$h$-Adaptive FEM for Transport Problems

Dmitri Kuzmin, Matthias Möller, Stefan Turek

Institute of Applied Mathematics, LS III
Dortmund University of Technology, Germany
matthias.moeller@math.tu-dortmund.de

Lake Tahoe, January 5, 2009
Overview

1 Motivation

2 Dynamic mesh adaptation
   ■ Red-green refinement
   ■ Mesh re-coarsening
   ■ Numerical examples

3 Goal-oriented error estimation
   ■ Error splitting
   ■ Error localization
   ■ Numerical examples

4 Conclusions and outlook
Transport problems

Scalar conservation law

\[
\frac{\partial u}{\partial t} + \nabla \cdot f(u) = 0
\]
Transport problems

Scalar conservation law

\[ \frac{\partial u}{\partial t} + \nabla \cdot f(u) = 0 \]

Convection-diffusion equation

\[ \frac{\partial u}{\partial t} + \nabla \cdot (\mathbf{v} u - d \nabla u) = 0 \]
Transport problems

Scalar conservation law
\[ \frac{\partial u}{\partial t} + \nabla \cdot f(u) = 0 \]

Convection-diffusion equation
\[ \frac{\partial u}{\partial t} + \nabla \cdot (\mathbf{v} u - d \nabla u) = 0 \]

Galerkin FEM
\[ M_C \frac{du}{dt} = Ku \]

Algebraic flux correction, talks by D. Kuzmin, M. Gurris

p-adaptation between first- and second-order approximations
h-adaptation improves resolution of flow features (e.g., shocks)
Transport problems

Scalar conservation law
\[ \frac{\partial u}{\partial t} + \nabla \cdot f(u) = 0 \]

Convection-diffusion equation
\[ \frac{\partial u}{\partial t} + \nabla \cdot (vu - d\nabla u) = 0 \]

Low-order scheme
\[ M_L \frac{du}{dt} = Ku + Du = Lu \]

Algebraic flux correction, talks by D. Kuzmin, M. Gurris

p-adaptation between first- and second-order approximations

h-adaptation improves resolution of flow features (e.g., shocks)
Transport problems

Scalar conservation law
\[ \frac{\partial u}{\partial t} + \nabla \cdot f(u) = 0 \]

High-resolution scheme
\[ M_L \frac{du}{dt} = Lu + \bar{f}(u) \]

Convection-diffusion equation
\[ \frac{\partial u}{\partial t} + \nabla \cdot (v u - d \nabla u) = 0 \]
Transport problems

Scalar conservation law

\[
\frac{\partial u}{\partial t} + \nabla \cdot f(u) = 0
\]

High-resolution scheme

\[
M_L \frac{du}{dt} = Lu + \bar{f}(u)
\]

Convection-diffusion equation

\[
\frac{\partial u}{\partial t} + \nabla \cdot (\mathbf{v} u - d\nabla u) = 0
\]

Compressible Euler equations

\[
\frac{\partial}{\partial t} \begin{pmatrix} \rho \\ \rho \mathbf{v} \\ \rho E \end{pmatrix} + \nabla \cdot \begin{pmatrix} \rho \mathbf{v} \\ \rho \mathbf{v} \otimes \mathbf{v} + pI \\ (\rho E + p)\mathbf{v} \end{pmatrix} = 0
\]

Algebraic flux correction

\[ \leftrightarrow \text{talks by D. Kuzmin, M. Gurris} \]

- \( p \)-adaptation between first- and second-order approximations
- \( h \)-adaptation improves resolution of flow features (e.g., shocks)
(Un)structured meshes?

**Unstructured meshes**
- mesh generation for complex domains
- prevent distorted cells near singular points
- overhead costs due to indirect addressing

**Structured grids**
- efficient hardware oriented numerics
- orthogonal grids to resolve boundary layers
- unflexible/impractical for complex domains
(Un)structured meshes?

**Unstructured meshes**
- mesh generation for complex domains
- prevent distorted cells near singular points
- overhead costs due to indirect addressing

**Structured grids**
- efficient hardware oriented numerics
- orthogonal grids to resolve boundary layers
- unflexible/impractical for complex domains

AFC schemes can handle hybrid meshes
Design goals for $h$-adaptation

- conforming triangulations based on hybrid initial mesh
- no deterioration of grid quality due to mesh refinement
- mesh re-coarsening ‘undoes’ subdivision of elements
- adaptive hierarchy of locally nested meshes is generated
- vertices/structure of initial triangulation is preserved
- efficient data structures for dynamic mesh adaptation
The **red-green** strategy revisited

