{-1}$ Incomplete Factorization
- $C = A^*$ Normal Equations
- $C =$ Multigrid V-cycle PCG-MG

Preconditoned Polynomial Methods

Preconditioning $\Leftrightarrow$ Matrix Splitting

Preconditoned Polynomial Methods

Preconditioning $\Leftrightarrow$ Matrix Splitting

- Any matrix splitting can be used as a preconditioning

Preconditoned Polynomial Methods

Preconditioning $\Leftrightarrow$ Matrix Splitting

- Any matrix splitting can be used as a preconditioning
- Any linear process, C, can be used as a preconditioning

Preconditoned Polynomial Methods

Preconditioning $\Leftrightarrow$ Matrix Splitting

- Any matrix splitting can be used as a preconditioning
- Any linear process, C, can be used as a preconditioning
- Any preconditioning can accelerated by a polynomial method

Numerical PDEs

- In general, if A comes from a PDE, optimal preconditioning requires a Multilevel algorithm
  - Optimal $\Rightarrow$ condition of CA is independent of the mesh

Numerical PDEs

- In general, if A comes from a PDE, optimal preconditioning requires a Multilevel algorithm
  - Optimal $\Rightarrow$ condition of CA is independent of the mesh
  - Optimal $\Rightarrow$ work grows linearly with the problem size

Numerical PDEs

- In general, if A comes from a PDE, optimal preconditioning requires a Multilevel algorithm
  - Optimal $\Rightarrow$ condition of CA is independent of the mesh
  - Optimal $\Rightarrow$ work grows linearly with the problem size

If you want to solve a problem with billions of unknowns on $128,000$ *processors, you will need a multilevel algorithm somewhere.*

Recent Developments

- A lot of recent activity in multilevel algorithms

Recent Developments

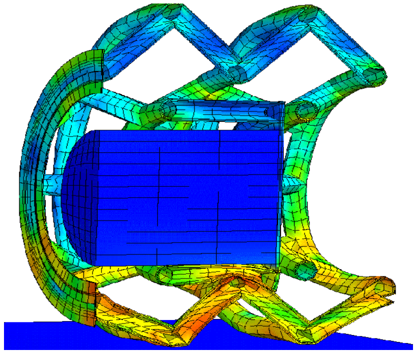
- A lot of recent activity in multilevel algorithms
- Especially in Algebraic Multigrid (AMG)

Recent Developments

- A lot of recent activity in multilevel algorithms
- Especially in Algebraic Multigrid (AMG)
  - More robust
  - More effective

Recent Developments

- A lot of recent activity in multilevel algorithms
- Especially in Algebraic Multigrid (AMG)
  - More robust
  - More effective



DYNA3D

Multigrid

Basic Components

- Simple relaxation or smoothing
  - Usually a matrix splitting or simple preconditioned one-step like damped Jacobi, Gauss/Sedel or block Gauss/Seidel
  - Resolves error in direction of eigenvectors with large eigenvalues
- Coarse-grid correction
  - Lower dimensional or simpler problem
  - Resolves error left by relaxation
- Recursion
  - Coarse-grid problem is solved by multigrid

Multigrid: example

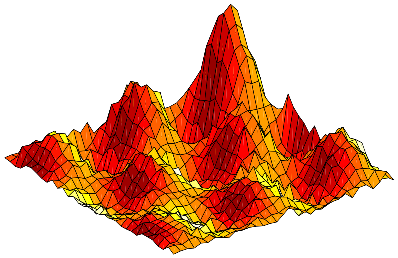
Example: discrete forms of second-order elliptic operators

$$-\nabla \cdot A \nabla u + cu = f$$

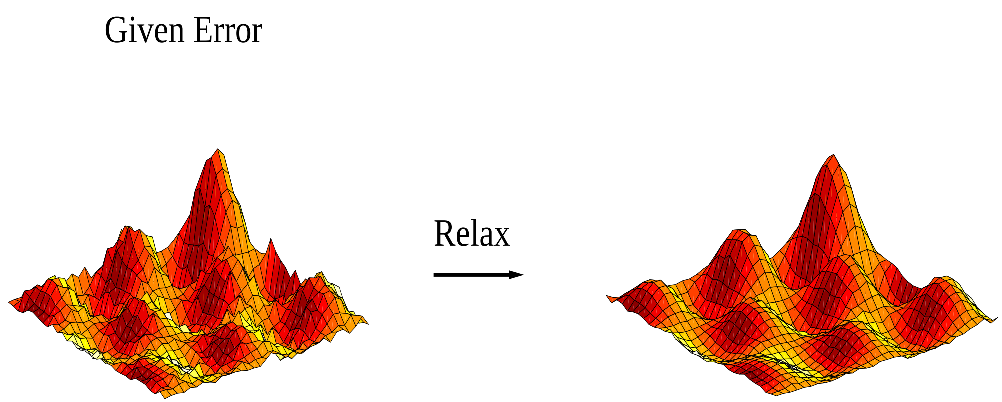
- Large eigenvalues are associated with high frequency eigenvectors
- Simple iterative methods leave error geometrically smooth
- Coarse grid problem is a version of fine grid problem

