Adaptive AMG as a Preconditioner

Scott MacLachlan

maclachl@colorado.edu

Department of Applied Mathematics, University of Colorado at Boulder

In collaboration with Tom Manteuffel and Steve McCormick

Overview

- Need a solver whose performance doesn't significantly degrade as problem size increases
- Multigrid methods obtain optimal efficiency through complementarity
- Krylov methods accelerate this performance
- The interaction between Krylov and adaptive multigrid methods may be exploited to attempt to reduce the overall time-to-solution or improve black-box performance

Multigrid

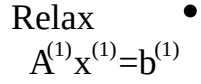
Multigrid Methods achieve optimality through complementarity

Multigrid

Multigrid Methods achieve optimality through complementarity

Multigrid Components

- Relaxation



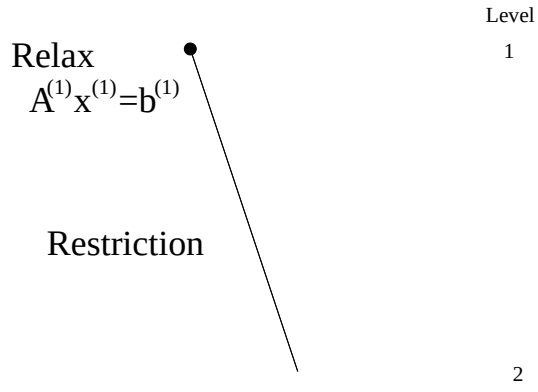
- Use a smoothing process (such as Gauss-Seidel) to eliminate oscillatory errors
- Remaining error satisfies $Ae = r$

Multigrid

Multigrid Methods achieve optimality through complementarity

Multigrid Components

- Relaxation
- Restriction



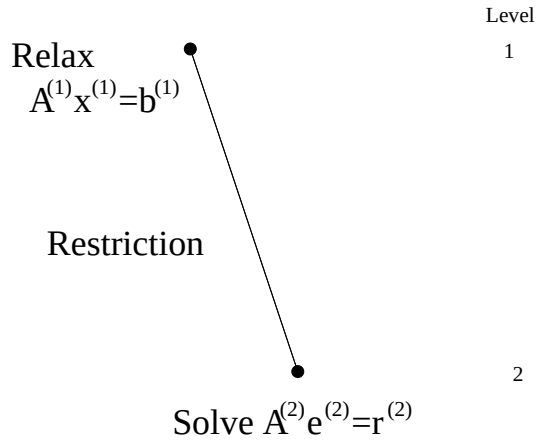
- Transfer residual to coarse grid

Multigrid

Multigrid Methods achieve optimality through complementarity

Multigrid Components

- Relaxation
- Restriction
- Coarse-Grid Correction



- Use coarse-grid correction to eliminate smooth errors
- To solve for error on coarse grid, use residual equation

