Adaptive Mesh Refinement for Multigrid on Semi-Structured Grids

Tobias Gradl, Ulrich Rüde

Dept. of Computer Science, System Simulation,
University of Erlangen-Nuremberg

Copper Mountain Conference on Multigrid Methods
March 23, 2009
The HHG Framework

Adaptive Mesh Refinement
Combining finite element and multigrid methods

FE mesh may be unstructured.
What nodes to remove for coarsening? Not straightforward!
Why not start from the coarse grid?
Combining finite element and multigrid methods

FE mesh may be unstructured.
What nodes to remove for coarsening? Not straightforward!
Why not start from the coarse grid?

The Hierarchical Hybrid Grids (HHG) concept

- Benjamin Bergen*: prototype
- Tobias Gradl: tuning, extensions and applications

Combining finite element and multigrid methods

FE mesh may be unstructured.
What nodes to remove for coarsening? Not straightforward!
Why not start from the coarse grid?

The Hierarchical Hybrid Grids (HHG) concept

- Benjamin Bergen*: prototype
- Tobias Gradl: tuning, extensions and applications

Combining finite element and multigrid methods

FE mesh may be unstructured.
What nodes to remove for coarsening? Not straightforward!
Why not start from the coarse grid?

The Hierarchical Hybrid Grids (HHG) concept

- Benjamin Bergen*:
  prototype
- Tobias Gradl: tuning,
  extensions and applications

Combining finite element and multigrid methods

FE mesh may be unstructured.
What nodes to remove for coarsening? Not straightforward!
Why not start from the coarse grid?

The Hierarchical Hybrid Grids (HHG) concept

- Benjamin Bergen*: prototype
- Tobias Gradl: tuning, extensions and applications

Combining finite element and multigrid methods

FE mesh may be unstructured. What nodes to remove for coarsening? Not straightforward! Why not start from the coarse grid?

The Hierarchical Hybrid Grids (HHG) concept

- Benjamin Bergen*: prototype
- Tobias Gradl: tuning, extensions and applications

Combining finite element and multigrid methods

FE mesh may be unstructured.
What nodes to remove for coarsening? Not straightforward!
Why not start from the coarse grid?

The Hierarchical Hybrid Grids (HHG) concept

- Benjamin Bergen*: prototype
- Tobias Gradl: tuning, extensions and applications

Properties of the HHG approach

**Advantages**

- Multigrid is straightforward
- Very memory efficient
- Massive performance benefits on current computer architectures
- Subserves parallelization

⇒ $10^{11}$ unknowns are possible
Properties of the HHG approach

Advantages

- Multigrid is straightforward
- Very memory efficient
- Massive performance benefits on current computer architectures
- Subserves parallelization

⇒ $10^{11}$ unknowns are possible

Limitation

- Coarse input grid needed
- Adaptivity?
HHG on parallel computers

Mesh is split up at **coarsest level**
→ Vertices, Edges, Faces, Volumes

Facilitates parallelization for **message passing infrastructures** (distributed memory parallel computers)

Largest run: $3 \times 10^{11}$ unknowns solved in 1.5 minutes on 9170 cores on *SGI Altix 4700*. 
Testing on HLRB II

HLRB II at
Leibniz-Rechenzentrum
Garching

- 9728 processor cores (1.6 GHz Intel Itanium2)
- 62 Tflop/s peak performance
- 39 Tbytes of main memory

### Performance on HLRB II

<table>
<thead>
<tr>
<th>Cores</th>
<th>Unknowns</th>
<th>Avg. time per V-cycle (sec)</th>
<th>Time to solution ($|r| &lt; 10^{-6} \cdot |r_0|)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>$2.1 \times 10^9$</td>
<td>4.93</td>
<td>59.2</td>
</tr>
<tr>
<td>504</td>
<td>$1.7 \times 10^{10}$</td>
<td>5.44</td>
<td>65.3</td>
</tr>
<tr>
<td>2040</td>
<td>$6.8 \times 10^{10}$</td>
<td>5.60</td>
<td>67.2</td>
</tr>
<tr>
<td>4080</td>
<td>$1.7 \times 10^{11}$</td>
<td>5.68</td>
<td>68.2</td>
</tr>
<tr>
<td>6120</td>
<td>$2.1 \times 10^{11}$</td>
<td>6.33</td>
<td>76.0</td>
</tr>
<tr>
<td>8152</td>
<td>$2.7 \times 10^{11}$</td>
<td>7.43 *</td>
<td>89.2</td>
</tr>
<tr>
<td>9170</td>
<td>$3.1 \times 10^{11}$</td>
<td>7.75 *</td>
<td>93.0</td>
</tr>
</tbody>
</table>

*: including high-density partitions

$3.1 \times 10^{11}$ is 1000 m$^3$ resolved at 1.5 mm, enough for 20 kHz sound. Current finite element acoustic simulations (G. Müller, TUM): 4 m$^3$ at 400 Hz (i.e. 7.5 cm resolution and approx. $10^5$ unknowns).
Adaptive refinement: motivation