Multigrid: example

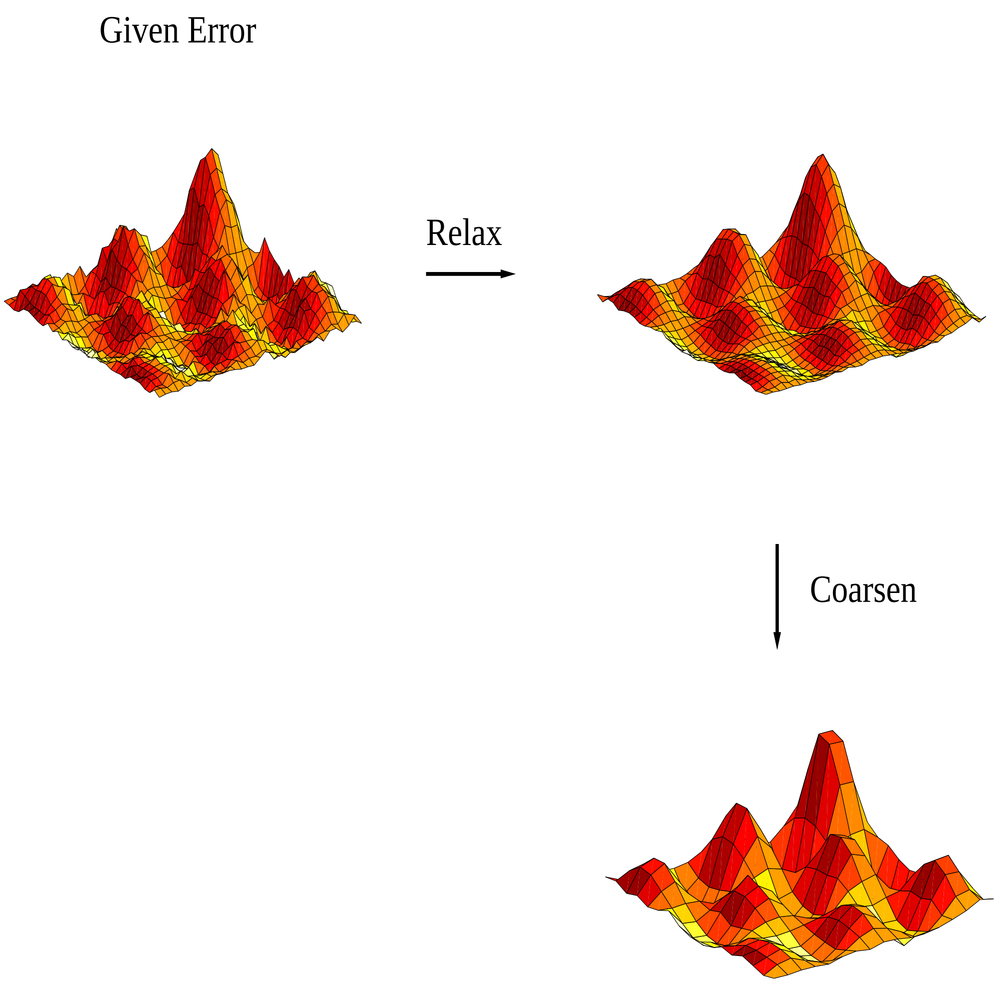
Given Error



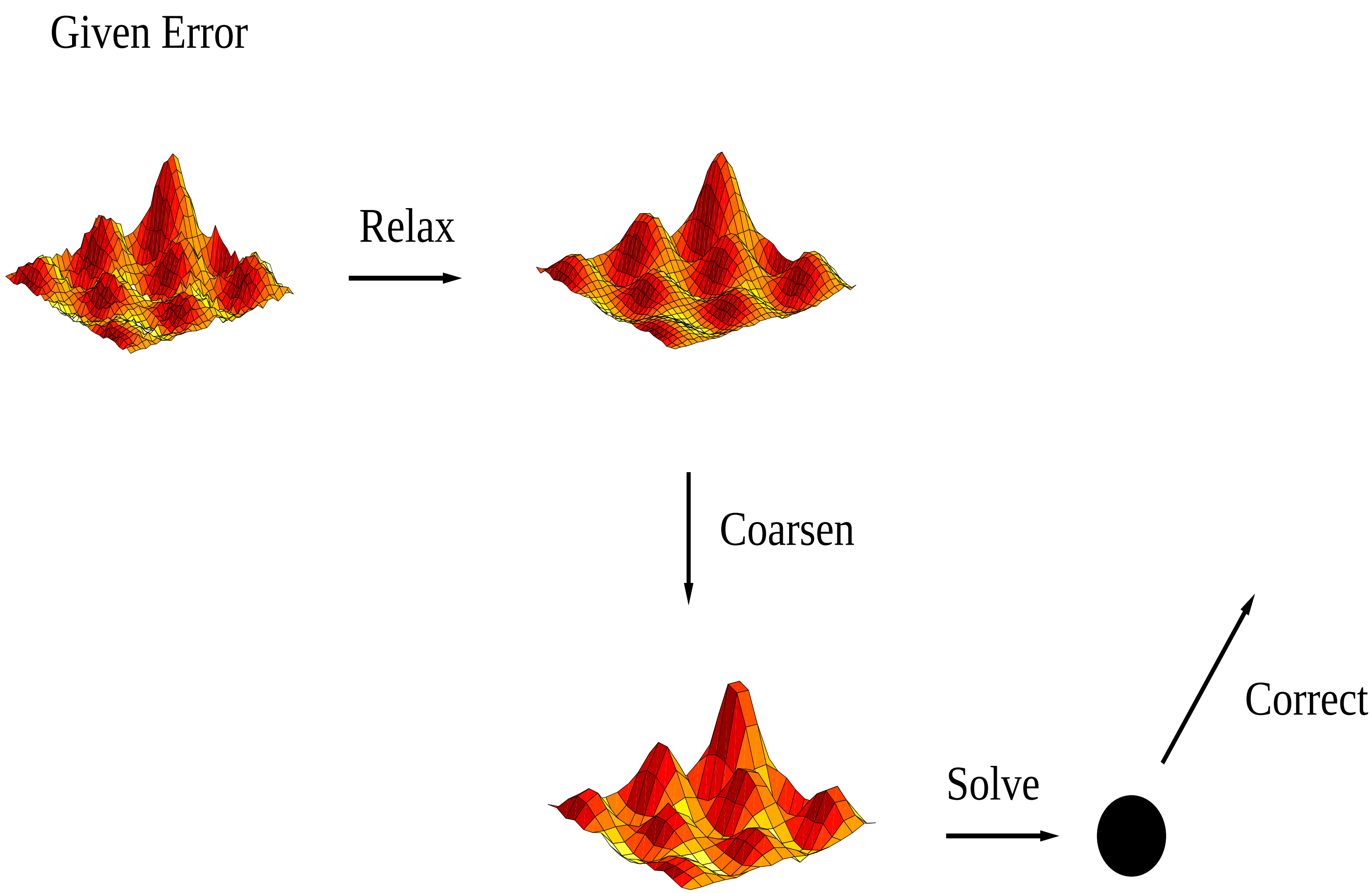
Multigrid: example



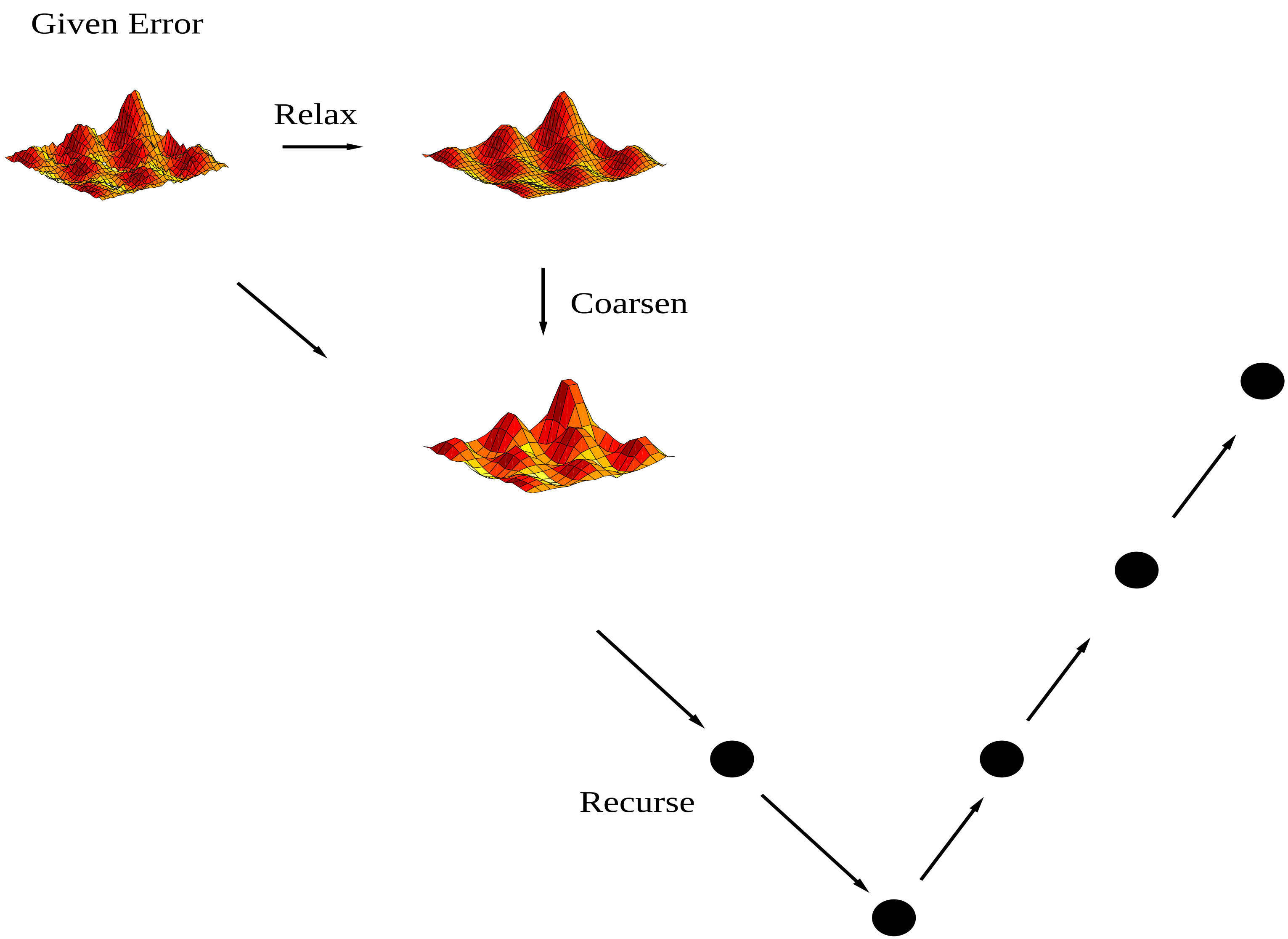
Multigrid: example



Multigrid: example



Multigrid: example



Basic Components (again)

Multigrid algorithm is determined by

- Relaxation
- Interpolation from coarse grid to fine grid (P)
- Restriction from fine grid to coarse grid (R)
- Coarse-grid operator (A_c)

Basic Components (again)

Multigrid algorithm is determined by

- Relaxation
- Interpolation from coarse grid to fine grid (P)
- Restriction from fine grid to coarse grid (R)
- Coarse-grid operator (A_c)

In variational MG,

$$A_c = RA_fP$$

Multigrid Flavors

- Geometric Multigrid (GMG)
- Algebraic Multigrid (AMG)

Multigrid Flavors

Geometric multigrid (GMG)

- Coarse-grid problem is geometrically determined
- It is usually a smaller version of the fine grid problem
- Interpolation and restriction usually determined by the operator

Multigrid Flavors

Algebraic Multigrid (AMG)

- Directly address the matrix A without presumed knowledge of
  - Geometry
  - Operator

Multigrid Flavors

Algebraic Multigrid (AMG)

- Directly address the matrix A without presumed knowledge of
- Assume simple relaxation
  - For example, Damped Jacobi, Gauss/Seidel

Multigrid Flavors

Algebraic Multigrid (AMG)

- Directly address the matrix A without presumed knowledge of
- Assume simple relaxation
- Coarse-grid problem is chosen to resolve the “Algebraicaly smooth” error
  - Defined to be the error that relaxation does not resolve

Multigrid Flavors

Algebraic Multigrid (AMG)

- Directly address the matrix A without presumed knowledge of
- Assume simple relaxation
- Coarse-grid problem is chosen to resolve the “Algebraicaly smooth” error
- Work focuses on selection of a coarse grid and the intergrid transfer operators (R and P)

Multigrid Flavors

Algebraic Multigrid (AMG)

- Directly address the matrix A without presumed knowledge of
- Assume simple relaxation
- Coarse-grid problem is chosen to resolve the “Algebraicaly smooth” error
- Work focuses on selection of a coarse grid and the intergrid transfer operators (R and P)
- The coarse-grid operator is formed variationally ($A_c = RA_fP$)

Multigrid Flavors

- Geometric Multigrid (GMG)
- Algebraic Multigrid (AMG)

Multigrid Flavors

- Geometric Multigrid (GMG)
- Algebraic Multigrid (AMG)
  - AMG
  - Smoothed Aggregation (SA)
  - Adaptive AMG (αAMG, αSA)

AMG Principles

AMG is characterized by choice of the Coarse Grid, Interpolation, P, and Restriction, R.

