Solving PDEs with Multigrid Methods

Scott MacLachlan

maclachl@colorado.edu

Department of Applied Mathematics, University of Colorado at Boulder

Support and Collaboration

- This work has been supported by the University of Colorado Chancellor's Fellowship program, the DOE SciDAC TOPS program, the Center for Applied Scientific Computing at Lawrence Livermore National Lab, and Los Alamos National Laboratory.
- This work has been performed in collaboration with Steve McCormick, Tom Manteuffel, John Ruge, and Marian Brezina at CU-Boulder, Rob Falgout at CASC, and J. David Moulton at LANL.

Outline

- Modern Scientific Computing
- Multigrid Methods
- Self-Correcting Multigrid Methods
- Current and Future Work

Why Compute?

- Interested in modeling physical processes
  - Diffusion (Heat, Energy, Chemical)
  - Fluid Flow
  - Particle Transport
  - Elastic Materials
- Can describe these processes through differential equations (both ODEs and PDEs)
- Cannot write down closed form solutions
- Need to find (approximate) solutions in other ways

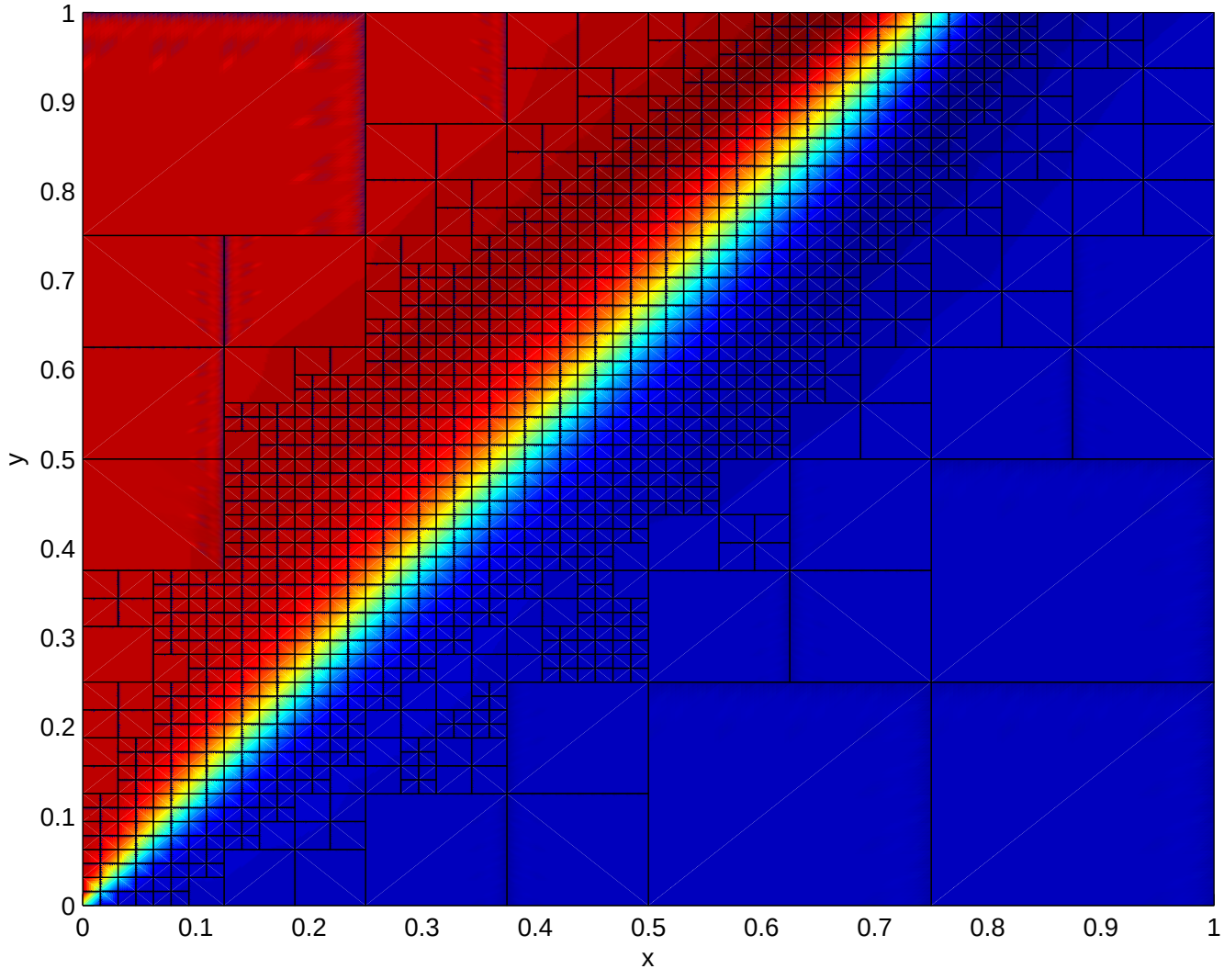
Scientific Computation

- Interested in simulating complex physical systems with parameters, and hence solutions, which vary on multiple scales
- Accuracy constraints lead to discretizations with tens of millions, or even billions, of degrees of freedom (DOFs)
- Need scalability, both algorithmic and parallel

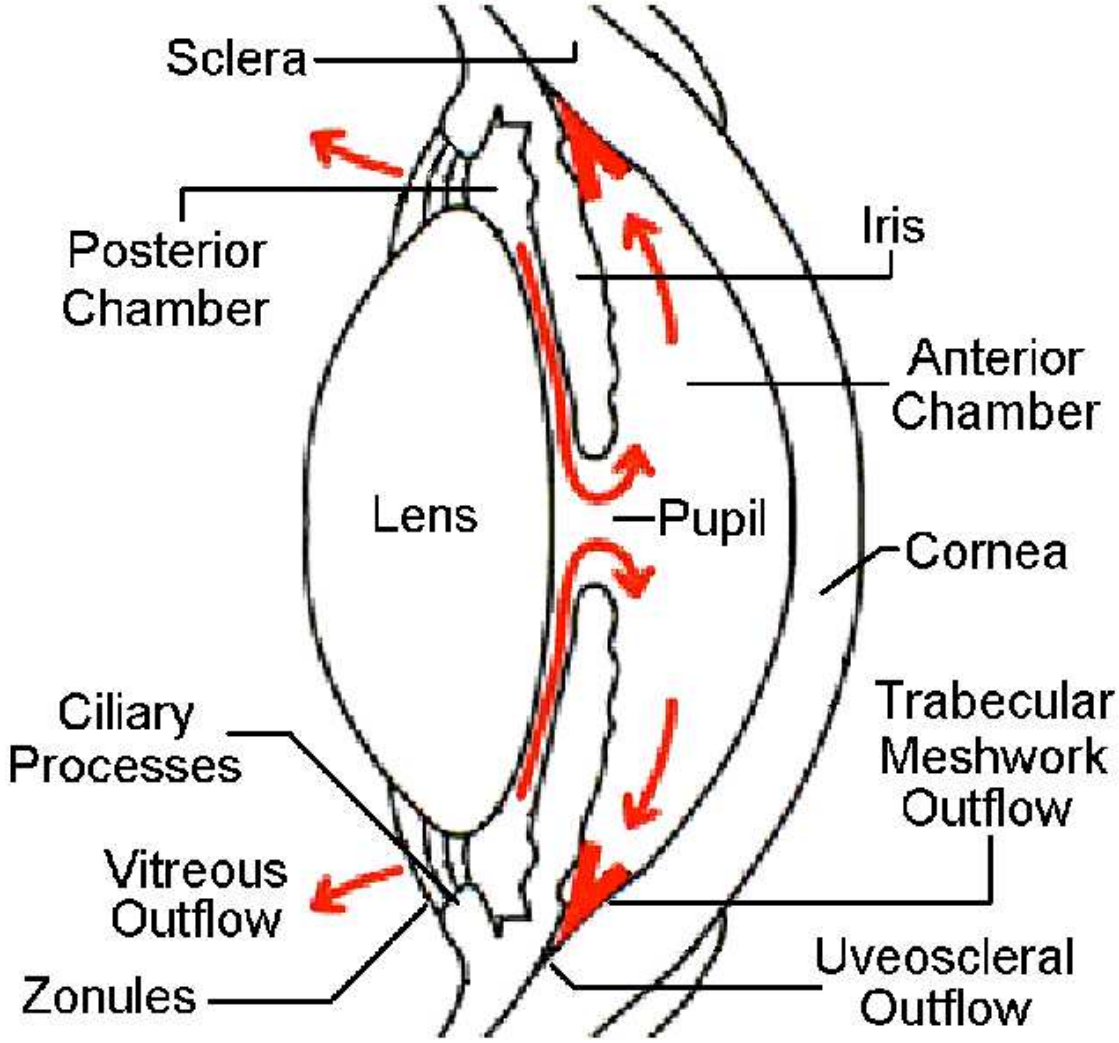
Challenges - Problem Size

- 3D Tsunami Model: 200 million cells, 3 weeks on 1200 processors
- Protein Folding: 18,000 atoms, 10 microsecond simulation, 6 months on 84 processors
- Transport: 500 million - 1 billion degrees of freedom
- Even with optimal methods, three-dimensional problems can be very expensive to solve

