

An



Multigrid Tutorial

Van Emden Henson

*Center for Applied Scientific Computing
Lawrence Livermore National Laboratory*

vhenson@llnl.gov

<http://www.casc.gov/CASC/people/henson>

April 10, 1999



AMG: What is Algebraic Multigrid??

- Any multilevel method where geometry is not used (and may not be available) to build coarse grids, interpolation and restriction, or coarse-grid operators.
- “Classical” AMG was introduced by Brandt, McCormick and Ruge in 1982. It was explored early on by Stueben in 1983, and popularized by Ruge and Stuben in 1987.
- This tutorial will describe only the classical AMG algorithm.

AMG: What is Algebraic Multigrid??

- Many other algorithms qualify under the definition given. Some whose approaches are closely related to “classical AMG”:
 - Chang
 - Griebel, Neunhoeffer, Regler
 - Huang
 - Krechel, Stueben
 - Zaslavsky
- Work close to the original, but using different approaches to coarsening or interpolation:
 - Fuhrmann
 - Kickinger

AMG: What is Algebraic Multigrid??

- Other approaches that are important, novel, historical, or weird:
 - Multigraph methods (Bank & Smith)
 - Aggregation methods (Braess; Chan & Zikatanov & Xu)
 - Smoothed Aggregation methods (Mandel & Brezina & Vanek)
 - Black Box Multigrid (Dendy, Dendy & Bandy)
 - Algebraic Multilevel Recursive Solver (Saad)
 - Element based algebraic multigrid (Chartier; Cleary et al)
 - MultiCoarse correction with Suboptimal Operators (Sokol)
 - Multilevel block ILU methods (Jang & Saad; Bank & Smith & Wagner; Reusken)
 - AMG based on Element Agglomeration (Jones & Vassilevski)
 - Sparse Approximate Inverse Smoothers (Tang & Wan)
 - Algebraic Schur-Complement approaches (Axelsson & Vassilevski & Neytcheva)

Highlights of Multigrid: The 1-d Model Problem

- Poisson's equation: $-\Delta u = f$ in $[0,1]$, with boundary conditions $u(0) = u(1) = 0$.
- Discretized as:
$$\frac{-u_{i-1} + 2u_i - u_{i+1}}{h^2} = f_i \quad u_0 = u_N = 0$$
- Leads to the Matrix equation $Au = f$, where
$$A = \frac{1}{h^2} \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}, \quad u = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{N-2} \\ u_{N-1} \end{pmatrix}, \quad f = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_{N-2} \\ f_{N-1} \end{pmatrix}$$

Highlights of Multigrid: Weighted Jacobi Relaxation

- Consider the iteration:

$$u_i^{(new)} \leftarrow (1 - \omega) u_i^{(old)} + \frac{\omega}{2h^2} (u_{i-1}^{(old)} + u_{i+1}^{(old)} + f_i)$$

- Letting $A = D + L + U$, the matrix form is:

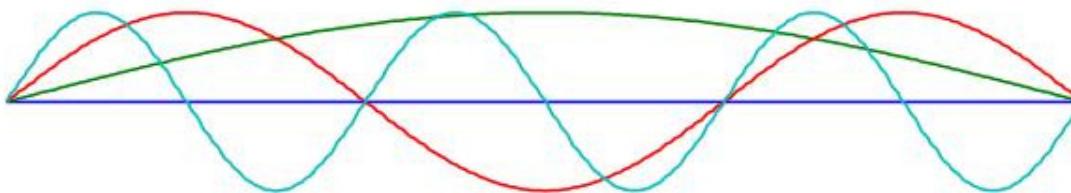
$$\begin{aligned} u^{(new)} &= \left[(1 - \omega)I - \omega D^{-1}(L + U) \right] u^{(old)} + \omega D^{-1}f \\ &= G_\omega u^{(old)} + \omega D^{-1}f \end{aligned}$$

- It is easy to see that if $e \equiv u^{(exact)} - u^{(approx)}$, then

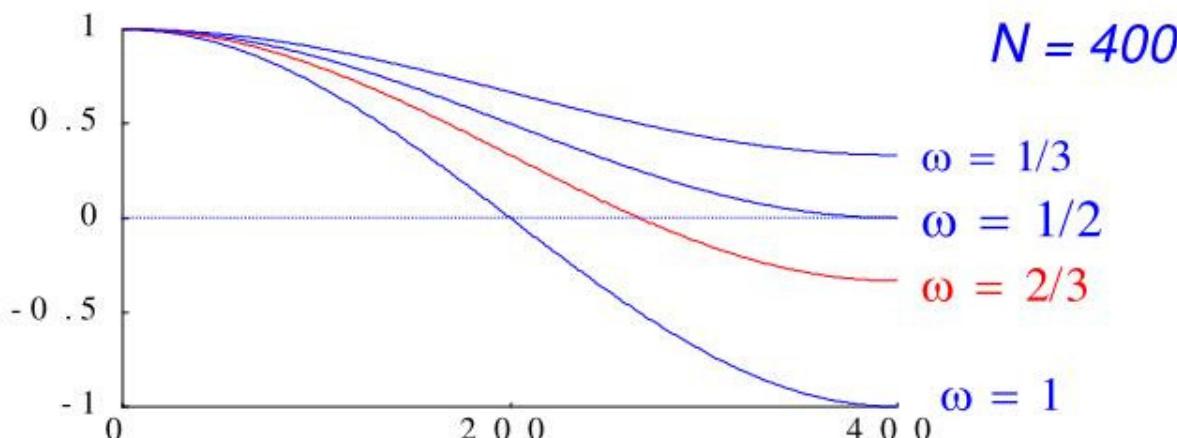
$$e^{(new)} = G_\omega e^{(old)}$$

Highlights of Multigrid: Relaxation Typically Stalls

- The eigenvectors of G_ω are the same as those of A , and are Fourier Modes: $v_i = \sin(ik\pi/N)$, $k = 1, 2, \dots, N-1$



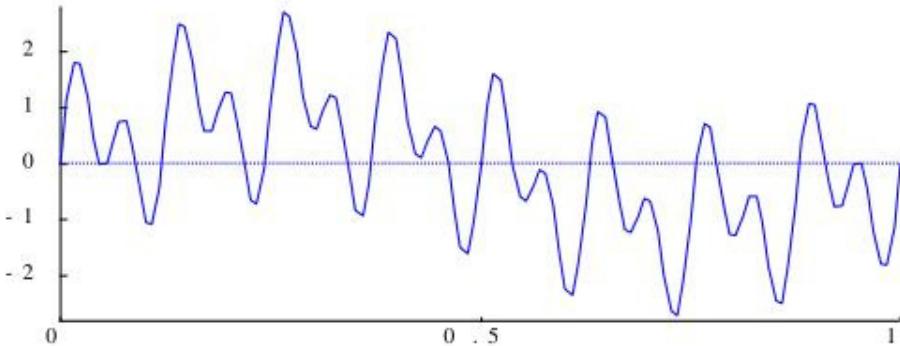
- The eigenvalues of G_ω are $1 - 2\omega \sin^2(k\pi/2N)$, so the effect of relaxation on the modes is:



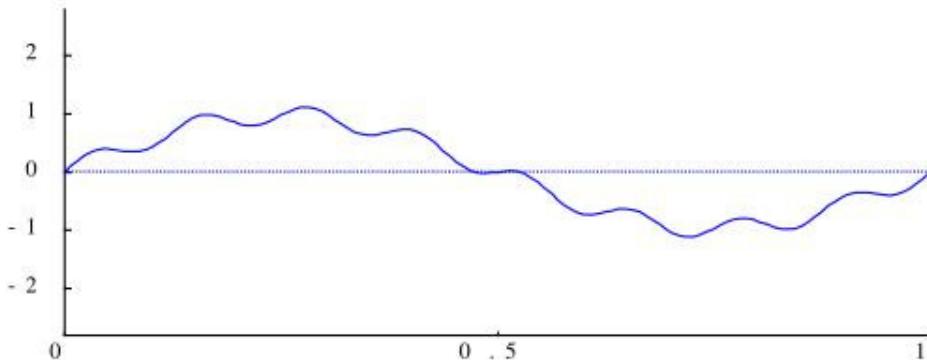
Note: No value of ω will damp out the low frequency waves

Highlights of Multigrid: Relaxation Smooths the Error

- Initial error.



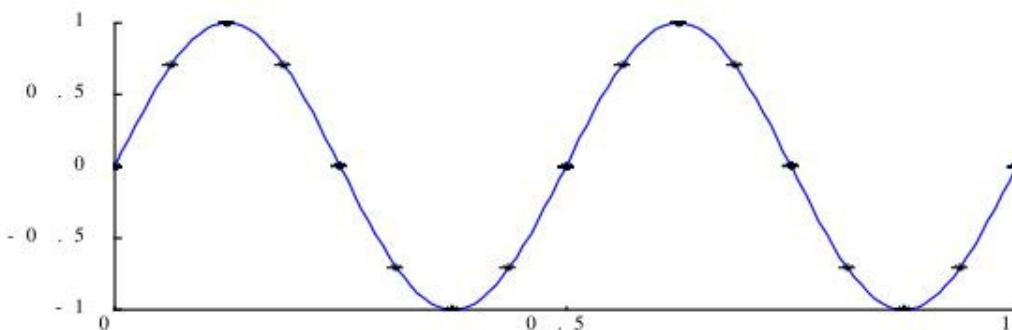
- Error after 35 iteration sweeps:



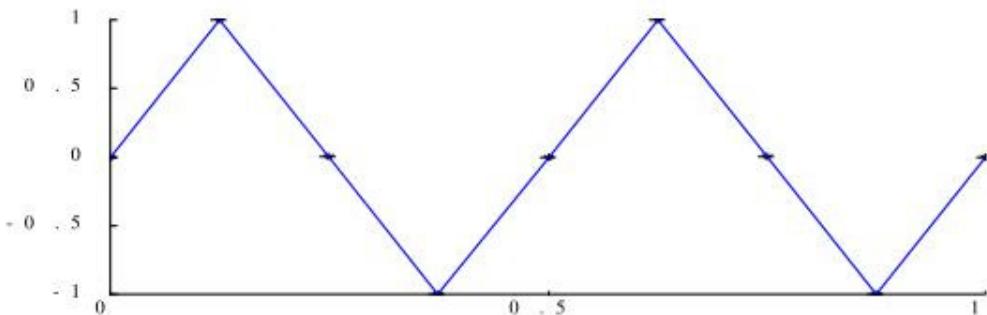
Many relaxation schemes have the smoothing property, where oscillatory modes of the error are eliminated effectively, but smooth modes are damped very slowly.

Highlights of Multigrid: Smooth error can be represented on a coarse grid

- A smooth function:



- Can be represented by linear interpolation from a coarser grid:



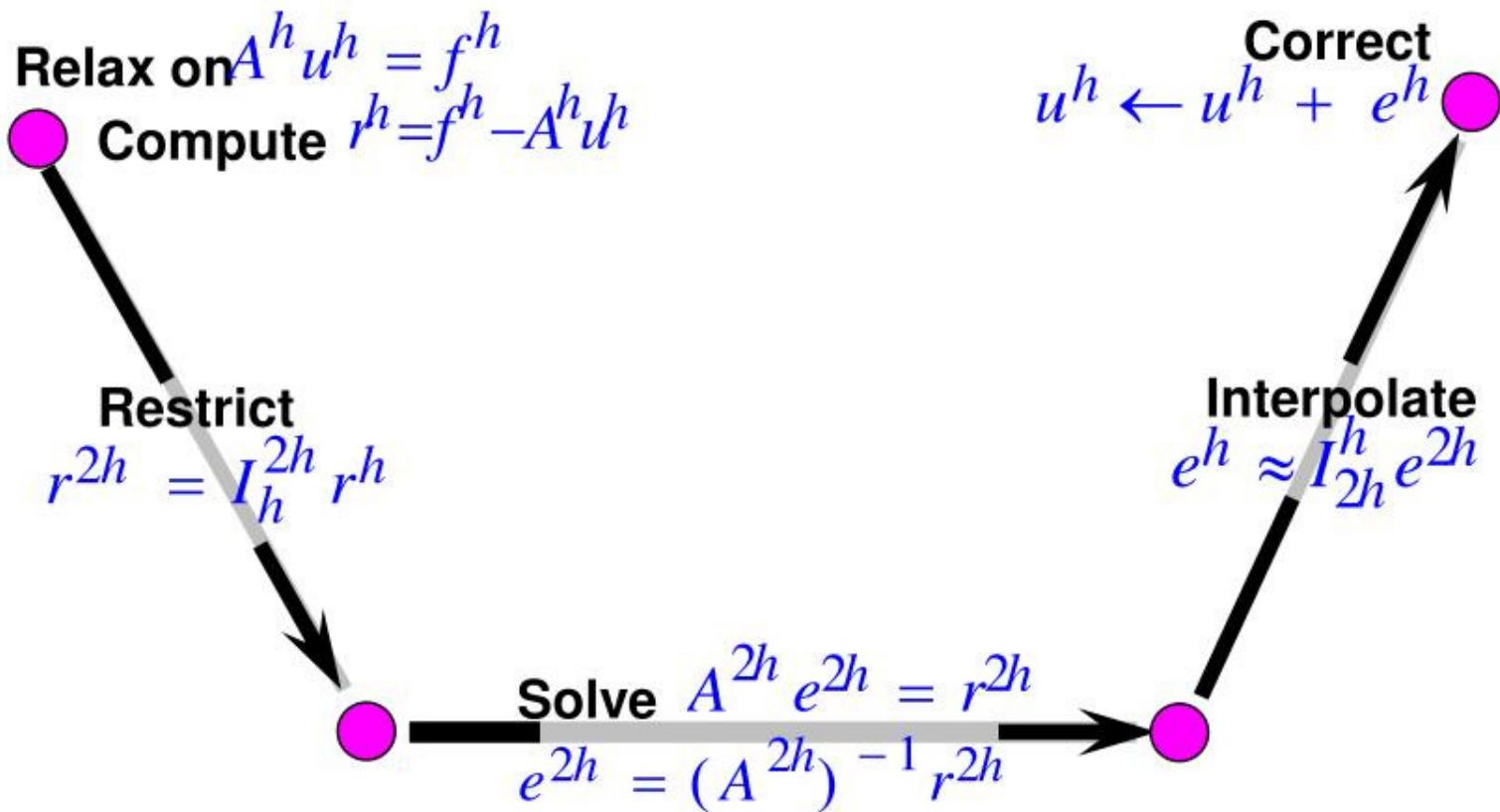
On the coarse grid, the smooth error appears to be relatively higher in frequency: in the example it is the 4-mode, out of a possible 16, on the fine grid, $1/4$ the way up the spectrum. On the coarse grid, it is the 4-mode out of a possible 8, hence it is $1/2$ the way up the spectrum.

Relaxation will be more effective on this mode if done on the coarser grid!!

Highlights of Multigrid: Coarse-grid Correction

- Perform relaxation on $A^h u^h = f^h$ on fine grid until error is smooth.
- Compute residual, $r^h = f^h - A^h u^h$ and transfer to the coarse grid $r^{2h} = I_h^{2h} r^h$.
- Solve the coarse-grid residual equation to obtain the error: $A^{2h} e^{2h} = r^{2h}, \quad \therefore e^{2h} = (A^{2h})^{-1} r^{2h}$
- Interpolate the error to the fine grid and correct the fine-grid solution: $u^h \leftarrow u^h + I_{2h}^h e^{2h}$

Highlights of Multigrid: Coarse-grid Correction



Highlights of Multigrid: Tools Needed

- **Interpolation and restriction operators:**

$$I_{2h}^h = \begin{pmatrix} 0.5 \\ 1.0 \\ 0.5 & 0.5 \\ 1.0 & 1.0 \\ 0.5 & 0.5 \\ 1.0 \\ 0.5 \end{pmatrix}, \quad I_h^{2h} = \begin{pmatrix} 0 & 1 & 0 \\ & 0 & 1 & 0 \\ & & 0 & 1 & 0 \end{pmatrix}, \quad I_h^{2h} = \begin{pmatrix} 0.25 & 1.0 & 0.25 \\ & 0.25 & 1.0 & 0.25 \\ & & 0.25 & 1.0 & 0.25 \end{pmatrix}$$

Linear

Injection

Full-weighting

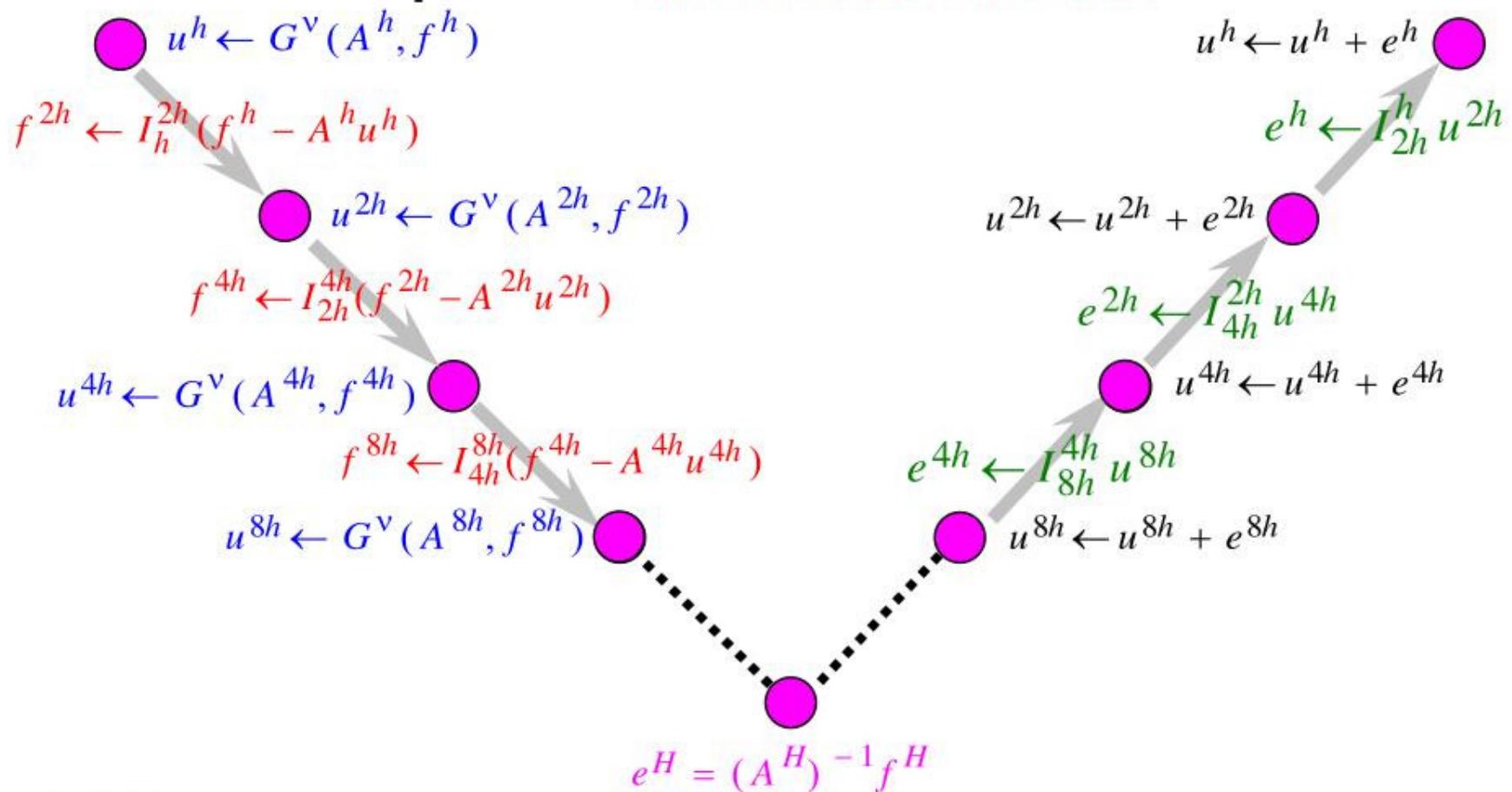
Interpolation

- **Coarse-grid Operator A^{2h} .** Two methods:
 - (1) Discretize equation at larger spacing
 - (2) Use Galerkin Formula:

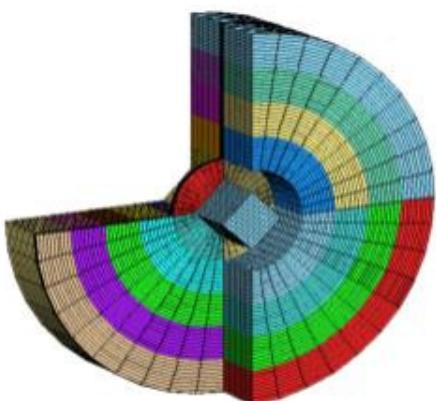
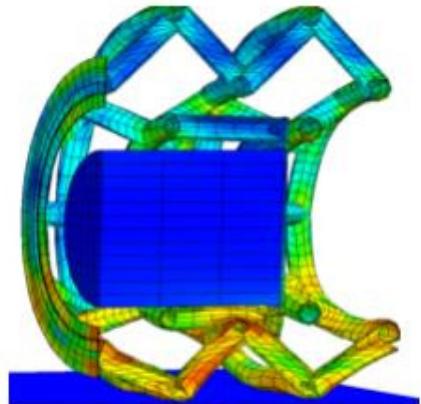
$$A^{2h} = I_h^{2h} A^h I_{2h}^h$$