### Refinement algorithm in 2D

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>subdivide marked elements regularly</td>
<td>(red refinement)</td>
</tr>
<tr>
<td>2</td>
<td>eliminate ‘hanging nodes’ by transition cells</td>
<td>(green refinement)</td>
</tr>
</tbody>
</table>

#### Diagrams

- **Green refinement**
- **Blue refinement**
- **Red refinement**
The red-green strategy revisited

Refinement algorithm in 2D  

1. subdivide marked elements regularly (red refinement)
2. eliminate ‘hanging nodes’ by transition cells (green refinement)

admissible types of green refinement

red refinement
Mesh genealogy

Triangulation $\mathcal{T}_m(\mathcal{E}_m, \mathcal{V}_m), m = 0, 1, 2, \ldots$ consists of

$\mathcal{E}_m = \{\Omega_k : k = 1, \ldots, N_E\}$ and $\mathcal{V}_m = \{v_i : i = 1, \ldots, N_V\}$
Mesh genealogy

Triangulation $T_m(\mathcal{E}_m, \mathcal{V}_m)$, $m = 0, 1, 2, \ldots$ consists of

$\mathcal{E}_m = \{\Omega_k : k = 1, \ldots, N_E\}$ and $\mathcal{V}_m = \{v_i : i = 1, \ldots, N_V\}$

- nodal **generation function** $g : \mathcal{V}_m \rightarrow \mathbb{N}_0$ is defined recursively

\[
g(v_i) := \begin{cases} 
0 & \text{if } v_i \in \mathcal{V}_0 \\
\max_{v_j \in \Gamma_{kl}} g(v_j) + 1 & \text{if } v_i \in \Gamma_{kl} := \bar{\Omega}_k \cap \bar{\Omega}_l \\
\max_{v_j \in \partial \Omega_k} g(v_j) + 1 & \text{if } v_i \in \Omega_k \setminus \partial \Omega_k
\end{cases}
\]
Mesh genealogy

Triangulation $T_m(\mathcal{E}_m, \mathcal{V}_m)$, $m = 0, 1, 2, \ldots$ consists of

$\mathcal{E}_m = \{\Omega_k : k = 1, \ldots, N_E\}$ and $\mathcal{V}_m = \{v_i : i = 1, \ldots, N_V\}$

- **Nodal generation function** $g : \mathcal{V}_m \to \mathbb{N}_0$ is defined recursively

$$g(v_i) := \begin{cases} 
0 & \text{if } v_i \in \mathcal{V}_0 \\
\max_{v_j \in \Gamma_{kl}} g(v_j) + 1 & \text{if } v_i \in \Gamma_{kl} := \bar{\Omega}_k \cap \bar{\Omega}_l \\
\max_{v_j \in \partial \Omega_k} g(v_j) + 1 & \text{if } v_i \in \Omega_k \setminus \partial \Omega_k 
\end{cases}$$
Mesh genealogy

Triangulation $\mathcal{T}_m(\mathcal{E}_m, \mathcal{V}_m)$, $m = 0, 1, 2, \ldots$ consists of
\[ \mathcal{E}_m = \{ \Omega_k : k = 1, \ldots, N_E \} \quad \text{and} \quad \mathcal{V}_m = \{ v_i : i = 1, \ldots, N_V \} \]

- nodal **generation function** $g : \mathcal{V}_m \rightarrow \mathbb{N}_0$ is defined recursively

\[
g(v_i) := \begin{cases} 
0 & \text{if } v_i \in \mathcal{V}_0 \\
\max_{v_j \in \Gamma_{kl}} g(v_j) + 1 & \text{if } v_i \in \Gamma_{kl} := \bar{\Omega}_k \cap \bar{\Omega}_l \\
\max_{v_j \in \partial \Omega_k} g(v_j) + 1 & \text{if } v_i \in \Omega_k \setminus \partial \Omega_k
\end{cases}
\]
Mesh genealogy

Triangulation $\mathcal{T}_m(\mathcal{E}_m, \mathcal{V}_m), m = 0, 1, 2, \ldots$ consists of

$\mathcal{E}_m = \{\Omega_k : k = 1, \ldots, N_E\}$ and $\mathcal{V}_m = \{v_i : i = 1, \ldots, N_V\}$

- **nodal generation function** $g : \mathcal{V}_m \rightarrow \mathbb{N}_0$ is defined recursively

$$g(v_i) := \begin{cases} 
0 & \text{if } v_i \in \mathcal{V}_0 \\
\max_{v_j \in \Gamma_{kl}} g(v_j) + 1 & \text{if } v_i \in \Gamma_{kl} := \tilde{\Omega}_k \cap \tilde{\Omega}_l \\
\max_{v_j \in \partial \Omega_k} g(v_j) + 1 & \text{if } v_i \in \Omega_k \setminus \partial \Omega_k
\end{cases}$$
Mesh genealogy