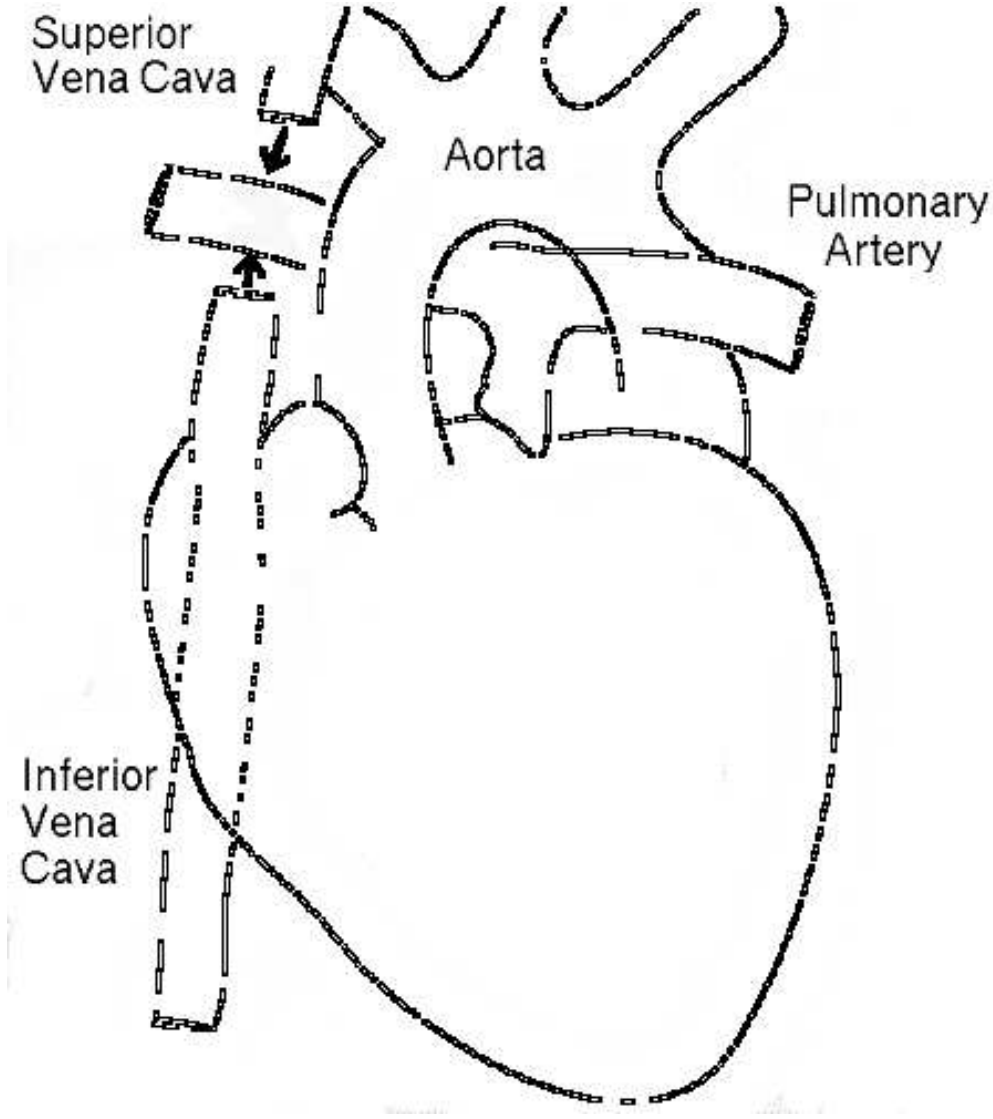
Challenges - Local Refinement



Challenges - Complex Geometry



Challenges - Coupled Systems



Properties of Discretizations

- We consider (primarily) discretizations of the underlying differential equations via finite elements or finite differences
- The matrices from these discretizations tend to be
  - Sparse (number of nonzeros per row doesn't change with n)
  - Ill-conditioned
  - Symmetric (if DE is)
  - Positive-Definite (if DE is)

Direct Methods

- Interested in solving $Ax = b$
- Gauss Elimination involves factoring linear system into an upper- and a lower-triangular part
- Naive cost is $O(N^3) = O(n^9)$ for a 3-dimensional problem
- Utilizing bandedness of our discretization matrix can reduce cost to $O(n^7)$

Stationary Iterative Methods

- Stationary iterative methods choose approximations $B = A^{-1}$ and iterate using the error equation
- If $Ae = A(x - x^k) = b - Ax^k$, then $x^{k+1} = x^k + B(b - Ax^k)$
- The Jacobi iteration chooses B to be the diagonal of A
- The Gauss-Seidel iteration chooses B to be the lower-triangular part of A
- SOR chooses B to try and minimize $\rho(I - BA)$

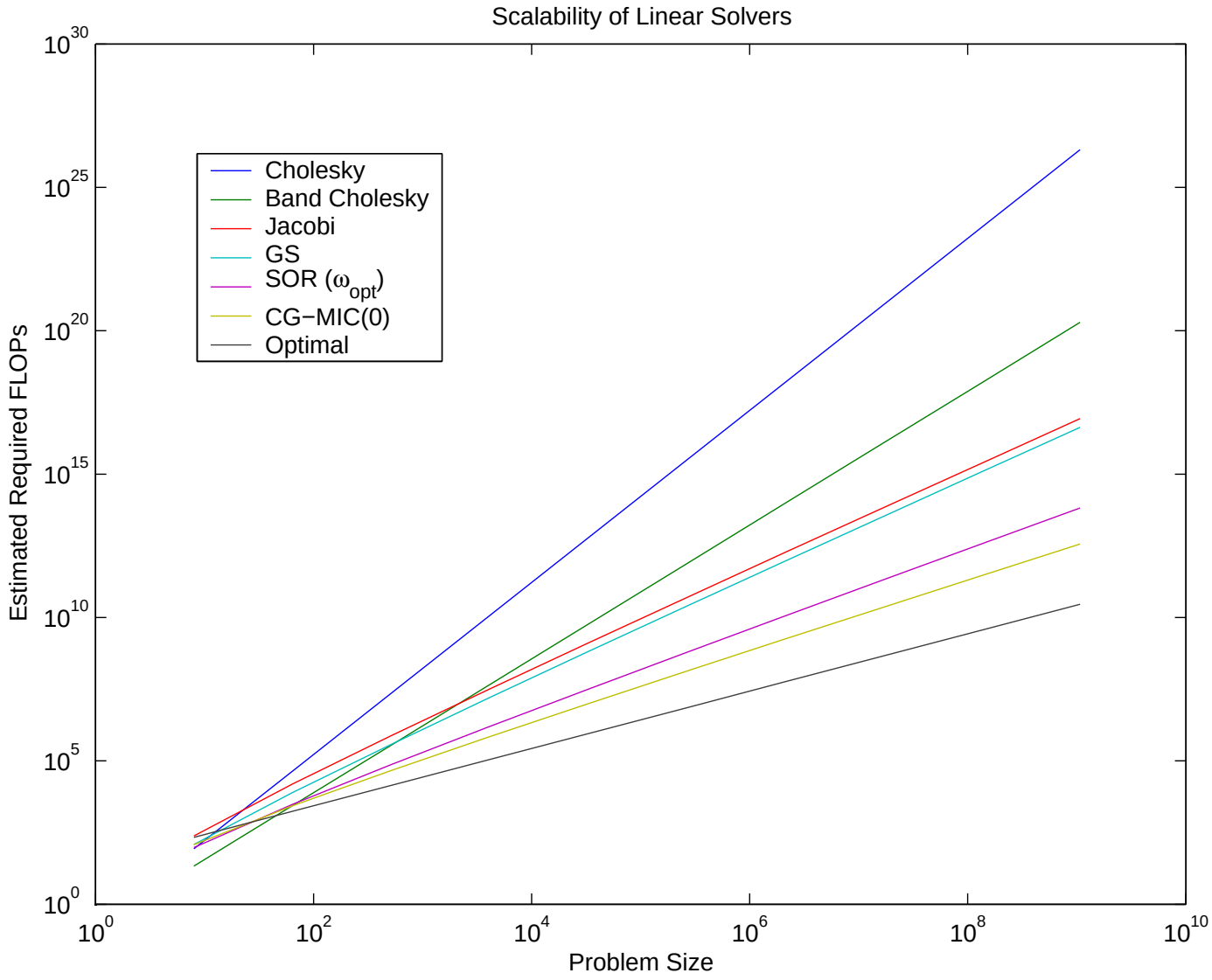
Stationary Iterative Methods . . .