Highlights of Multigrid: Recursion: the (∇ , 0) V-cycle

- Major question: How do we “solve” the coarse-grid residual equation? Answer: *recursion!*



Algebraic multigrid: for unstructured-grids



- Automatically defines coarse “grid”
 - AMG has two distinct phases:
 - setup phase: define MG components
 - solution phase: perform MG cycles
 - AMG approach is opposite of geometric MG
 - fix relaxation (point Gauss-Seidel)
 - choose coarse “grids” and prolongation, P , so that error not reduced by relaxation is in $\text{range}(P)$
 - define other MG components so that coarse-grid correction eliminates error in $\text{range}(P)$ (i.e., use Galerkin principle)
- (in contrast, geometric MG fixes coarse grids, then defines suitable operators and smoothers)

AMG has two phases:

- **Setup Phase**
 - Select Coarse “grids,” $\Omega^{m+1}, m = 1, 2, \dots$
 - Define interpolation, $I_{m+1}^m, m = 1, 2, \dots$
 - Define restriction and coarse-grid operators
$$I_m^{m+1} = (I_{m+1}^m)^T \quad A^{m+1} = I_m^{m+1} A^m I_{m+1}^m$$
- **Solve Phase**
 - Standard multigrid operations, e.g., V-cycle, W-cycle, FMG, etc

In AMG, we choose relaxation first:

- Typically, pointwise Gauss-Seidel is used

$$A = (D + L + U)$$

- The iteration is developed:

$$Ax = b$$

$$(D + L)x = b - Ux$$

$$x^{new} = (D + L)^{-1}b - (D + L)^{-1}Ux^{old}$$

- Add and subtract $(D + L)^{-1}(D + L)x^{old}$ to get:

$$x^{new} = x^{old} + (D + L)^{-1}r^{old}$$

Gauss-Seidel relaxation error propagation:

- The iteration is:

$$x^{new} = x^{old} + (D+L)^{-1} r^{old}$$

- Subtracting both sides from the exact solution:

$$x^{exact} - x^{new} = x^{exact} - (x^{old} + (D+L)^{-1} r^{old})$$

$$e^{new} = e^{old} - (D+L)^{-1} r^{old}$$

- Using $r = A e$ this can be written as:

$$e^{new} = [I - (D+L)^{-1} A] e^{old}$$

An observation: error that is slow to converge \Rightarrow “small” residuals

- Consider the iterative method error recurrence

$$e^{k+1} = (I - Q^{-1}A) e^k$$

- Error that is slow to converge satisfies

$$\begin{aligned}(I - Q^{-1}A) e \approx e &\Rightarrow Q^{-1}A e \approx 0 \\ &\Rightarrow r \approx 0\end{aligned}$$

- Perhaps a better viewpoint is

$$(I - Q^{-1}A) e \approx e \Rightarrow \langle Q^{-1}A e, Ae \rangle \ll \langle e, Ae \rangle$$

Some implications of slow convergence

- For most iterations (e.g., Jacobi or Gauss-Seidel) this last holds if $\langle D^{-1}Ae, Ae \rangle \ll \langle e, Ae \rangle$. (1)
- Hence $\sum_{i=1}^N \frac{r_i^2}{a_{ii}} \ll \sum_{i=1}^N r_i e_i$ implying that, on average,
$$|r_i| \ll a_{ii} |e_i|$$
- An implication is that, if e is an error slow to converge, then locally at least, e_i can be well-approximated by an average of its neighbors:

In Multigrid, error that is slow to converge is geometrically smooth

- Combining the algebraic property that slow convergence implies “small residuals” with the observation above, in AMG we DEFINE smooth error:
- Smooth error is that error which is slow to converge under relaxation, that is,

$$(I - Q^{-1}A) e \approx e$$

or, more precisely,

$$\|(I - Q^{-1}A) e\|_A \approx \|e\|_A$$

But sometimes, smooth error **isn't!** (example from Klaus Stueben)

- Consider the problem

$$-(au_x)_x - (bu_y)_y + cu_{xy} = f(x, y)$$

- on the unit square, using a regular Cartesian grid, with finite difference stencils and values for **a, b, and c**:

$a=1$ $b=1000$ $c=0$	$A=1$ $b=1$ $c=2$
$a=1$ $b=1$ $c=0$	$a=1000$ $b=1$ $c=0$

$$u_{xx} = h^{-2} \begin{bmatrix} 1 & -2 & 1 \end{bmatrix}$$

$$u_{yy} = \frac{1}{h^2} \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

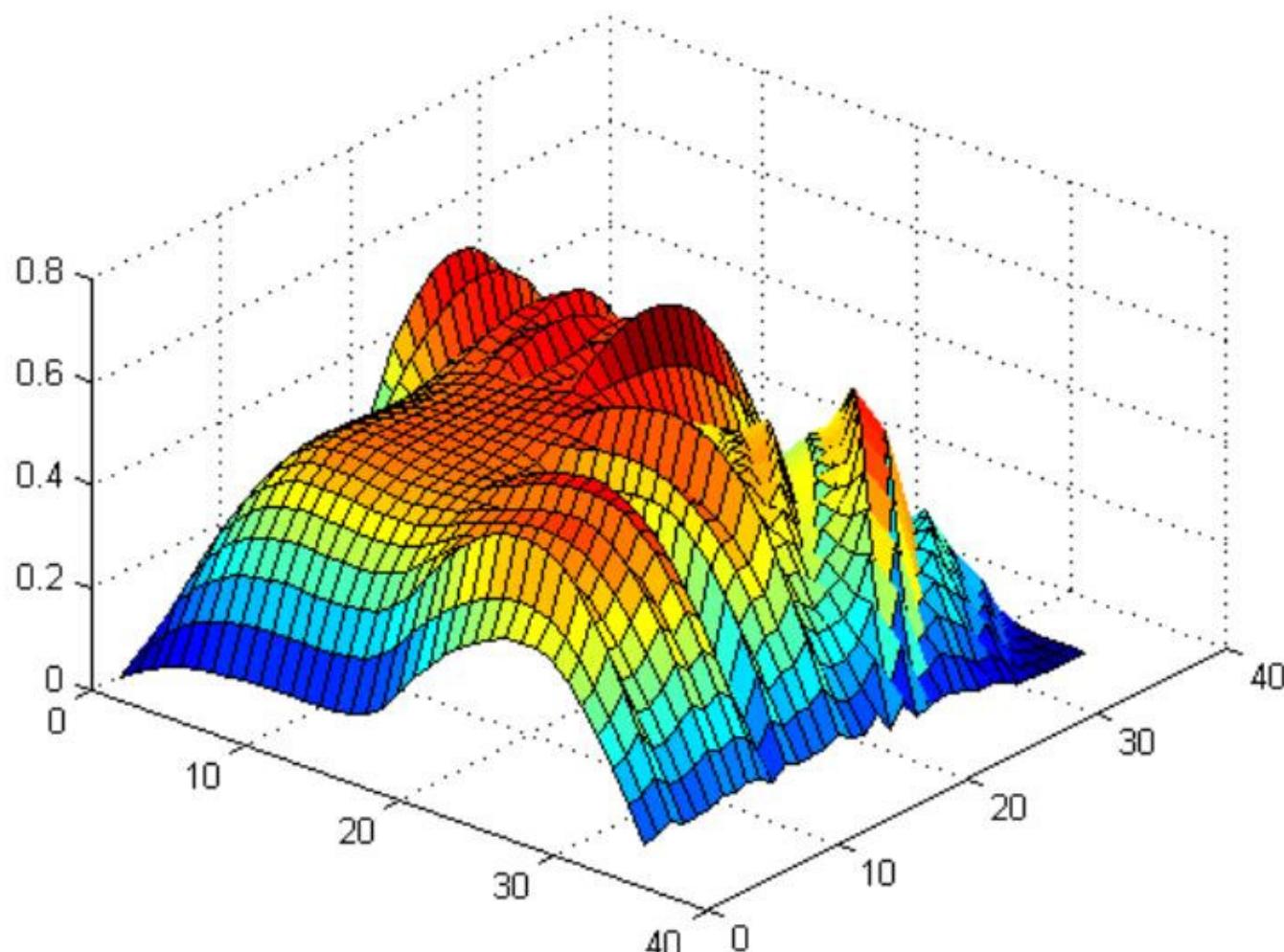
$$u_{xy} = \frac{1}{2h^2} \begin{bmatrix} -1 & 1 \\ 1 & -2 & 1 \\ 1 & 1 & -1 \end{bmatrix}$$

But sometimes, smooth error isn't!

$$-(a u_x)_x - (b u_y)_y + c u_{xy} = f(x, y)$$

- Using a zero right-hand side and a random initial guess, after 8 sweeps of Gauss-Seidel iteration the error is unchanging in norm. By our definition, the error is smooth. And it looks like this:

Smooth error for

$$-(a u_x)_x - (b u_y)_y + c u_{xy} = f(x, y)$$


AMG uses dependence (influence) to determine MG components

- We need to choose a subset of the gridpoints (coarse grid) that can be used 1) to represent smooth errors, and 2) to interpolate these errors to the fine grid.
- Intuitively, a point u_j is a good candidate for a C-point if its value is important in determining the value of another point, u_i in the *i*th equation.
- If the a_{ij} coefficient is “large” compared to the other off-diagonal coefficients in the *i*th equation then u_j influences u_i (or u_i depends on u_j).

Dependence and smooth error

- For M-matrices, we define "*i depends on j*" by

$$-a_{ij} \geq \theta \max_{k \neq i} \{-a_{ik}\}, \quad 0 < \theta \leq 1$$

alternatively, "*j influences i.*"

- It is easy to show from (1) that smooth error satisfies $\langle Ae, e \rangle \ll \langle De, e \rangle$ (2)

Dependence and smooth error

- For M-matrices, we have from (2)

$$\frac{1}{2} \sum_{i \neq j} \left(\frac{-a_{ij}}{2a_{ii}} \right) \left(\frac{e_i - e_j}{e_i} \right)^2 \ll 1$$

- If e_i does **not** depend on e_j then the inequality may be satisfied because a_{ij} is “small”.
 - If e_i does depend on e_j , then a_{ij} need not be small, and the inequality must be satisfied by
- This implies that **smooth error varies slowly in the direction of dependence.**

Some useful definitions

- The set of **dependencies** of a variable u_i , that is, the variables upon whose values the value of u_i depends, is defined as

$$S_i = \left\{ j : -a_{ij} > \max_{k \neq i} \{ -a_{ik} \} \right\}$$

- The set of points that u_i influences is denoted:

$$S_i^T \equiv \{ j : i \in S_j \}$$

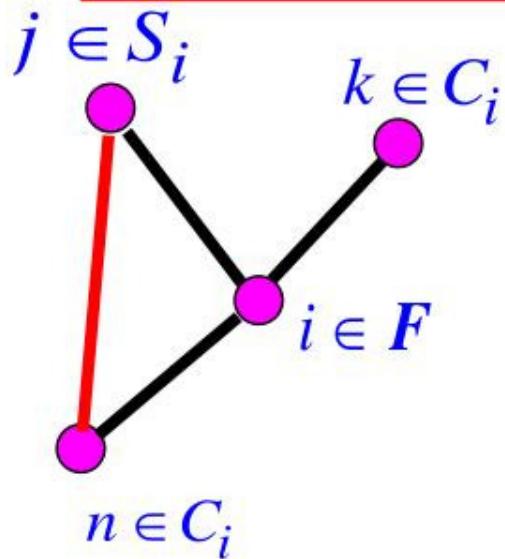
More useful definitions

- The set of coarse-grid variables is denoted C .
- The set of fine-grid variables is denoted F .
- The set of coarse-grid variables used to interpolate the value of the fine-grid variable u_i , called the *coarse interpolatory set for i* , is denoted C_i .

Two Criteria for Choosing the Coarse Grid Points

- First Criterion: F - F dependence

— (C1) For each $i \in F$, each point $j \in S_i$ should either be in C itself or should depend on at least one point in C_i .



Since the value of u_i depends on the value of u_j , the value of u_j must be represented on the coarse-grid for good interpolation. If j isn't a C -point, it should depend on a point in C_i so its value is "represented" in the interpolation.

Two Criteria for Choosing the Coarse Grid Points

- Second Criterion: Maximal Subset

- (C2) C should be a maximal subset with the property that no C -point depends on another.
- (C1) tends to increase the number of C -points. In general, the more C -points on Ω^H , the better the h-level convergence.
- But more C -points means more work for relaxation and interpolation.
- (C2) is designed to limit the size (and work) of the coarse grid.

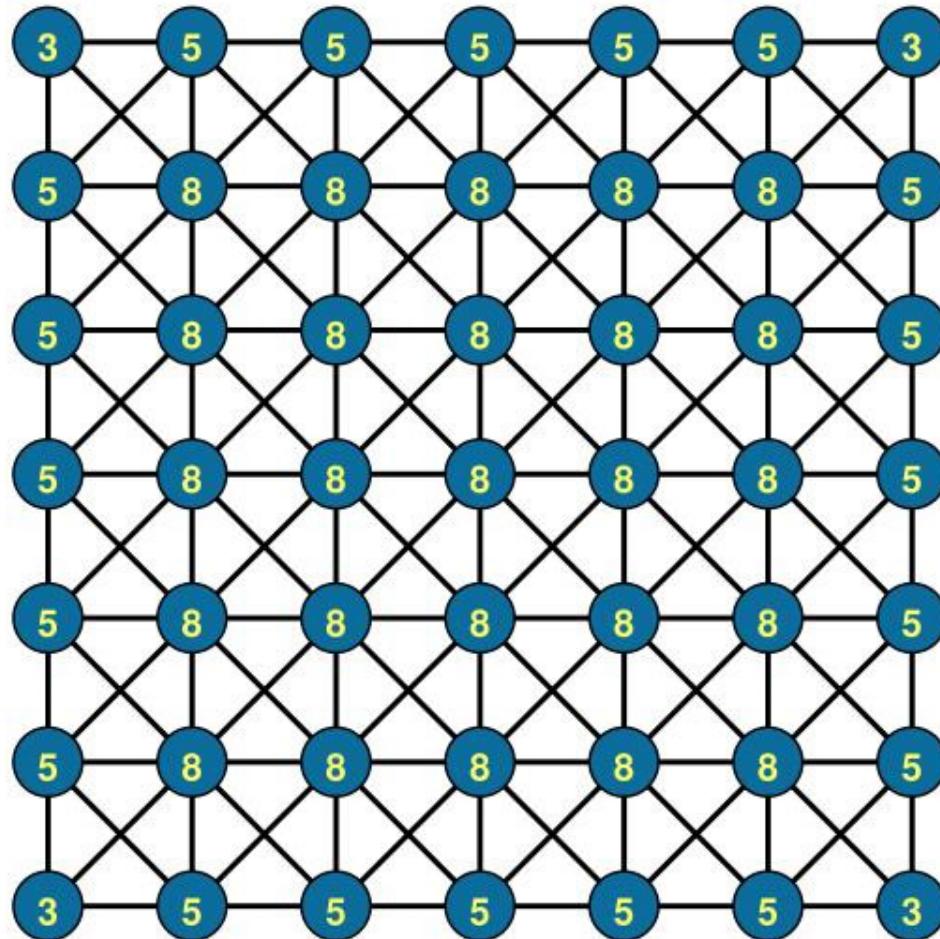
Two Criteria for Choosing the Coarse Grid Points

- It is sometimes not possible to satisfy both criteria simultaneously (an example will be seen shortly).
- In those cases, we choose to satisfy **(C1)**, the requirement that F-F dependencies be represented in the coarse-interpolatory set, while using **(C2)** as a guide.
- This choice leads to somewhat larger coarse grids, but tends to preserve good convergence properties.

Choosing the Coarse Grid Points

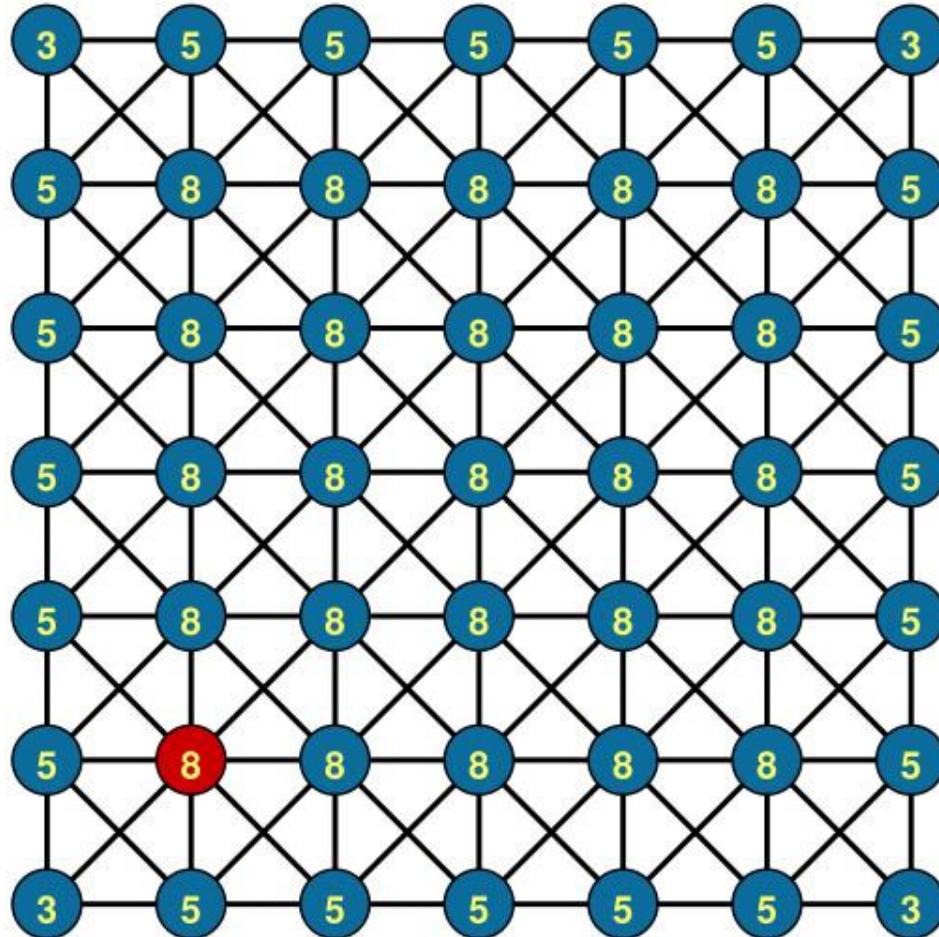
- Assign to each gridpoint k a “value” equal to the number of points that depend on k .
- Choose the first point with global maximum value as a C -point.
- The new C -point can be used to interpolate values of points it influences. Assign them all as F -points.
- Other points influencing these new F -points can be used in their interpolation. Increment their value.
- Repeat until all points are C - or F -points.

