Energy Minimization in Parallel Setting

Andrey Prokopenko
Jeremie Gaidamour, Jonathan Hu, Ray Tuminaro

16th Copper Mountain Conference on Multigrid Methods
21 March 2013

Sandia National Laboratories
Outline

- Introduction
- Energy-minimization based AMG
  - Motivations
  - Algorithm
- Parallel implementation
- Numerical results
- Conclusion
AMG

- Iterative method for solving linear equations
- Commonly used as a preconditioner
- Idea: capture error at multiple resolutions using grid transfer operator:
  - **Smoothing** damps the oscillatory error (high energy)
  - **Coarse grid correction** reduces the smooth error (low energy)

```
Solving Au = f with initial guess v

Pre-smoothing

Calculate the residual r = f – Au

Restrict r to a coarser grid

Determine the error e by solving Ae = r on the coarser grid

Recursive loop

Interpolate e to the original grid

Correct v (v + e)

Post-smoothing
```

\[ \Omega^0 \rightarrow \Omega^1 \rightarrow \Omega^2 \rightarrow \cdots \rightarrow \Omega^{[\text{maxlevel}]} \]

Restriction

Prolongation
Prolongator requirements

Few desired properties

- **preservation of null space**: the span of basis functions on each coarse level should contain zero energy modes

- **minimization of energy**: basis functions on the coarse levels should have as small energy as possible

- **bounded intersection**: the supports of the basis functions on the coarse levels should overlap as little as possible.
Smoothed Aggregation

SA prolongator is constructed in a few steps

• Construct aggregates
  – Select a set of root nodes
  – Group unknowns into aggregates

• Construct tentative prolongator and coarse nullspace
  – Restrict fine nullspace onto aggregates
  – Do QR decomposition
    We satisfy $P_{tent}B_c = B$

• Decrease energy of $P_{tent}$ by smoothing
  $P = (I - \omega D^{-1}A)P_{tent}$
  May not satisfy $P_{SA}B_c = B$
Energy minimization
Energy minimization is a general framework.

**Idea:** construct the prolongator $P$ by minimizing the energy of each column $P_k$ while enforcing constraints.

**Find $P$:**

$$P = \arg\min \sum \|P_k\|_\chi$$

**subject to**

- specified sparsity pattern;
- nullspace preservation.

**Advantages:**

- Flexibility (input):
  - accept any sparsity pattern (arbitrary basis function support)
  - enforce constraints: important modes requiring accurate interpolation
  - choice of norm for minimization and search space
- Robustness
Constraint matrix

- Sparsity pattern
- $B, B_c$ fine and coarse mode(s) requiring accurate interpolation

Preservation of the nullspace: for instance $P_1 = I$

$$N = \begin{bmatrix} * & * & * \\ * & 0 & * \\ 0 & * & * \end{bmatrix} \quad PB_c = B \Leftrightarrow \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \\ p_{31} & p_{32} \\ p_{41} & p_{42} \end{bmatrix} \begin{bmatrix} b_{11}^c \\ b_{21}^c \end{bmatrix} = \begin{bmatrix} b_{11} \\ b_{21} \\ b_{31} \\ b_{41} \end{bmatrix}$$

- Representation of the constraints in the algorithm:
Constraint matrix

Two nullspace vectors:

\[ P \begin{bmatrix} b_{11}^c & b_{12}^c \\ b_{21}^c & b_{22}^c \end{bmatrix} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \\ b_{41} & b_{42} \end{bmatrix} \]
Energy-minimization algorithm

Find $P$:

$$P = \arg\min \sum \|P_k\|_X$$

subject to

- specified sparsity pattern;
- nullspace preservation.