For simplification, assume A symmetric and $R = P^t$

AMG Principles

Divide degrees of freedom into the Coarse DOF and Fine DOF

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

AMG Principles

Divide degrees of freedom into the Coarse DOF and Fine DOF

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

After relaxation error is algebraically smooth

$$||A\underline{e}|| << ||\underline{e}||$$

error in direction of large eigenvalues has been reduced

AMG Principles

Divide degrees of freedom into the Coarse DOF and Fine DOF

$$A = \left[\begin{array}{cc} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{array} \right]$$

After relaxation error is algebraically smooth

$$\|A\underline{e}\| << \|\underline{e}\|$$

error in direction of large eigenvalues has been reduced

$$\left[\begin{array}{cc} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{array} \right] \left(\begin{array}{c} \underline{e}_f \\ \underline{e}_c \end{array} \right) \simeq \left(\begin{array}{c} \underline{0} \\ \underline{0} \end{array} \right)$$

AMG Principles

$$\begin{bmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{bmatrix} \begin{pmatrix} \underline{e}_f \\ \underline{e}_c \end{pmatrix} \simeq \begin{pmatrix} \underline{0} \\ \underline{0} \end{pmatrix}$$

AMG Principles

$$\begin{bmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{bmatrix} \begin{pmatrix} \underline{e}_f \\ \underline{e}_c \end{pmatrix} \simeq \begin{pmatrix} \underline{0} \\ \underline{0} \end{pmatrix}$$

$$A_{ff}\underline{e}_f + A_{fc}\underline{e}_c = 0$$

AMG Principles

$$\begin{bmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{bmatrix} \begin{pmatrix} \underline{e}_f \\ \underline{e}_c \end{pmatrix} \simeq \begin{pmatrix} \underline{0} \\ \underline{0} \end{pmatrix}$$

$$A_{ff}\underline{e}_f + A_{fc}\underline{e}_c = 0$$

$$\underline{e}_f = -A_{ff}^{-1} A_{fc} \underline{e}_c$$

AMG Principles

$$\begin{bmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{bmatrix} \begin{pmatrix} \underline{e}_f \\ \underline{e}_c \end{pmatrix} \simeq \begin{pmatrix} \underline{0} \\ \underline{0} \end{pmatrix}$$

$$A_{ff}\underline{e}_f + A_{fc}\underline{e}_c = 0$$

$$\underline{e}_f = -A_{ff}^{-1} A_{fc} \underline{e}_c$$

Perfect Interpolation

$$\begin{pmatrix} \underline{e}_f \\ \underline{e}_c \end{pmatrix} = \begin{bmatrix} -A_{ff}^{-1} A_{fc} \\ I \end{bmatrix} \underline{e}_c = P\underline{e}_c$$

AMG Principles

Perfect Interpolation

$$\begin{pmatrix} \underline{e}_f \\ \underline{e}_c \end{pmatrix} = \begin{bmatrix} -A_{ff}^{-1} A_{fc} \\ I \end{bmatrix} \underline{e}_c = P \underline{e}_c$$

AMG Principles

Perfect Interpolation

$$\begin{pmatrix} \underline{e}_f \\ \underline{e}_c \end{pmatrix} = \begin{bmatrix} -A_{ff}^{-1} A_{fc} \\ I \end{bmatrix} \underline{e}_c = P \underline{e}_c$$

After relaxation

$$\begin{aligned} AP\underline{e}_c &= \underline{r} \\ P^t AP\underline{e}_c &= P^t \underline{r} \\ A_c \underline{e}_c &= \underline{r}_c \end{aligned}$$

AMG Principles

Perfect Interpolation

$$\begin{pmatrix} \underline{e}_f \\ \underline{e}_c \end{pmatrix} = \begin{bmatrix} -A_{ff}^{-1} A_{fc} \\ I \end{bmatrix} \underline{e}_c = P \underline{e}_c$$

After relaxation

$$\begin{aligned} AP\underline{e}_c &= \underline{r} \\ P^t AP\underline{e}_c &= P^t \underline{r} \\ A_c \underline{e}_c &= \underline{r}_c \end{aligned}$$

A_c is the Schur Complement

$$A_c = A_{cc} - A_{cf} A_{ff}^{-1} A_{fc}$$

A_c is Dense

AMG Principles

Problem: A_{ff}^{-1} is Dense $\Rightarrow A_c$ is Dense

AMG Principles

Problem: A_{ff}^{-1} is Dense $\Rightarrow A_c$ is Dense

Solution: Sparse Approximation of A_{ff}^{-1}

- $A_{ff}^{-1} \longrightarrow D_{ff}^{-1}$ Diagonal of A_{ff}
- $A_{ff}^{-1} \longrightarrow \hat{D}_{ff}^{-1}$ Lumped Diagonal of A_{ff}
- $A_{ff}^{-1} \longrightarrow C_{ff}$ Sparse approximate inverse of A_{ff}

AMG Principles

Problem: A_{ff}^{-1} is Dense $\Rightarrow A_c$ is Dense

Solution: Sparse Approximation of A_{ff}^{-1}

- $A_{ff}^{-1} \to D_{ff}^{-1}$ Diagonal of A_{ff}
- $A_{ff}^{-1} \to \hat{D}_{ff}^{-1}$ Lumped Diagonal of A_{ff}
- $A_{ff}^{-1} \to C_{ff}$ Sparse approximate inverse of A_{ff}

For example: simple iteration on $A_{ff} = D_{ff} - B_{ff}$

$$A_{ff}^{-1} \to (I + D_{ff}^{-1} B_{ff}) D_{ff}^{-1}$$

Iterated Interpolation, Long Range Interpolation, Compatible Relaxation

AMG Principles

Problem: A_{ff}^{-1} is Dense $\Rightarrow A_c$ is Dense

Solution: Sparse Approximation of A_{ff}^{-1}

- $A_{ff}^{-1} \rightarrow D_{ff}^{-1}$ Diagonal of A_{ff}
- $A_{ff}^{-1} \rightarrow \hat{D}_{ff}^{-1}$ Lumped Diagonal of A_{ff}
- $A_{ff}^{-1} \rightarrow C_{ff}$ Sparse approximate inverse of A_{ff}

Are any of these any good?

Weak Approximation Property

Interpolation must approximate an eigenvector up to the same accuracy as the size of the corresponding eigenvalue

Weak Approximation Property

Interpolation must approximate an eigenvector up to the same accuracy as the size of the corresponding eigenvalue

Weak approximation property: there exists constant C

$$M(P,\underline{u}) := \min_{\underline{v}} \frac{\|\underline{u} - P\underline{v}\|^2}{\langle A\underline{u},\, \underline{u}\rangle} \leq \frac{C}{\|A\|}$$

Weak Approximation Property

Interpolation must approximate an eigenvector up to the same accuracy as the size of the corresponding eigenvalue

Weak approximation property: there exists constant C

$$M(P,\underline{u}) := \min_{\underline{v}} \frac{\|\underline{u} - P\underline{v}\|^2}{\langle A\underline{u},\, \underline{u}\rangle} \leq \frac{C}{\|A\|}$$

Two-grid Convergence Factor

$$\rho \leq 1 - O(\frac{1}{C})$$

Measure can be enforced locally

Strength of Connection

Attempt to identify connections between unknowns that are important

Strength of Connection

Attempt to identify connections between unknowns that are important

Strength of Connection: Original definition: i is strongly depends on the set

$$S_i := \{j \ : \ |a_{ij}| \geq \theta \max_{k \neq i} |a_{ik}|\}$$

for some parameter θ. (e.g. θ = .25)

Strength of Connection

Attempt to identify connections between unknowns that are important

Strength of Connection: Original definition: i is strongly depends on the set

$$S_i := \{ j \; : \; |a_{ij}| \geq \theta \max_{k \neq i} |a_{ik}| \}$$

for some parameter θ. (e.g. θ = .25)

New, more general, definitions of strength derived from local approximation of A^{-1}

Strength of Connection

Attempt to identify connections between unknowns that are important

Strength of Connection: Original definition: i is strongly depends on the set

$$S_i := \{j \;:\; |a_{ij}| \geq \theta \max_{k \neq i} |a_{ik}|\}$$

for some parameter θ. (e.g. θ = .25)

Strength of connection fundamental in choosing the coarse grid

AMG Alphabet Soup

AMG	Classical AMG (84)
SA	Soothed Aggregation (96)
AMGe	finite element AMG (01)
AMG$\not{e}$	element free AMGe (02)
ρAMGe	spectral AMGe (03)

AMG Alphabet Soup

AMG	Classical AMG (84)
SA	Soothed Aggregation (96)
AMGe	finite element AMG (01)
AMG$\not e$	element free AMGe (02)
ρAMGe	spectral AMGe (03)

Adaptive Algorithms

AMG	adaptive AMG (84)
BAMG	Bootstrap AMG (01)
αSA	adaptive Soothed Aggregation (04)
CR	Compatible Relaxation (04)
αAMG	adaptive AMG (06)
αAMGr	adaptive AMGr (06)

Cassical AMG

- Classical or RS - AMG (84)