Triangulation $T_m(\mathcal{E}_m, \mathcal{V}_m)$, $m = 0, 1, 2, \ldots$ consists of

$\mathcal{E}_m = \{\Omega_k : k = 1, \ldots, N_E\}$ and $\mathcal{V}_m = \{v_i : i = 1, \ldots, N_V\}$

- Nodal generation function $g : \mathcal{V}_m \rightarrow \mathbb{N}_0$ is defined recursively

$$
g(v_i) := \begin{cases} 
0 & \text{if } v_i \in \mathcal{V}_0 \\
\max_{v_j \in \Gamma_{kl}} g(v_j) + 1 & \text{if } v_i \in \Gamma_{kl} := \tilde{\Omega}_k \cap \tilde{\Omega}_l \\
\max_{v_j \in \partial \Omega_k} g(v_j) + 1 & \text{if } v_i \in \Omega_k \setminus \partial \Omega_k
\end{cases}
$$

- Represents number of subdivisions $\Rightarrow$ prescribe maximum depth
- Characterizes elements and their relation to neighboring cells
Mesh re-coarsening

Coarsening algorithms (classical approach)

1. identify (patches of) elements which can be coarsened
2. delete elements/vertices and re-triangulate subdomain
Mesh re-coarsening

Coarsening algorithms (classical approach)

1. identify (patches of) elements which can be coarsened
2. delete elements/vertices and re-triangulate subdomain
Mesh re-coarsening

Re-coarsening algorithms (vertex-based approach)

1. ‘lock’ vertices step-by-step which must not be removed
2. delete ‘free’ vertices/elements and restore macro cells

Result:
Vertex \( v_i \) is locked if \( d(v_i) \leq 0 \); otherwise it can be deleted.
All vertices of the initial mesh are locked by construction!
Mesh re-coarsening

Re-coarsening algorithms (vertex-based approach)

1. ‘lock’ vertices step-by-step which must not be removed
2. delete ‘free’ vertices/elements and restore macro cells

1. initialize $d(v_i) := g(v_i), \forall v_i \in V_m \Rightarrow d(v_i) = 0, \forall v_i \in V_0$
Mesh re-coarsening

Re-coarsening algorithms (vertex-based approach)

1. ‘lock’ vertices step-by-step which must not be removed
2. delete ‘free’ vertices/elements and restore macro cells

1. initialize $d(v_i) := g(v_i)$, $\forall v_i \in V_m \Rightarrow d(v_i) = 0$, $\forall v_i \in V_0$

2. vertex $v_i \in V_m$ is locked, i.e. $d(v_i) := -|d(v_i)|$ if
   - $v_i$ belongs to an element which is marked for refinement
   - $v_i$ belongs to a red element which should not be coarsened
### Mesh re-coarsening

#### Re-coarsening algorithms (vertex-based approach)

1. ‘lock’ vertices step-by-step which must not be removed
2. delete ‘free’ vertices/elements and restore macro cells

<table>
<thead>
<tr>
<th>1</th>
<th>Initialize $d(v_i) := g(v_i), \forall v_i \in \mathcal{V}_m \Rightarrow d(v_i) = 0, \forall v_i \in \mathcal{V}_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Vertex $v_i \in \mathcal{V}_m$ is locked, i.e. $d(v_i) := -</td>
</tr>
<tr>
<td></td>
<td>$v_i$ belongs to an element which is marked for refinement</td>
</tr>
<tr>
<td></td>
<td>$v_i$ belongs to a red element which should not be coarsened</td>
</tr>
<tr>
<td></td>
<td>there is an edge $ij$ such that $g(v_i) &lt; g(v_j)$ for some $v_j \in \mathcal{V}_m$</td>
</tr>
</tbody>
</table>

Result: Vertex $v_i$ is locked if $d(v_i) \leq 0$; otherwise it can be deleted. All vertices of the initial mesh are locked by construction!
Mesh re-coarsening

Re-coarsening algorithms (vertex-based approach)

1. ‘lock’ vertices step-by-step which must not be removed
2. delete ‘free’ vertices/elements and restore macro cells

1. initialize $d(v_i) := g(v_i), \forall v_i \in V_m \Rightarrow d(v_i) = 0, \forall v_i \in V_0$
2. vertex $v_i \in V_m$ is locked, i.e. $d(v_i) := -|d(v_i)|$ if
   - $v_i$ belongs to an element which is marked for refinement
   - $v_i$ belongs to a red element which should not be coarsened
   - there is an edge $ij$ such that $g(v_i) < g(v_j)$ for some $v_j \in V_m$
3. vertices are locked to preclude the creation of blue elements
Mesh re-coarsening