Varying mesh density required by

- geometry (e.g. walls of a room)
Adaptive refinement: motivation

Varying mesh density required by

- geometry (e.g. walls of a room)
- physics (e.g. singularities)
Two approaches

- Hanging nodes → non-conforming grids
Two approaches

- Hanging nodes $\rightarrow$ non-conforming grids
- Red-green $\rightarrow$ conforming grids
Two approaches

- Hanging nodes → non-conforming grids

- Red-green → conforming grids
Two approaches

- Hanging nodes $\rightarrow$ non-conforming grids
  
- Red-green $\rightarrow$ conforming grids
Two approaches

- Hanging nodes $\rightarrow$ non-conforming grids

- Red-green $\rightarrow$ conforming grids
Comparison: red-green vs. hanging nodes

Both have (dis-)advantages (some of them HHG specific):

- Red-green: impossible for purely quadrilateral or hexahedral grids
- Hanging nodes: too large refined areas (in HHG)
- Red-green: more elements on coarsest level (in HHG)
Comparison: red-green vs. hanging nodes

Both have (dis-)advantages (some of them HHG specific):

- Red-green: impossible for purely quadrilateral or hexahedral grids
- Hanging nodes: too large refined areas (in HHG)
- Red-green: more elements on coarsest level (in HHG)

The ideal HHG grid has *few coarse level elements.*
⇒ Use refinement with hanging nodes whenever possible.
Refinement with hanging nodes in HHG
Refinement with hanging nodes in HHG
Refinement with hanging nodes in HHG
Refinement with hanging nodes in HHG
Refinement with hanging nodes in HHG
Refinement with hanging nodes in HHG
Residual computation at refinement boundaries

\[ r_i^c = \sum_{j \in \mathcal{N}^F(i)} \left( w_j r_j^F(u) \right) \]
Residual computation at refinement boundaries

\[ r_i^C = \sum_{j \in N^F(i)} \left( w_j r_j^F(u) \right) \]
\[ = \sum_{j \in N^F_S(i)} \left( w_j r_j^F(u) \right) + \sum_{j \in N^F_I(i)} \left( w_j r_j^F(u) \right) \]
Residual computation at refinement boundaries

\[ r_i^c = \sum_{j \in N^F(i)} (w_j r_j^F(u)) \]

\[ = \sum_{j \in N_S^F(i)} (w_j r_j^F(u)) + \sum_{j \in N_I^F(i)} (w_j r_j^F(\tilde{u})) \]
Residual computation at refinement boundaries

\[
\begin{align*}
    r_i^c &= \sum_{j \in N^F(i)} \left( w_j r_j^F(u) \right) \\
    &= \sum_{j \in N_S^F(i)} \left( w_j r_j^F(u) \right) + \sum_{j \in N_I^F(i)} \left( w_j r_j^F(\tilde{u}) \right)
\end{align*}
\]

Using \( \tilde{r}_i^c = \sum_{j \in N^F(i)} \left( w_j r_j^F(\tilde{u}) \right) = 0 \).
Residual computation at refinement boundaries

\[ r_i^c = \sum_{j \in N^F(i)} \left( w_j r_j^F(u) \right) \]

\[ = \sum_{j \in N^F_S(i)} \left( w_j r_j^F(u) \right) + \sum_{j \in N^F_I(i)} \left( w_j r_j^F(\tilde{u}) \right) \]

Using \( \tilde{r}_i^c = \sum_{j \in N^F(i)} \left( w_j r_j^F(\tilde{u}) \right) = 0 \), i.e.

\[- \sum_{j \in N^F_S(i)} \left( w_j r_j^F(\tilde{u}) \right) = \sum_{j \in N^F_I(i)} \left( w_j r_j^F(\tilde{u}) \right).\]
Residual computation at refinement boundaries

\[ r_i^c = \sum_{j \in N^F(i)} \left( w_j r_j^F(u) \right) \]
\[ = \sum_{j \in N^F_S(i)} \left( w_j r_j^F(u) \right) + \sum_{j \in N^F_I(i)} \left( w_j r_j^F(\tilde{u}) \right) \]
\[ = \sum_{j \in N^F_S(i)} \left( w_j \left( r_j^F(u) - r_j^F(\tilde{u}) \right) \right) \]

Using \( \tilde{r}_i^c = \sum_{j \in N^F(i)} \left( w_j r_j^F(\tilde{u}) \right) = 0 \), i.e.

\[ - \sum_{j \in N^F_S(i)} \left( w_j r_j^F(\tilde{u}) \right) = \sum_{j \in N^F_I(i)} \left( w_j r_j^F(\tilde{u}) \right). \]
Thank you for your attention!

Any questions?

The development of HHG was funded by

- the Elite Network of Bavaria within the International Doctorate Program “Identification, Optimization and Control with Applications in Modern Technologies”
- KONWIHR