Ruge AMG: start



- select C-pt with maximal measure
- select neighbors as F-pts
- update measures of F-pt neighbors

Ruge AMG: select C-pt 1

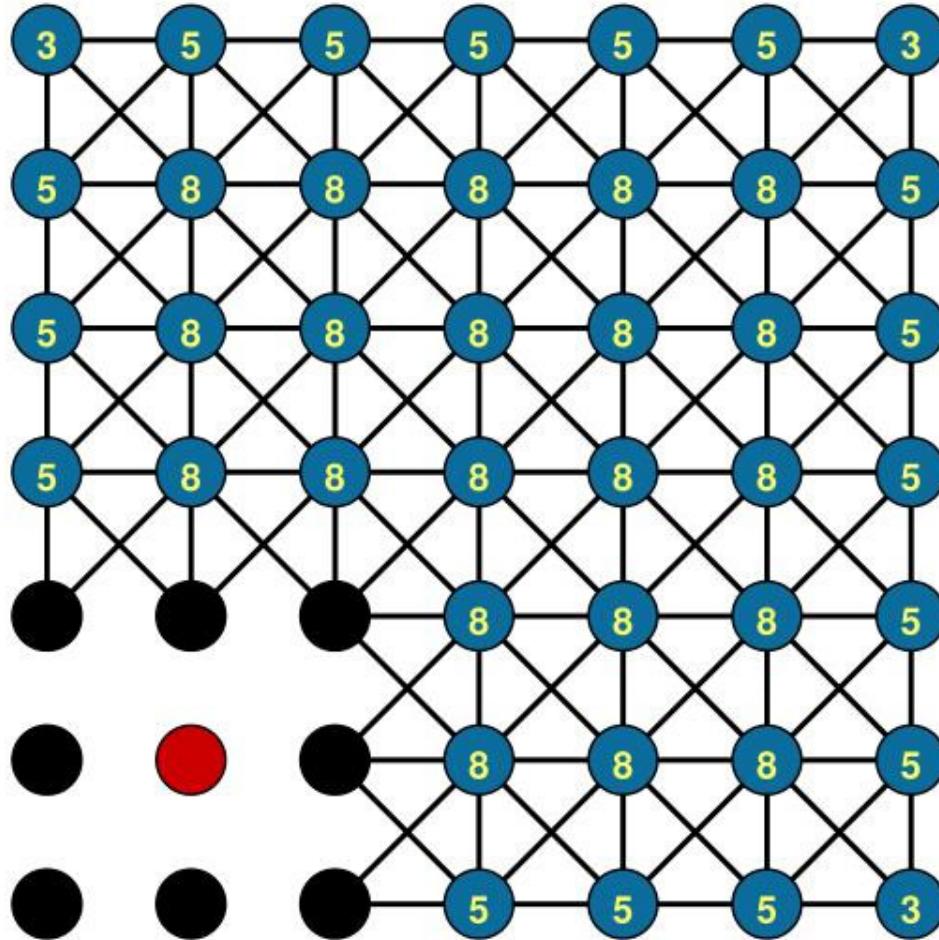


→ select next C-pt
with maximal
measure

→ select neighbors
as F-pts

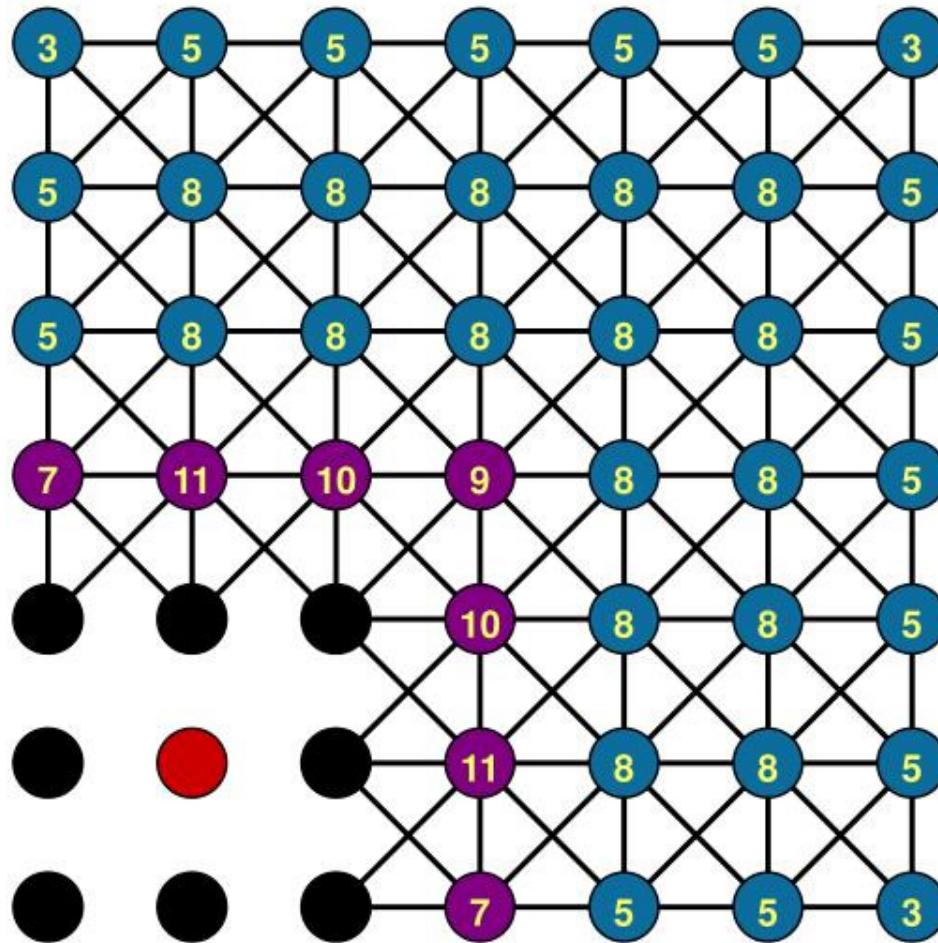
→ update measures
of F-pt neighbors

Ruge AMG: select F-pt 1



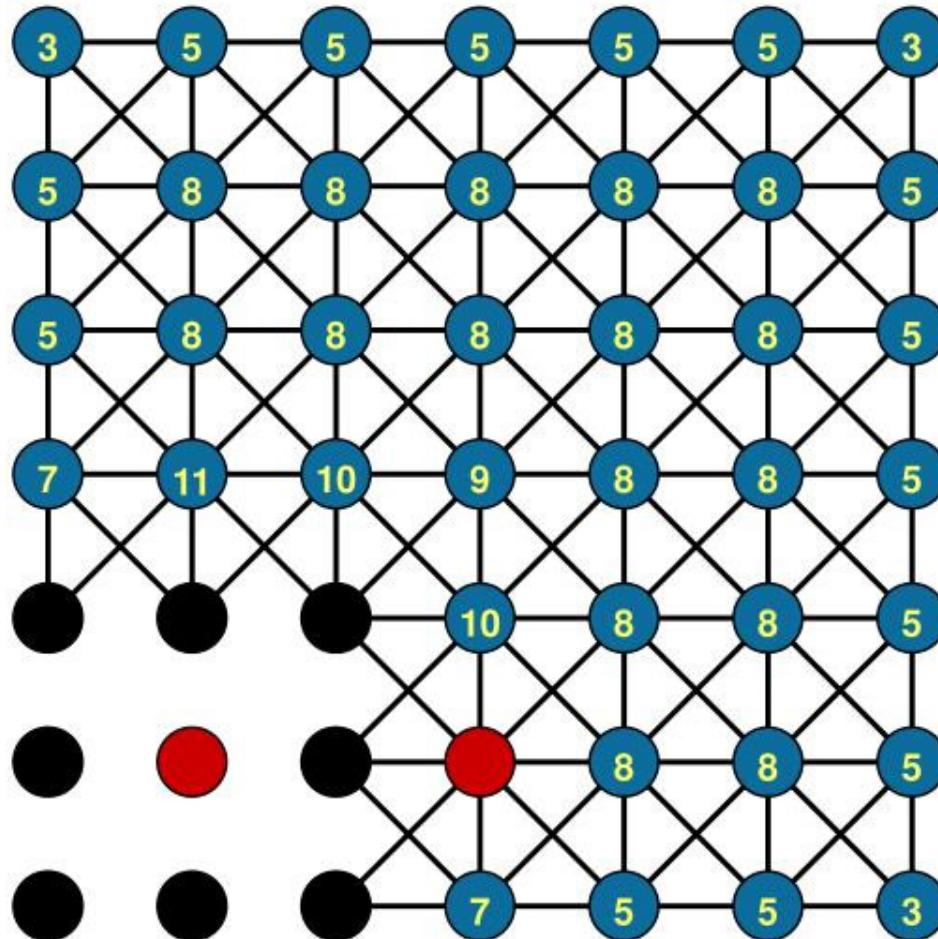
- select C-pt with maximal measure
- select neighbors as F-pts
- update measures of F-pt neighbors

Ruge AMG: update F-pt neighbors 1



- select C-pt with maximal measure
- select neighbors as F-pts
- update measures of F-pt neighbors

Ruge AMG: select C-pt 2

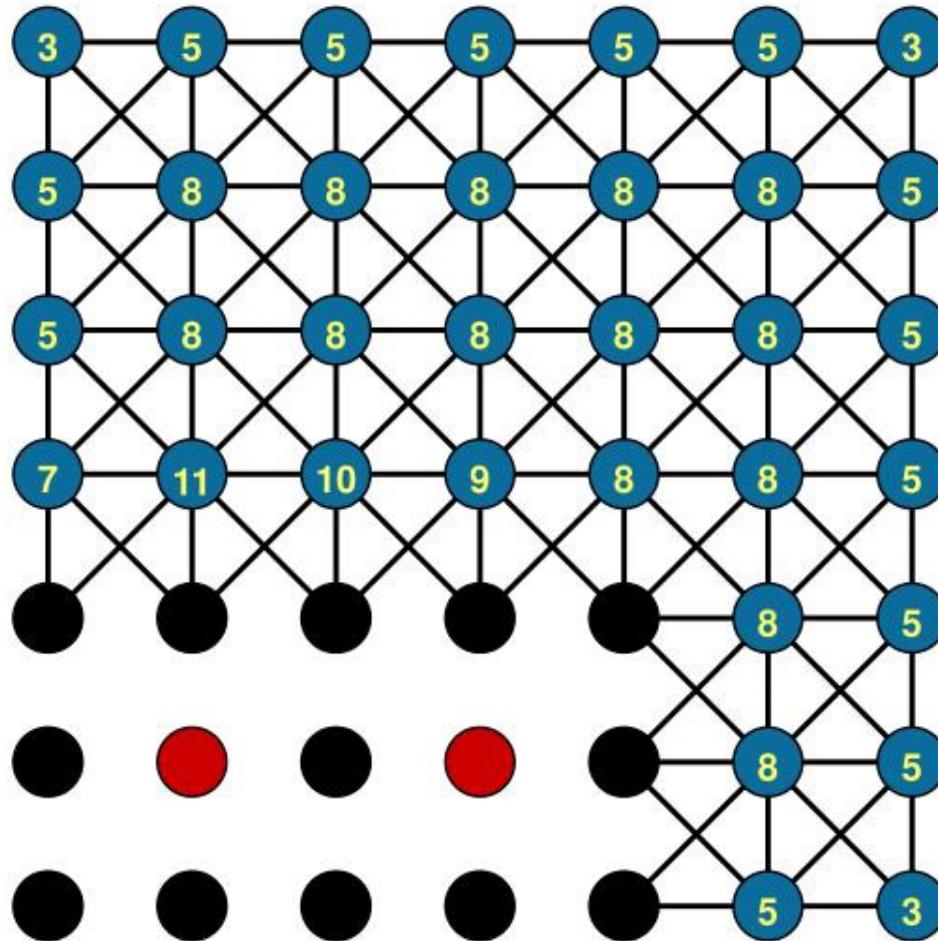


→ select next C-pt
with maximal
measure

→ select neighbors
as F-pnts

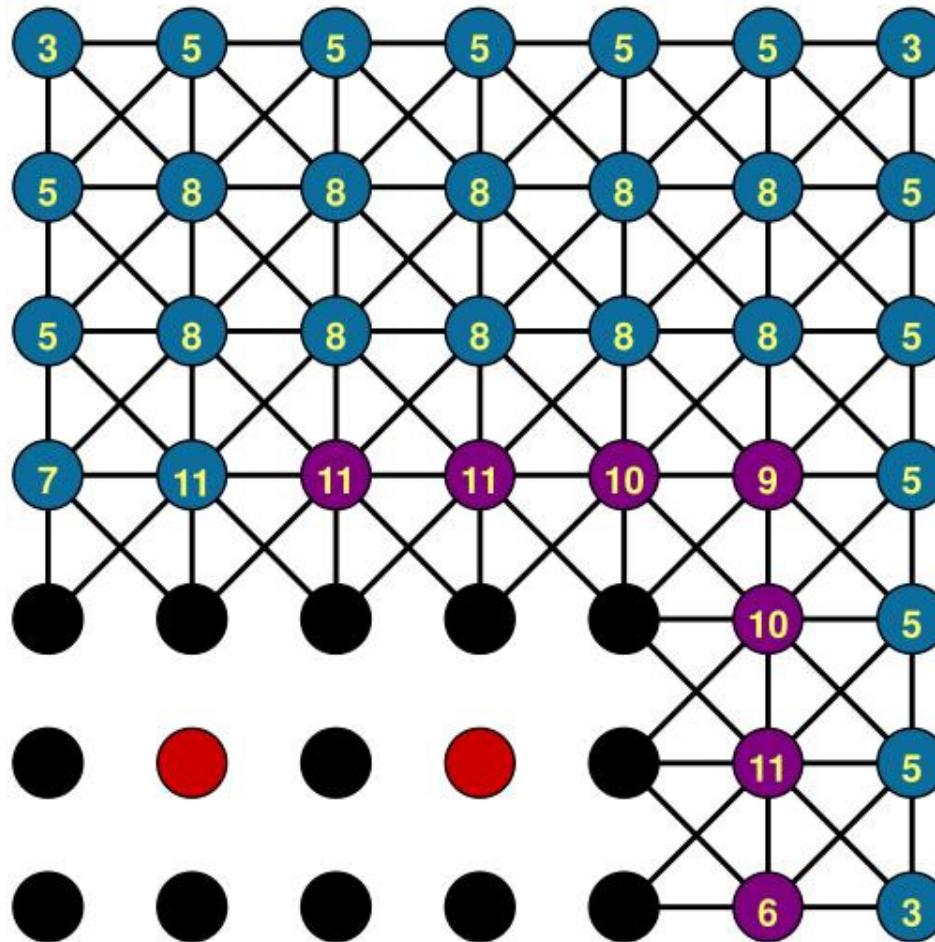
→ update measures
of F-pt neighbors

Ruge AMG: select F-pt 2



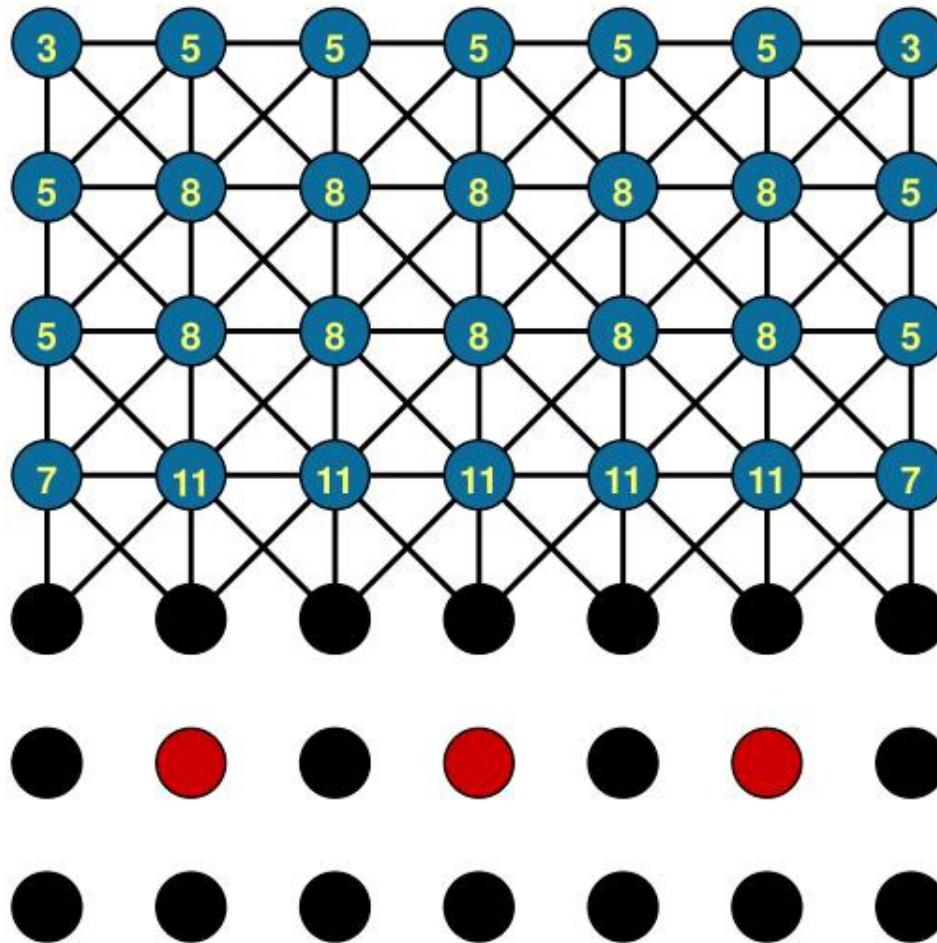
- select next C-pt with maximal measure
- select neighbors as F-pnts
- update measures of F-pt neighbors

Ruge AMG: update F-pt neighbors 2



- select next C-pt with maximal measure
- select neighbors as F-ppts
- update measures of F-pt neighbors