- Jacobi and Gauss-Seidel converge to the level of discretization error in $O(n^5)$ operations for the 3-dimensional Poisson problem
- SOR with an optimal parameter choice converges in $O(n^4)$ operations

Krylov Methods

- Krylov methods find the optimal approximation to the solution in a given subspace
- Iteratively increase the size of the subspace to improve accuracy
- For 3D Poisson, the Conjugate Gradient algorithm converges in $O(n^4)$ operations (without any parameter choice)

Scalability



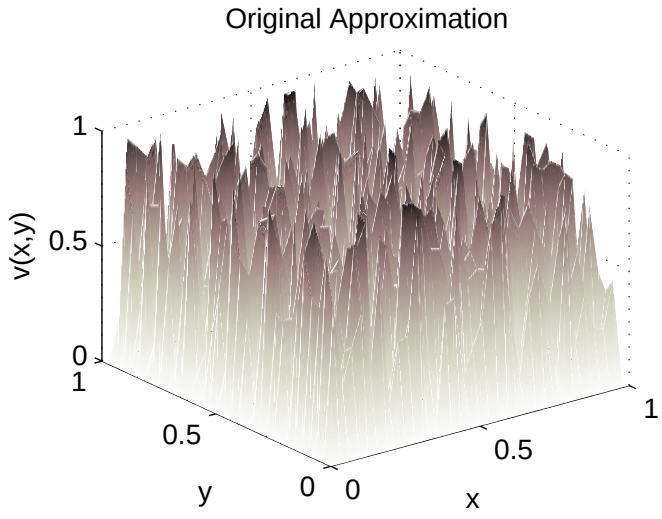
Scalability

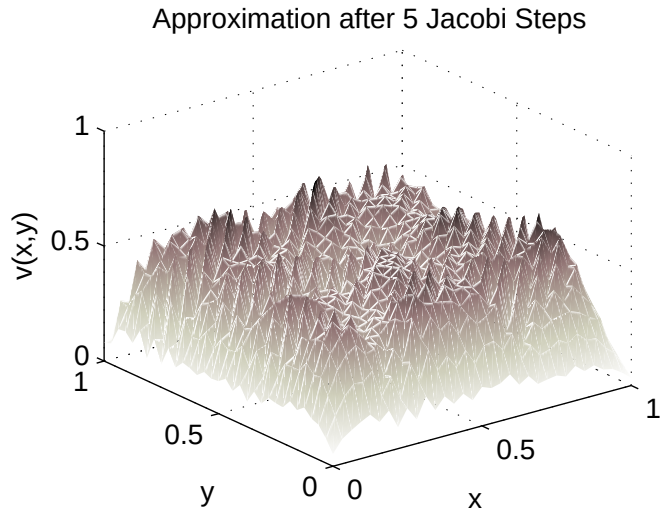
- Because the problems we look to solve are so large, even the cost of $O(N^{\frac{4}{3}})$ is too much
- If $n = 1000$, then $N^{\frac{4}{3}} = 10^{12}$
- An algorithm is said to be scalable (or fast) if it requires only $O(N)$ or $O(N \log N)$ operations
- We must have scalable algorithms in order to solve problems of interest at resolutions of interest

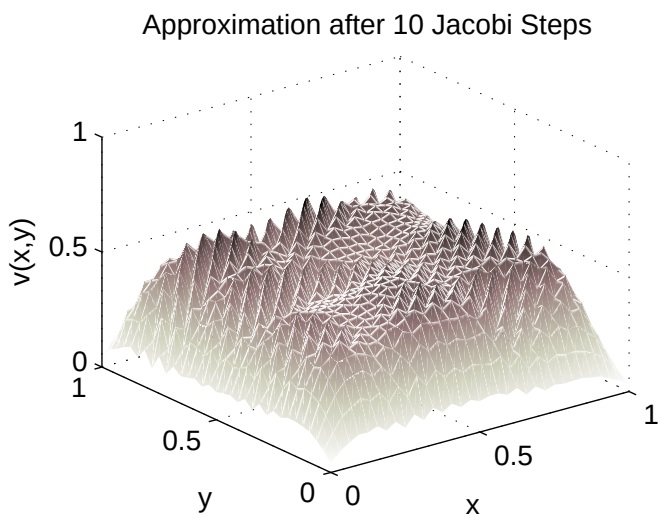
Stationary Iterative Methods . . .

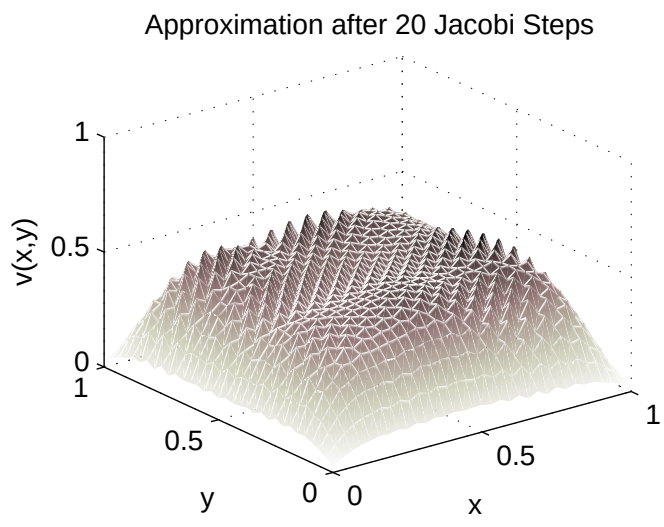
- Jacobi and Gauss-Seidel converge to the level of discretization error in $O(n^5)$ operations for the 3-dimensional Poisson problem
- SOR with an optimal parameter choice converges in $O(n^4)$ operations
- But, Jacobi and Gauss-Seidel resolve some components much faster than others
- In particular, for Poisson the geometrically smoothest components of u are the slowest to be resolved
- For this reason, Jacobi and Gauss-Seidel are often called smoothers - they smooth the error in the approximation