Re-coarsening algorithms (vertex-based approach)

1. ‘lock’ vertices step-by-step which must not be removed
2. delete ‘free’ vertices/elements and restore macro cells

1. initialize \( d(v_i) := g(v_i), \forall v_i \in V_m \Rightarrow d(v_i) = 0, \forall v_i \in V_0 \)
2. vertex \( v_i \in V_m \) is locked, i.e. \( d(v_i) := -|d(v_i)| \) if
   - \( v_i \) belongs to an element which is marked for refinement
   - \( v_i \) belongs to a red element which should not be coarsened
   - there is an edge \( ij \) such that \( g(v_i) < g(v_j) \) for some \( v_j \in V_m \)
3. vertices are locked to preclude the creation of blue elements

Result: Vertex \( v_i \) is locked if \( d(v_i) \leq 0 \); otherwise it can be deleted.
Mesh re-coarsening

Re-coarsening algorithms (vertex-based approach)

1. ‘lock’ vertices step-by-step which must not be removed
2. delete ‘free’ vertices/elements and restore macro cells

1. initialize \( d(v_i) := g(v_i) \), \( \forall v_i \in V_m \) \( \Rightarrow \) \( d(v_i) = 0, \forall v_i \in V_0 \)
2. vertex \( v_i \in V_m \) is locked, i.e. \( d(v_i) := -|d(v_i)| \) if
   - \( v_i \) belongs to an element which is marked for refinement
   - \( v_i \) belongs to a red element which should not be coarsened
   - there is an edge \( ij \) such that \( g(v_i) < g(v_j) \) for some \( v_j \in V_m \)
3. vertices are locked to preclude the creation of blue elements

Result: Vertex \( v_i \) is locked if \( d(v_i) \leq 0 \); otherwise it can be deleted. All vertices of the initial mesh are locked by construction!
Step-by-step illustration

Refinement algorithm: initial mesh
Refinement algorithm: mark elements for regular refinement
Step-by-step illustration

Refinement algorithm: perform regular refinement
Refinement algorithm: mark elements for regular refinement
Step-by-step illustration

**Refinement algorithm:** perform regular refinement + transition cells
Step-by-step illustration

Re-coarsening algorithm: vertices from initial mesh are locked
Step-by-step illustration

**Re-coarsening algorithm:** keep cells and lock connected vertices
Step-by-step illustration

**Re-coarsening algorithm:** lock vertices if there are younger neighbors
Step-by-step illustration

Re-coarsening algorithm: lock vertices to preclude blue elements
Re-coarsening algorithm: remove vertices and update elements
Solid body rotation

FEM-FCT scheme, Crank-Nicolson time-stepping, $\Delta t = 10^{-3}$

$$\frac{\partial u}{\partial t} + \nabla \cdot (v u) = 0 \quad \text{in} \quad (0, 1)^2 \times (0, T) \quad u = 0 \quad \text{on} \quad \Gamma_D$$

- dynamic mesh adaptation
  - every 5 time steps
  - protective layers
- approximate $\nabla u \approx g(\nabla u_h)$
  $$\| \nabla u - \nabla u_h \|_{L^2(\Omega)}^2 \approx \sum_k \eta_k$$
  where $\eta_k = \| g - \nabla u_h \|_{L^2(\Omega_k)}^2$
Solid body rotation

FEM-FCT scheme, Crank-Nicolson time-stepping, $\Delta t = 10^{-3}$

$$\frac{\partial u}{\partial t} + \nabla \cdot (\mathbf{v} u) = 0 \quad \text{in} \quad (0, 1)^2 \times (0, T)$$

$$u = 0 \quad \text{on} \quad \Gamma_D$$

Initial/exact solution

$1/512 \leq h \leq 1/8$
Solid body rotation

FEM-FCT scheme, Crank-Nicolson time-stepping, \( \Delta t = 10^{-3} \)

\[
\frac{\partial u}{\partial t} + \nabla \cdot (\mathbf{v} u) = 0 \quad \text{in} \quad (0, 1)^2 \times (0, T) \\
u = 0 \quad \text{on} \quad \Gamma_D
\]
Double Mach reflection