Cassical AMG

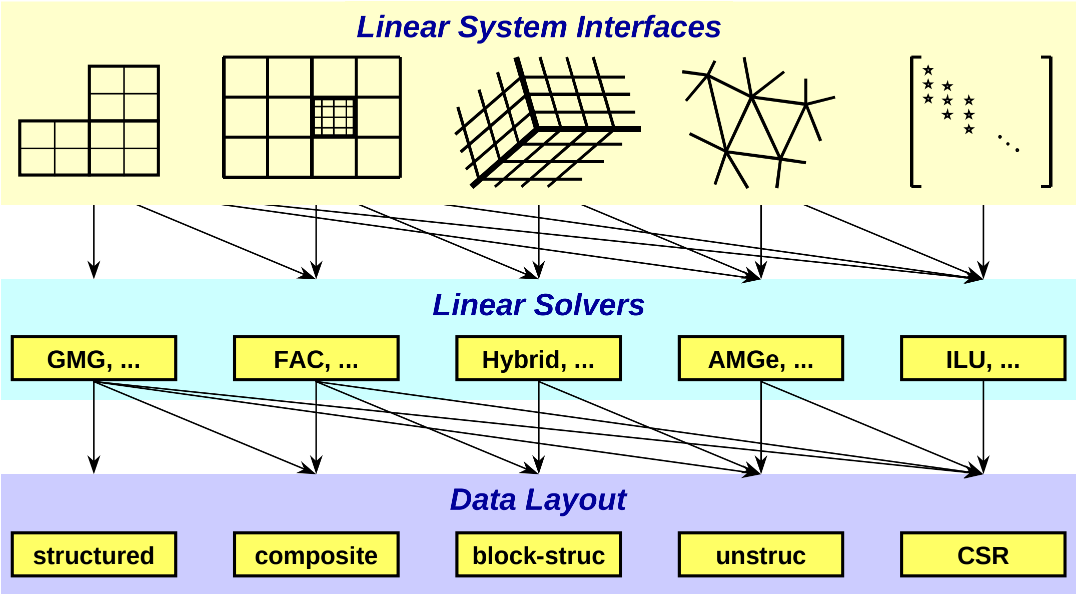
- Classical or RS - AMG (84)
  - Developed by Brandt/McCormick/Ruge (84)
  - Implemented by Ruge/Stuben (85)

Cassical AMG

- Classical or RS - AMG (84)
- Workhorse in many applications

Cassical AMG

- Classical or RS - AMG (84)
- Workhorse in many applications
- Implemented as BoomerAMG in HYPRE



Cassical AMG

- Classical or RS - AMG (84)
- Workhorse in many applications
- Implemented as BoomerAMG in HYPRE
- Weaknesses:
  - Systems of PDEs
  - Singularities
  - Operator complexity in 3D

Cassical AMG

- Classical or RS - AMG (84)
- Workhorse in many applications
- Implemented as BoomerAMG in HYPRE
- Weaknesses:
  - Systems of PDEs
  - Singularities
  - Operator complexity in 3D
- Based on M-matrix principles

AMG Alphabet Soup

AMG	Classical AMG (84)
SA	Soothed Aggregation (96)
AMGe	finite element AMG (01)
AMG$\not{e}$	element free AMGe (02)
ρAMGe	spectral AMGe (03)

Adaptive Algorithms

AMG	adaptive AMG (84)
BAMG	Bootstrap AMG (01)
αSA	adaptive Soothed Aggregation (04)
CR	Compatible Relaxation (04)
αAMG	adaptie AMG (06)
αAMGr	adaptive AMGr (06)

Smoothed Aggregation

Smoothed Aggregation (SA) Brezina, Mandel, Vanek (96)

Smoothed Aggregation

Smoothed Aggregation (SA) Brezina, Mandel, Vanek (96)

- Requires knowledge of one (or more) global (near) null-space vector(s), $\underline{v}_j$.

Smoothed Aggregation

Smoothed Aggregation (SA) Brezina, Mandel, Vanek (96)

- Requires knowledge of one (or more) global (near) null-space vector(s), $\underline{v}_j$.
- Divide the Graph of A into disjoint aggregates, $\{\mathcal{A}_i\}$

Smoothed Aggregation

Smoothed Aggregation (SA) Brezina, Mandel, Vanek (96)

- Requires knowledge of one (or more) global (near) null-space vector(s), $\underline{v}_j$.
- Divide the Graph of A into disjoint aggregates, $\{\mathcal{A}_i\}$
- Associate one (or more) coarse-level DOF with each aggregate

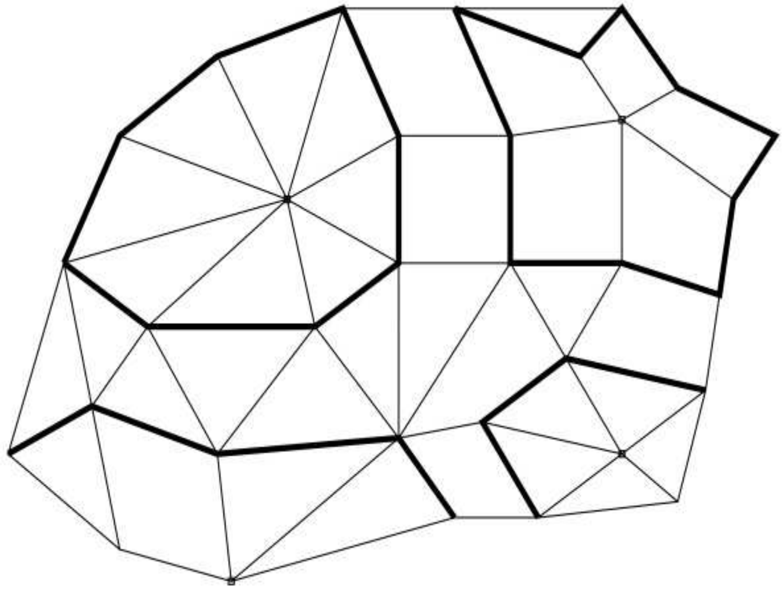
Smoothed Aggregation (SA) Brezina, Mandel, Vanek (96)

- Requires knowledge of one (or more) global (near) null-space vector(s), $\underline{v}_j$.
- Divide the Graph of A into disjoint aggregates, $\{\mathcal{A}_i\}$
- Associate one (or more) coarse-level DOF with each aggregate
- Construct a tentative interpolation matrix, $\hat{P}$, by chopping up the near null-space vector(s)

Smoothed Aggregation: Interpolation

Null-space vector: $\underline{v} = (v_1, v_2, \ldots, v_n)^t$

$$\hat{P} = \begin{bmatrix} v_1 & & \\ \vdots & & \\ v_{n_{f_1}} & & \\ & v_{n_2} & \\ & \vdots & \\ & v_{n_{f_2}} & \\ & & v_{n_c} \\ & & \vdots \\ & & v_{n_{f_c}} \end{bmatrix}$$



Note: $\underline{v}$ is in Range(P)

Smoothed Aggregation: Interpolation

Normalize

$$\hat{P}^t \hat{P} = I$$

Smoothed Aggregation: Interpolation

Normalize

$$\hat{P}^t \hat{P} = I$$

Smooth $\hat{P}$

$$P = (I - \alpha A)\hat{P}$$

Smoothed Aggregation: Interpolation

Normalize

$$\hat{P}^t \hat{P} = I$$

Smooth $\hat{P}$

$$P = (I - \alpha A)\hat{P}$$

Construct coarse-grid operator

$$A_c = P^t A P = \hat{P}^t (I - \alpha A) A (I - \alpha A) \hat{P}$$

Smoothed Aggregation: Interpolation

Normalize

$$\hat{P}^t \hat{P} = I$$

Smooth $\hat{P}$

$$P = (I - \alpha A)\hat{P}$$

Construct coarse-grid operator

$$A_c = P^t A P = \hat{P}^t (I - \alpha A) A (I - \alpha A) \hat{P}$$

- Choose α to reduce the condition of A_c

Smoothed Aggregation: Interpolation

Normalize

$$\hat{P}^t \hat{P} = I$$

Smooth $\hat{P}$

$$P = (I - \alpha A)\hat{P}$$

Construct coarse-grid operator

$$A_c = P^t A P = \hat{P}^t (I - \alpha A) A (I - \alpha A) \hat{P}$$

- Choose α to reduce the condition of A_c

Recurse

Smoothed Aggregation

$$A_c = P^t A P = \hat{P}^t (I - \alpha A) A (I - \alpha A) \hat{P}$$

- Reduces the condition of A_c
- Maintains good approximation of null-space vector, $\underline{v}$
  - Null-space, $\underline{v}$, still in the range of P
  - Other near null-space vectors still well approximated by P
- Yields aggressive coarsening

Multiple Null Space Vectors

Accommodate multiple (near) null-space vectors, $V = [\underline{v}_1, \ldots, \underline{v}_k]$

$$V_j = \begin{bmatrix} v_{1n_j} & \cdot & v_{kn_j} \\ \vdots & & \vdots \\ v_{1n_{f_j}} & \cdot & v_{kn_{f_j}} \end{bmatrix}$$

$$\hat{P} = \begin{bmatrix} V_1 & & \\ & V_2 & \\ & & V_{n_c} \end{bmatrix}$$

Multiple Null Space Vectors

Accommodate multiple (near) null-space vectors, $V = [\underline{v}_1, \ldots, \underline{v}_k]$

$$V_j = \begin{bmatrix} v_{1n_j} & \cdot & v_{kn_j} \\ \vdots & & \vdots \\ v_{1n_{f_j}} & \cdot & v_{kn_{f_j}} \end{bmatrix}$$

$$\hat{P} = \begin{bmatrix} V_1 & & \\ & V_2 & \\ & & V_{n_c} \end{bmatrix}$$

Normalize

$$\hat{P}^t \hat{P} = I$$

Smooth $\hat{P}$

$$P = (I - \alpha A)\hat{P}$$

Coarse-grid operator

$$A_c = P^t A P$$

Recurse

Smoothed Aggregation

- *Very effective for systems, like linear Elasticity, where (near) null-space (rigid body motions) is known.*