Smoother Performance









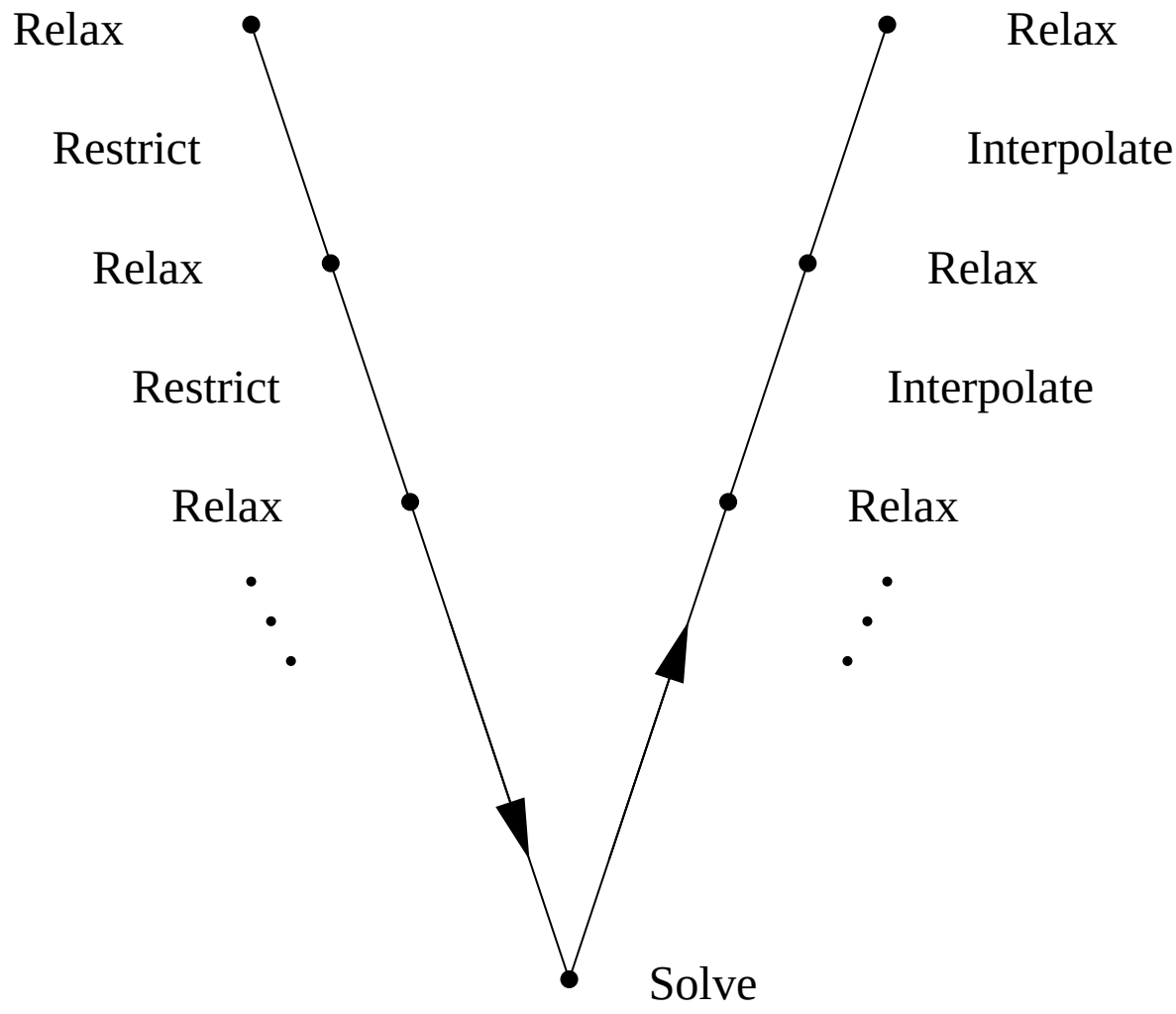
Complementing Relaxation

- If the error left after relaxing is smooth, it can be accurately represented using fewer degrees of freedom
- Problems with fewer degrees of freedom can be solved with less effort
- Error which appears smooth across many degrees of freedom is oscillatory when represented on fewer degrees of freedom

Multigrid Basics

- Multigrid methods obtain optimal efficiency through complementarity
- Use a smoothing process (such as Gauss-Seidel) to eliminate oscillatory errors
- Use coarse grid correction to eliminate smooth errors
- Obtain optimal efficiency through recursion

The V-Cycle

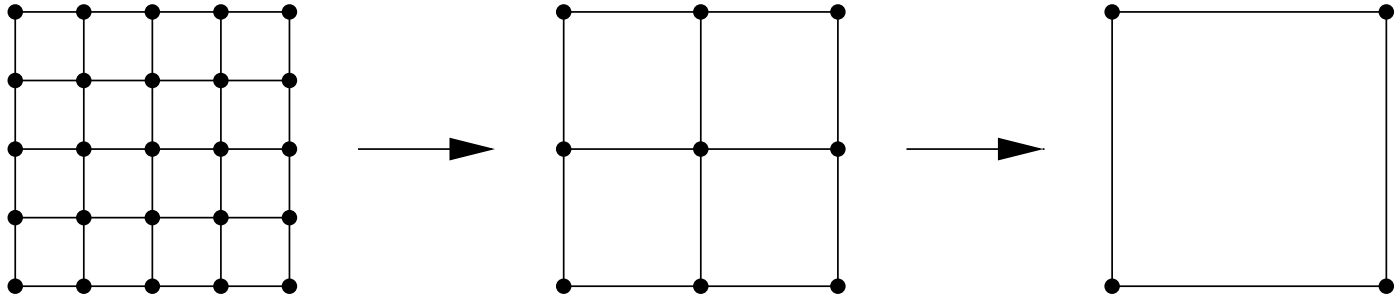

Relax
Restrict
Relax
Restrict
Relax
Solve
Relax
Interpolate
Relax
Interpolate
Relax

Multigrid Operators

- Multigrid V-Cycle requires transfers of residuals and corrections from one grid to the next
- Accomplished through Interpolation (Prolongation) and Restriction operators
- Often pick a form of interpolation (P) and take restriction $R = P^T$ (theoretical benefits)
- Smoothing on coarse grids requires operators on those grids
- These operators must well-approximate the fine grid operator

Geometric Multigrid

- Multigrid algorithms can be broadly classified by how they pick their coarse grids
- If we start with a geometrically regular grid, coarse grids can easily be chosen



Geometric Multigrid

- Interpolation that is accurate for geometrically smooth functions is easy to choose
- Can use linear/bilinear/trilinear averaging to get values at fine-grid points that are not also coarse-grid points
- Restriction can be chosen either by simply taking the fine-grid values at coarse-grid points (injection), or as the transpose of interpolation
- Coarse grid equations can be chosen by rediscretizing the PDE on the coarser grid or . . .

Variational Multigrid

- Multigrid with $R = P^T$ and $A_c = RAP$ is called a *variational formulation*
- Terminology comes from minimization form of $Ax = b$:

$$F(v) = \frac{1}{2}\langle Av, v\rangle - \langle b, v\rangle$$

$$x = \arg\min_{v \in \mathcal{H}} F(v)$$

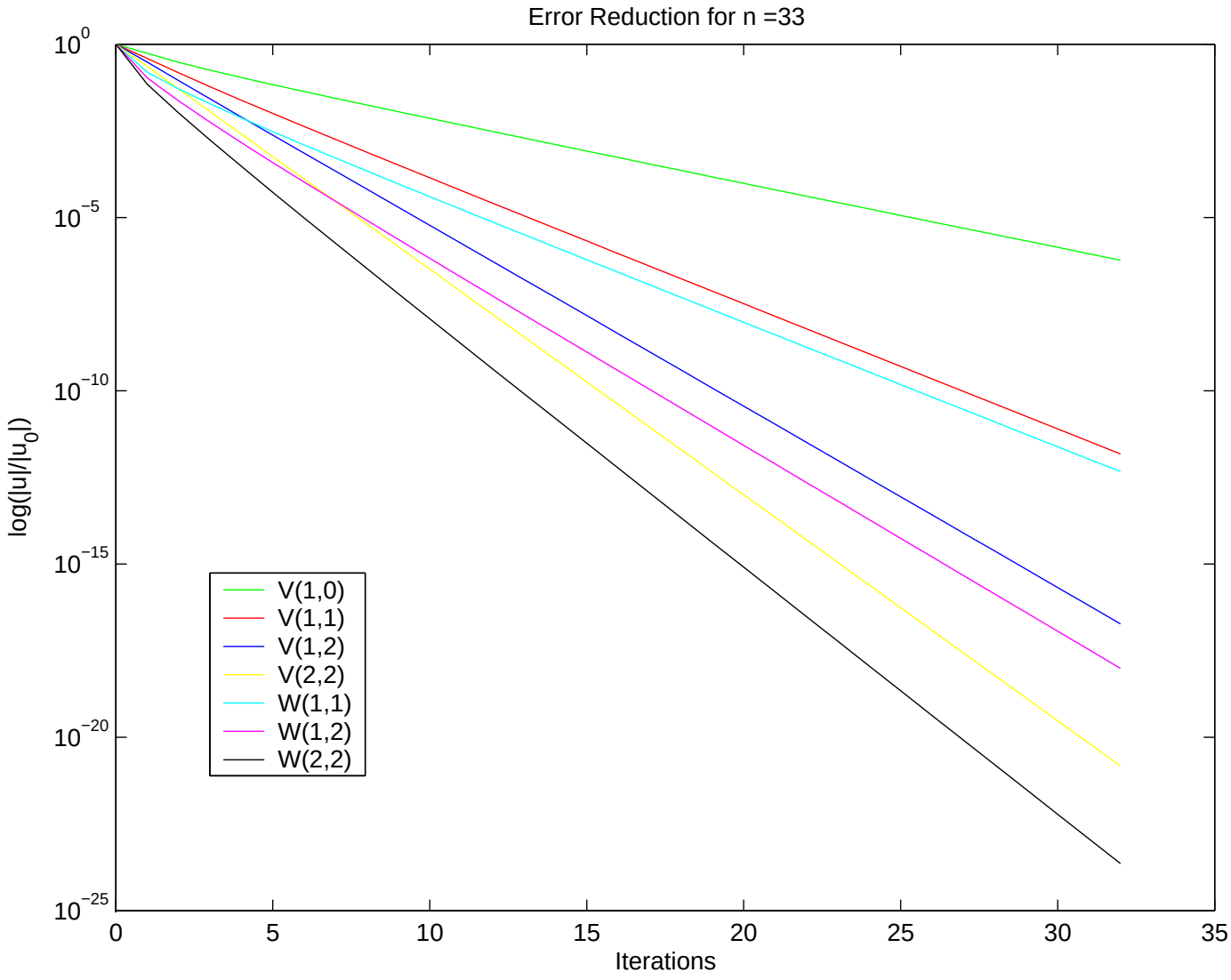
- Given an approximation v to the solution on the fine level, it can be shown that the optimal coarse grid correction Pw solves