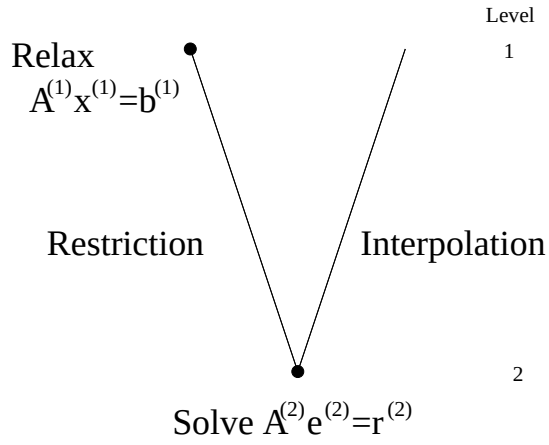
$$A^{(2)}e^{(2)} = r^{(2)}$$

Multigrid

Multigrid Methods achieve optimality through complementarity

Multigrid Components

- Relaxation
- Restriction
- Coarse-Grid Correction
- Interpolation



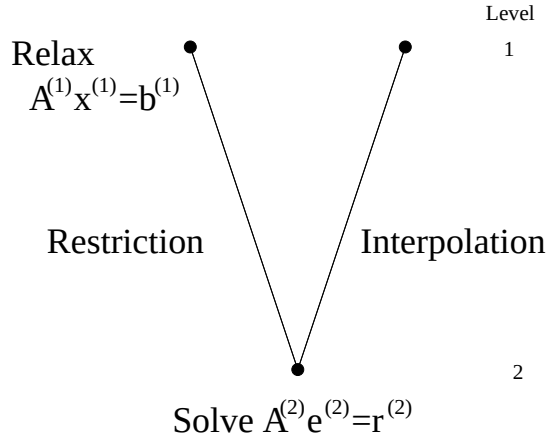
- Transfer correction to fine grid

Multigrid

Multigrid Methods achieve optimality through complementarity

Multigrid Components

- Relaxation
- Restriction
- Coarse-Grid Correction
- Interpolation
- Relaxation



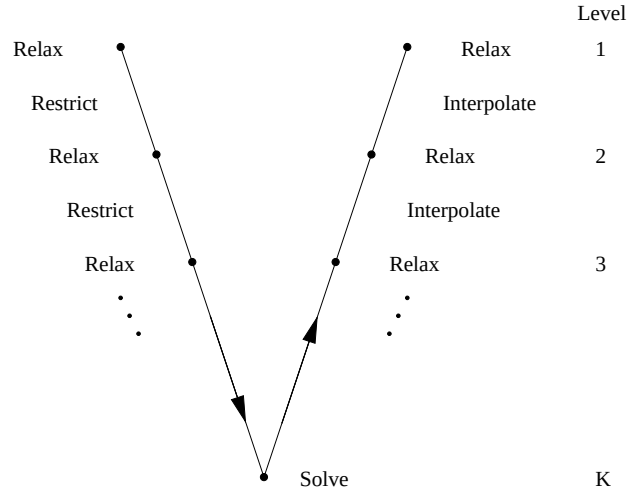
- Relax once again to remove oscillatory error introduced in coarse-grid correction

Multigrid

Multigrid Methods achieve optimality through complementarity

Multigrid Components

- Relaxation
- Restriction
- Coarse-Grid Correction
- Interpolation
- Relaxation



Obtain optimal efficiency through recursion

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
- Classical AMG assumes point-wise relaxation is slow to resolve errors that are locally constant (algebraically-smooth error)
- Interpolation must then be chosen so that it is accurate for such error
- Efficiency of multigrid comes in the coarse-grid correction step where all errors with similar local character are effectively reduced as well.

Algebraic Multigrid Interpolation

- Point-wise relaxation stalls on errors with small residuals - when $Ae \approx 0$
- Given a fine/coarse splitting $\Omega = F \cup C$, this means

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

- Interpolation defined by rewriting e_j for $j \in F_i$ in terms of e_i and e_k, for $k \in C_i$
- Assumption on algebraically-smooth error suggests

$$e_j \approx \frac{\sum_{k \in C_i} a_{jk}e_k}{\sum_{k \in C_i} a_{jk}}$$

Adaptive Multigrid Interpolation

- Seek to eliminate the assumption on the character of algebraically-smooth error
- Replace the weighted average used in collapsing strong F-F connections with one that also accounts for a prototype of algebraically-smooth error

Adaptive Multigrid Interpolation

- Seek to eliminate the assumption on the character of algebraically-smooth error
- Replace the weighted average used in collapsing strong F-F connections with one that also accounts for a prototype of algebraically-smooth error

$$e_j \approx \frac{\sum_{k \in C_i} a_{jk} e_k}{\sum_{k \in C_i} a_{jk}} \rightarrow e_j \approx \frac{\sum_{k \in C_i} a_{jk} v_j e_k}{\sum_{k \in C_i} a_{jk} v_k}$$

Model Problems

- Consider bilinear FE discretization of $-\nabla \cdot \mathcal{K}(x,y)\nabla p = f$ on $[0,1]^2$
  - Problem 1: $\mathcal{K}(x,y) = 1$ (Laplace) with Dirichlet BCs
  - Problem 2: $\mathcal{K}(x,y) = 10^{-8}$ on 20% of elements (chosen randomly) and $\mathcal{K}(x,y) = 1$ otherwise. Neumann BCs along $y = 0$ and $y = 1$, Dirichlet BCs along $x = 0$ and $x = 1$
- Also consider scaled versions of these matrices, $\tilde{A} = DAD$, where $d_{ii} = 10^r$, for r chosen randomly between 0 and 5 (Problems 1r and 2r)

Calibrated AMG performance

Asymptotic Convergence Factor

grid	Prob 1	Prob 1r	Prob 2	Prob 2r
64^2	0.07	0.07	0.19	0.19
128^2	0.07	0.07	0.20	0.21
256^2	0.08	0.08	0.24	0.24
512^2	0.08	0.08	0.29	0.29

Total Time to Solution

64^2	0.04s	0.03s	0.03s	0.05s
128^2	0.25s	0.22s	0.31s	0.27s
256^2	0.89s	0.91s	1.09s	1.22s
512^2	3.52s	3.64s	4.84s	5.35s

Calibrated AMG tuning

- These results are achieved because of hand-tuning of the number of relaxations performed on each level to determine the approximation of algebraically-smooth error used in the definition of interpolation

Pre-relaxations on Finest Grid and Coarse Grids

grid	Prob 1		Prob 1r		Prob 2		Prob 2r	
64^2	4	2	8	4	4	2	11	5
128^2	4	2	10	5	5	3	18	9
256^2	6	3	15	7	8	4	27	14
512^2	9	4	22	11	12	6	43	21

Adaptive approach

- Do a small, fixed number of relaxations on the finest grid to expose algebraically-smooth error
- Form interpolation, inject smooth-error approximation and relax on coarse-grid problem
- Repeat until reach coarsest grid
- On subsequent setup cycles, use current V-cycles instead of relaxation to expose error not adequately reduced by the current method
- Form interpolation to fit this error (if necessary, in addition to the error currently being fit)