Solve $AP = 0$ in a constrained Krylov space

- Definition of energy $\|\cdot\|_X$ depends on Krylov method
  - $A$ for CG
  - $A^TA$ for GMRES
Energy minimization algorithm

Construct aggregates
\[ \mathcal{N} = |A||P^{(0)}| \]

\[ D = \text{diag}(A) \]
\[ R = -AP^{(0)} \]
\[ R = \text{enforce}(R, \mathcal{N}) \]
\[ R = \text{project}(R, X) \]

\[ \text{for } i \text{ to iter do} \]
\[ Z = D^{-1}R \]
\[ \gamma = \langle R, Z \rangle_F \]
\[ \text{if } i \text{ is 1 then} \]
\[ Y = Z \]
\[ \text{else} \]
\[ \beta = \gamma / \gamma_{old}; \]
\[ Y = Z + \beta Y \]
\[ \text{end if} \]
\[ \gamma_{old} = \gamma \]
\[ Y_A = AY \]
\[ Y_A = \text{enforce}(Y_A, \mathcal{N}) \]
\[ Y_A = \text{project}(Y_A, B_c) \]
\[ \alpha = \gamma / \langle Y, Y_A \rangle_F \]
\[ P^{(i)} = P^{(i-1)} + \alpha Y \]
\[ R = R - \alpha Y_A \]

▷ Select sparsity pattern
▷ Diagonal preconditioner
  ▷ Initial residual
  ▷ Enforce sparsity on \( R \)
  ▷ Enforce \( RB_c = 0 \)
▷ New search direction
▷ Enforce sparsity on \( Y_A \)
  ▷ Enforce \( Y_A B_c = 0 \)
▷ Update prolongator
▷ Update residual
A Special Case of Energy Minimization: SA

• Assume an initial guess $P_0$ satisfying $B = P_0 B_c$, i.e., it satisfies constraints of interpolating nullspace.
• Improve $P_0$ with one step of damped Jacobi:
  \[ P = (I - \omega D^{-1} A) P_0 \]
  
• $P$ still interpolates the nullspace. $P$ can be rewritten as
  \[ P = P_0 - \omega D^{-1} A P_0 = P_0 + \Delta P \]

  Note that $\Delta P B_c = 0$.

• SA can be viewed as one step of energy minimization with constraints specifying nullspace interpolation but not sparsity pattern enforcement.
Energy-minimization – Elasticity 3D

Lots of choices. We focus on 3 DOFs/nodes on the coarse grid

• 6 rigid body modes (3 translations & 3 rotations)
• CG to solve $A P = 0$ (effectively defines energy)
• $P_0$ & sparsity pattern are smoothed aggregation inspired
  – Initial Guess: tentative prolongator
  – Sparsity Pattern: $|S||P_{tent}|$, where S is either A, or filtered A
• Filtered matrix is defined using distance Laplacian + dropping for sparsity pattern
• A is still used to define energy (as opposed to filtered A)
Comparison with Smoothed Aggregation

- SA: 6 DOFs/node
- Energy Minimization: 3 DOFs/node, 6 nullspace vectors

Tab. : Iteration count and complexity (lower complexity = faster run time) for increasing mesh sizes and stretch factors.

<table>
<thead>
<tr>
<th>Mesh</th>
<th>$\epsilon = 1$</th>
<th></th>
<th>$\epsilon = 10$</th>
<th></th>
<th>$\epsilon = 100$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SA</td>
<td>Emin</td>
<td>SA</td>
<td>Emin</td>
<td>SA</td>
<td>Emin</td>
</tr>
<tr>
<td>$10^3$</td>
<td>6</td>
<td>1.30</td>
<td>8</td>
<td>2.81</td>
<td>9</td>
<td>3.21</td>
</tr>
<tr>
<td>$15^3$</td>
<td>8</td>
<td>1.19</td>
<td>10</td>
<td>2.32</td>
<td>12</td>
<td>2.54</td>
</tr>
<tr>
<td>$20^3$</td>
<td>8</td>
<td>1.24</td>
<td>10</td>
<td>2.59</td>
<td>13</td>
<td>3.05</td>
</tr>
<tr>
<td>$25^3$</td>
<td>9</td>
<td>1.26</td>
<td>11</td>
<td>2.76</td>
<td>14</td>
<td>3.04</td>
</tr>
<tr>
<td>$30^3$</td>
<td>10</td>
<td>1.22</td>
<td>12</td>
<td>2.52</td>
<td>15</td>
<td>3.06</td>
</tr>
<tr>
<td>$35^3$</td>
<td>10</td>
<td>1.24</td>
<td>12</td>
<td>2.66</td>
<td>16</td>
<td>3.03</td>
</tr>
<tr>
<td>$40^3$</td>
<td>10</td>
<td>1.26</td>
<td>12</td>
<td>2.77</td>
<td>16</td>
<td>3.21</td>
</tr>
</tbody>
</table>