$$(P^T AP)w = P^T(b - Av)$$

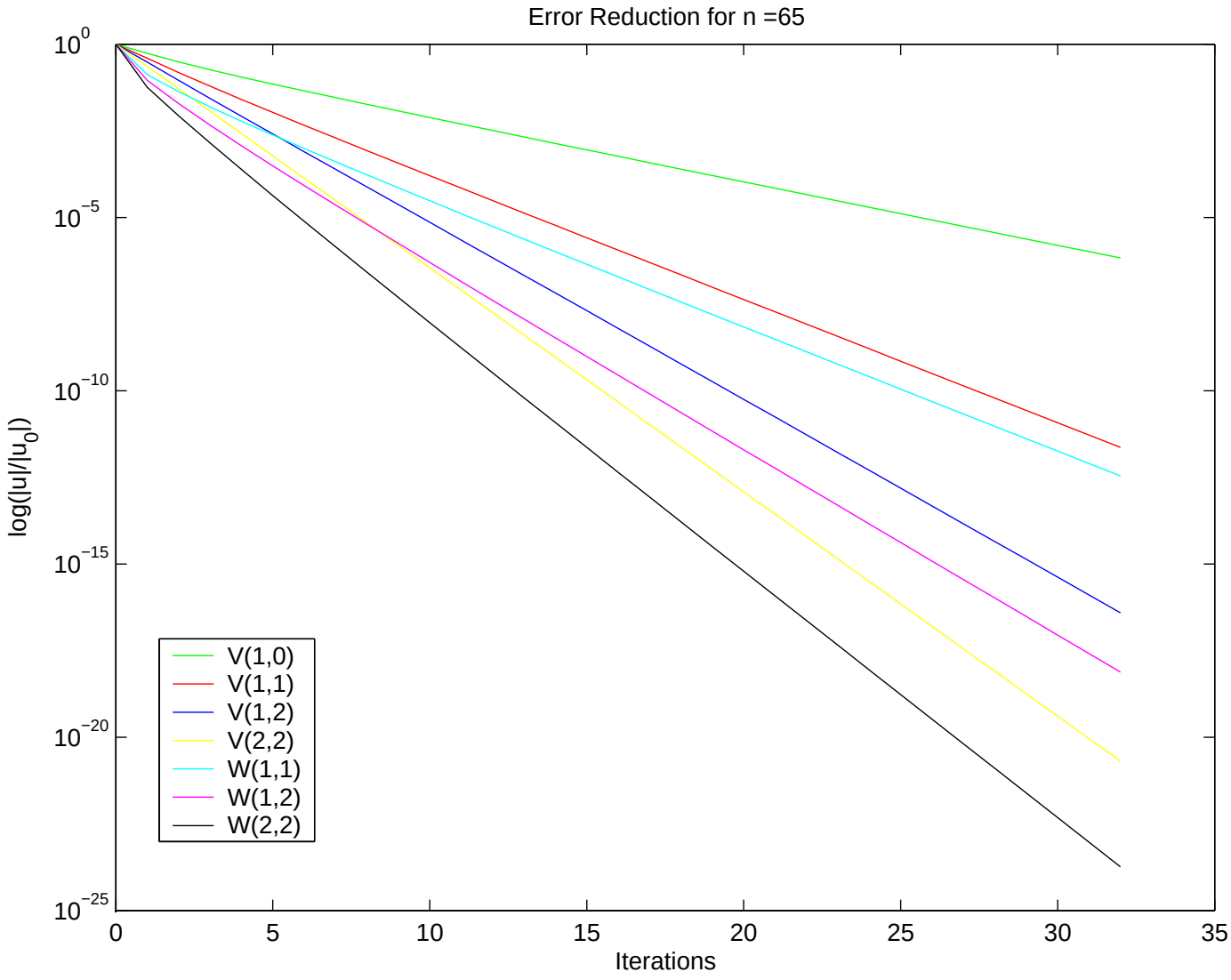
Theoretical Results

- Convergence in a fixed, finite number of V-cycles for finite differences
- Convergence in a fixed, finite number of V-cycles for finite element discretizations for H^1-elliptic operators