- **Initial conditions:** left and right states for a Mach 10 shock

\[
\begin{bmatrix}
\rho_{\text{pre}} \\
\mathbf{u}_{\text{pre}} \\
\mathbf{v}_{\text{pre}} \\
\rho_{\text{pre}}
\end{bmatrix} =
\begin{bmatrix}
8.0 \\
8.25 \cos(30^\circ) \\
-8.25 \sin(30^\circ) \\
116.5
\end{bmatrix}
\quad \begin{bmatrix}
\rho_{\text{post}} \\
\mathbf{u}_{\text{post}} \\
\mathbf{v}_{\text{post}} \\
\rho_{\text{post}}
\end{bmatrix} =
\begin{bmatrix}
1.4 \\
0.0 \\
0.0 \\
1.0
\end{bmatrix}
\]

- **Boundary conditions:** separation point \( x_s(t) = \frac{1}{6} + \frac{1+20t}{\sqrt{3}} \)

\[\Gamma_{\text{pre}} = \{ x < x_s(t), y = 1 \}, \quad \Gamma_{\text{post}} = \{ x \geq x_s(t), y = 1 \}\]
Goal-oriented error estimation

\[
\begin{aligned}
\nabla \cdot (v u - d \nabla u) &= f \quad \text{in } \Omega \\
u &= b \quad \text{on } \Gamma
\end{aligned}
\]
Goal-oriented error estimation

\[
\begin{cases}
\nabla \cdot (\mathbf{v} u - d \nabla u) = f & \text{in } \Omega \\
u = b & \text{on } \Gamma
\end{cases}
\]

Primal problem: find \( u \in H^1_b(\Omega) \)
\[
a(w, u) = (w, f) \quad \forall w \in H^1_0(\Omega)
\]

Dual problem: find \( z \in H^1_0(\Omega) \)
\[
a(z, w) = j(w) \quad \forall w \in H^1_0(\Omega)
\]

\[
a(w, u) = \int_{\Omega} w \nabla \cdot (\mathbf{v} u) \, dx
\]
\[+ \int_{\Omega} \nabla w \cdot (d \nabla u) \, dx
\]

Dual weighted residual error

Galerkin orthogonality error needs to be estimated

can be computed
Goal-oriented error estimation

\[
\begin{aligned}
\nabla \cdot (\mathbf{v} u - d \nabla u) &= f \quad \text{in } \Omega \\
u &= b \quad \text{on } \Gamma
\end{aligned}
\]

Primal problem: find \( u \in H^1_b(\Omega) \)
\[
a(w, u) = (w, f) \quad \forall w \in H^1(\Omega)
\]

Dual problem: find \( z \in H^1_0(\Omega) \)
\[
a(z, w) = j(w) \quad \forall w \in H^1(\Omega)
\]

Error representation: \( u \approx \bar{u} = \sum_j \tilde{u}_j \varphi_j \)
\[
j(u - \bar{u}) = (z, f) - a(z, \bar{u}) = \rho(z, \bar{u})
\]

\[a(w, u) = \int_\Omega w \nabla \cdot (\mathbf{v} u) \, dx + \int_\Omega \nabla w \cdot (d \nabla u) \, dx\]
Goal-oriented error estimation

\[
\begin{aligned}
\nabla \cdot (vu - d \nabla u) &= f \quad \text{in } \Omega \\
u &= b \quad \text{on } \Gamma
\end{aligned}
\]

Primal problem: find \( u \in H^1_b(\Omega) \)
\[
a(w, u) = (w, f) \quad \forall w \in H^1_0(\Omega)
\]

Dual problem: find \( z \in H^1_0(\Omega) \)
\[
a(z, w) = j(w) \quad \forall w \in H^1_0(\Omega)
\]

Error representation:
\[
u \approx \bar{u} = \sum_j \bar{u}_j \varphi_j, \quad z \approx \bar{z} = \sum_i \bar{z}_i \varphi_i
\]
\[
j(u - \bar{u}) = (z, f) - a(z, \bar{u}) = \rho(z - \bar{z}, \bar{u}) + \rho(\bar{z}, \bar{u})
\]
Goal-oriented error estimation

\[ \begin{cases} \nabla \cdot (\mathbf{v} u - d \nabla u) = f & \text{in } \Omega \\ u = b & \text{on } \Gamma \end{cases} \]

- \( a(w, u) = \int_{\Omega} w \nabla \cdot (\mathbf{v} u) \, dx \)
  \[ + \int_{\Omega} \nabla w \cdot (d \nabla u) \, dx \]

**Primal problem:** find \( u \in H^1_b(\Omega) \)

\[ a(w, u) = (w, f) \quad \forall w \in H^1_0(\Omega) \]

**Dual problem:** find \( z \in H^1_0(\Omega) \)

\[ a(z, w) = j(w) \quad \forall w \in H^1_0(\Omega) \]

