Adapting Algebraic Multigrid

Scott MacLachlan

maclachl@colorado.edu

Department of Applied Mathematics, University of Colorado at Boulder

In collaboration with: Marian Brezina, Rob Falgout, Tom Manteuffel, Steve McCormick, and John Ruge

The Basics

- Need a solver whose performance doesn't significantly degrade as problem size increases
- Multigrid methods obtain optimal efficiency through complementarity
- Use a smoothing process (such as Gauss-Seidel) to eliminate oscillatory errors
- Use a coarse grid correction process to eliminate smooth errors
- Obtain optimal efficiency through recursion

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

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

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 $D^{-1}x^{(1)}$

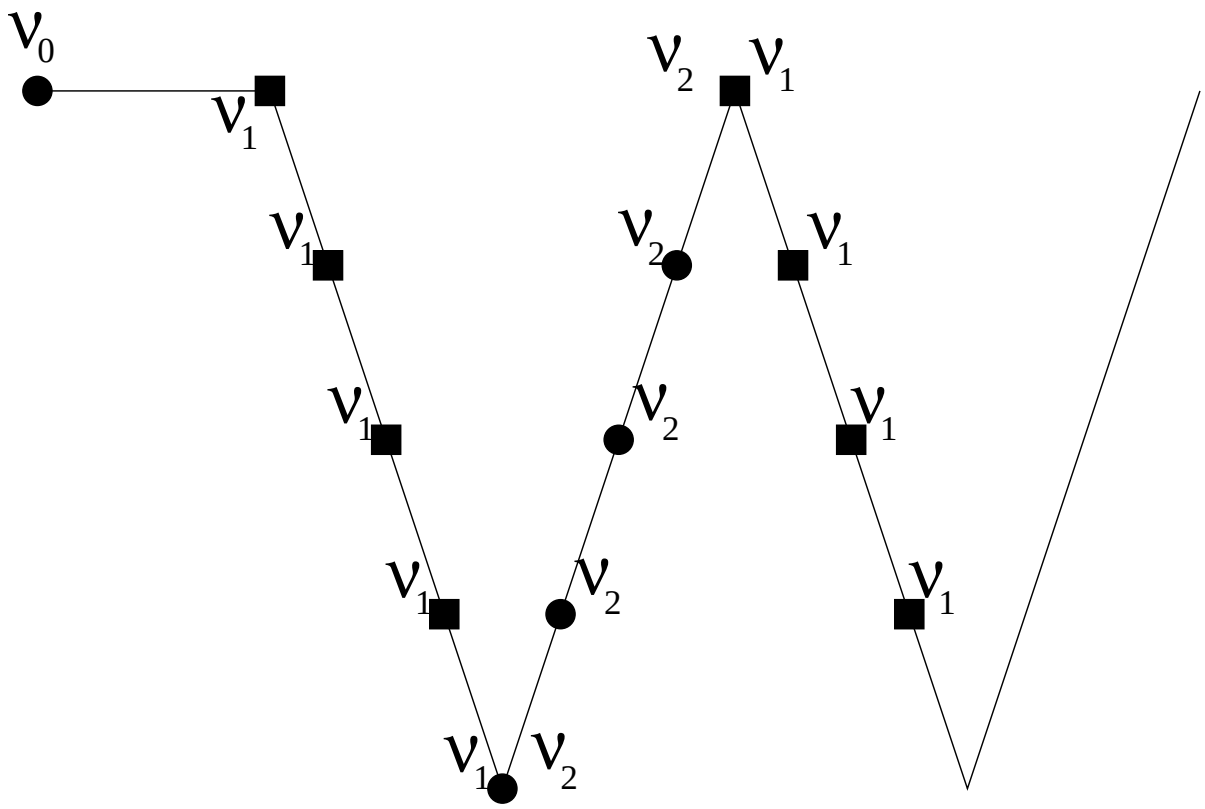
Determining $x^{(1)}$

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

- One relaxation on the fine grid costs 1 work unit
- One relaxation on each grid of a 2D, full-coarsening based V-cycle costs $\frac{4}{3}$ work units
- We chose to study a system where 2 preliminary V-cycles are used to determine $x^{(1)}$, with interpolation and coarse-grid operators computed only on the downward side of the cycle
- We perform ν_0 relaxations on the finest grid, then 2 V-cycles, with ν_1 relaxations on the downward side and ν_2 relaxations on the upward side of the cycle

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.
- Cost of relaxation in setup is then approximated as $\nu_0 + \frac{8}{3}\nu_1 + \frac{4}{3}\nu_2$ work units
- 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

Setup Choices

- Having chosen a 2 V-cycle setup procedure, two choices are needed
  1. The distribution of the relaxation effort between ν_0, ν_1, and ν_2
  2. How much relaxation is necessary for a robust algorithm