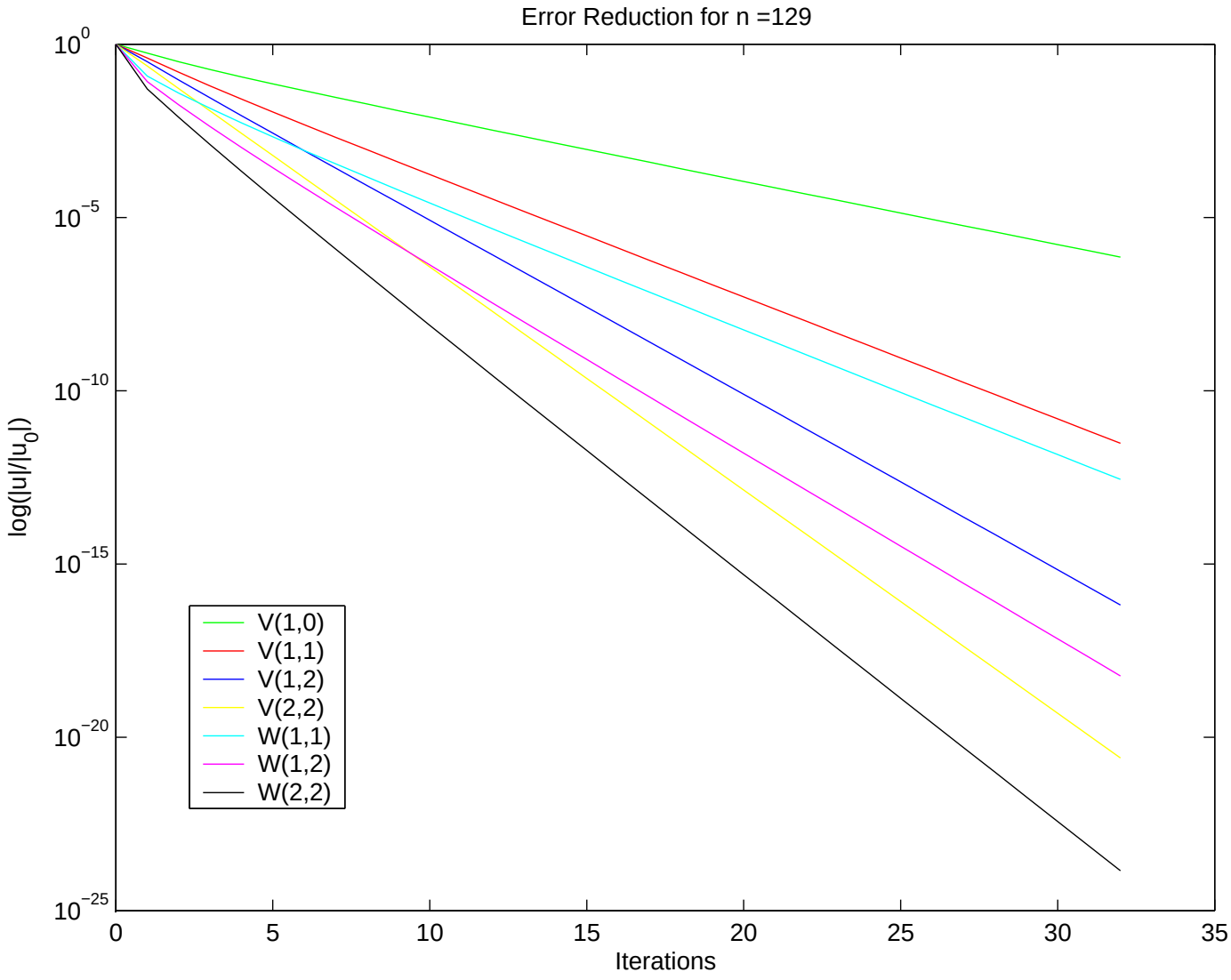
Numerical Results



Numerical Results



Numerical Results



Complications

- Difficult to work out appropriate interpolation for arbitrary geometries
- Some problems don’t have associated geometry (e.g. graph problems)
- Linear interpolation is not appropriate across material boundaries (discontinuities in PDE coefficients)
- Linear interpolation is inefficient in cases of strong anisotropy or convection

Philosophy

- All of the above problems can be solved by tweaking the standard, geometric multigrid algorithm
- Different smoothers and different interpolations can be used
- Each problem requires its own tuning
- Instead, we concentrate on developing an algorithm which is nearly-optimal on a larger number of problems

Algebraic Multigrid

- In the absence of geometric information, choices must be made based on algebraic information
- Interpolation and coarse grids must be chosen based on the ability to interpolate a suitable correction
- Coarse grid operators must be chosen based on the fine-grid operator - Galerkin coarsening may be the most natural choice

Smoothness

- Without geometric information, we can't talk about a vector being "smooth" in the same sense
- We define a vector, e, to be algebraically smooth if it is slow to be reduced by relaxation on $Ae = 0$
- For Jacobi, the condition becomes $\langle D^{-1}Ae, Ae\rangle \ll \langle e, Ae\rangle$
- In general, we think of e as being algebraically smooth if $Ae \ll e$

Influence and Dependence

- Classical (Ruge-Stueben) AMG is all about keeping track of how one gridpoint affects another
- Two gridpoints, i and j are said to be strongly connected if a_{ij} is large
- In particular, we say i strongly influences j if

$$|a_{ij}| > \theta \max_{k \neq j} |a_{kj}|$$

- We say i strongly depends on j if

$$|a_{ij}| > \theta \max_{k \neq i} |a_{ik}|$$

Coarsening Heuristics

- An good choice of a coarse grid is one which can be effectively used to complement relaxation
- That is, we want to choose a coarse grid to allow us to correct the algebraically smooth components on the fine grid
- Ideally, to interpolate to a point i, we would want to have values at all points that it strongly depends on
- In practice, this would yield far too many coarse-grid points
- Instead, we say that for each point j that strongly influences i, either j is a coarse grid point or it is itself strongly dependent on one coarse-grid neighbour of i

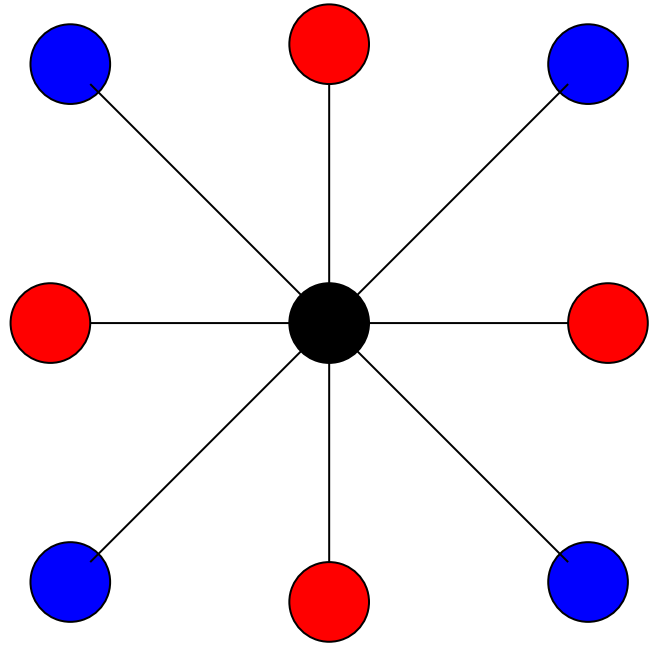
Coarsening Heuristics

- We must also, however, balance the desire for a good interpolation with the need to have a small coarse-grid
- To do this, we insist that the set of coarse points is a maximal subset of the fine-grid such that no coarse-grid point strongly depends on another coarse-grid point
- Implementing these heuristics is accomplished using a colouring algorithm

Defining Interpolation

- For each fine-grid point, i, we want to interpolate its values from neighbouring coarse-grid points

N_i, the Neighbourhood of i



Fine Grid Points

Coarse Grid Points

Defining Interpolation

- For each fine-grid point, i, we want to interpolate its values from neighbouring coarse-grid points
- We consider an interpolation operator that must be accurate for algebraically smooth components, so we start by considering $Ae = 0$, or

$$a_{ii}e_i = -\sum_{j \in C_i} a_{ij}e_j - \sum_{k \in F_i} a_{ik}e_k$$

- Must get rid of connections to points $k \in F_i$

Defining Interpolation

- Points $k \in F_i$ can be either strongly or weakly connected to i
- If k is weakly connected, it isn't important in interpolation, so collapse to the diagonal (i.e. consider $e_k \approx e_i$)
- If k is strongly connected, then we've ensured it is strongly dependent on something in C_i
- So, approximate e_k by a weighted average of $e_j, j \in C_i$

$$e_k \approx \frac{\sum_{j \in C_i} a_{kj} e_j}{\sum_{j \in C_i} a_{kj}}$$

Improvements

- Resulting algorithm can easily handle jumps in coefficients
- No need to know underlying geometry
- Can be adapted to handle anisotropy
- Can be modified to handle more complicated problems, e.g. Elasticity, Stokes Flow, Maxwell's Equations, Hyperbolic PDEs, . . .

Numerical Results

- We start with 2 test problems on $[0,1]^2$, both from bilinear FE discretizations
- Problem 1 is Poisson with pure Dirichlet Boundary Conditions
- Problem 2 is $-\nabla \cdot D(x,y)\nabla p(x,y) = 0$ with Dirichlet BCs on the left and right and Neumann BCs on top and bottom, and

$$D(x,y) = \begin{cases} 10^2 & (x,y) \in [\frac{1}{3}, \frac{2}{3}]^2 \\ 1 & \text{otherwise} \end{cases}$$

Numerical Results

Convergence Factors for AMG

h	Problem 1	Problem 2
$1/32$	0.09	0.14
$1/64$	0.10	0.13
$1/128$	0.14	0.16
$1/256$	0.13	0.15
$1/512$	0.15	0.21

Complications

- Each new type of problem requires a new adaptation
- Coupled Systems become complicated - should tune AMG to each piece of the system
- Very hard to predict what tuning will be necessary
- Many knobs to turn

AMG Assumptions

- Algebraic Multigrid methods attempt to mimic geometric methods in their choices of interpolation operators and coarse grids
- Typically use a fixed, pointwise relaxation scheme
- Classical (Ruge-Stueben) AMG assumes that algebraically smooth error varies slowly along strong connections
- This is equivalent to assuming that algebraically smooth error is essentially (locally) constant

AMG Weaknesses

- AMG assumes the slowest-resolved components are near-constant
- For standard (e.g. finite difference, Galerkin FE) discretizations of scalar differential operators this is usually true
- If discretizations are non-standard or the resulting matrices are scaled, AMG cannot achieve good performance

Importance of Interpolation

- Complementarity is key in multigrid - error components that are not quickly reduced by relaxation must be reduced by coarse-grid correction
- A component can only be corrected from the coarse-grid if it is properly interpolated from that grid
- Interpolation must be most accurate for components that relaxation is slowest to resolve

Choosing Interpolation

- Seek to define interpolation to fit an algebraically smooth vector
- Algebraic smoothness means

$$(Ae)_i \approx 0$$

$$\text{or } a_{ii}e_i \approx -\sum_{j \in N_i} a_{ij}e_j$$

$$= -\sum_{j \in C_i} a_{ij}e_j - \sum_{k \in F_i} a_{ik}e_k$$

- To define interpolation, need to collapse connections from F_i to C_i

Choosing Interpolation . . .