Smoothed Aggregation

- *Very effective for systems, like linear Elasticity, where (near) null-space (rigid body motions) is known.*
- *Effective in the context of irregular meshes*

Smoothed Aggregation

- *Very effective for systems, like linear Elasticity, where (near) null-space (rigid body motions) is known.*
- *Effective in the context of irregular meshes*
- *Aggressive coarsening yields good complexity*

Smoothed Aggregation

- *Very effective for systems, like linear Elasticity, where (near) null-space (rigid body motions) is known.*
- *Effective in the context of irregular meshes*
- *Aggressive coarsening yields good complexity*
- *Amenable to parallel implementation*

Smoothed Aggregation

- *Very effective for systems, like linear Elasticity, where (near) null-space (rigid body motions) is known.*
- *Effective in the context of irregular meshes*
- *Aggressive coarsening yields good complexity*
- *Amenable to parallel implementation*
- *Conceptionally straightforward*

SA and AMG

Compare SA to AMG

- SA constructs P column by column
- AMG constructs P row by row
- Both attempt to accurately interpolate algebraically smooth vectors
- Both try to reduce the complexity (number of nonzeros) of the coarse-grid operator

AMG Alphabet Soup

AMG	Classical AMG (84)
SA	Soothed Aggregation (96)
AMGe	finite element AMG (01)
AMG$\cancel{e}$	element free AMGe (02)
ρAMGe	spectral AMGe (03)

Adaptive Algorithms

AMG	adaptive AMG (84)
BAMG	Bootstrap AMG (01)
αSA	adaptive Soothed Aggregation (04)
CR	Compatible Relaxation (04)
αAMG	adaptive AMG (06)
αAMGr	adaptive AMGr (06)

AMG Gang

Ball State	I. Livshits	Delft	S. MacLachlan
Boulder	M. Brezina	Davidson College	T. Chartier
	T. Manteuffel	FIT	J. Jones
	S. McCormick	Penn State	J. Brannick
	J. Ruge		J. Xu
	G. Sanders		L. Zikatanov
	B. Sheehan	SNL	J. Hu
LLNL	A. Baker		R. Tuminaro
	A. Cleary	Urbana-Champaign	D. Alber
	R. Falgout		L. Olson
	V. Henson	Utah	O. Livne
	T. Kolev	Weizmann Institute	A. Brandt
	B. Lee		
	P. Vassilevski		
	U. Yang		

AMG Alphabet Soup

AMG	Classical AMG (84)
SA	Soothed Aggregation (96)
AMGe	finite element AMG (01)
AMG$\cancel{e}$	element free AMGe (02)
ρAMGe	spectral AMGe (03)

Adaptive Algorithms

AMG	adaptive AMG (84)
BAMG	Bootstrap AMG (01)
αSA	adaptive Soothed Aggregation (04)
CR	Compatible Relaxation (04)
αAMG	adaptive AMG (06)
αAMGr	adaptive AMGr (06)

AMGe

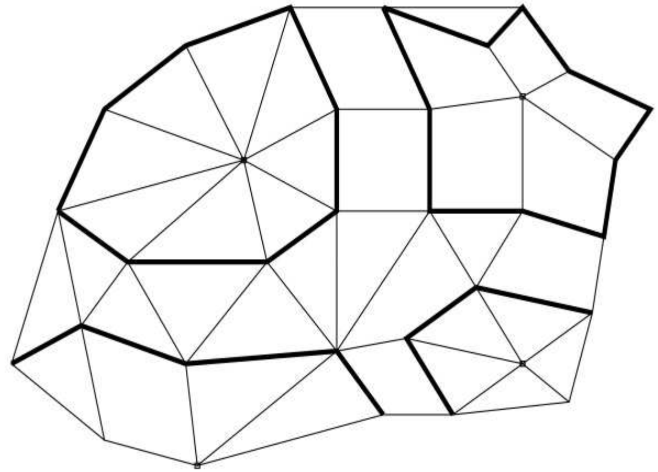
AMGe Finite element AMG (04)

AMGe

AMGe

Finite element AMG (04)

- Uses local stiffness matrices
- Aggregates elements like SA
- Uses local null-space to determine local interpolation properties

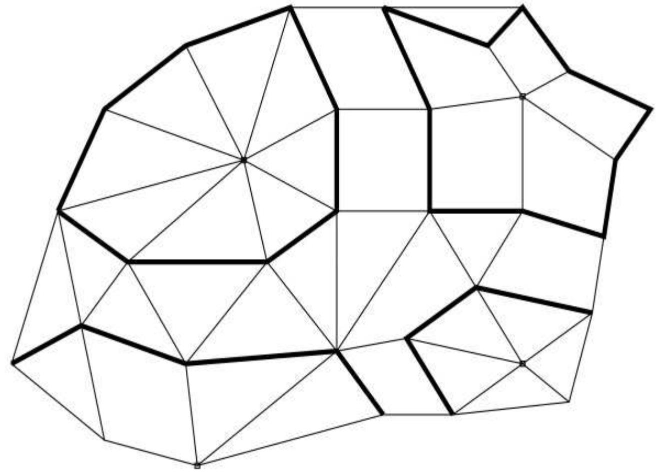


AMGe

AMGe

Finite element AMG (04)

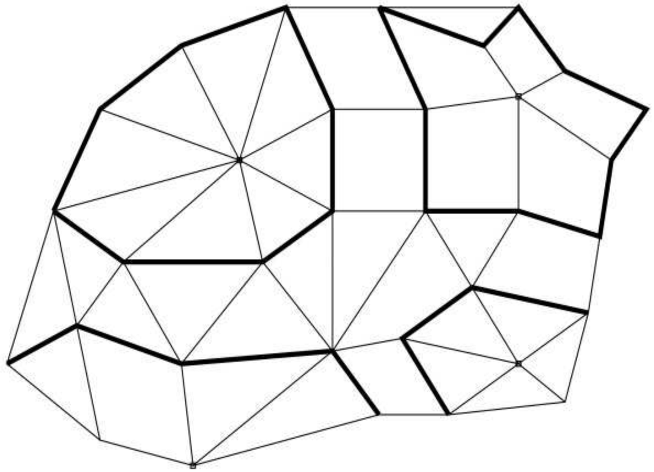
- Uses local stiffness matrices
- Aggregates elements like SA
- Uses local null-space to determine local interpolation properties
- Effective for
  - Anisotropic Problems
  - Systems PDEs

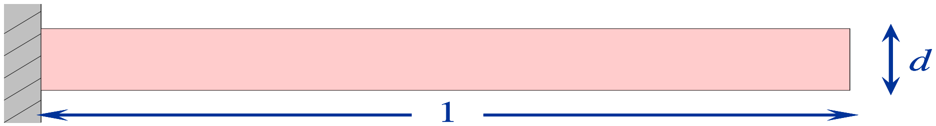


AMGe

Finite element AMG (04)

- Uses local stiffness matrices
- Aggregates elements like SA
- Uses local null-space to determine local interpolation properties





	AMG	AMGe
ρ	.98	.26

AMG Alphabet Soup

AMG	Classical AMG (84)
SA	Soothed Aggregation (96)
AMGe	finite element AMG (01)
AMG$\cancel{e}$	element free AMGe (02)
ρAMGe	spectral AMGe (03)

Adaptive Algorithms

AMG	adaptive AMG (84)
BAMG	Bootstrap AMG (01)
αSA	adaptive Soothed Aggregation (04)
CR	Compatible Relaxation (04)
αAMG	adaptive AMG (06)
αAMGr	adaptive AMGr (06)

Spectral AMGe

ρAMGe

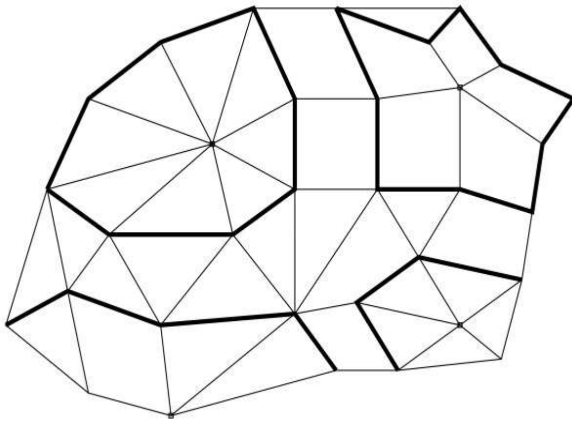
Spectral AMGe (02)

Spectral AMGe

ρAMGe

Spectral AMGe (02)

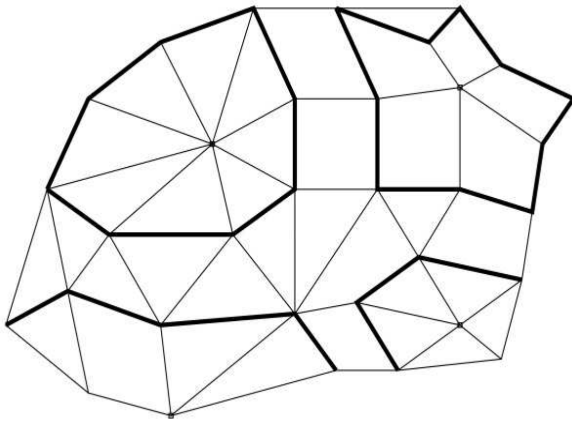
- Based on local stiffness matrices like AMGe
- Aggregates elements like SA
- Creates local columns in interpolation matrix based on local null-space