complexity: $\sum_i \frac{nnz(A_i)}{nnz(A)}$

3.85x
Parallel implementation
Energy minimization algorithm

Construct aggregates
\[ \mathcal{N} = |A||P^{(0)}| \]

\[ D = \text{diag}(A) \]
\[ R = -AP^{(0)} \]
\[ R = \text{enforce}(R, \mathcal{N}) \]
\[ R = \text{project}(R, X) \]

\[ \textbf{for } i \to \text{iter} \textbf{ do} \]
\[ Z = D^{-1}R \]
\[ \gamma = \langle R, Z \rangle_F \]
\[ \textbf{if } i \text{ is 1 then} \]
\[ Y = Z \]
\[ \textbf{else} \]
\[ \beta = \gamma / \gamma_{old}; \]
\[ Y = Z + \beta Y \]
\[ \textbf{end if} \]
\[ \gamma_{old} = \gamma \]
\[ Y_A = AY \]
\[ Y_A = \text{enforce}(Y_A, \mathcal{N}) \]
\[ Y_A = \text{project}(Y_A, B_c) \]
\[ \alpha = \gamma / \langle Y, Y_A \rangle_F \]
\[ P^{(i)} = P^{(i-1)} + \alpha Y \]
\[ R = R - \alpha Y_A \]

\[ \triangleright \text{Select sparsity pattern} \]
\[ \triangleright \text{Diagonal preconditioner} \]
\[ \triangleright \text{Initial residual} \]
\[ \triangleright \text{Enforce sparsity on } R \]
\[ \triangleright \text{Enforce } RB_c = 0 \]

\[ \triangleright \text{New search direction} \]

\[ \triangleright \text{Enforce sparsity on } Y_A \]
\[ \triangleright \text{Enforce } Y_AB_c = 0 \]

\[ \triangleright \text{Update prolongator} \]
\[ \triangleright \text{Update residual} \]
Energy minimization algorithm

Construct aggregates
\[ \mathcal{N} = |A| |P^{(0)}| \]

\[ D = \text{diag}(A) \]
\[ R = -AP^{(0)} \]
\[ R = \text{enforce}(R, \mathcal{N}) \]
\[ R = \text{project}(R, X) \]

▷ Select sparsity pattern
▷ Diagonal preconditioner
  ▷ Initial residual
  ▷ Enforce sparsity on \( R \)
  ▷ Enforce \( RB_c = 0 \)

\begin{verbatim}
for \( i \) to \( \text{iter} \) do
  \[ Z = D^{-1}R \]
  \[ \gamma = \langle R, Z \rangle_F \]
  if \( i \) is 1 then
    \[ Y = Z \]
  else
    \[ \beta = \gamma / \gamma_{\text{old}}; \]
    \[ Y = Z + \beta Y \]
  end if
  \[ \gamma_{\text{old}} = \gamma \]
  \[ Y_A = AY \]
  \[ Y_A = \text{enforce}(Y_A, \mathcal{N}) \]
  \[ Y_A = \text{project}(Y_A, B_c) \]
  \[ \alpha = \gamma / \langle Y, Y_A \rangle_F \]
  \[ P^{(i)} = P^{(i-1)} + \alpha Y \]
  \[ R = R - \alpha Y_A \]
end if
\end{verbatim}

▷ New search direction
▷ Enforce sparsity on \( Y_A \)
  ▷ Enforce \( Y_AB_c = 0 \)
▷ Update prolongator
▷ Update residual
Parallel aggregation

Two choices: coupled and uncoupled aggregation
• Uncoupled aggregation aggregates only inside a subdomain
• Coupled aggregation allows aggregates to cross subdomain boundary
• Coupled aggregation is more expensive, but has convergence similar to the serial case
Coupled aggregation