Adaptive AMG performance

- Perform 5 iterations of the current solver on each level of each setup cycle
- On first cycle, this is Gauss-Seidel relaxation
- On subsequent cycles, it is the current V-cycle

Setup Iteration Count and Total Time to Solution

grid	Prob 1		Prob 1r		Prob 2		Prob 2r	
64^2	1	0.08s	1	0.07s	1	0.09s	2	0.11s
128^2	1	0.32s	1	0.34s	1	0.41s	2	0.54s
256^2	1	1.29s	2	1.82s	2	2.19s	2	1.96s
512^2	1	5.46s	2	7.29s	2	8.16s	4	12.91s

Setup vs Solution phases

- This approach relies on the separation of a setup phase of the algorithm from the solution of the linear system, $Ax = b$, of interest
- The costs of the setup phase include the computation of the multigrid hierarchy, but also (possibly many) relaxation sweeps
- These relaxations are “lost” - they do not directly contribute to the solution of $Ax = b$

Preconditioning

- Preconditioning $Ax = b$ with an approximate inverse of A, M^{-1}, within CG gives a series of residuals, r_k, and generalized residuals, $s_k = M^{-1} r_k$
- The matrix, $S_k = (s_0, s_1, \ldots, s_k)$, diagonalizes M and tridiagonalizes A
- Using relaxation as a preconditioner for CG applied to $Ax = b$ then also generates a simple system whose eigenvalues are those of an orthogonal section of $M^{-\frac{1}{2}} A M^{-\frac{1}{2}}$

Lanczos as a Smoother

- Solving $S_k A S_k v = \lambda_{\text{min}} S_k M S_k v$ gives the smoothest vector relative to $M^{-1}A$ in the space spanned by $\{s_j\}_{j=0}^{k}$
- This vector, an approximation to the largest eigenmode of $I - M^{-1}A$, represents the component that is slowest to be reduced by the preconditioner
- That component can be used in the same way as the algebraically-smooth error in the adaptive AMG approach

Adaptive Preconditioning

- By allowing the preconditioner (MG), M^{-1}, to vary, we can adapt the overall solution process
- Starting with a simple preconditioner,
  - Run a given number of PCG iterations
  - Find the smoothest vector in the Lanczos process
  - Adapt the preconditioner to better reduce this vector
- Overall efficiency depends on both the performance of the (adapted) preconditioner and the speed at which the Lanczos process converges

PCG-Lanczos Cycle

- Start with no knowledge of algebraically-smooth error, so use relaxation-preconditioned CG on $Ax = b$ to generate initial approximations to x and to the algebraically-smooth error
- Use the smooth-error approximation to create an interpolation operator and coarse-grid problem
- Recurse until sufficiently coarsened, interpolate and add correction to fine-grid solution
- Now, use the MG-preconditioned CG to solve $Ax = b$.

Adaptive PCG-Lanczos Cycle

- If performance is inadequate
  - Use Lanczos to extract new smooth vector (missed by both relaxation and current coarse-grid correction)
  - Readapt MG cycle to account for this error, using the existing MG solver (instead of relaxation) on all levels to expose error not currently being efficiently reduced
- Now test solver as MG-preconditioned CG on $Ax = b$

Numerical Results

- Increment the size of the space for Lanczos by 10 vectors
- Adapt MG whenever a smoother vector (measured in RQ_A) is available
- Assume scalar PDE-based problem, so adaptation is based only on the single smoothest vector available

Adaptation Count and Total Time to Solution

grid	Prob 1		Prob 1r		Prob 2		Prob 2r	
64^2	2	0.11s	2	0.12s	2	0.13s	2	0.12s
128^2	2	0.81s	2	0.82s	2	0.80s	2	0.68s
256^2	3	4.31s	3	3.83s	5	6.65s	5	7.95s
512^2	5	26.50s	4	20.98s	8	49.23s	7	41.68s

Analysis

- Preconditioning approach allows us to construct an efficient multigrid preconditioner while we solve $Ax = b$
- Significant cost is incurred by iterations of poor preconditioners constructed along the way
- In general, smoothing the residual of $Ax = b$ takes more work than smoothing the error in $Ax = 0$
- Adaptations are most effective when preconditioner is very bad - algebraically-smooth error is faster to be resolved

Conclusions

- Hand-tuned setup phase in a calibrated MG algorithm results in scalable solvers
- Adaptive tuning yields a less-efficient algorithm, but still scalable and better than no tuning
- Preconditioning approach allows tuning while solving $Ax = b$, but requires significantly more effort to properly expose algebraically-smooth error in the space $\{s_j\}_{j=0}^{k}$
- For discretized systems of PDEs, such as linear elasticity, an adaptive or preconditioning approach will be crucial to development of a representative set of algebraically-smooth components