**Error representation:**

\[ u \approx \bar{u} = \sum_j \bar{u}_j \varphi_j, \quad z \approx \bar{z} = \sum_i \bar{z}_i \varphi_i \]

\[ j(u - \bar{u}) = (z, f) - a(z, \bar{u}) = \rho(z - \bar{z}, \bar{u}) + \rho(\bar{z}, \bar{u}) \]

Galerkin orthogonality error can be computed
Goal-oriented error estimation

\[
\begin{cases}
\nabla \cdot (v u - d \nabla u) = f & \text{in } \Omega \\
u = b & \text{on } \Gamma
\end{cases}
\]

Primal problem: find \( u \in H^1_b(\Omega) \)

\[
a(w, u) = (w, f) \quad \forall w \in H^1_0(\Omega)
\]

Dual problem: find \( z \in H^1_0(\Omega) \)

\[
a(z, w) = j(w) \quad \forall w \in H^1_0(\Omega)
\]

Error representation:

\[
u \approx \tilde{u} = \sum_j \tilde{u}_j \varphi_j, \quad z \approx \tilde{z} = \sum_i \tilde{z}_i \varphi_i\]

\[
j(u - \tilde{u}) = (z, f) - a(z, \tilde{u}) = \rho(z - \tilde{z}, \tilde{u}) + \rho(\tilde{z}, \tilde{u})
\]

Dual weighted residual error needs to be estimated

Galerkin orthogonality error can be computed
Error splitting

- Approximate dual solution \( z \approx \hat{z} = \sum_i \bar{z}_i \psi_i \)

\[
\rho(\hat{z} - \bar{z}, \bar{u}) = \int_{\Omega} (\hat{z} - \bar{z})(f - \nabla \cdot (\mathbf{v} \bar{u})) \, dx - d \int_{\Omega} \nabla (\hat{z} - \bar{z}) \cdot \nabla \bar{u} \, dx
\]
Error splitting

- Approximate dual solution \( z \approx \hat{z} = \sum_i \bar{z}_i \psi_i \)

\[
\rho(\hat{z} - \bar{z}, \bar{u}) = \int_\Omega (\hat{z} - \bar{z})(f - \nabla \cdot (v \bar{u})) \, dx - d \int_\Omega \nabla(\hat{z} - \bar{z}) \cdot \nabla \bar{u} \, dx
\]

- Continuous gradient approximation \( g(\bar{u}) \approx \nabla \bar{u} \)

\[
0 = \int_\Omega (\hat{z} - \bar{z}) \nabla \cdot g(\bar{u}) \, dx + \int_\Omega \nabla(\hat{z} - \bar{z}) \cdot g(\bar{u}) \, dx
\]
Error splitting

- Approximate dual solution $z \approx \hat{z} = \sum_i \tilde{z}_i \psi_i$

$$\rho(\hat{z} - \bar{z}, \bar{u}) = \int_{\Omega} (\hat{z} - \bar{z})(f - \nabla \cdot (v \bar{u})) \, dx - d \int_{\Omega} \nabla(\hat{z} - \bar{z}) \cdot \nabla \bar{u} \, dx$$

- Continuous gradient approximation $g(\bar{u}) \approx \nabla \bar{u}$

$$0 = \int_{\Omega} (\hat{z} - \bar{z}) \nabla \cdot g(\bar{u}) \, dx + \int_{\Omega} \nabla(\hat{z} - \bar{z}) \cdot g(\bar{u}) \, dx$$

Computable DWR error

$$\rho(\hat{z} - \bar{z}, \bar{u}) = \int_{\Omega} (\hat{z} - \bar{z})(f - \nabla \cdot (v \bar{u} - d g(\bar{u}))) \, dx \quad \text{residual error}$$

$$+ d \int_{\Omega} \nabla(\hat{z} - \bar{z}) \cdot (g(\bar{u}) - \nabla \bar{u}) \, dx \quad \text{diffusive flux error}$$
### Goal-oriented estimate

\[
j(u - \bar{u}) \approx \rho(w, \bar{u}) + \rho(\tilde{z}, \bar{u}), \quad w = \hat{z} - \tilde{z}
\]

\[
|\rho(w, \bar{u})| \leq \Phi = \sum_i \Phi_i, \quad |\rho(\tilde{z}, \bar{u})| \leq \Psi = \sum_i \Psi_i
\]
Node-based error localization

**Goal-oriented estimate**

\[ j(u - \bar{u}) \approx \rho(w, \bar{u}) + \rho(\tilde{z}, \bar{u}), \quad w = \hat{z} - \bar{z} \]

\[ |\rho(w, \bar{u})| \leq \Phi = \sum_i \Phi_i, \quad |\rho(\tilde{z}, \bar{u})| \leq \Psi = \sum_i \Psi_i \]