- Seek to define interpolation to fit an algebraically smooth vector
- If $k \in F_i$ is connected to a set of $j \in C_i$, we want to write

$$e_k = \sum_{j \in C_i} w_{kj} e_j$$

- Then, using the definition of algebraic smoothness, we have

$$a_{ii} e_i \approx - \sum_{j \in C_i} a_{ij} e_j - \sum_{k \in F_i} a_{ik} e_k$$

$$a_{ii} e_i \approx - \sum_{j \in C_i} a_{ij} e_j - \sum_{k \in F_i} \sum_{j \in C_i} a_{ik} w_{kj} e_j$$

Choosing w_{kj}

- If we have a vector, $x^{(1)}$, such that $(Ax^{(1)})_k \approx 0$ and so

$$a_{kk}x_k^{(1)} = -\sum_{j \in C_i} a_{kj}x_j^{(1)} - \sum_{j \notin C_i} a_{kj}x_j^{(1)}$$

- Eliminate extra terms by replacing matrix entry a_{kk} with arbitrary d_{kk}

$$d_{kk}x_k^{(1)} = -\sum_{j \in C_i} a_{kj}x_j^{(1)}$$

Choosing w_{kj} ...

- Taking the value of d_{kk} given here, we can write

$$x_k^{(1)} = -\sum_{j \in C_i} \frac{a_{kj}}{d_{kk}} x_j^{(1)} = \sum_{j \in C_i} \frac{a_{kj} x_k^{(1)}}{\sum_{j' \in C_i} a_{kj'} x_{j'}^{(1)}} x_j^{(1)}$$

- Use this formula to collapse all algebraically smooth error

$$e_k = \sum_{j \in C_i} \left(\frac{a_{kj} x_k^{(1)}}{\sum_{j' \in C_i} a_{kj'} x_{j'}^{(1)}} \right) e_j = \sum_{j \in C_i} w_{kj} e_j$$

Adaptive Interpolation

So, we define interpolation to a fine grid point i as

$$e_i = -\sum_{j \in C_i} \frac{a_{ij} + \sum_{k \in F_i} a_{ik} w_{kj}}{a_{ii}} e_j$$

$$= -\sum_{j \in C_i} \frac{a_{ij} + \sum_{k \in F_i} a_{ik} \left(\frac{a_{kj} x_k^{(1)}}{\sum_{j' \in C_i} a_{kj'} x_{j'}^{(1)}} \right)}{a_{ii}} e_j$$

Relation to Ruge-Stueben

- Ruge-Stueben AMG takes $x^{(1)} = 1$
- Substituting this into our interpolation formula gives

$$e_i = -\sum_{j \in C_i} \frac{a_{ij} + \sum_{k \in F_i} a_{ik} \left(\frac{a_{kj}}{\sum_{j' \in C_i} a_{kj'}} \right)}{a_{ii}} e_j$$

- This is the same as the AMG strong-connection-only interpolation formula

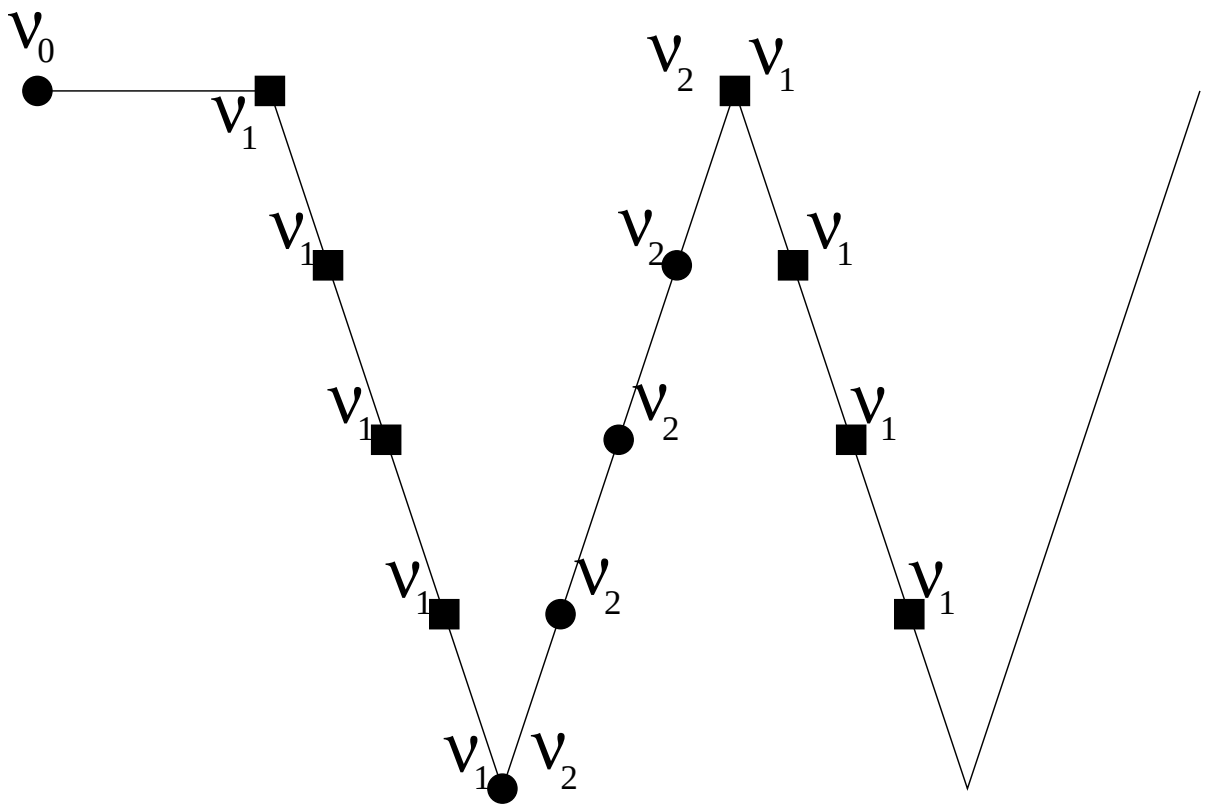
Scaling Invariance

- Combining our interpolation with pointwise relaxation leads to an algorithm that is nearly insensitive to any diagonal scaling
- In particular, if A is scaled to DAD, and $x^{(1)}$ is scaled to $D^{-1}x^{(1)}$, then we achieve the same convergence rates for the scaled problem as for the unscaled problem
- Difficulty lies in generating the scaled vector $D^{-1}x^{(1)}$

Determining $x^{(1)}$

- Choosing a good interpolation operator requires a good approximation, $x^{(1)}$, to the algebraically-smoothest vector of a given matrix A
- Such an approximation could be determined by sufficient relaxation on a random initial guess with a zero right-hand side
- In practice, this requires too much computation to be feasible
- Instead, we use preliminary V-cycles to accelerate the exposure of components for which $Ax \approx 0$

Determining $x^{(1)}$ ...