Spectral AMGe

ρAMGe

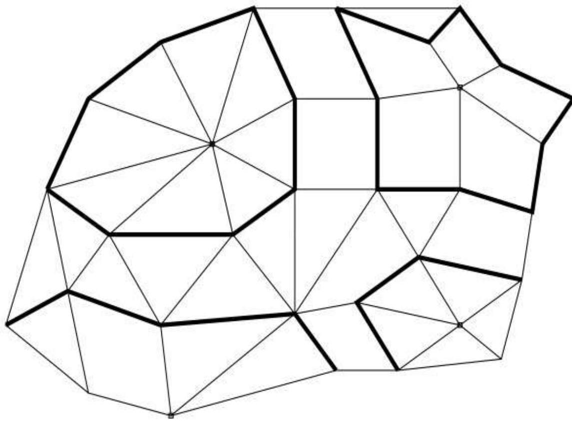
Spectral AMGe (02)



- Based on local stiffness matrices like AMGe
- Aggregates elements like SA
- Creates local columns in interpolation matrix based on local null-space
- Blends rather than smooths columns of P

Spectral AMGe

ρAMGe | Spectral AMGe (02)



- Based on local stiffness matrices like AMGe
- Aggregates elements like SA
- Creates local columns in interpolation matrix based on local null-space
- Blends rather than smooths columns of P
- Effective when global null-space vectors not available, but local stiffness matrices are available

AMG Alphabet Soup

AMG	Classical AMG (84)
SA	Soothed Aggregation (96)
AMGe	finite element AMG (01)
AMG $\not{e}$	element free AMGe (02)
ρAMGe	spectral AMGe (03)

Adaptive Algorithms

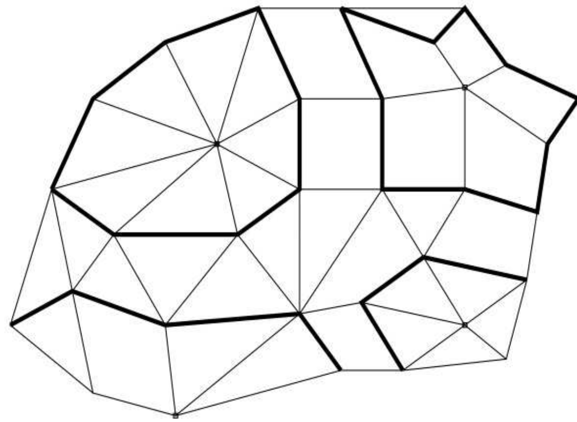
AMG	adaptive AMG (84)
BAMG	Bootstrap AMG (01)
αSA	adaptive Soothed Aggregation (04)
CR	Compatible Relaxation (04)
αAMG	adaptive AMG (06)
αAMGr	adaptive AMGr (06)

Element free AMGe

AMG $\not{e}$ element free AMGe (02)

Element free AMGe

AMG $\not{e}$ element free AMGe (02)

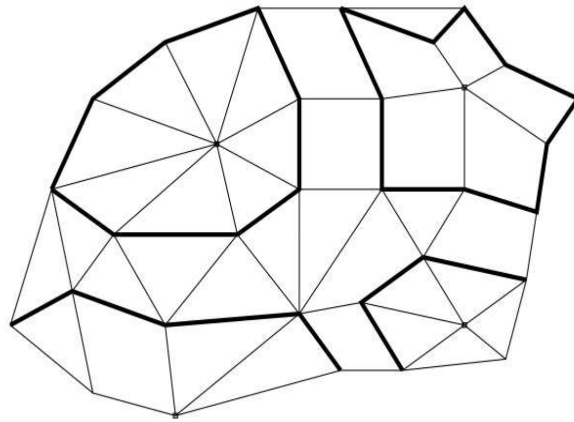


- Based on principles of AMGe
- Aggregates elements like SA
- Creates local stiffness matrices from neighboring elements

Element free AMGe

AMG $\not{e}$ element free AMGe (02)

- Based on principles of AMGe
- Aggregates elements like SA
- Creates local stiffness matrices from neighboring elements



- Effective when local stiffness matrices are not available

AMG Alphabet Soup

AMG	Classical AMG (84)
SA	Soothed Aggregation (96)
AMGe	finite element AMG (01)
AMG$\not e$	element free AMGe (02)
ρAMGe	spectral AMGe (03)

Adaptive Algorithms

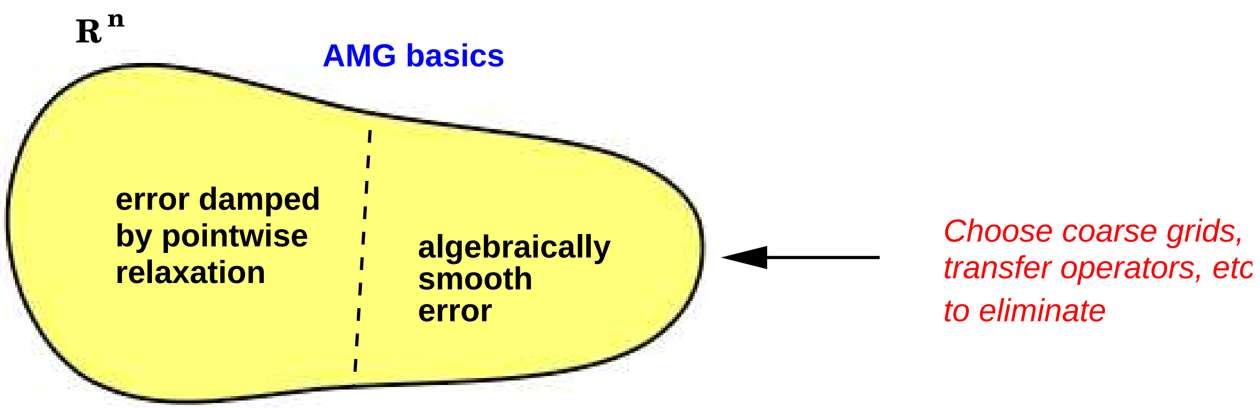
AMG	adaptive AMG (84)
BAMG	Bootstrap AMG (01)
αSA	adaptive Soothed Aggregation (04)
CR	Compatible Relaxation (04)
αAMG	adaptive AMG (06)
αAMGr	adaptive AMGr (06)

Adaptive AMG

- AMG methods employ (relatively) simple relaxation

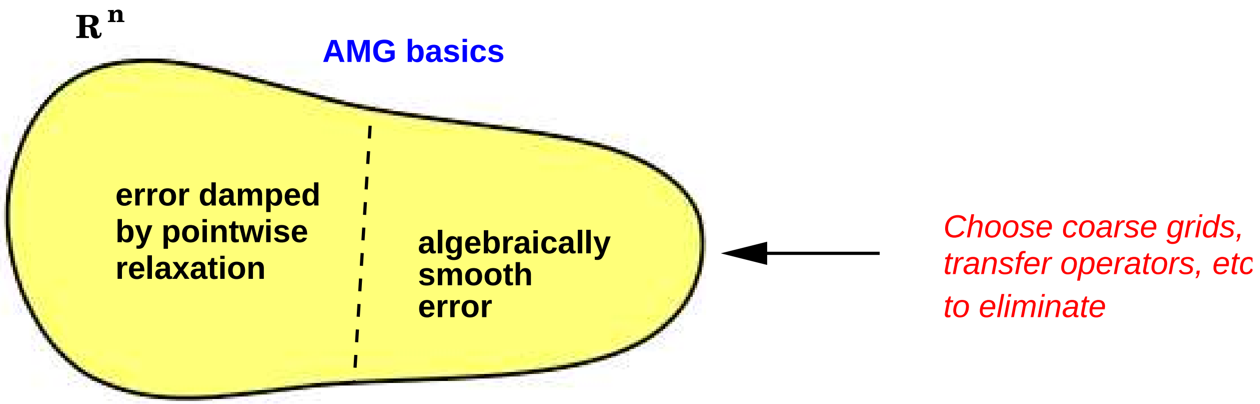
Adaptive AMG

- AMG methods employ (relatively) simple relaxation
- The coarse-grid problem must capture all modes not effectively reduced by relaxation



Adaptive AMG

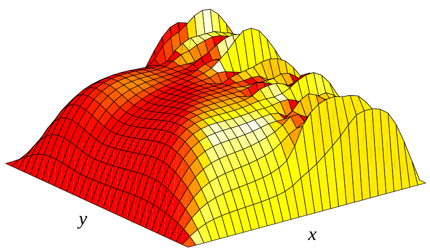
- AMG methods employ (relatively) simple relaxation
- The coarse-grid problem must capture all modes not effectively reduced by relaxation

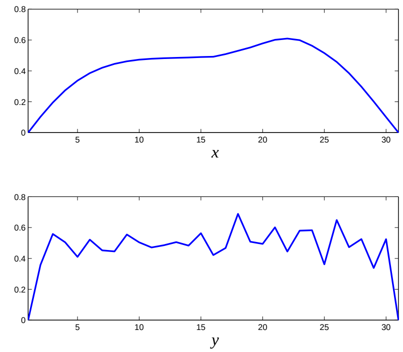


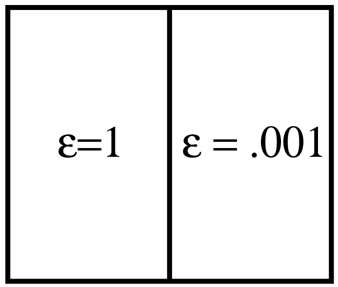
- Algebraically smooth vectors are not necessarily geometrically smooth