- **Galerkin error**

\[
\tilde{z} = \sum_i \tilde{z}_i \varphi_i \quad \Rightarrow \quad |\rho(\tilde{z}_i \varphi_i, \bar{u})| = \Psi_i
\]

\[
\Psi_i = \left| \int_{\Omega} \tilde{z}_i \{ \varphi_i (f - \nabla \cdot (v \bar{u})) - \nabla \varphi_i \cdot (d \nabla \bar{u}) \} \, dx \right|
\]
### Goal-oriented estimate

\begin{align*}
  j(u - \bar{u}) &\approx \rho(w, \bar{u}) + \rho(\bar{z}, \bar{u}), \quad w = \hat{z} - \bar{z} \\
  \left| \rho(w, \bar{u}) \right| &\leq \Phi = \sum_i \Phi_i, \quad \left| \rho(\bar{z}, \bar{u}) \right| \leq \Psi = \sum_i \Psi_i
\end{align*}

- **Galerkin error**
  \[
  \bar{z} = \sum_i \bar{z}_i \varphi_i \quad \Rightarrow \quad \left| \rho(\bar{z}_i \varphi_i, \bar{u}) \right| = \Psi_i
  \]

\[
  \Psi_i = \left| \int_\Omega \bar{z}_i \left\{ \varphi_i (f - \nabla \cdot (v \bar{u})) - \nabla \varphi_i \cdot (d \nabla \bar{u}) \right\} \, dx \right|
\]

- **DWR error**
  \[
  \hat{z} = \sum_i \bar{z}_i \psi_i, \quad \hat{z} - \bar{z} = \sum_i w_i, \quad \left| \rho(w_i, \bar{u}) \right| = \Phi_i
  \]

\[
  \Phi_i = \int_\Omega \left| w_i (f - \nabla \cdot (v \bar{u} - d g(\bar{u}))) \right| \, dx + d \int_\Omega \left| \nabla w_i \cdot (g(\bar{u}) - \nabla \bar{u}) \right| \, dx
\]
Node-based error localization

Goal-oriented estimate

\[ j(u - \bar{u}) \approx \rho(w, \bar{u}) + \rho(\bar{z}, \bar{u}), \quad w = \hat{z} - \bar{z} \]

\[ |\rho(w, \bar{u})| \leq \Phi = \sum_i \Phi_i, \quad |\rho(\bar{z}, \bar{u})| \leq \Psi = \sum_i \Psi_i \]

- Galerkin error

\[ \bar{z} = \sum_i \bar{z}_i \varphi_i \quad \Rightarrow \quad |\rho(\bar{z}_i \varphi_i, \bar{u})| = \Psi_i \]

\[ \Psi_i = |\int_\Omega \bar{z}_i \{ \varphi_i (f - \nabla \cdot (v\bar{u})) - \nabla \varphi_i \cdot (d\nabla \bar{u}) \} \, dx| \]

- DWR error

\[ \hat{z} = \sum_i \bar{z}_i \psi_i, \quad \hat{z} - \bar{z} = \sum_i w_i, \quad |\rho(w_i, \bar{u})| = \Phi_i \]

\[ w_i = \bar{z}_i (\psi_i - \varphi_i) \quad \text{(Schmich & Vexler, 2008)} \]
Node-based error localization

Goal-oriented estimate
\[ j(u - \bar{u}) \approx \rho(w, \bar{u}) + \rho(\bar{z}, \bar{u}), \quad w = \hat{z} - \bar{z} \]

\[ |\rho(w, \bar{u})| \leq \Phi = \sum_i \Phi_i, \quad |\rho(\bar{z}, \bar{u})| \leq \Psi = \sum_i \Psi_i \]

- Galerkin error
  \[ \bar{z} = \sum_i \bar{z}_i \varphi_i \Rightarrow |\rho(\bar{z}_i \varphi_i, \bar{u})| = \Psi_i \]
  \[ \Psi_i = \left| \int_\Omega \bar{z}_i \{ \varphi_i (f - \nabla \cdot (\mathbf{v} \bar{u})) - \nabla \varphi_i \cdot (d \nabla \bar{u}) \} \, dx \right| \]

- DWR error
  \[ \hat{z} = \sum_i \bar{z}_i \psi_i, \quad \hat{z} - \bar{z} = \sum_i w_i, \quad |\rho(w_i, \bar{u})| = \Phi_i \]

Alternative:
\[ w_i = \varphi_i (\hat{z} - \bar{z}), \quad \sum_i \varphi_i \equiv 1 \]
A posteriori error estimate

Conversion to element contributions

\[ |j(u - \bar{u})| \leq \Phi + \Psi =: \eta, \quad \eta = \sum_k \eta_k = \sum_i \Phi_i + \Psi_i \]
A posteriori error estimate