Ruge AMG: select C-pt, F-pts, update neighbors 3

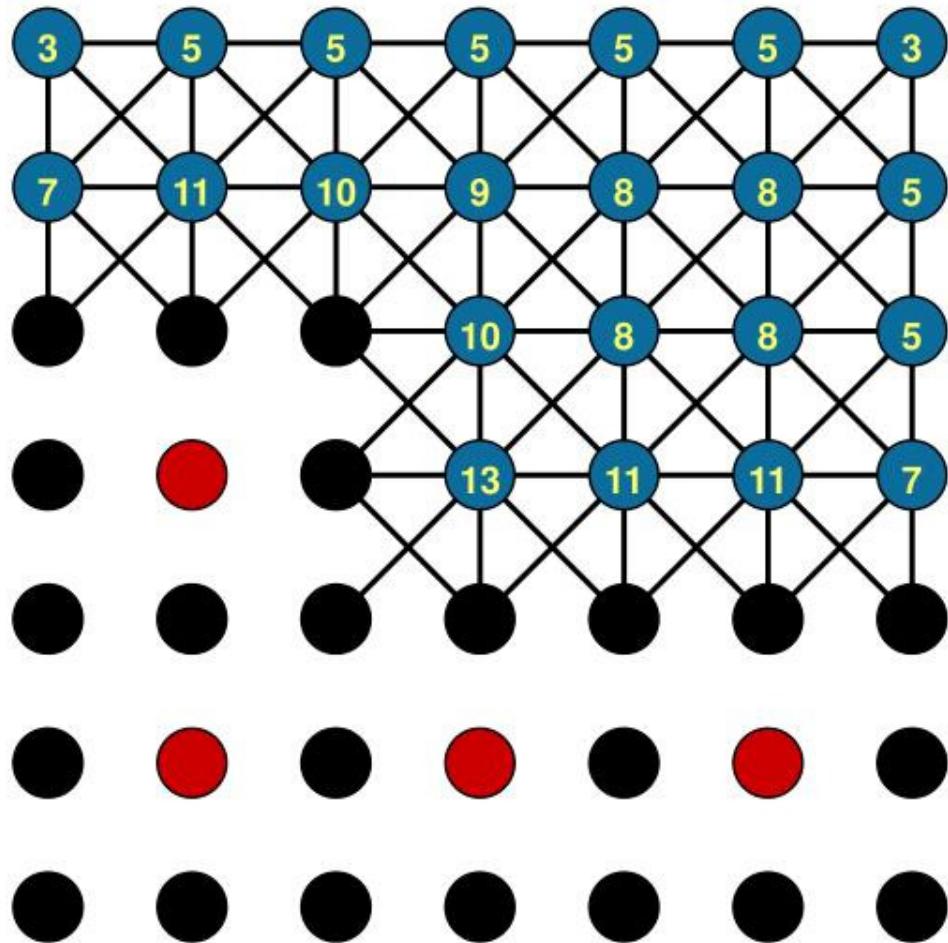


→ select next C-pt
with maximal
measure

→ select neighbors
as F-pts

→ update measures
of F-pt neighbors

Ruge AMG: select C-pt, F-pts, update neighbors 4

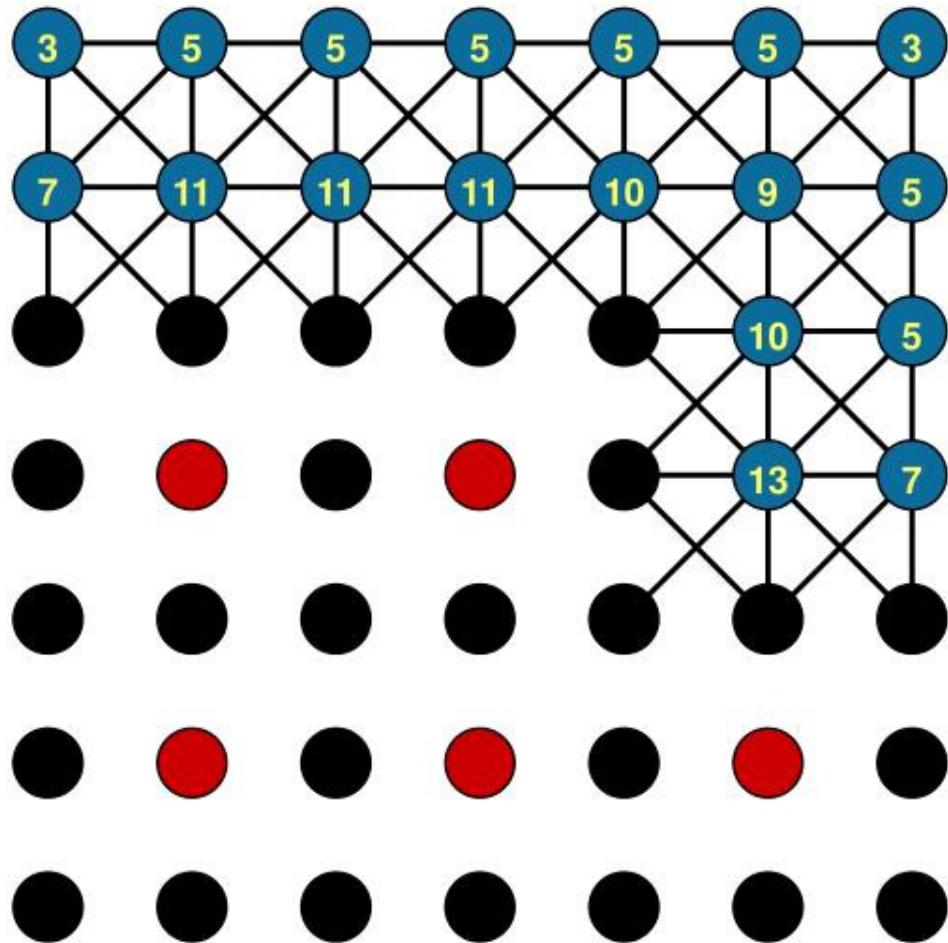


→ select next C-pt
with maximal
measure

→ select neighbors
as F-pts

→ update measures
of F-pt neighbors

Ruge AMG: select C-pt, F-pts, update neighbors 5

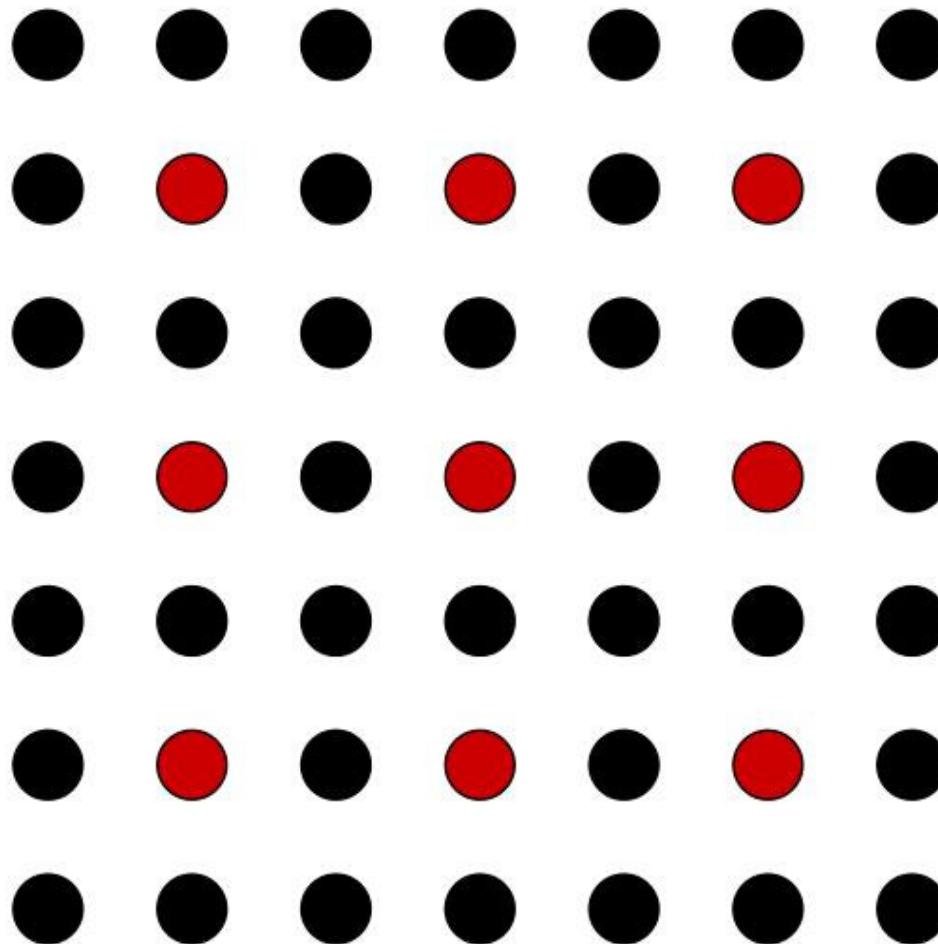


→ select next C-pt
with maximal
measure

→ select neighbors
as F-pts

→ update measures
of F-pt neighbors

Ruge AMG: select C-pt, F-pts, update neighbors 6,7,8,9



→ select next C-pt
with maximal
measure

→ select neighbors
as F-pts

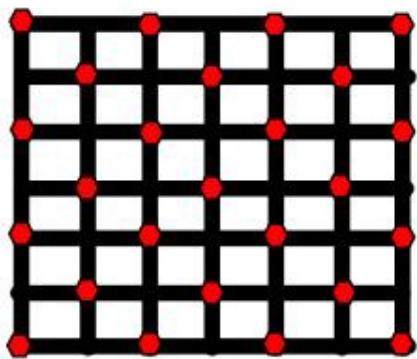
→ update measures
of F-pt neighbors

Examples: Laplacian Operator

5-pt FD, 9-pt FE (quads), and 9-pt FE (stretched quads)

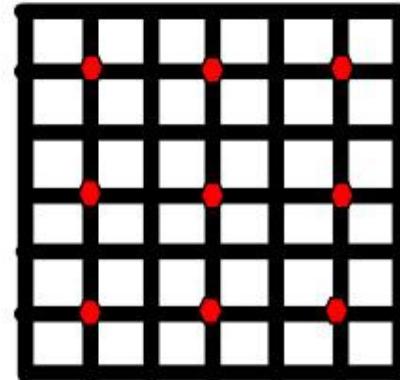
5-pt FD

$$\begin{pmatrix} & -1 \\ -1 & 4 & -1 \\ & -1 \end{pmatrix}$$

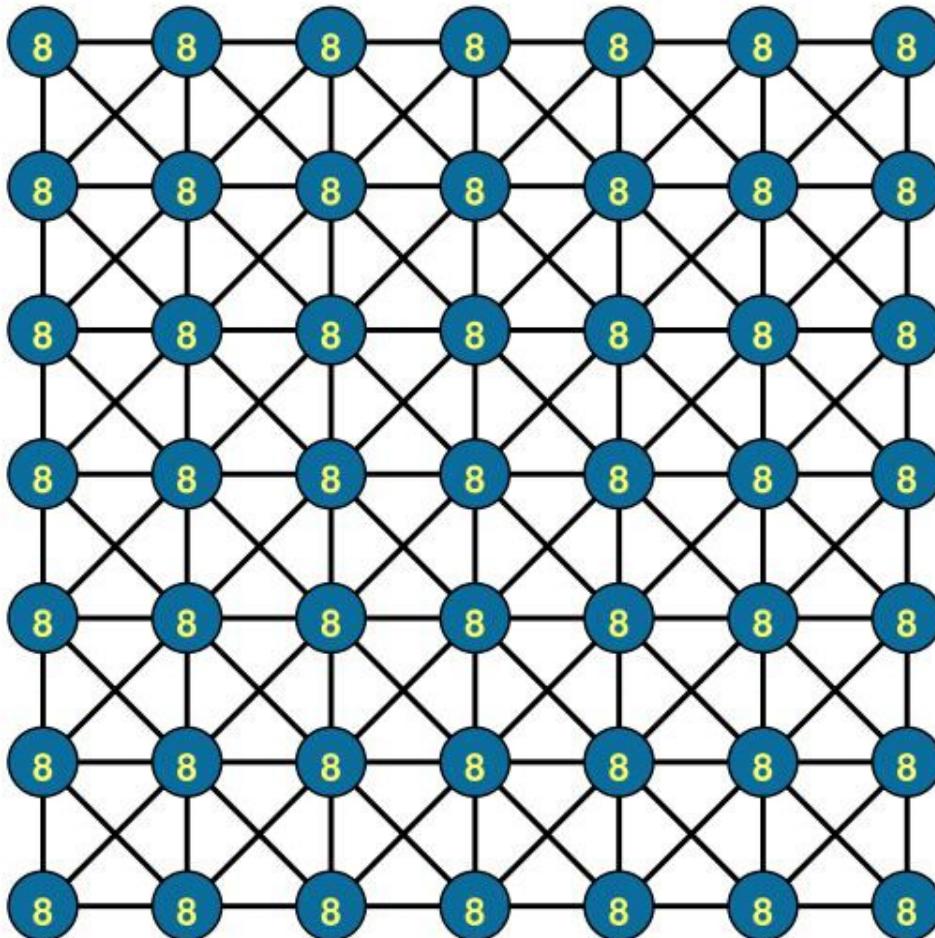


9-pt FE (quads)

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

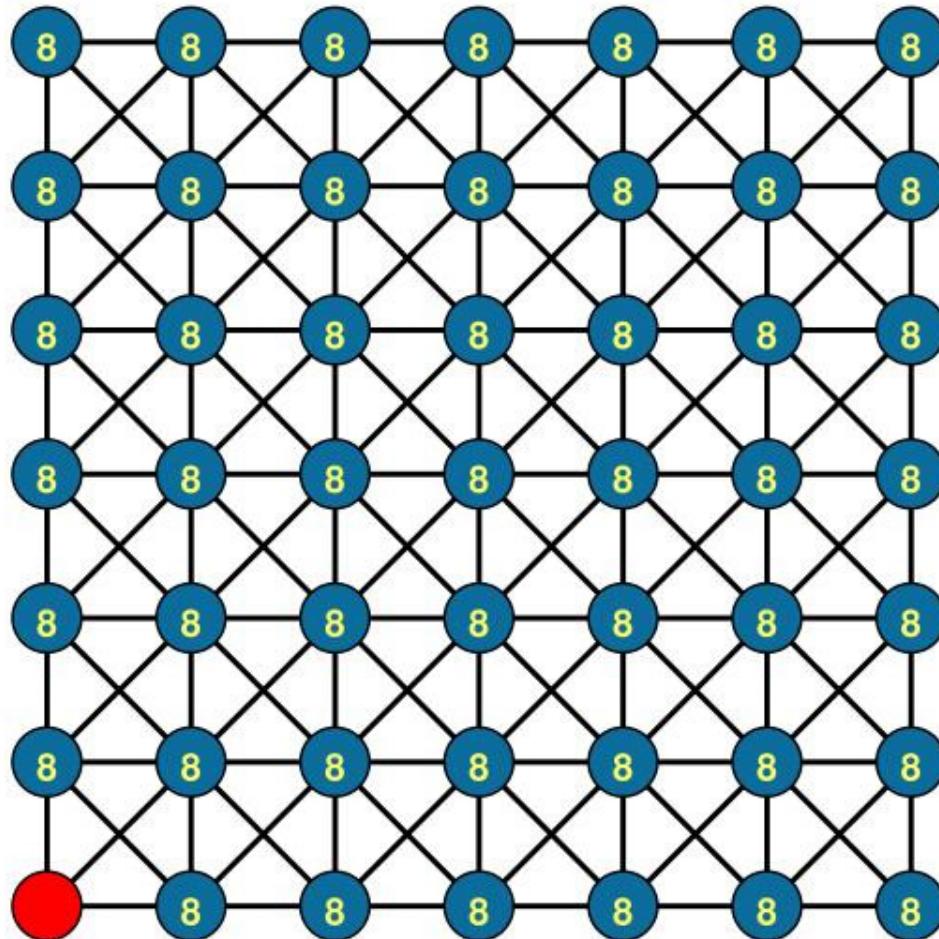


Coarse-grid selection: 9pt Laplacian, periodic boundary conditions



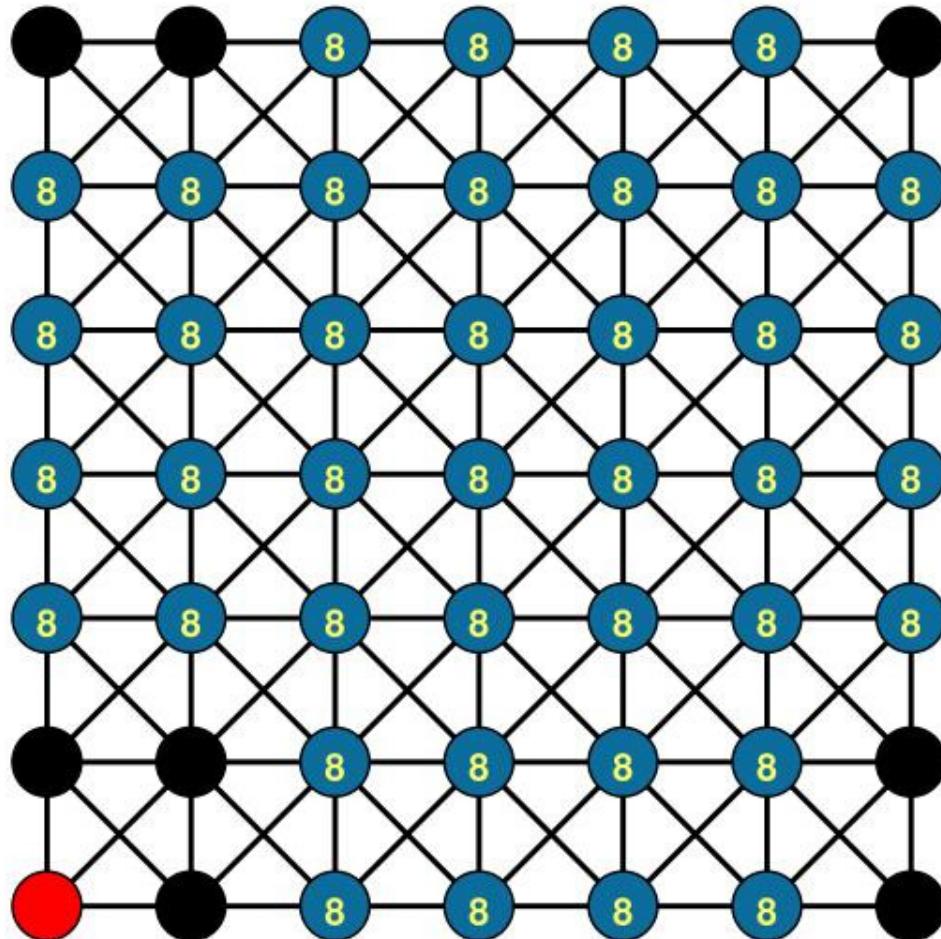
- select C-pt with maximal measure
- select neighbors as F-pts
- update measures of F-pt neighbors

Coarse-grid selection: 9pt Laplacian, periodic boundary conditions



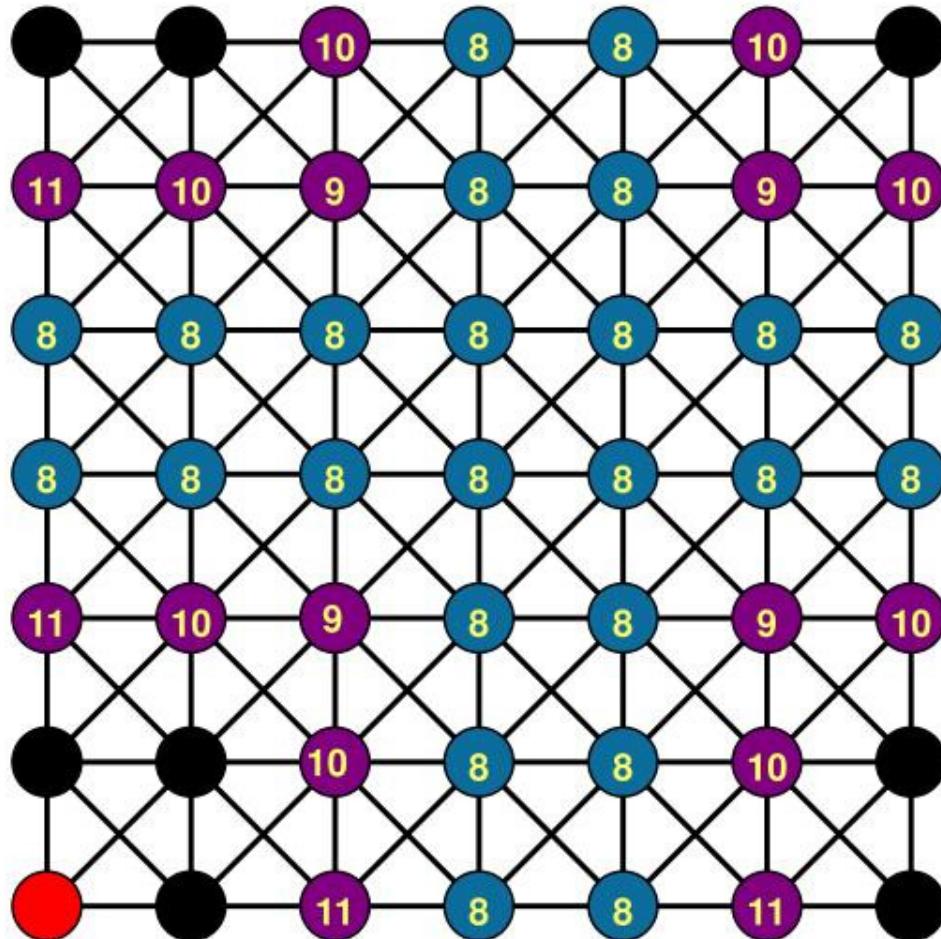
- select C-pt with maximal measure
- select neighbors as F-pts
- update measures of F-pt neighbors

Coarse-grid selection: 9pt Laplacian, periodic boundary conditions



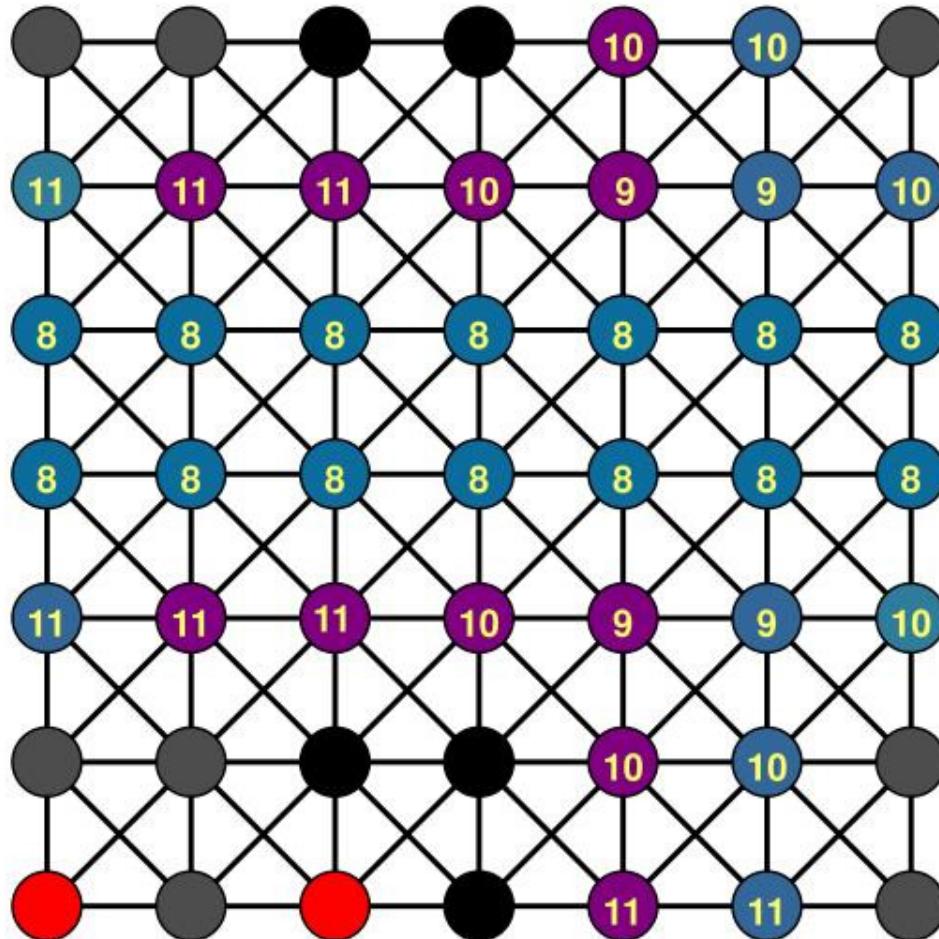
- select C-pt with maximal measure
- select neighbors as F-pts
- update measures of F-pt neighbors