Algebraically smooth error can be oscillatory

- Error after seven Gauss/Seidel iterations on $-u_{xx} - \epsilon u_{yy} = f$



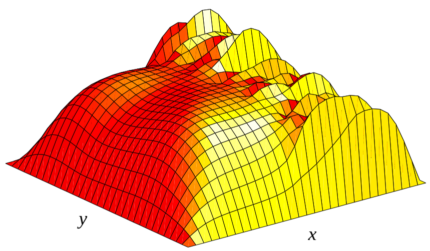


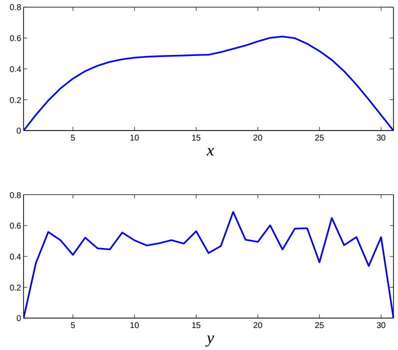


Algebraically smooth error can be oscillatory

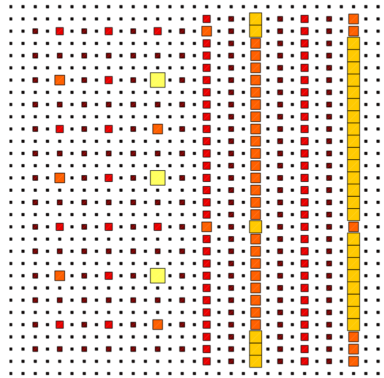
- Error after seven Gauss/Seidel iterations on

  $-u_{xx} - \epsilon u_{yy} = f$





- Adaptive AMG can “follow physics”



Adpative Principles

- Let current method tell you what type of error is not being reduced effectively

Adpative Principles

- Let current method tell you what type of error is not being reduced effectively
- Adjust AMG components to capture this error

Adpative Principles

- Let current method tell you what type of error is not being reduced effectively
- Adjust AMG components to capture this error
- Do no harm: make sure change does not awaken previously reduced errors

Adpative Principles

- Let current method tell you what type of error is not being reduced effectively
- Adjust AMG components to capture this error
- Do no harm: make sure change does not awaken previously reduced errors
- Do as much of the work as possible on the coarser grids

Adpative Principles

- Let current method tell you what type of error is not being reduced effectively
- Adjust AMG components to capture this error
- Do no harm: make sure change does not awaken previously reduced errors
- Do as much of the work as possible on the coarser grids
- Test the current method and modify as necessary

Adaptive Smoothed Aggregation

- Given A, choose simple relaxation, call it the current method, C

Adaptive Smoothed Aggregation

- Given A, choose simple relaxation, call it the current method, C

- Iterate with the current method on $CA\underline{x} = \underline{0}$
  - If it is acceptable, stop

Adaptive Smoothed Aggregation

- Given A, choose simple relaxation, call it the current method, C
- Iterate with the current method on $CA\underline{x} = \underline{0}$
  - If it is acceptable, stop
- Approximate largest eigenvalue/vector of $(I - CA)$
  - Can be accomplished with a multilevel process

Adaptive Smoothed Aggregation

- Given A, choose simple relaxation, call it the current method, C
- Iterate with the current method on $CA\underline{x} = \underline{0}$
  - If it is acceptable, stop
- Approximate largest eigenvalue/vector of $(I - CA)$
  - Can be accomplished with a multilevel process
- Construct new coarse interpolation, P, and coarse-grid operator, A_c

Adaptive Smoothed Aggregation

Current approximation to the Null-space vector: $\underline{v} = (v_1, v_2, \ldots, v_n)^t$

$$\hat{P} = \begin{bmatrix} v_1 & & \\ \vdots & & \\ v_{n_{f_1}} & & \\ & v_{n_2} & \\ & \vdots & \\ & v_{n_{f_2}} & \\ & & v_{n_c} \\ & & \vdots \\ & & v_{n_{f_c}} \end{bmatrix}$$

Normalize

$$\hat{P}^t \hat{P} = I$$

Smooth $\hat{P}$

$$P = (I - \alpha A)\hat{P}$$

Coarse-grid operator

$$A_c = P^t A P$$

Recurse

Adaptive Smoothed Aggregation

- Recursively construct V-cycle, call it the current method, C
  - Don't come back until your finished!

Adaptive Smoothed Aggregation

- Recursively construct V-cycle, call it the current method, C
  - Don't come back until your finished!

- Iterate with the current method on $CA\underline{x} = \underline{0}$
  - If acceptable, stop
  - Better approximation to null-space $\underline{v}_1$
  - Add new column to each aggregate $\underline{v}_1, \underline{v}_2$

Adaptive Smoothed Aggregation

- Recursively construct V-cycle, call it the current method, C
  - Don't come back until your finished!
- Iterate with the current method on $CA\underline{x} = \underline{0}$
  - If acceptable, stop
  - Better approximation to null-space $\underline{v}_1$
  - Add new column to each aggregate $\underline{v}_1, \underline{v}_2$
- Recurse

Adaptive AMG

Adaptive Flavors

AMG	adaptive AMG (84)
BAMG	Bootstrap AMG (01)
αSA	adaptive Soothed Aggregation (04)
CR	Compatible Relaxation (04)
αAMG	adaptive AMG (06)
αAMGr	adaptive AMGr (06)

Adaptive AMG

Adaptive Flavors

AMG	adaptive AMG (84)
BAMG	Bootstrap AMG (01)
αSA	adaptive Soothed Aggregation (04)
CR	Compatible Relaxation (04)
αAMG	adaptive AMG (06)
αAMGr	adaptive AMGr (06)

All depend on determining a local representation of algebraically smooth vectors

αAMG and αAMGr

Perfect Interpolation

$$A = \begin{bmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{bmatrix} \qquad P = \begin{bmatrix} -A_{ff}^{-1} A_{fc} \\ I \end{bmatrix}$$

αAMG and αAMGr

Perfect Interpolation

$$A = \begin{bmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{bmatrix} \qquad P = \begin{bmatrix} -A_{ff}^{-1} A_{fc} \\ I \end{bmatrix}$$

Choose diagonal matrix Δ_{ff}

$$\Delta_{ff} A_{fc} \underline{v}_1 = A_{ff}^{-1} A_{fc} \underline{v}_1$$

αAMG and αAMGr

Perfect Interpolation

$$A = \begin{bmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{bmatrix} \qquad P = \begin{bmatrix} -A_{ff}^{-1} A_{fc} \\ I \end{bmatrix}$$

Choose diagonal matrix Δ_{ff}

$$\Delta_{ff} A_{fc} \underline{v}_1 = A_{ff}^{-1} A_{fc} \underline{v}_1$$

- Adaptive approximation to smallest eigenvalue/vector(s), $\underline{v}_1$

Compatible Relaxation

CR

Livne(04), Brannick(05)

$$A = \begin{bmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{bmatrix} \qquad P = \begin{bmatrix} -A_{ff}^{-1} A_{fc} \\ I \end{bmatrix}$$

- Principle: Coarse grid is adequate if A_{ff} is well conditioned

Compatible Relaxation

CR

Livne(04), Brannick(05)

$$A = \begin{bmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{bmatrix} \qquad P = \begin{bmatrix} -A_{ff}^{-1} A_{fc} \\ I \end{bmatrix}$$

- Principle: Coarse grid is adequate if A_{ff} is well conditioned
- Use simple relaxation on A_{ff}, together with a greedy independent set algorithm, to choose coarse grid

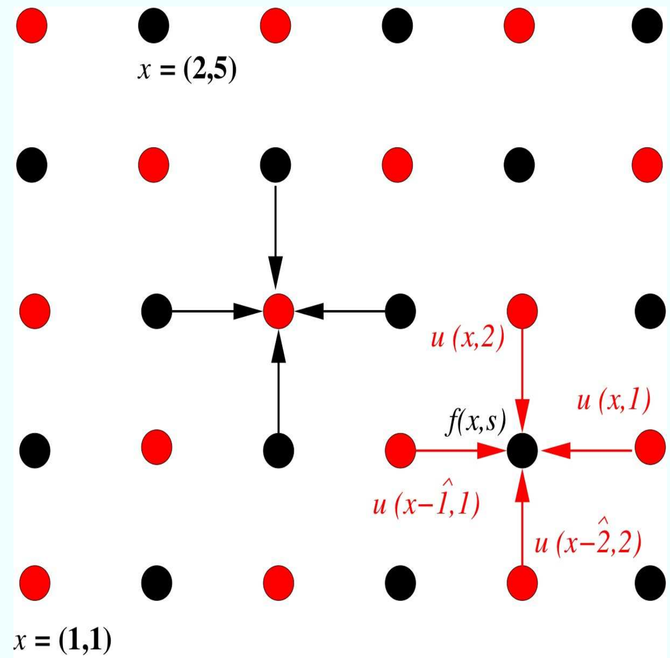
Adaptive Algebraic Multigrid

αAMG and αSA surprisingly effective on a wide range of problems

- Highly irregular meshes
- Strongly anisotropic
- Adaptively refined meshes
- Discontinuous coefficients (heterogeneous material)
- Singularities
- Hyperbolic problems
- QCD