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

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

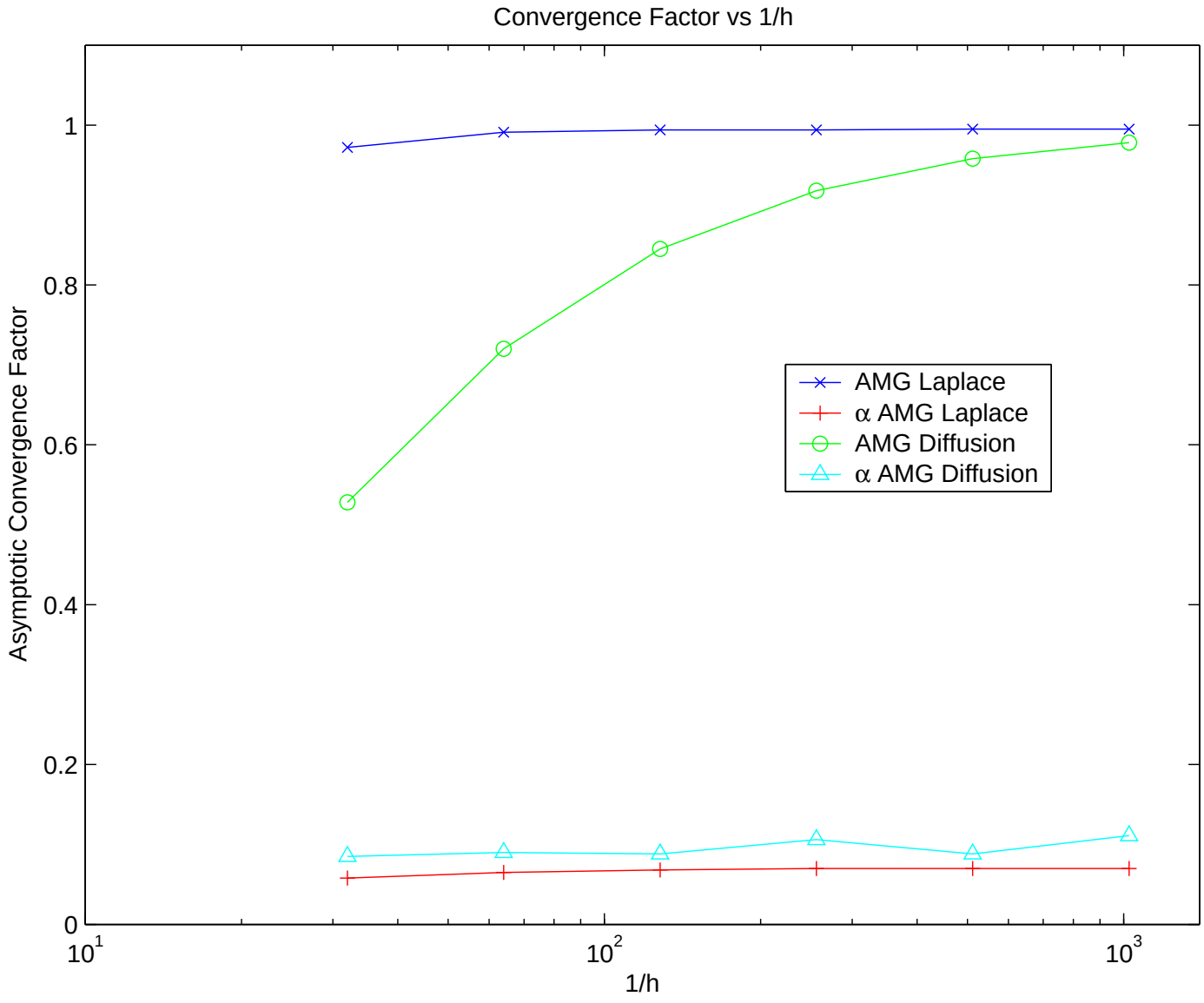
h	Problem 1	Problem 2	Problem 3	Problem 4
$1/32$	12.9	14.7	12.9	14.7
$1/64$	13.4	15.6	13.5	15.4
$1/128$	13.6	14.9	13.7	14.7
$1/256$	13.9	16.4	13.9	25.2
$1/512$	13.9	15.8	13.9	16.4
$1/1024$	13.9	23.2	13.9	25.6

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



Conclusions

- Cost of classical AMG cannot be beat for problems where $x^{(1)} = 1$
- Our interpolation formula does offer an improvement on classical AMG
- Proper distribution and amount of relaxation during setup is crucial to achieving a robust algorithm
- Cost of robustness is not prohibitive

Future Work

- Extension to systems is straight-forward
- Fully algebraic code is under development
- Seek to include more advanced coarsening (e.g. compatible relaxation)
- Consider altering construction to take advantage of more robust smoothing