Algebraic Multigrid

Presented to
ATPESC 2017 Participants

Ulrike Meier Yang

Q Center, St. Charles, IL (USA)
August 7, 2017
Outline

• Why multigrid methods?
• Algebraic multigrid software
• How does multigrid work?
• Hypre software library – interfaces and solvers
  – Why different interfaces?
  – Solvers and data structures
• Effect of complexity on performance
• Hands-on exercises
Multigrid linear solvers are optimal \( O(N) \) operations, and hence have good scaling potential

- Weak scaling – want constant solution time as problem size grows in proportion to the number of processors
Available multigrid software

• ML, MueLu included in Trilinos
• GAMG in PETsc
• The hypre library provides various algebraic multigrid solvers, including multigrid solvers for special problems e.g. Maxwell equations, ...

• All of these provide different flavors of multigrid and provide excellent performance for suitable problems
• Focus here on hypre
The *hypre* software library provides structured and unstructured multigrid solvers

- Used in many applications
  - Elasticity / Plasticity
  - Electromagnetics
  - Magneto-hydrodynamics
  - Quantum Chromodynamics
  - Facial surgery

- Displays **excellent weak scaling and parallelization properties** on BG/Q type architectures
Multigrid (MG) uses a sequence of coarse grids to accelerate the fine grid solution.

**Multigrid V-cycle**

- **smoothing** (relaxation)
- **restriction**
- **prolongation** (interpolation)

**Error on the fine grid**

**Error approximated on a smaller coarse grid**

**Algebraic multigrid (AMG)** only uses matrix coefficients.

No actual grids!
# AMG Building Blocks

## Setup Phase:

- Select coarse “grids”
- Define interpolation: \( P^{(m)}, \ m = 1,2,... \)
- Define restriction: \( R^{(m)}, \ m = 1,2,..., \) often \( R^{(m)} = (P^{(m)})^T \)
- Define coarse-grid operators: \( A^{(m+1)} = R^{(m)} A^{(m)} P^{(m)} \)

## Solve Phase:

\[
\begin{align*}
\text{Relax } & A^{(m)} u^m = f^m \\
\text{Compute } & r^m = f^m - A^{(m)} u^m \\
\text{Restrict } & r^{m+1} = R^{(m)} r^m \\
\text{Solve } & A^{(m+1)} e^{m+1} = r^{m+1} \\
\text{Interpolate } & e^m = P^{(m)} e^{m+1} \\
\text{Correct } & u^m \leftarrow u^m + e^m
\end{align*}
\]
(Conceptual) linear system interfaces are necessary to provide “best” solvers and data layouts.

**Linear System Interfaces**

- Structured
- Composite
- Block-struc
- Unstruc
- CSR

**Linear Solvers**

- PFMG, ...
- FAC, ...
- Split, ...
- MLI, ...
- AMG, ...

**Data Layouts**

- Structured
- Composite
- Block-struc
- Unstruc
- CSR
Why multiple interfaces? The key points

• Provides natural “views” of the linear system

• Eases some of the coding burden for users by eliminating the need to map to rows/columns

• Provides for more efficient (scalable) linear solvers

• Provides for more effective data storage schemes and more efficient computational kernels
**hypre** supports these system interfaces

- **Structured-Grid** (Struct)
  - logically rectangular grids

- **Semi-Structured-Grid** (SSstruct)
  - grids that are mostly structured
  - Examples: block-structured grids, structured adaptive mesh refinement grids, overset grids
  - Finite elements

- **Linear-Algebraic** (IJ)
  - general sparse linear systems
SMG and PFMG are semicoarsening multigrid methods for structured grids

- **Interface**: Struct, SStruct
- **Matrix Class**: Struct

- SMG uses plane smoothing in 3D, where each plane “solve” is effected by one 2D V-cycle
- SMG is very robust
- PFMG uses simple pointwise smoothing, and is less robust

- Constant-coefficient versions!
Structured-Grid System Interface (Struct)

- Appropriate for scalar applications on structured grids with a fixed stencil pattern
- Grids are described via a global $d$-dimensional *index space* (singles in 1D, tuples in 2D, and triples in 3D)
- A *box* is a collection of cell-centered indices, described by its “lower” and “upper” corners
- The grid is a collection of boxes
- Matrix coefficients are defined via stencils

$$
\begin{bmatrix}
S_4 \\
S_1 \\
S_3 \\
S_0 \\
S_2
\end{bmatrix}
= \begin{bmatrix}
-1 & 4 & -1 \\
-1 & 1 & -1 \\
-1 & 0 & -1 \\
-1 & 3 & -1
\end{bmatrix}
$$
StructMatrix data structure

- Stencil
\[
\begin{pmatrix}
S_4 & S_1 & S_0 & S_2 \\
S_1 & S_0 & S_2 & S_3 \\
\end{pmatrix}
= \begin{pmatrix}
-1 & -1 \\
-1 & 4 & -1 \\
\end{pmatrix}
\]

- Grid boxes: \([(-3,1), (-1,2)] \]
\([ (0,1), (2,4) ]\)

- Data Space: grid boxes + ghost layers:
\([(-4,0), (0,3)] , \([-1,0), (3,5) \]

- Data stored

- Operations applied to stencil entries per box (corresponds to matrix (off) diagonals from a matrix point of view)
BoomerAMG is an algebraic multigrid method for unstructured grids

- **Interface**: SStruct, IJ
- **Matrix Class**: ParCSR

- Originally developed as a general matrix method (i.e., assumes given only $A$, $x$, and $b$)
- Various coarsening, interpolation and relaxation schemes
- Automatically coarsens “grids”
- Can solve systems of PDEs if additional information is provided
- Can also be used through PETSc and Trilinos
ParCSRMatrix data structure

- Based on compressed sparse row (CSR) data structure
- Consists of two CSR matrices:
  - One containing local coefficients connecting to local column indices
  - The other (Offd) containing coefficients with column indices pointing to off processor rows
- Also contains a mapping between local and global column indices for Offd
- Requires much indirect addressing, integer computations, and computations of relationships between processes etc,
Complexity issues

- Coarse-grid selection in AMG can produce unwanted side effects
- Operator (RAP) “stencil growth” reduces efficiency
- Not so much an issue for SMG and PFMG, for which stencil growth is limited (to at most 27 points per stencil in 3D)

- For BoomerAMG we will therefore also consider complexities:
  - Operator complexity: \( C_{op} = \frac{\sum_{i=0}^{L} \text{nnz} (A_i)}{\text{nnz}(A_0)} \)
  - Generally would like this to be less than 2, close to 1
  - Affects flops and memory
- Can ameliorate with more aggressive coarsening
Algebraic multigrid as preconditioner

- Generally algebraic multigrid methods are used as preconditioners to Krylov methods, such as conjugate gradient (CG) or GMRES
- This often leads to additional performance improvements
Hands-on Exercises

- Equation: \( \varphi - \Delta \cdot \beta \nabla \varphi = \text{RHS} \), Dirichlet boundary conditions
- Grid: 128 x 128 x 128, block structured, consisting of (at least) 8 subgrids

RHS: 

solution:
• This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC.

• LLNL-PRES-736166