Coarse-grid selection: 9pt Laplacian, periodic boundary conditions



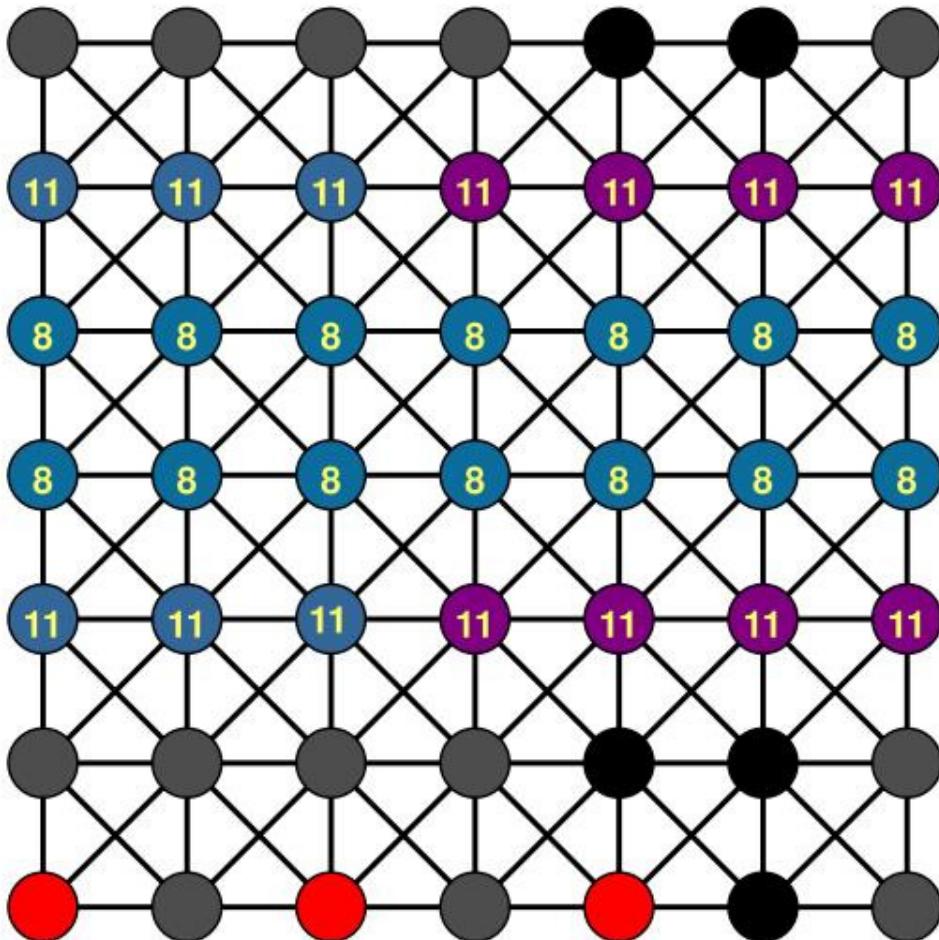
- select C-pt with maximal measure
- select neighbors as F-pnts
- update measures of F-pt neighbors

Coarse-grid selection: 9pt Laplacian, periodic boundary conditions



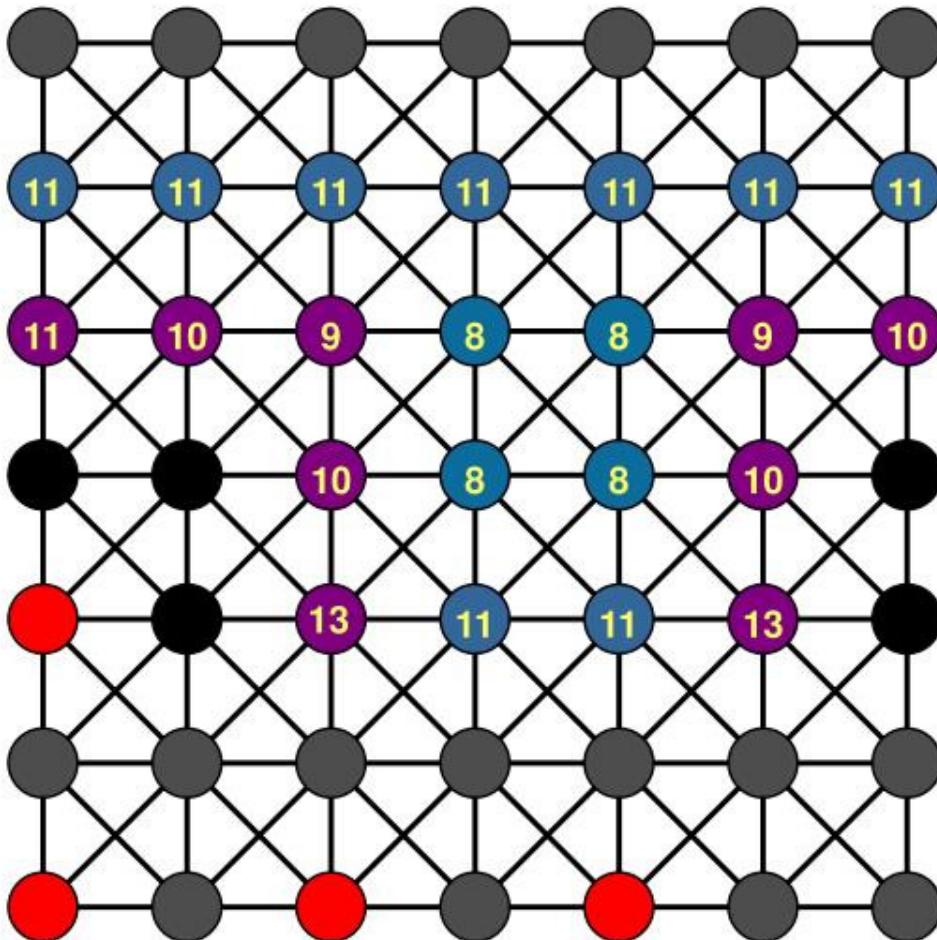
- select C-pt with maximal measure
- select neighbors as F-pts
- update measures of new F-pt neighbors

Coarse-grid selection: 9pt Laplacian, periodic boundary conditions



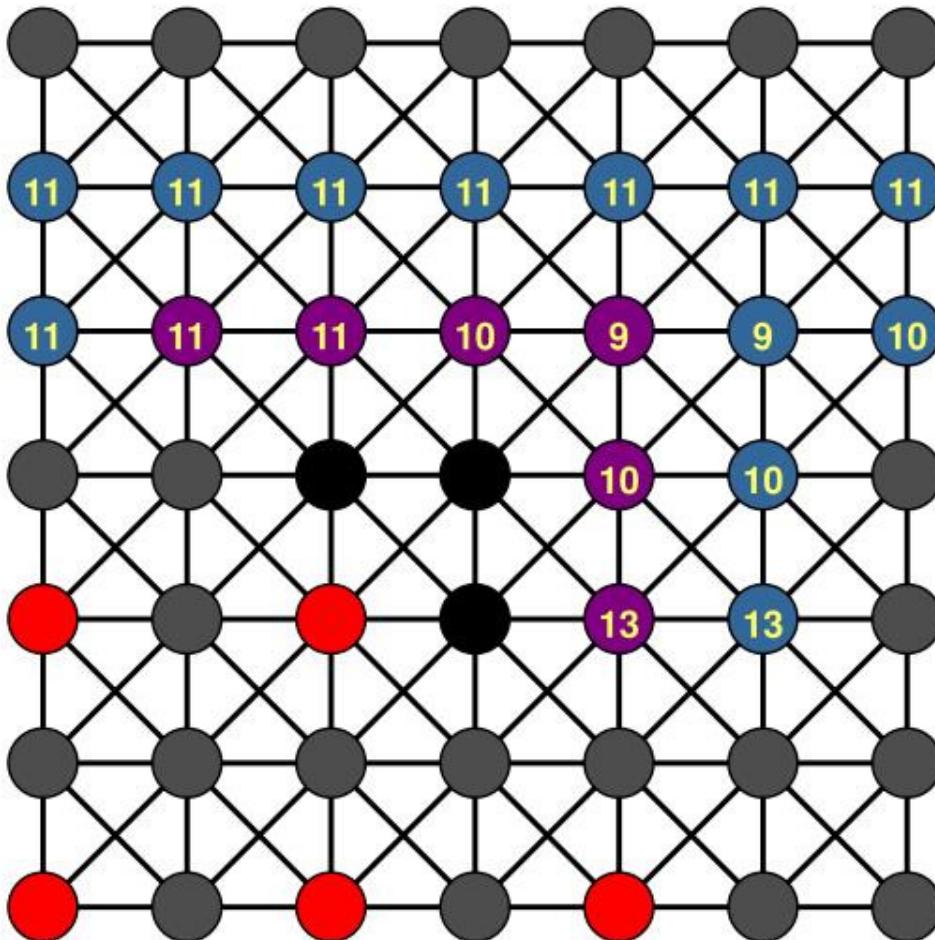
- select C-pt with maximal measure
- select neighbors as F-pts
- update measures of new F-pt neighbors

Coarse-grid selection: 9pt Laplacian, periodic boundary conditions



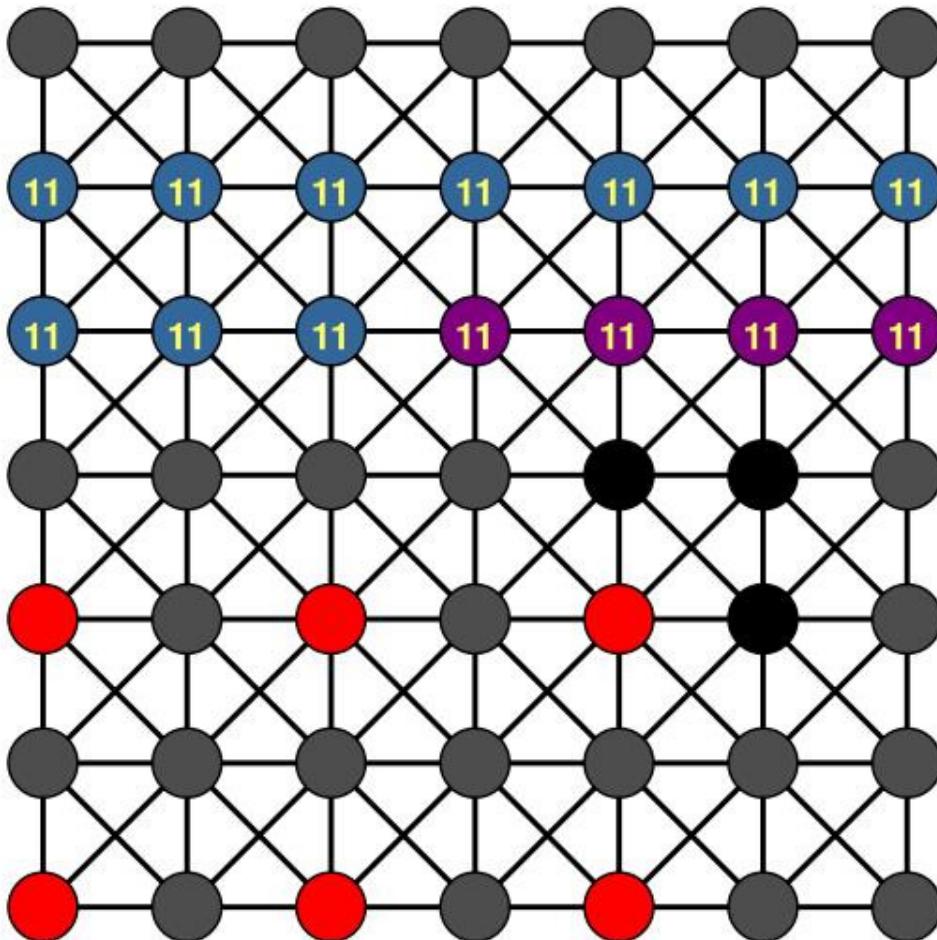
- select C-pt with maximal measure
- select neighbors as F-pts
- update measures of new F-pt neighbors

Coarse-grid selection: 9pt Laplacian, periodic boundary conditions



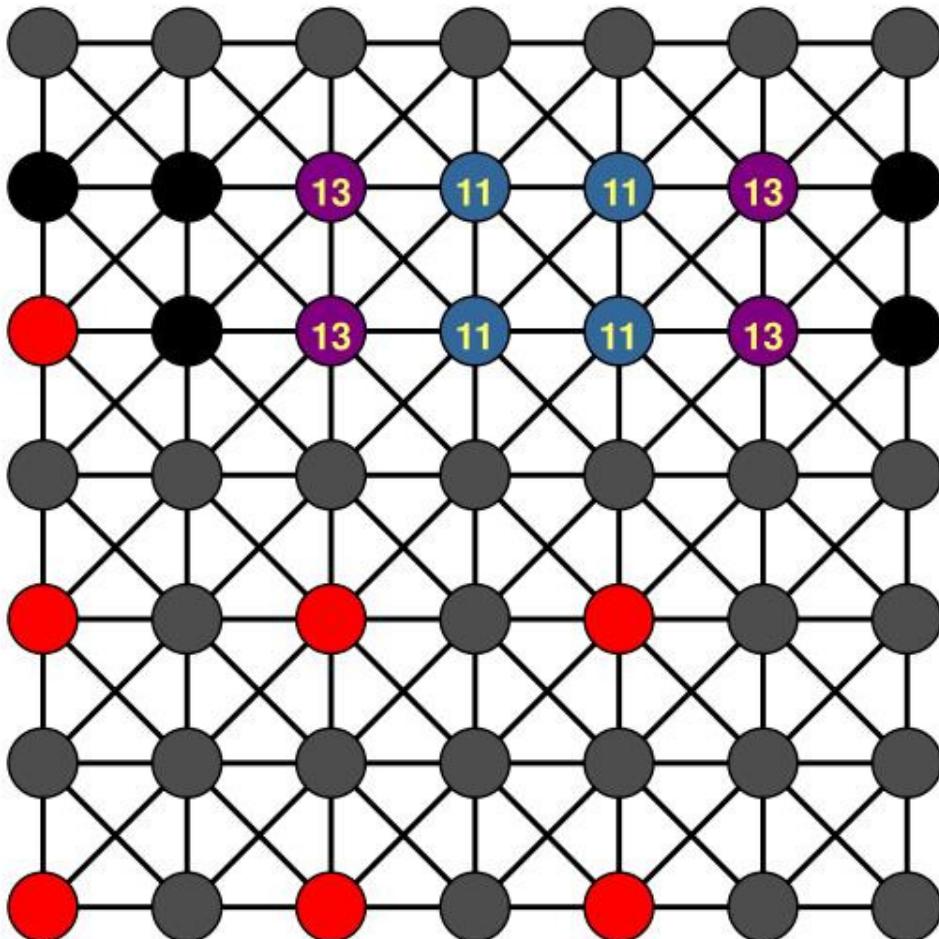
- select C-pt with maximal measure
- select neighbors as F-pts
- update measures of new F-pt neighbors

Coarse-grid selection: 9pt Laplacian, periodic boundary conditions



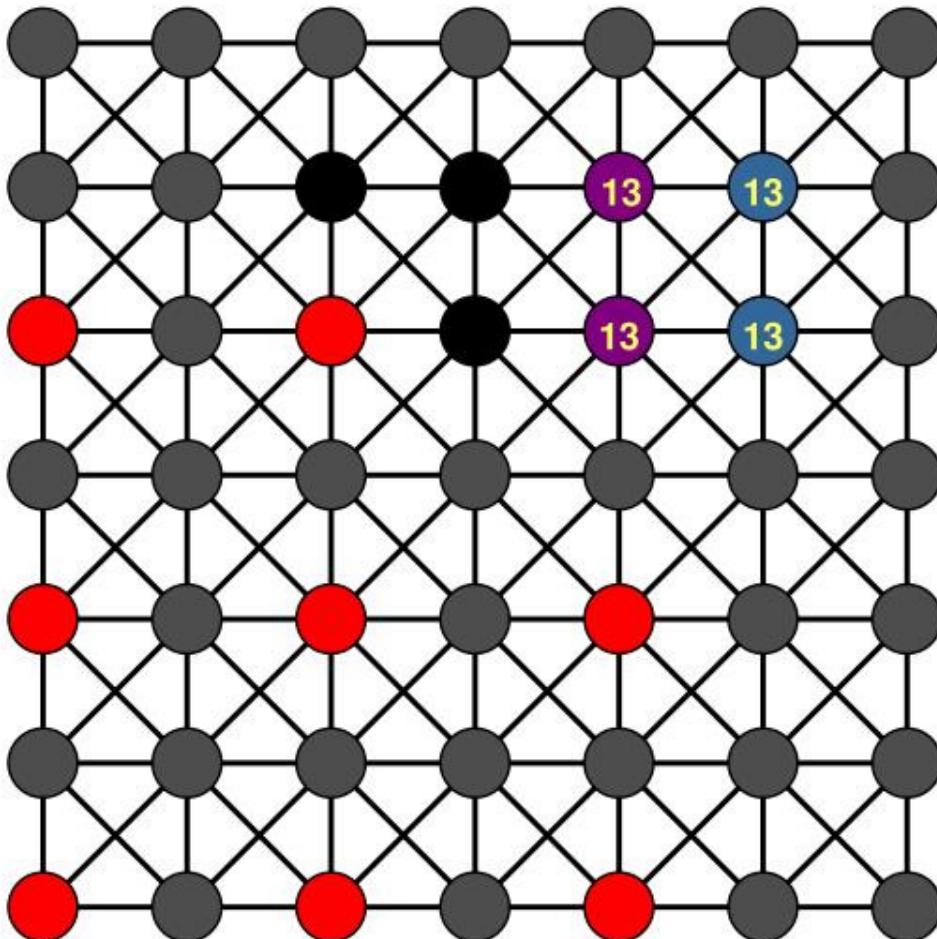
- select C-pt with maximal measure
- select neighbors as F-pts
- update measures of new F-pt neighbors

Coarse-grid selection: 9pt Laplacian, periodic boundary conditions



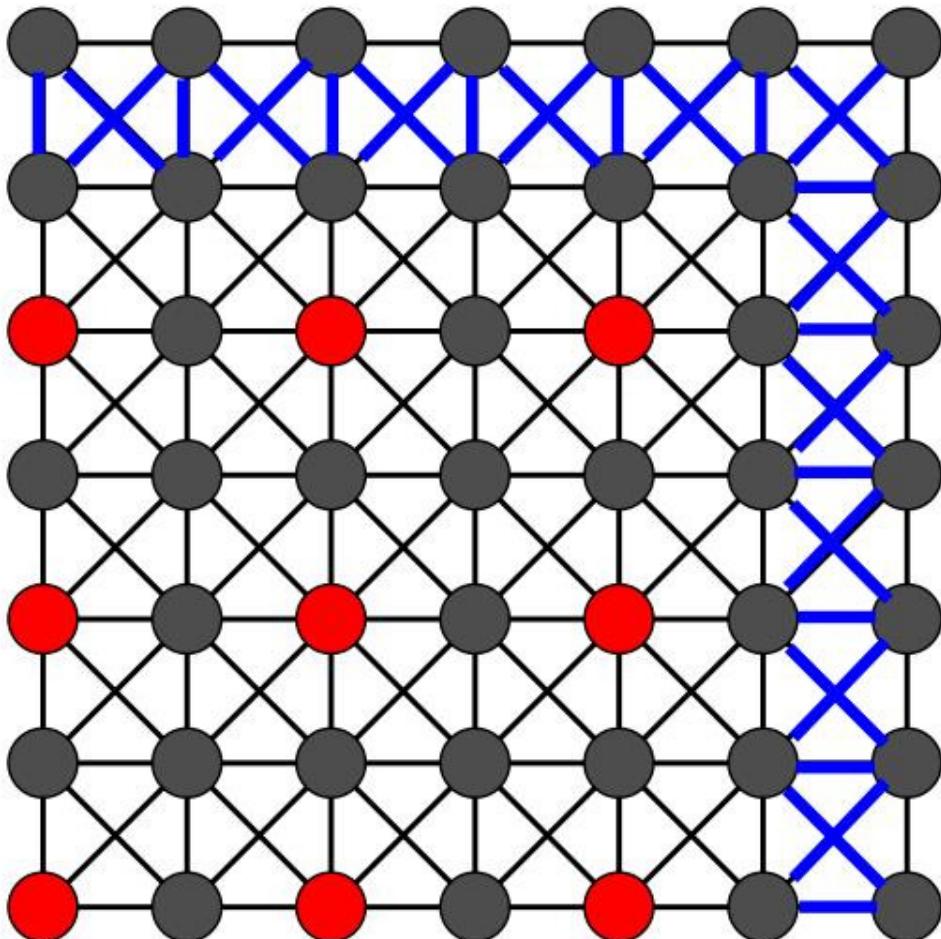
- select C-pt with maximal measure
- select neighbors as F-pts
- update measures of new F-pt neighbors

Coarse-grid selection: 9pt Laplacian, periodic boundary conditions



- select C-pt with maximal measure
- select neighbors as F-pts
- update measures of new F-pt neighbors

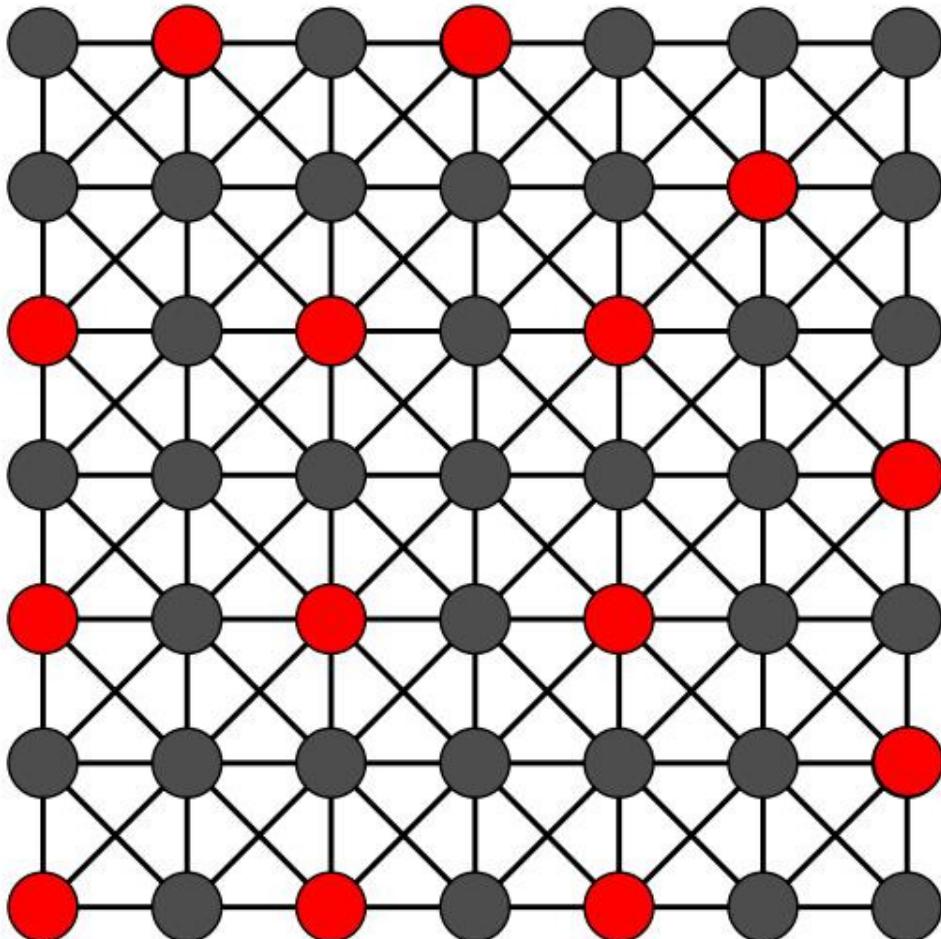
Coarse-grid selection: 9pt Laplacian, periodic boundary conditions



→ Modulo periodicity,
it's the same
coarsening as in
the Dirichlet case.