- In 2D, total cost of relaxation can then be approximated by $\nu_0 + \frac{8}{3}\nu_1 + \frac{4}{3}\nu_2$ work units

Test Problems

- We start with 2 test problems on $[0,1]^2$, both from bilinear FE discretizations
- Problem 1 is Poisson with pure Dirichlet Boundary Conditions
- Problem 2 is $-\nabla \cdot D(x,y)\nabla p(x,y) = 0$ with Dirichlet BCs on the left and right and Neumann BCs on top and bottom, and

$$D(x,y) = \begin{cases} 10^2 & (x,y) \in [\frac{1}{3}, \frac{2}{3}]^2 \\ 1 & \text{otherwise} \end{cases}$$

Test Problems

- The second pair of problems come from diagonally scaling Problems 1 and 2
- To scale, we use the node-wise scaling function

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

- This function gives variable scaling on each node, but does not change its character with h

Numerical Results

- Coarse grids are chosen geometrically, based on full-coarsening
- Coarse grid operators are determined by the Galerkin condition.
- Compute asymptotic convergence factor, then use this to estimate number of V(1,1)-cycles needed to reduce error by 10^{-6}
- From number and cost of cycles ($\frac{8}{3}$ work units), can estimate total cost of solution stage

AMG-Equivalent Results

- By fixing $x^{(1)} = 1$, we can generate results indicative of AMG's performance

Work Units for standard AMG

h	Problem 1	Problem 2	Problem 3	Problem 4
$1/32$	12.9	14.5	1297	59.4
$1/64$	13.4	15.6	4075	112.1
$1/128$	13.6	14.9	6122	218.7
$1/256$	13.8	16.4	6122	430.6
$1/512$	13.9	15.2	7350	858.6
$1/1024$	13.9	16.7	7350	1656

Distributing Relaxation

- To choose how to distribute relaxation, we fix the number of work units allotted to the relaxation in the setup phase

$$\nu_0 + \frac{8}{3}\nu_1 + \frac{4}{3}\nu_2 = 12$$

- Best results were achieved for $\nu_0 = 4, \nu_1 = 2, \nu_2 = 2$, with good results also seen for $\nu_0 = 4, \nu_1 = 3, \nu_2 = 0$ and $\nu_0 = 4, \nu_1 = 1, \nu_2 = 4$
- Poor results were achieved with $\nu_0 = 0, \nu_1 = 3, \nu_2 = 3$ and $\nu_0 = 4, \nu_1 = 0, \nu_2 = 6$

Work units for solution

$$\nu_0 = 0, \nu_1 = 3, \nu_2 = 3$$

h	Problem 1	Problem 2	Problem 3	Problem 4
$1/32$	12.9	14.8	12.9	14.7
$1/64$	13.4	15.6	13.5	15.3
$1/128$	13.6	14.7	13.8	15.4
$1/256$	13.8	16.4	13.9	30.3
$1/512$	13.9	24.0	13.9	25.8
$1/1024$	749.0	926.1	103.7	977.2

Work units for solution

$$\nu_0 = 4, \nu_1 = 2, \nu_2 = 2$$

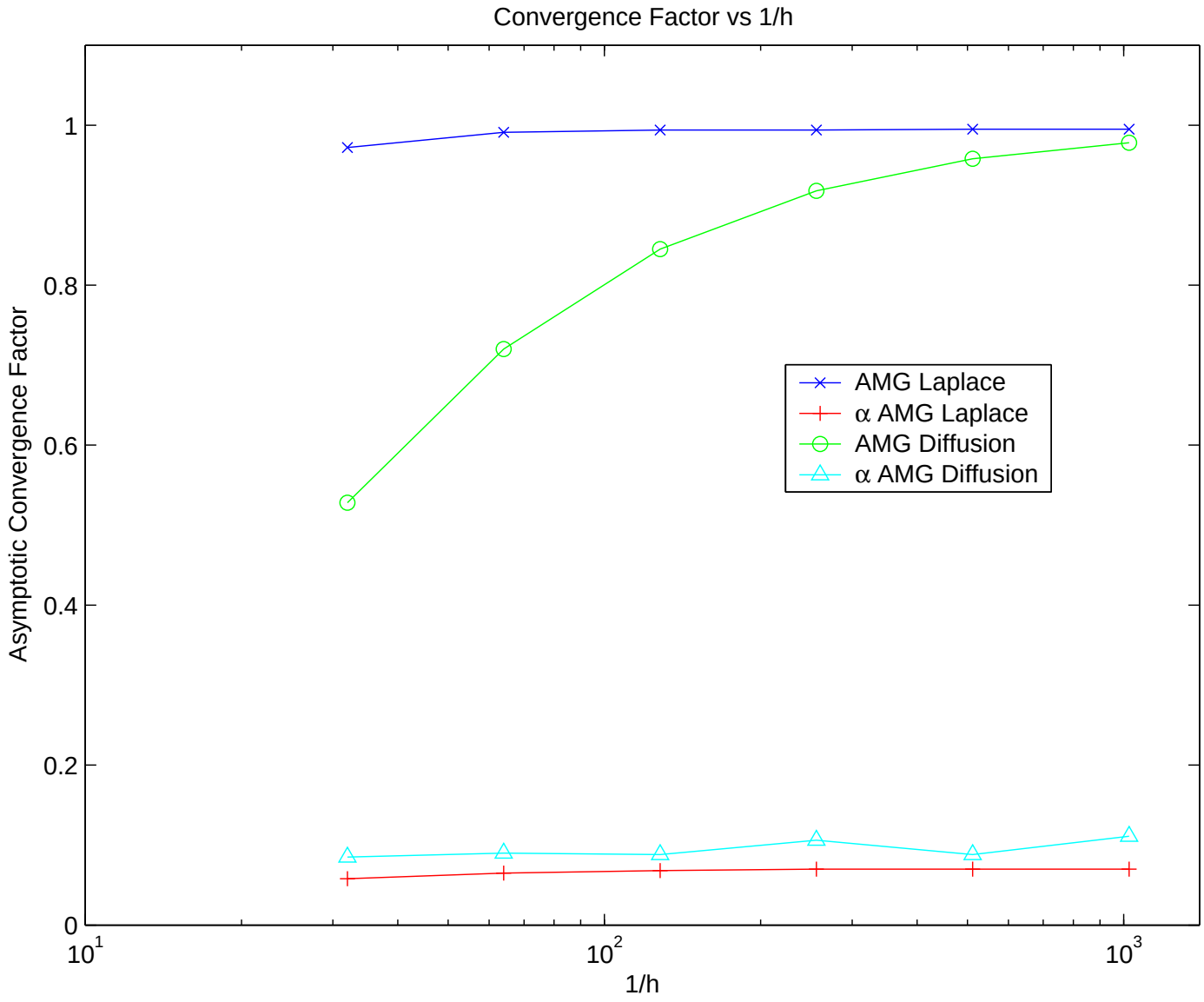
h	Problem 1	Problem 2	Problem 3	Problem 4
$1/32$	12.9	14.7	12.9	14.7
$1/64$	13.4	15.6	13.5	15.4
$1/128$	13.6	14.9	13.7	14.7
$1/256$	13.9	16.4	13.9	25.2
$1/512$	13.9	15.8	13.9	16.4
$1/1024$	13.9	23.2	13.9	25.6

Work units for solution

$$\nu_0 = 6, \nu_1 = 3, \nu_2 = 3$$

h	Problem 1	Problem 2	Problem 3	Problem 4
$1/32$	12.9	14.9	12.9	14.9
$1/64$	13.4	15.6	13.5	15.3
$1/128$	13.6	15.2	13.7	15.3
$1/256$	13.8	16.4	13.8	16.4
$1/512$	13.9	15.2	13.9	15.2
$1/1024$	13.9	16.7	13.9	16.8

Convergence Factors



Current and Future Work

- Developing a theory for self-correcting AMG
- Developing a fully-algebraic version
- Investigating better coarsening procedures (Compatible Relaxation)
- Natural extension to systems
- Alternate smoothers

Summary

- Applications driving need for solvers for large problems
- Classical iterative methods do not scale appropriately for the sizes we are considering
- Multigrid (multiscale) methods do offer optimal efficiency

Summary

- For regular grids, with smooth PDE coefficients, geometric MG works well
- For irregular grids, discontinuous coefficients, algebraic MG works well
- For coupled systems, exotic bases, adaptive algebraic MG offers hope
- All are $O(N)$ algorithms, constants are non-trivial, but not prohibitive