Adaptive AMG for Lattice QCD

- Quantum Chromodynamics (QCD) calculations involve huge linear systems and large-scale (petascale) computing
- Requires solving the complex and non-hermitian discretized Dirac operator
- Each equation may be solved 1000s times

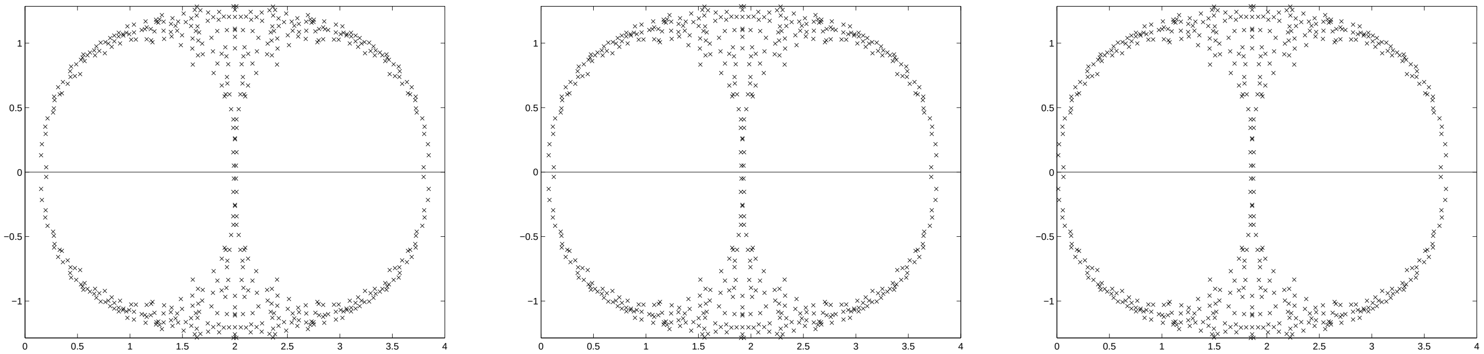


$$M(\mathcal{U}) = D(\mathcal{U}) - m_0 I$$

$$= \begin{bmatrix} A - m_0 I & B \\ -B* & A - m_0 I \end{bmatrix}$$

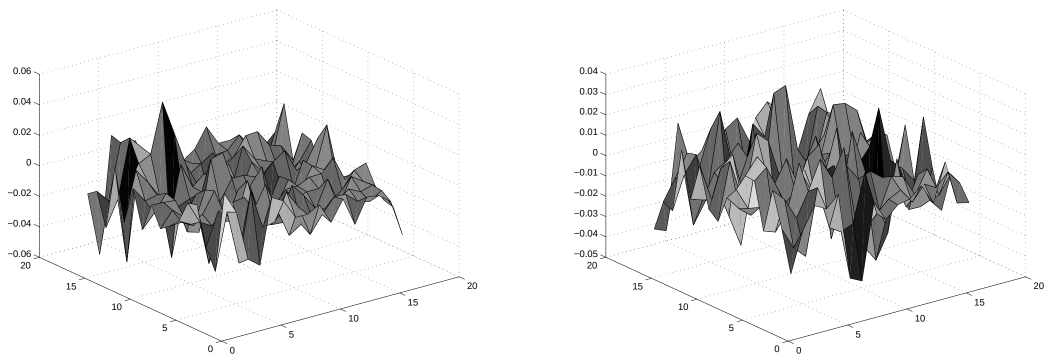
QCD: 2D Schwinger Model

- The system becomes extremely ill-conditioned for typical choices of m_0



$m_0 = 0$ $\longrightarrow$ $m_0 \approx m_{cr}$

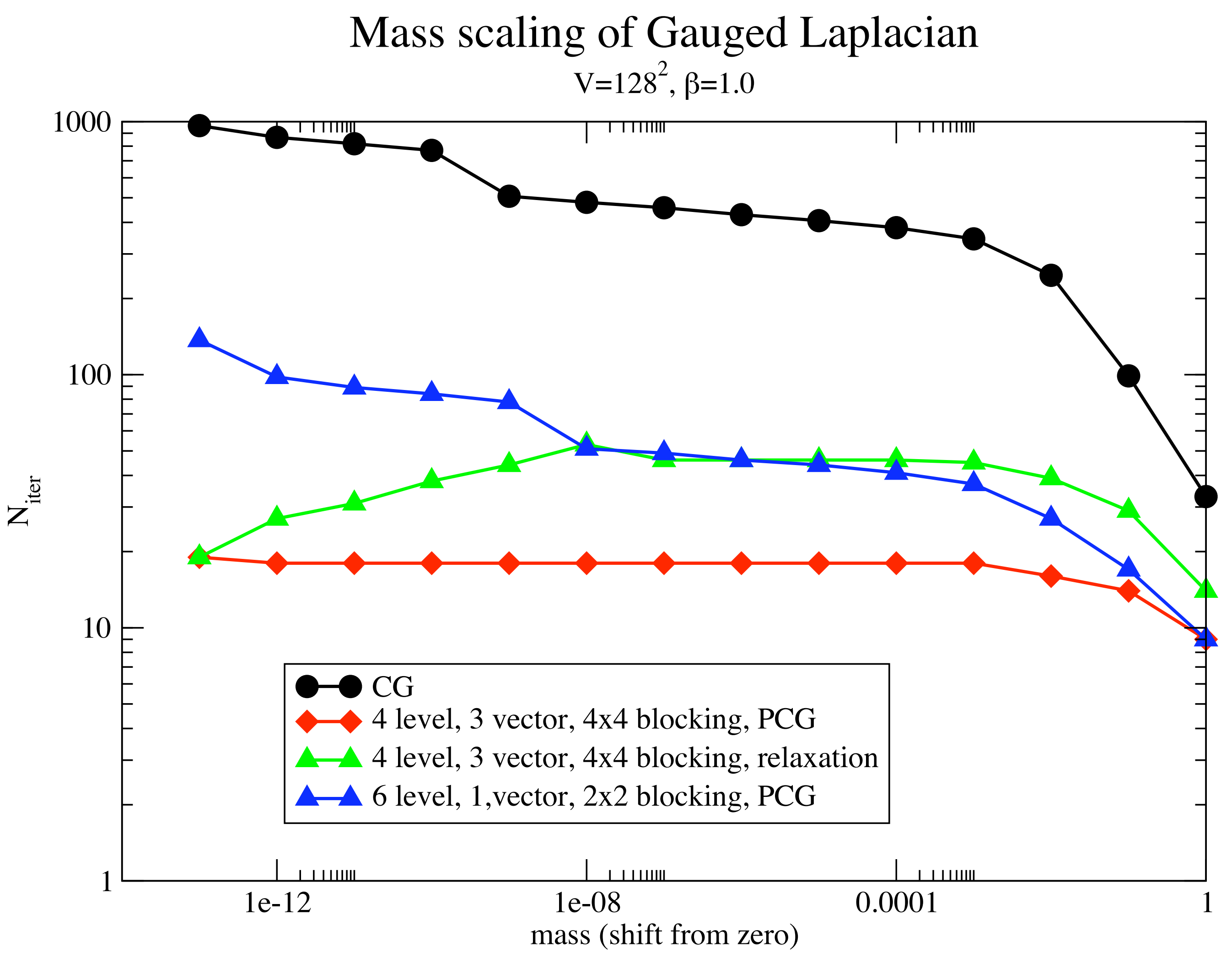
- Near null space is unknown and oscillatory



2D Schwinger Model

- Form the normal equations and apply αSA
- Set-up requires 100s of Work Units
  - (WU = matrix vector multiply)
- Interpolation requires 8 – 10 columns on each aggregate
- For small mass shift, faster than the current method (Diagonally scaled PCG) on even one right-hand side

QCD


Mass scaling of Gauged Laplacian
V=128^2, β=1.0
1000
100
10
1
N_iter
1e-12
1e-08
0.0001
1
mass (shift from zero)
CG
4 level, 3 vector, 4x4 blocking, PCG
4 level, 3 vector, 4x4 blocking, relaxation
6 level, 1,vector, 2x2 blocking, PCG

Real Problem

- Real Problem: 4D model – preliminary results promising

Real Problem

- Real Problem: 4D model – preliminary results promising
- Real Real Problem: Dirac Equations

Real Problem

- Real Problem: 4D model – preliminary results promising
- Real Real Problem: Dirac Equations

αSA allows the QCD community to do problems that they could not do before

Conclusions

- Linear systems from PDEs require multilevel algorithms
- GMG optimal for structured grids
- AMG/SA effective for unstructured grids, known (near) null-space
- αAMG/SA greatly expand the domain of applicability

Conclusions

- Linear systems from PDEs require multilevel algorithms
- GMG optimal for structured grids
- AMG/SA effective for unstructured grids, known (near) null-space
- αAMG/SA greatly expand the domain of applicability
- Adaptive AMG/SA a group effort

AMG Gang

Ball State	I. Livshits	Delft	S. MacLachlan
Boulder	M. Brezina	Davidson College	T. Chartier
	T. Manteuffel	FIT	J. Jones
	S. McCormick	Penn State	J. Brannick
	J. Ruge		J. Xu
	G. Sanders		L. Zikatanov
	B. Sheehan	SNL	J. Hu
LLNL	A. Baker		R. Tuminaro
	A. Cleary	Urbana-Champaign	D. Alber
	R. Falgout		L. Olson
	V. Henson	Utah	O. Livne
	T. Kolev	Weizmann Institute	A. Brandt
	B. Lee		
	P. Vassilevski		
	U. Yang		