→ However, it has
many F-F
connections that do
not share a common
C-point

Coarse-grid selection: 9pt Laplacian, periodic boundary conditions

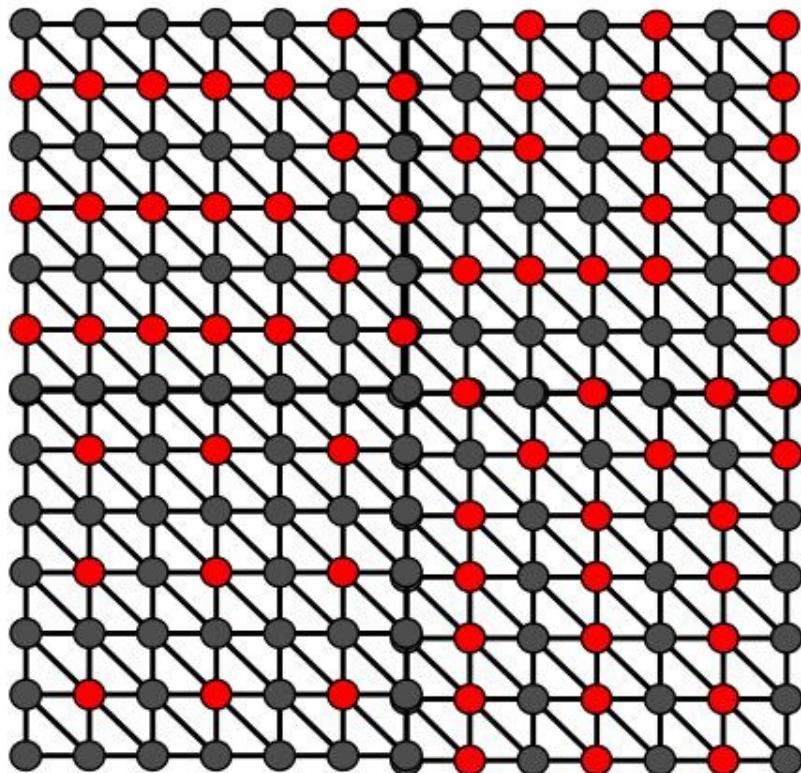


→ A second pass is made in which some F-points are made into C-points to enforce (C1).

→ Goals of the second pass include minimizing C-C connections, and minimizing the number of C-points converted to F-points.

How well does AMG coarsen:

$$-(au_x)_x - (bu_y)_y + cu_{xy} = f(x, y)$$

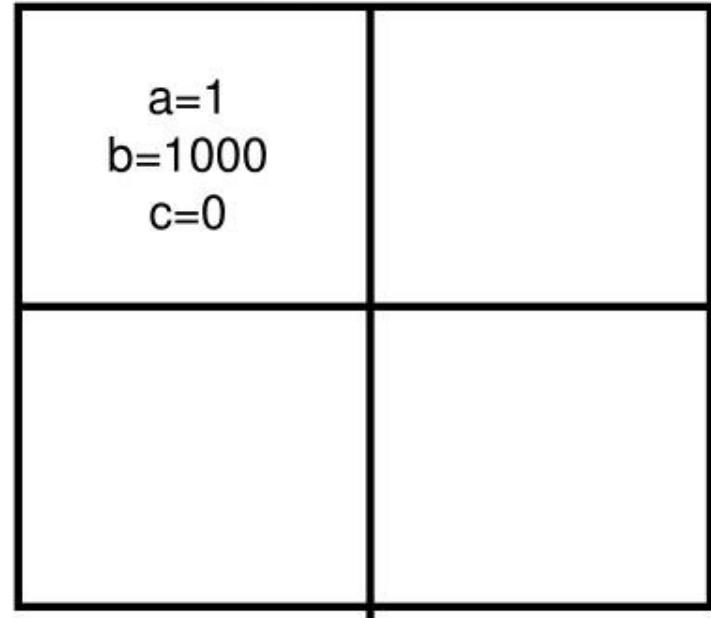
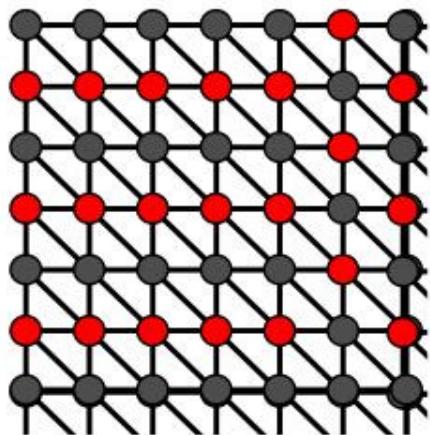


$a=1$ $b=1000$ $c=0$	$A=1$ $b=1$ $c=2$
$a=1$ $b=1$ $c=0$	$a=1000$ $b=1$ $c=0$

→ In each region, AMG coarsens only in the direction of dependence!

How well does AMG coarsen:

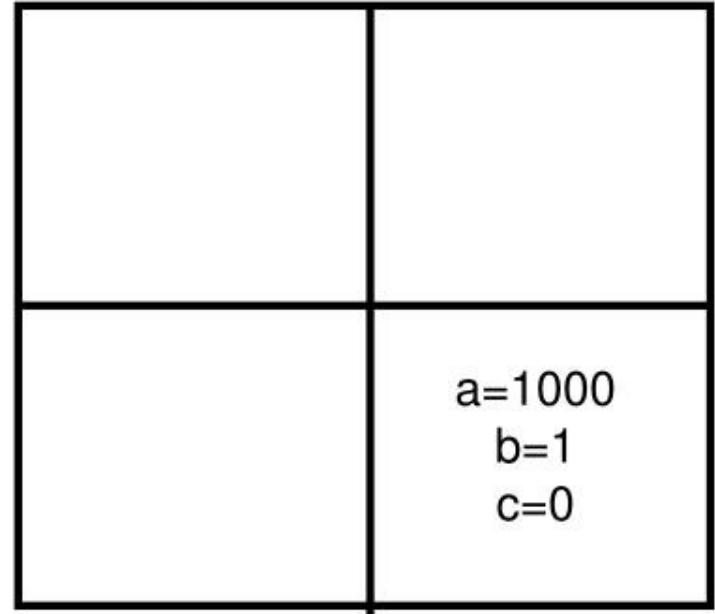
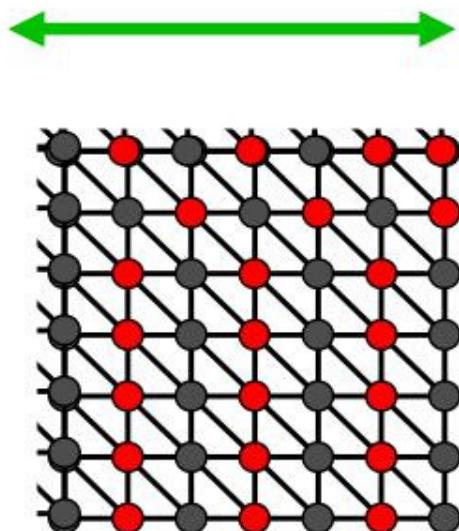
$$-(a u_x)_x - (b u_y)_y + c u_{xy} = f(x, y)$$



→ In each region, AMG coarsens only in the direction of dependence!

How well does AMG coarsen:

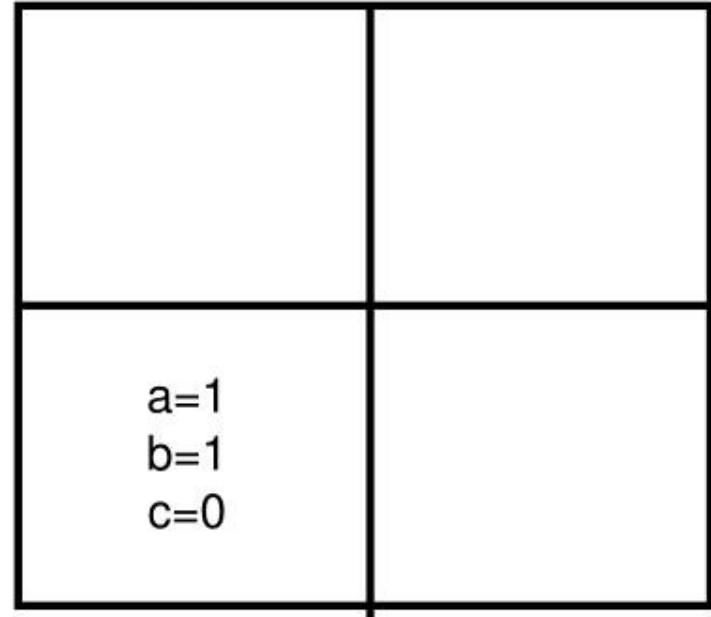
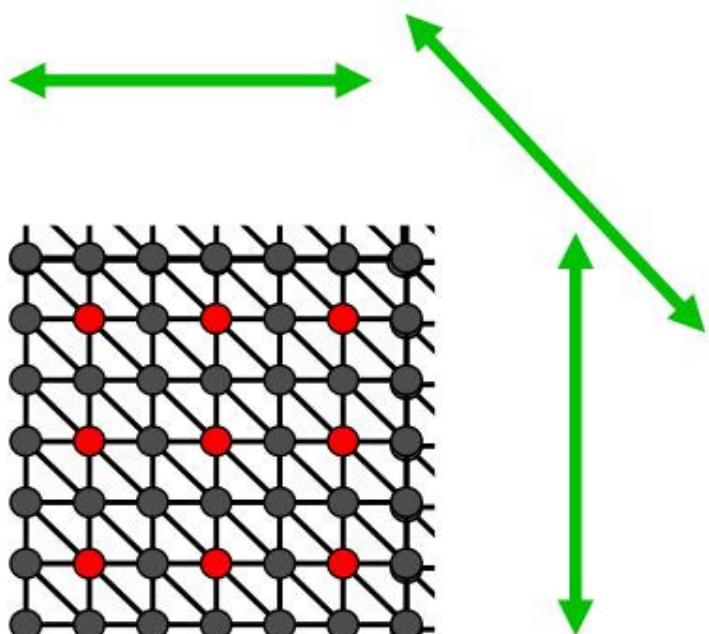
$$-(a u_x)_x - (b u_y)_y + c u_{xy} = f(x, y)$$



→ In each region, AMG coarsens only in the direction of dependence!

How well does AMG coarsen:

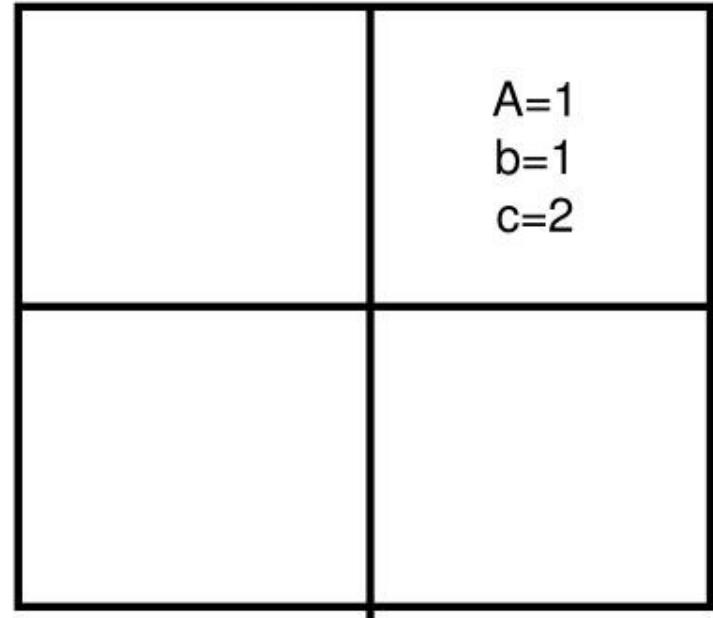
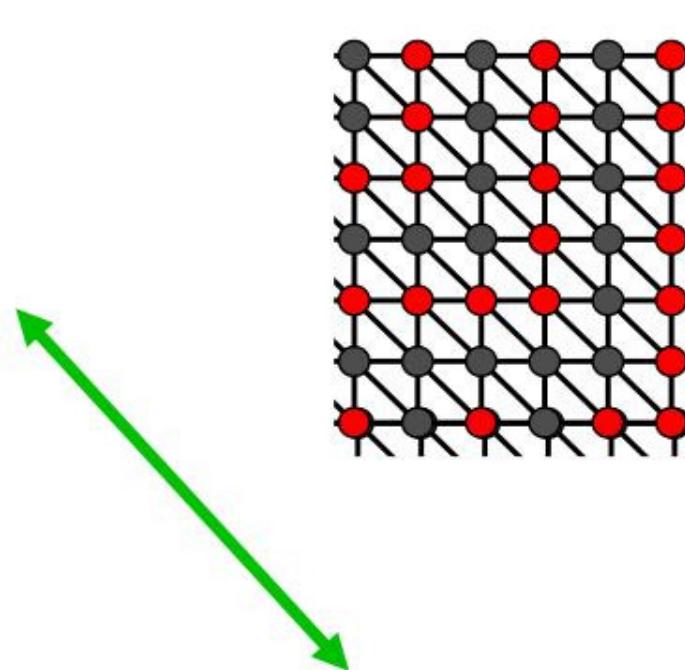
$$-(a u_x)_x - (b u_y)_y + c u_{xy} = f(x, y)$$



→ In each region, AMG coarsens only in the direction of dependence!

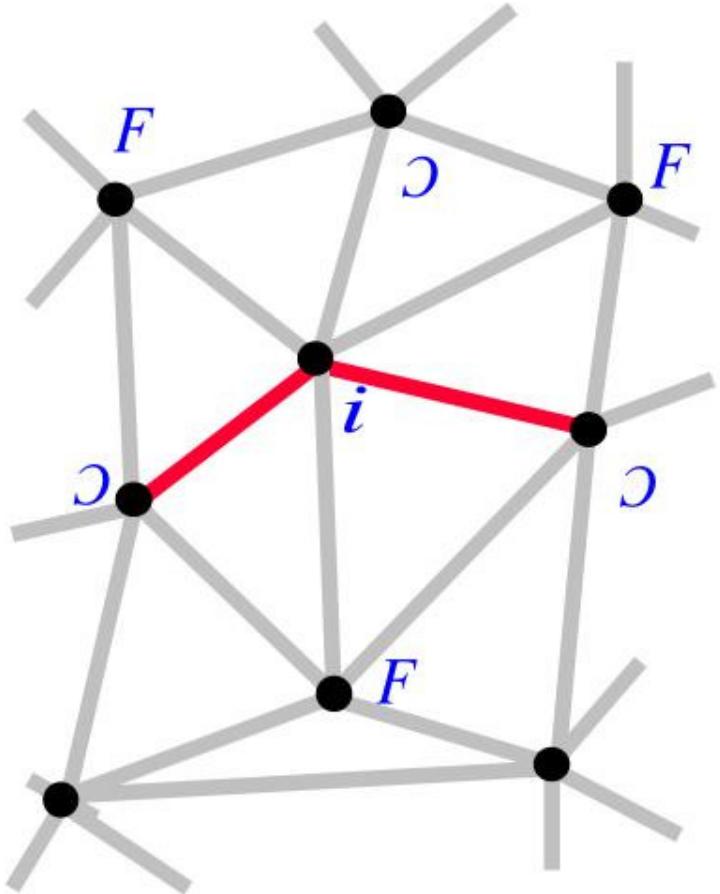
How well does AMG coarsen:

$$-(a u_x)_x - (b u_y)_y + c u_{xy} = f(x, y)$$



→ In each region, AMG coarsens only in the direction of dependence!

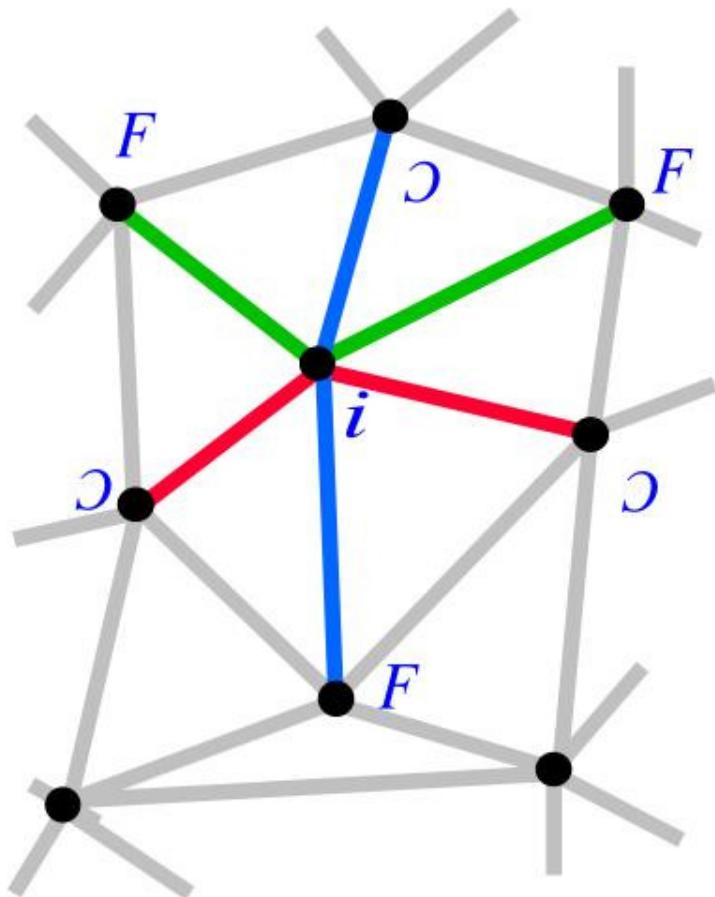
Prolongation



$$(Pe)_i = \begin{cases} e_i & , i \in C \\ \sum_{k \in C_i} \omega_{ik} e_k & , i \in F \end{cases}$$

The interpolated value at point i is just e_i if i is a C -point. If i is an F -point, the value is a weighted sum of the values of the points in the **coarse interpolatory set** C_i .

To define prolongation at i , we must examine the types of connections of u_i .



Sets of connection types:

C_i — i is dependent on these coarse interpolatory C -points.

D_i^s — i is dependent on these F -points.

D_i^w — i does not depend on these “weakly connected” points, which may be C - or F -points.

Prolongation is based on smooth error and dependencies (from M-matrices)

Recall that smooth error is characterized by “small” residuals:

$$r_i = a_{ii} e_i + \sum_{j \in N_i} a_{ij} e_j \approx 0$$

which we can rewrite as:

$$a_{ii} e_i \approx - \sum_{j \neq i} a_{ij} e_j$$

We base prolongation on this formula by “solving” for e_i and making some approximating substitutions.