Couple aggregation algorithm:

1. Construct uncoupled aggregation in each subdomain (local procedure)
   - Some nodes are left unaggregated

2. Assign unaggregated vertices to adjacent root nodes from neighbor subdomains
   - Might require some arbitration

3. Create new root nodes and aggregates if we have multiple adjacent unaggregated nodes

4. Sweep remaining nodes into existing aggregates
Constraints in parallel

Let $P$ have the following pattern and nullspace consist of two vectors

$$
P \begin{bmatrix}
    b_{11}^c & b_{12}^c \\
    b_{21}^c & b_{22}^c
\end{bmatrix} = \begin{bmatrix}
    b_{11} & b_{12} \\
    b_{21} & b_{22} \\
    b_{31} & b_{32} \\
    b_{41} & b_{42}
\end{bmatrix}
$$

$$
P = \begin{bmatrix}
    p_{11} & p_{12} \\
    p_{21} & 0 \\
    p_{31} & p_{32} \\
    0 & p_{41}
\end{bmatrix}
$$
What does each block correspond to?

Consider a row of $P$ with three nonzeros

Block of the constraint corresponding to the row
Energy minimization algorithm (updated)

Construct aggregates
\[ \mathcal{N} = |A||P^{(0)}| \]
Import ghost components of nullspace vectors

\[
D = \text{diag}(A) \\
R = -AP^{(0)} \\
R = \text{enforce}(R, \mathcal{N}) \\
R = \text{project}(R, X)
\]

\[
\text{for } i \text{ to iter do} \\
Z = D^{-1}R \\
\gamma = \langle R, Z \rangle_F \\
\text{if } i \text{ is 1 then} \\
Y = Z \\
\text{else} \\
\beta = \gamma / \gamma_{\text{old}}; \\
Y = Z + \beta Y \\
\text{end if} \\
\gamma_{\text{old}} = \gamma \\
Y_A = AY \\
Y_A = \text{enforce}(Y_A, \mathcal{N}) \\
Y_A = \text{project}(Y_A, B_c) \\
\alpha = \gamma / \langle Y, Y_A \rangle_F \\
P^{(i)} = P^{(i-1)} + \alpha Y \\
R = R - \alpha Y_A
\]

\[ \text{Select sparsity pattern} \]
\[ \text{Diagonal preconditioner} \]
\[ \text{Initial residual} \]
\[ \text{Enforce sparsity on } R \]
\[ \text{Enforce } R B_c = 0 \]
\[ \text{New search direction} \]
\[ \text{Enforce sparsity on } Y_A \]
\[ \text{Enforce } Y_A B_c = 0 \]
\[ \text{Update prolongator} \]
\[ \text{Update residual} \]
MueLu

• Future package of the Trilinos project (to replace ML)
  – Massively parallel
  – Multicore and GPU aware
  – Templated types for mixed precision calculation (32-bit – 64-bit) and type complex

• Objective is to solve problem with billions of DOF on 100Ks of cores...

• Leverage the Trilinos software stack:

  - Teuchos Utility package
  - Belos Krylov methods
  - Anasazi Eigen-solvers
  - Tifpack Algebraic precond.
  - MueLu MG solver
  - Tpetra – distributed linear algebra
  - Kokkos – single node kernels

• Currently in development...
Numerical results - Laplace 3D

- Laplace 3D, 7 point stencil
- Energy minimization
  - 2 CG iterations
  - Initial guess: tentative prolongator
  - Sparsity pattern: same as SA
Numerical results - Elasticity 3D

- Elasticity 3D, Poisson ratio 0.25
- Energy minimization
  - 2 CG iterations
  - Initial guess: tentative prolongator
  - Sparsity pattern: same as SA
Summary

• Energy minimization AMG is flexible

• Energy minimization AMG is suitable for parallelization
  – Standard parallel operations (MxM, BLAS1) are well known
  – Constraint application could be done locally storing ghost info

• Preliminary results show promise

European Trilinos User Group Meeting 2013
June 3rd - June 5th
Technical University of Munich, Munich, Germany