Prolongation is based on smooth error and dependencies (from M-matrices)

We begin by writing the smooth-error relation:

$$a_{ii} e_i \approx - \sum_{j \neq i} a_{ij} e_j$$

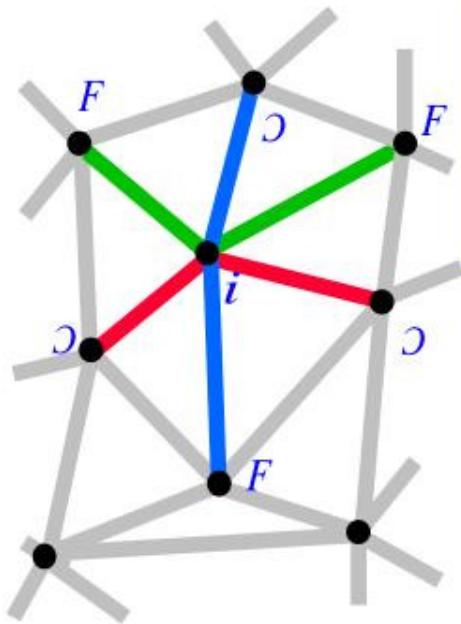
Identifying its component sums:

$$a_{ii} e_i \approx - \sum_{j \in C_i} a_{ij} e_j - \sum_{j \in D_i^s} a_{ij} e_j - \sum_{j \in D_i^w} a_{ij} e_j$$

*Coarse
interpolatory
set* *F-point
dependencies* *Weak
connections*

We must approximate e_j in each of the last two sums in terms of e_i or of e_j for $j \in C_i$.

For the **weak connections**:
let $e_j \approx e_i$.



$$a_{ii}e_i \approx -\sum_{j \in C_i} a_{ij}e_j - \sum_{j \in D_i^s} a_{ij}e_j - \sum_{j \in D_i^w} a_{ij}e_j$$

Coarse
interpolatory
set F-point
dependencies Weak
connections

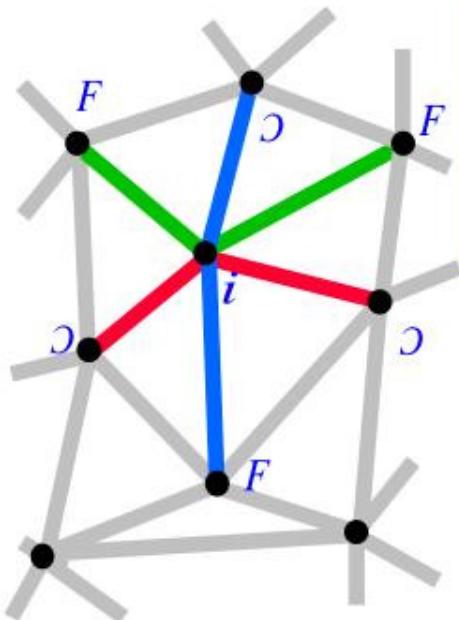
Effectively, this throws the weak connections onto the diagonal:

$$\left(a_{ii} + \sum_{j \in D_i^w} a_{ij} \right) e_i \approx -\sum_{j \in C_i} a_{ij}e_j - \sum_{j \in D_i^s} a_{ij}e_j$$

This approximation can't hurt too much:

- Since the connection is weak.
- If i depended on points in D_i^w , smooth error varies slowly in the direction of dependence

For the F-point dependencies: use a weighted avg. of errors in $C_i \cap C_j$.



$$\left(a_{ii} + \sum_{j \in D_i^w} a_{ij} \right) e_i \approx - \sum_{j \in C_i} a_{ij} e_j - \sum_{j \in D_i^s} a_{ij} e_j$$

Weak
connections
 Coarse
interpolatory set
 F-point
dependencies

Approximate e_j by a weighted average of the e_k in the coarse interpolatory set $C_i \cap C_j$.

$$e_j \approx \frac{\left(\sum_{k \in C_i} a_{jk} e_k \right)}{\left(\sum_{k \in C_i} a_{jk} \right)}$$

It is for this reason that the intersection of the coarse interpolatory sets of two F-points with a dependence relationship must be nonempty (**C1**).

Finally, the prolongation weights are defined

Making the previous substitution, and with a bit of messy algebra, the smooth error relation can be “solved” for e_i to yield the interpolation formula:

$$e_i \approx \sum_{j \in C_i} \omega_{ij} e_j$$

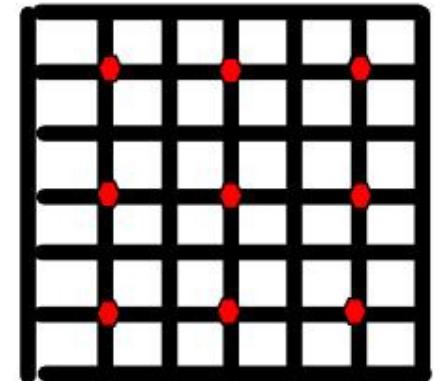
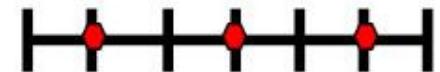
where the prolongation weights are given:

$$\omega_{ij} = - \frac{a_{ij} + \sum_{j \in D_i^s} \frac{a_{ik} a_{kj}}{\sum_{m \in C_i} a_{km}}}{a_{ii} + \sum_{n \in D_i^w} a_{in}}$$

Highlights of Multigrid:

Storage: f^h, u^h must be stored each level

- In 1-d, each coarse grid has about half the number of points as the finer grid.
- In 2-d, each coarse grid has about one-fourth the number of points as the finer grid.
- In d -dimensions, each coarse grid has about 2^{-d} the number of points as the finer grid.
- Storage cost: $2N^d(1 + 2^{-d} + 2^{-2d} + 2^{-3d} + \dots + 2^{-Md}) < \frac{2N^d}{1 - 2^{-d}}$ less than 2, 4/3, 8/7 the cost of storage on the fine grid for 1, 2, and 3-d problems, respectively.



AMG storage: grid complexity

- For AMG there is no simple predictor for total storage costs. u^m , f^m , and $A^m = I_{m-1}^m A^{m-1} I_m^{m-1}$ must be stored on all levels.
- Define σ^Ω , the **grid complexity**, as the total number of unknowns (gridpoints) on all levels, divided by the number of unknowns on the finest level. Total storage of the vectors u and f occupy $2\sigma^\Omega$ storage locations.

AMG storage: operator complexity

- Define σ^A , the **operator complexity**, as the total number of nonzero coefficients of all operators A^m divided by the number of nonzero coefficients in the fine-level operator A^0 . Total storage of the operators occupies σ^A storage locations.

AMG storage: interpolation

- We could define σ^I , an interpolation complexity, as the total number of nonzero coefficients of all operators I_m^{m+1} divided by the number of nonzero coefficients in the operator I_0^1 . This measure is not generally cited, however (like most multigrids, the AMG crowd tends to ignore the cost of intergrid transfers).
- Two measures that occasionally appear are κ^A , the average “stencil size,” and κ^I , the average number of interpolation points per F-point.

AMG Setup Costs: flops

- Flops in the setup phase are only a small portion of the work, which includes sorting, maintaining linked-lists, keeping counters, storage manipulation, and garbage collection.
- Estimates of the total flop count to define interpolation weights (ω^I) and the coarse-grid operators (ω^A) are:

$$\omega^A = N\kappa^I(2\kappa^I(\kappa^A - \kappa^I) + 3\kappa^I + \kappa^A)$$

and

$$\omega^I = N\kappa^I(3(\kappa^A - \kappa^I) - 2)$$

AMG setup costs: a bad rap

- Many geometric MG methods need to compute prolongation and coarse-grid operators
- The only additional expense in the AMG setup phase is the coarse grid selection algorithm
- AMG setup phase is only 10-25% more expensive than in geometric MG and may be considerably less than that!

Highlights of Multigrid: Computation Costs

- Let 1 Work Unit (WU) be the cost of one relaxation sweep on the fine-grid.
- Ignore the cost of restriction and interpolation (typically about 20% of the total cost). (See?)
- Consider a V-cycle with 1 pre-Coarse-Grid correction relaxation sweep and 1 post-Coarse-Grid correction relaxation sweep.
- Cost of V-cycle (in WU):

$$2(1 + 2^{-d} + 2^{-2d} + 2^{-3d} + \dots + 2^{-Md}) < \frac{2}{1 - 2^{-d}}$$

- Cost is about 4, 8/3, 16/7 WU per V-cycle in 1, 2, and 3 dimensions.

AMG Solve Costs: flops per cycle

- The approximate number of flops in one level m for one relaxation sweep, residual transfer, and interpolation are (respectively)

$$2N_m^A \quad 2N_m^A + 2\kappa^I N_m^F \quad N_m^C + 2\kappa^I N_m^F$$

where N_m^A is the number of coefficients in A^m and N_m^C, N_m^F are the numbers of C-, F-points on Ω^m .

- The total flop count for a (v_1, v_2) V-cycle, noting that $\sum_m N_m^F \approx N$ and letting $v = v_1 + v_2$ is approximately

$$N(2(v+1)\kappa^A\sigma^\Omega + 4\kappa^I + \sigma^\Omega - 1)$$

AMG Solve Costs: flops per cycle, again

- All that is very well, but in practice we find the solve phase is generally dominated by the cost of relaxation and computing the residual.
- Both of those operations are proportional to the number of nonzero entries in the operator matrix on any given level.
- Thus the best measure of the ratio of work done on all levels to the work done on the finest level is operator complexity:

$$\sigma^A$$

Highlights of Multigrid: difficulties-anisotropic operators and grids

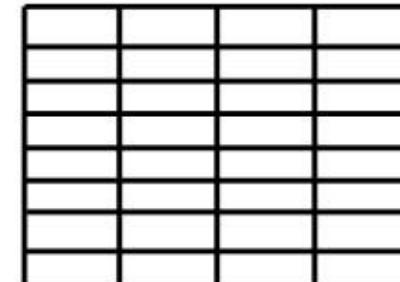
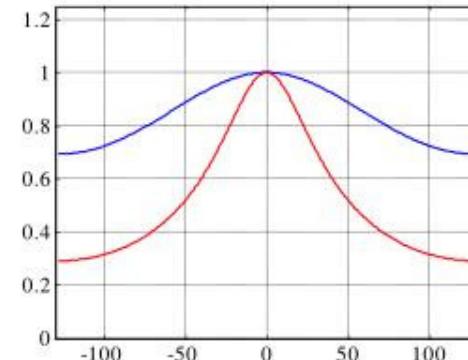
- Consider the operator :

$$-\alpha \frac{\partial^2 u}{\partial x^2} - \beta \frac{\partial^2 u}{\partial y^2} = f(x, y)$$

- If $\alpha \ll \beta$ then the GS-smoothing factors in the x - and y -directions are shown at right.

Note that GS relaxation does not damp oscillatory components in the x -direction.

- The same phenomenon occurs for grids with much larger spacing in one direction than the other:



Highlights of Multigrid: difficulties-discontinuous or anisotropic coefficients

- Consider the operator : $-\nabla \bullet (D(x,y) \nabla u)$ where

$$D(x,y) = \begin{pmatrix} d_{11}(x,y) & d_{12}(x,y) \\ d_{21}(x,y) & d_{22}(x,y) \end{pmatrix}$$

- Again, GS-smoothing factors in the x - and y -directions can be highly variable, and very often, GS relaxation does not damp oscillatory components in the one or both directions.
- Solutions: line-relaxation (where whole gridlines of values are found simultaneously), and/or semi-coarsening (coarsening only in the strongly coupled direction).

AMG does semi-coarsening automatically!

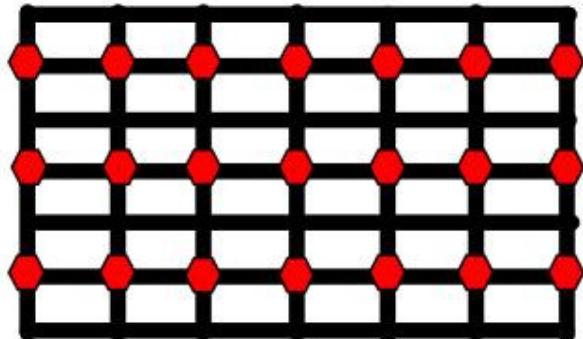
- Consider the operator :

$$-\alpha \frac{\partial^2 u}{\partial x^2} - \beta \frac{\partial^2 u}{\partial y^2} = f(x, y)$$

- In the limit, as $\alpha \rightarrow \infty$, the stencil becomes:

$$\begin{pmatrix} -1 & -4 & -1 \\ 2 & 8 & 2 \\ -1 & -4 & -1 \end{pmatrix}$$

- AMG automatically produces a semi-coarsened grid!!



AMG Convergence: there is theory (some)

- There is some theory, although it is of limited utility. It generally looks like:
- Theorem
 - Let $A^m \equiv A$ be SPD, and let the interpolation operator I_{m+1}^m be full rank, and let restriction and coarse-grid operators be defined by $I_m^{m+1} = (I_{m+1}^m)^T$ and $A^{m+1} = I_m^{m+1} A^m I_{m+1}^m$ and let there be smoothing operators G^m and coarse-grid correction operators

$$T^m = I^m - I_{m+1}^m (A^{m+1})^{-1} I_m^{m+1} A^m$$

AMG Convergence: there is theory (some)

- Theorem (continued)

- suppose that, for all e^m ,

$$\|G^m e^m\|_A^2 \leq \|e^m\|_A^2 - \delta \|T^m e^m\|_A^2$$

- holds for some $\delta > 0$ independently for all e^m and m .

Then $\delta \leq 1$, and, provided the coarsest problem is solved and at least one smoothing step is performed after each coarse-grid correction step, the V-cycle has a convergence factor wrt the energy norm bounded above by

$$\sqrt{1-\delta}.$$

How's it perform (vol I)?

Regular grids, plain, old, vanilla problems

- The Laplace Operator:

Stencil	Convergence	Complexity	Time	Setup
	per cycle		per Cycle	Times
5-pt	0.054	2.21	0.29	1.63
5-pt skew	0.067	2.12	0.27	1.52
9-pt (-1,8)	0.078	1.30	0.26	1.83
9-pt (-1,-4,20)	0.109	1.30	0.26	1.83

- Anisotropic Laplacian: $-\varepsilon U_{xx} - U_{yy}$

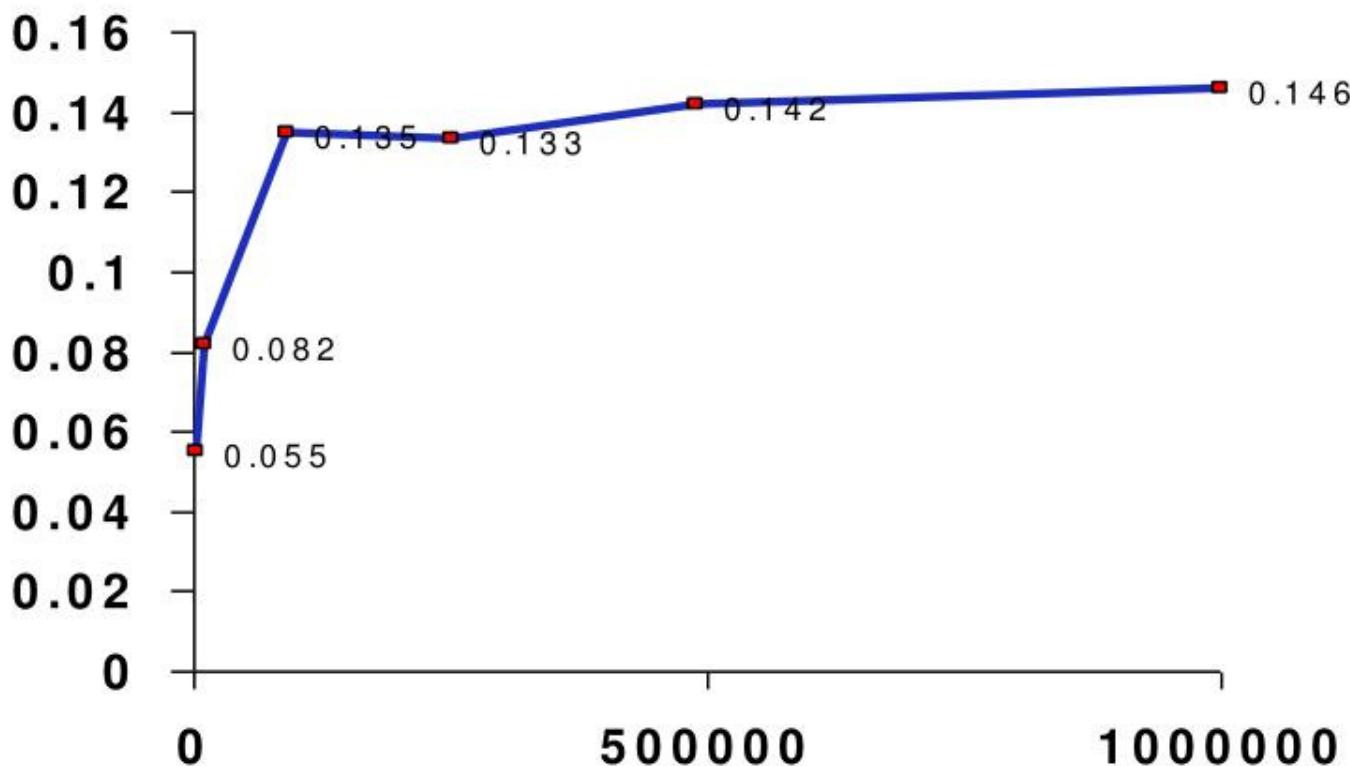
Epsilon	0.001	0.01	0.1	0.5	1	2	10	100	1000
Convergence/cycle	0.084	0.093	0.058	0.069	0.056	0.079	0.087	0.093	0.083

How's it perform (vol II)?

Structured Meshes, Rectangular Domains

- 5-point Laplacian on regular rectangular grids

Convergence factor (y-axis) plotted against number of nodes (x-axis)

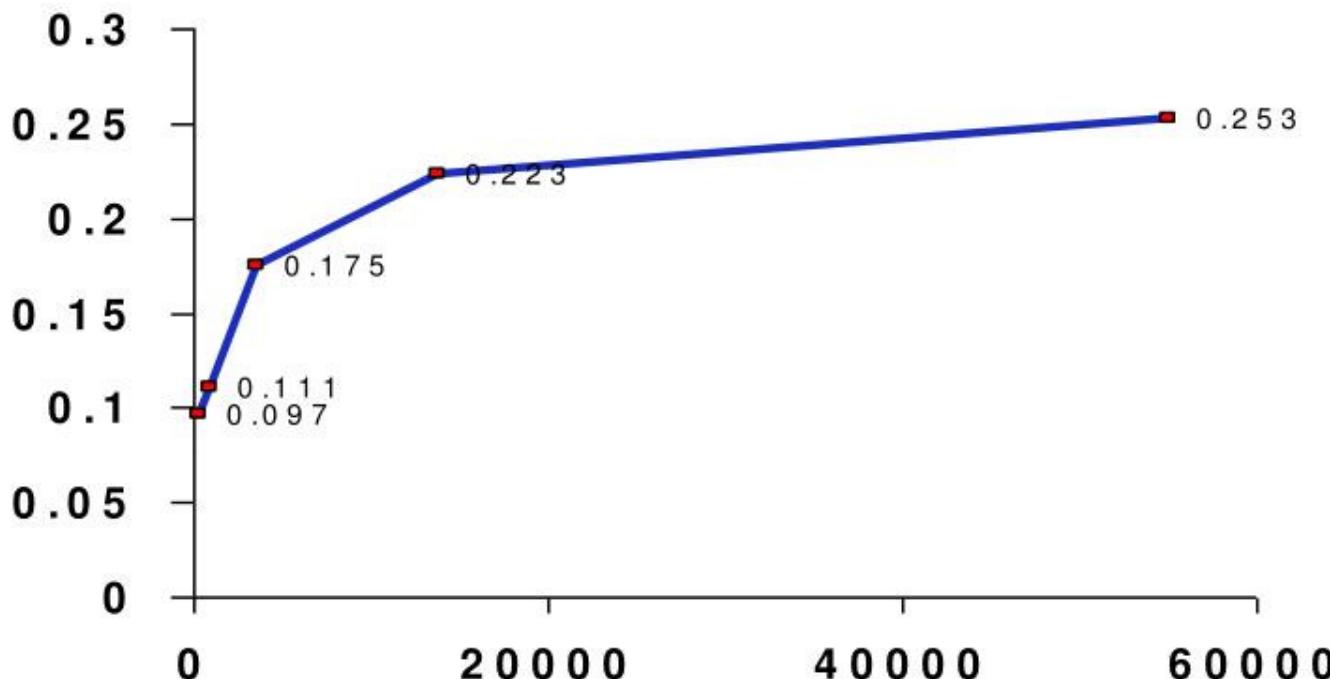


How's it perform (vol III)?

Unstructured Meshes, Rectangular Domains

- **Laplacian on random unstructured grids** (regular triangulations, 15-20% nodes randomly collapsed into neighboring nodes)

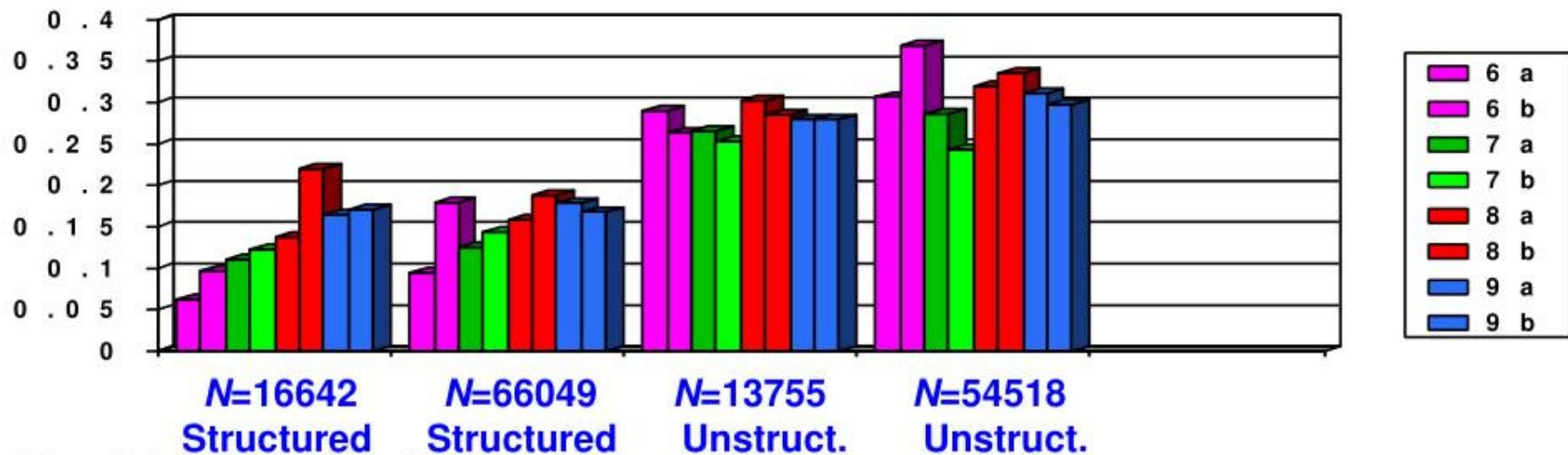
Convergence factor (y-axis) plotted against number of nodes (x-axis)



How's it perform (vol IV)?

Isotropic diffusion, Structured/Unstructured Grids

$\nabla \bullet (d(x,y) \nabla u)$ on structured, unstructured grids



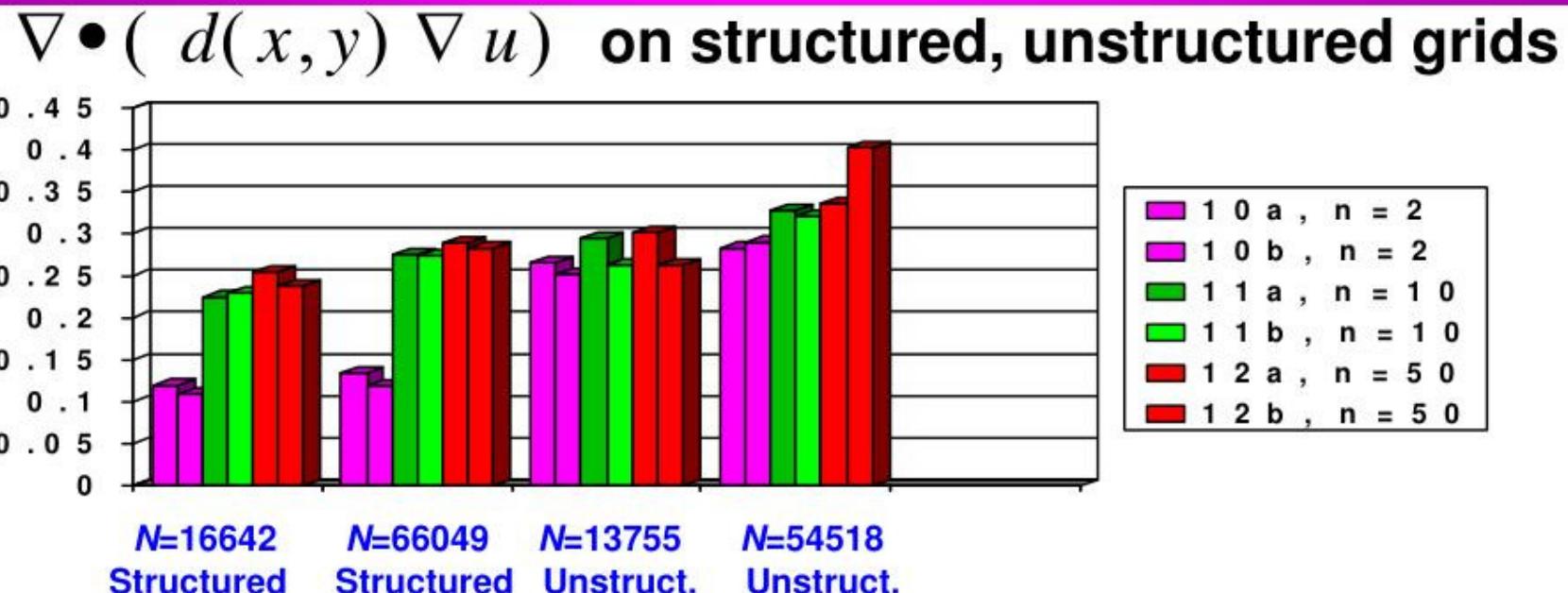
Problems used: "a" means parameter $c=10$, "b" means $c=1000$

$$6: d(x,y) = 1.0 + c |x - y| \quad 8: d(x,y) = \begin{cases} 1.0 & 0.125 \leq \max\{|x - 0.5|, |y - 0.5|\} \leq 0.25 \\ c & \text{otherwise} \end{cases}$$

$$7: d(x,y) = \begin{cases} 1.0 & x \leq 0.5 \\ c & x > 0.5 \end{cases} \quad 9: d(x,y) = \begin{cases} 1.0 & 0.125 \leq \sqrt{(x - 0.5)^2 + (y - 0.5)^2} \leq 0.25 \\ c & \text{otherwise} \end{cases}$$

How's it perform (vol IVa)?

Isotropic diffusion, Structured/Unstructured Grids



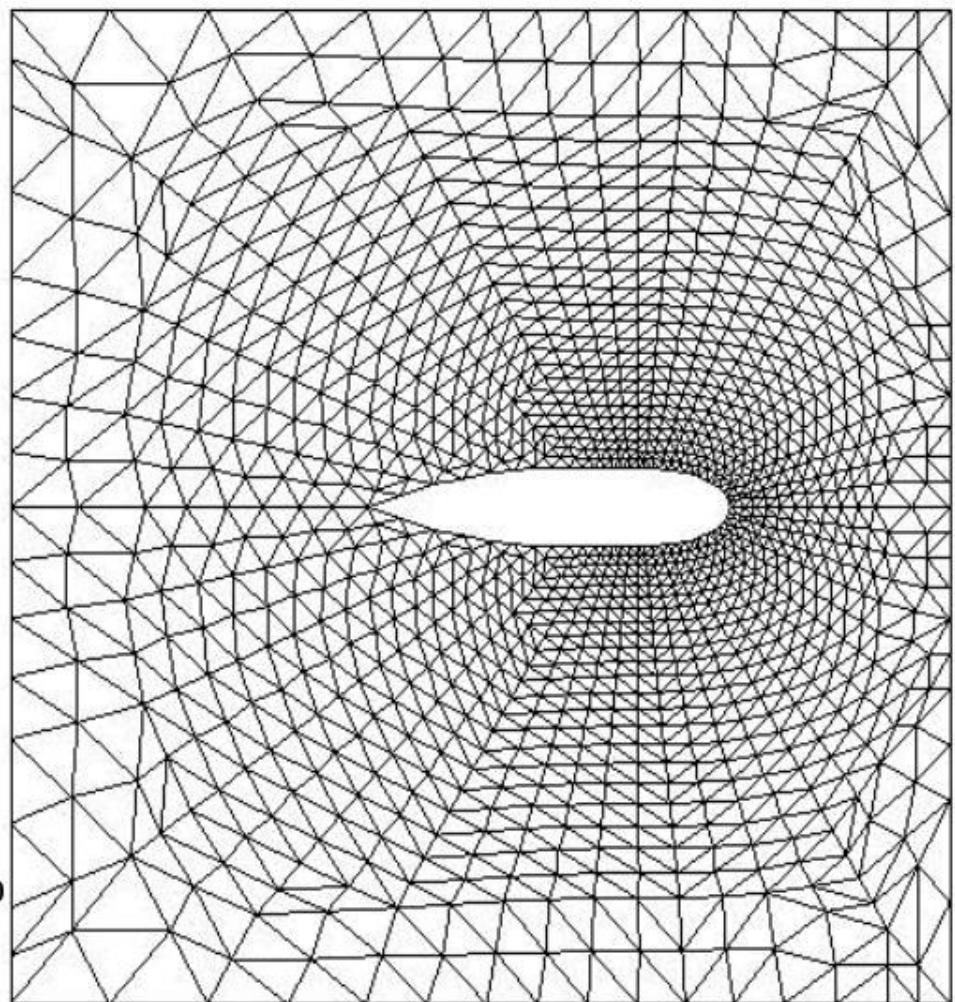
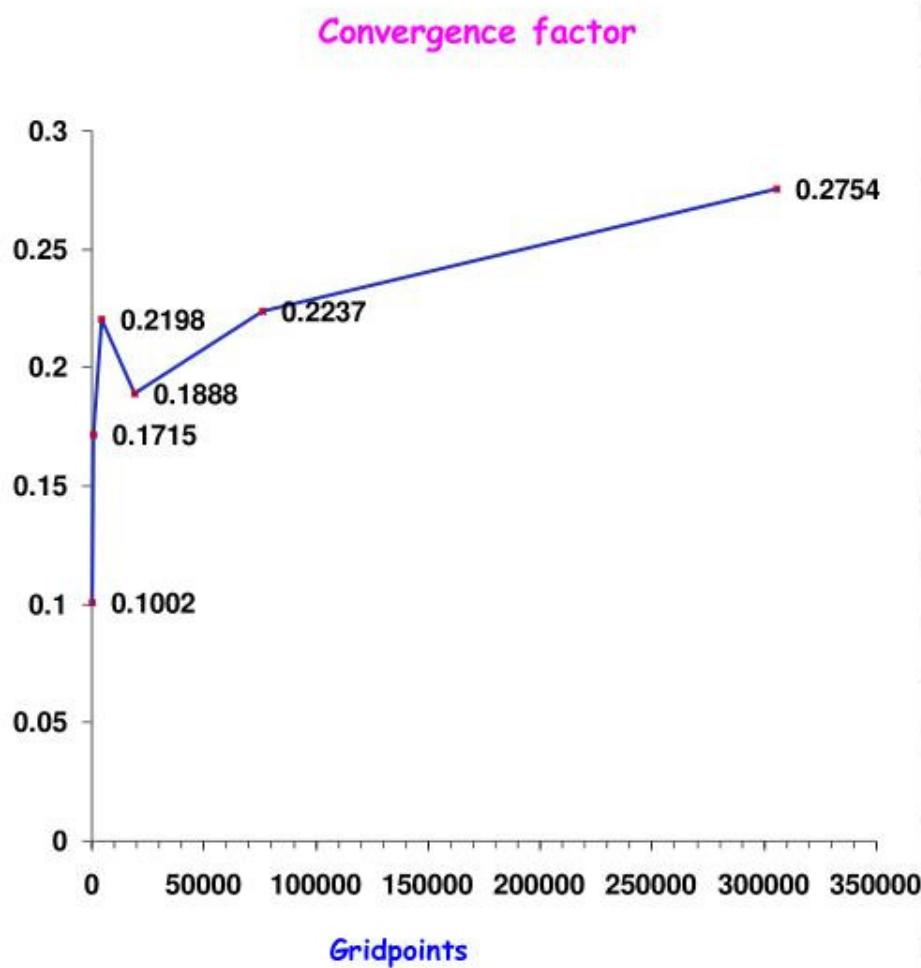
Problem used: "a" means parameter $c=10$, "b" means $c=1000$

"Checkerboard" of coefficients 1.0 and c , squares sized $1/n$:

$$d(x, y) = \begin{cases} 1.0 & \frac{i}{n} \leq x < \frac{i+1}{n}, \frac{j}{n} \leq y < \frac{j+1}{n}, i+j \text{ even} \\ c & \frac{i}{n} \leq x < \frac{i+1}{n}, \frac{j}{n} \leq y < \frac{j+1}{n}, i+j \text{ odd} \end{cases}$$

How's it perform (vol V)?

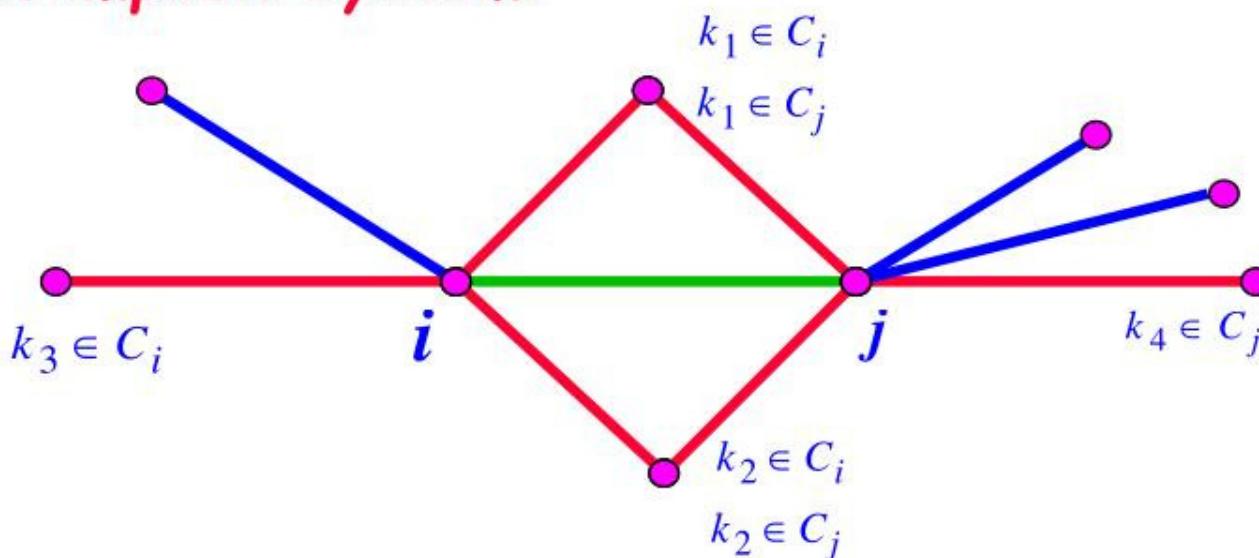
Laplacian operator, unstructured Grids



So, what could go wrong?

Strong F-F connections: weights are dependent on each other

- For point i the value e_j is interpolated from k_1, k_2 , and is needed to make the interpolation weights for approximating e_i .
- For point j the value e_i is interpolated from k_1, k_2 , and is needed to make the interpolation weights for approximating e_j .
- *It's an implicit system!*



Is there a fix?

- A Gauss-Seidel like **iterative** approach to weight definition is implemented. Usually two passes suffice. But does it work?
- Frequently, it does:
Convergence factors for
Laplacian, stretched quadrilaterals

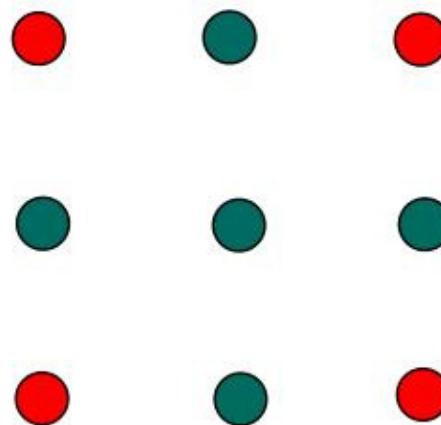
$$\Delta x = 10 \Delta y$$

$$\Delta x = 100 \Delta y$$

theta	Standard	Iterative
0.25	0.47	0.14
	0.24	0.14
0.5	0.83	0.82
	0.53	0.23

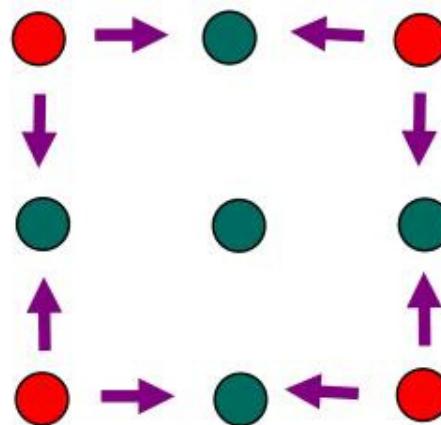
Another Fix: indirect interpolation (see Stueben's text for detail)

- The 5-point problem cannot give “full” coarsening because the F-point in the middle has no connection to any of the 4 C-points. Hence, there is no way to interpolate its value.



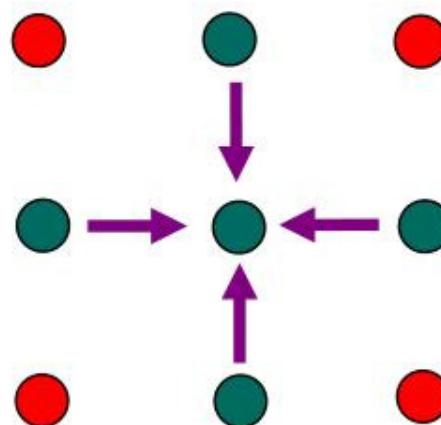
Another Fix: indirect interpolation (see Stueben's text for detail)

- Full coarsening could be achieved by indirect interpolation.
- First interpolate the F-points from the C-points.



Another Fix: indirect interpolation (see Stueben's text for detail)

- Full coarsening could be achieved by indirect interpolation.
- First interpolate the F-points from the C-points.
- Then interpolate the “middle” from the F-points.



- Similar treatment could be applied whenever F-F dependencies arise.

AMG for systems

- How can we do AMG on systems?

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}$$

- Naïve approach: “Block” approach (block Gauss-Seidel, using scalar AMG to “solve” at each cycle)

$$u \leftarrow (A_{11})^{-1}(f - A_{12}v)$$

$$v \leftarrow (A_{22})^{-1}(g - A_{21}u)$$

- Great Idea! Except that it doesn't work! (relaxation does not evenly smooth errors in both unknowns)

AMG for systems: a solution

- To solve the system problem, allow interaction between the unknowns at all levels:

$$A^k = \begin{pmatrix} A_{11}^k & A_{12}^k \\ A_{21}^k & A_{22}^k \end{pmatrix} \quad \text{and} \quad I_{k+1}^k = \begin{pmatrix} (I_{k+1}^k)_u & 0 \\ 0 & (I_{k+1}^k)_v \end{pmatrix}$$

- This is called the “*unknown*” approach.
- Results: 2-D elasticity, uniform quadrilateral mesh:

mesh spacing	0.125	0.0625	0.03135	0.015625
Convergence factor	0.22	0.35	0.42	0.44

So, what *else* can go wrong? *Ouch!* Thin body elasticity!

- Elasticity, 3-d, thin bodies!

$$u_{xx} + \frac{1-\nu}{2}(u_{yy} + u_{zz}) + \frac{1+\nu}{2}(v_{xy} + w_{xz}) = f_1$$

$$v_{yy} + \frac{1-\nu}{2}(v_{xx} + v_{zz}) + \frac{1+\nu}{2}(u_{xy} + w_{yz}) = f_2$$

$$w_{zz} + \frac{1-\nu}{2}(w_{xx} + w_{yy}) + \frac{1+\nu}{2}(u_{xz} + v_{yz}) = f_3$$

- Slide surfaces, Lagrange multipliers, force balance constraints:

$$\begin{pmatrix} S & T \\ U & V \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} F_1 \\ F_2 \end{pmatrix}$$

- S is “generally” positive definite, V can be zero, $U^T \neq T$.

Wanted:

Good solution
method
for this
problem.

REWARD

Needed: more robust methods for characterizing smooth error

- Consider quadrilateral finite elements on a stretched 2D Cartesian grid ($dx \rightarrow \infty$):

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

- Direction of dependence is not apparent here
- Iterative weight interpolation** will sometimes compensate for mis-identified dependence
- Elasticity problems are still problematic

Scalability is central for large-scale parallel computing

- A code is **scalable** if it can **effectively** use additional computational resources to solve larger problems
- Many specific factors contribute to scalability:
 - architecture of the parallel computer
 - parallel implementation of the algorithms
 - **convergence rates of iterative linear solvers**

Linear solver convergence can be discussed independent of parallel computing, and is often overlooked as a key scalability issue.

In Conclusion, AMG Rules!

- Interest in AMG methods is high, and probably still rising, because of the increasing importance of terra-scale simulations on unstructured grids.
- *AMG has been shown to be a robust, efficient solver on a wide variety of problems of real-world interest.*
- *Much research is underway to find effective ways of parallelizing AMG, which is essential to large-scale computing.*

Acknowledgment

- This work was performed under the auspices of the U. S. Department of Energy by Lawrence Livermore National Laboratory under contract number:

W-7405-Eng-48.

- Document Release number UCRL-MI-133749