Conversion to element contributions

\[ |j(u - \bar{u})| \leq \Phi + \Psi =: \eta, \quad \eta = \sum_k \eta_k = \sum_i \Phi_i + \Psi_i \]

- Continuous error function
  \[ \xi = \sum_i \xi_i \varphi_i, \quad \xi_i = \frac{\Phi_i + \Psi_i}{\int_\Omega \varphi_i \, dx} \]
A posteriori error estimate

Conversion to element contributions

\[ |j(u - \bar{u})| \leq \Phi + \Psi =: \eta, \quad \eta = \sum_k \eta_k = \sum_i \Phi_i + \Psi_i \]

- **Continuous error function**
  \[ \xi = \sum_i \xi_i \varphi_i, \quad \xi_i = \frac{\Phi_i + \Psi_i}{\int_{\Omega} \varphi_i \, dx} \]

- **Element contribution**
  \[ \eta_k = \int_{\Omega_k} \xi \, dx, \quad \forall \Omega_k \subset \Omega \]
A posteriori error estimate

Conversion to element contributions

\[ |j(u - \bar{u})| \leq \Phi + \Psi =: \eta, \quad \eta = \sum_k \eta_k = \sum_i \Phi_i + \Psi_i \]

- **Continuous error function**
  \[ \xi = \sum_i \xi_i \varphi_i, \quad \xi_i = \frac{\Phi_i + \Psi_i}{\int_{\Omega} \varphi_i \, dx} \]

- **Element contribution**
  \[ \eta_k = \int_{\Omega_k} \xi \, dx, \quad \forall \Omega_k \subset \Omega \]

- **Effectivity index**
  \[ I_{\text{eff}} = \frac{\eta}{|j(u - \bar{u})|} \]
A posteriori error estimate

Conversion to element contributions

\[ |j(u - \bar{u})| \leq \Phi + \Psi =: \eta, \quad \eta = \sum_k \eta_k = \sum_i \Phi_i + \Psi_i \]

- **Continuous error function**
  \[ \xi = \sum_i \xi_i \varphi_i, \quad \xi_i = \frac{\Phi_i + \Psi_i}{\int_{\Omega} \varphi_i \, dx} \]

- **Element contribution**
  \[ \eta_k = \int_{\Omega_k} \xi \, dx, \quad \forall \Omega_k \subset \Omega \]

- **Relative effectivity index**
  \[ I_{\text{rel}} = \left| \frac{\eta - |j(u - \bar{u})|}{j(u)} \right| \]
Convection-diffusion in 1D

\[ \text{Pe} \frac{du}{dt} - \frac{d^2u}{dx^2} = 0, \quad u(0) = 0, \quad u(1) = 1, \quad j(u) = \int_0^1 u \, dx \]

Discretization: central difference scheme, \( h = 1/10 \)

| Pe  | \( |j(u - \bar{u})| \)  | \( \Phi \)  | \( \Psi \)  | \( \eta \)  | \( I_{\text{rel}} \) |
|-----|----------------------|------------|------------|------------|-------------|
| 1   | 7.67e-04             | 7.80e-04   | 4.09e-16   | 7.80e-04   | 3.05e-05    |
| 10  | 2.84e-05             | 4.10e-05   | 3.56e-18   | 4.10e-05   | 1.25e-04    |
| 100 | –                    | –          | –          | –          | –           |
Convection-diffusion in 1D

\[ \text{Pe} \frac{du}{dt} - \frac{d^2 u}{dx^2} = 0, \quad u(0) = 0, \quad u(1) = 1, \quad j(u) = \int_0^1 u \, dx \]

Discretization: upwind difference scheme, \( h = 1/10 \)

| Pe | \( |j(u - \bar{u})| \) | \( \Phi \) | \( \Psi \) | \( \eta \) | \( I_{rel} \) |
|----|----------------|---------|---------|-------|-------|
| 1  | 4.52e-03       | 7.38e-04| 3.58e-03| 4.32e-03| 4.79e-04|
| 10 | 4.91e-02       | 3.06e-04| 4.76e-02| 4.79e-02| 1.21e-02|
| 100| 5.00e-02       | 1.59e-09| 5.00e-02| 5.00e-02| 1.21e-08|
Convection-diffusion in 1D

\[ \text{Pe} \frac{du}{dt} - \frac{d^2 u}{dx^2} = 0, \quad u(0) = 0, \quad u(1) = 1, \quad j(u) = \int_0^1 u \, dx \]

Discretization: TVD scheme, MC limiter, \( h = 1/10 \)

<table>
<thead>
<tr>
<th>Pe</th>
<th>( j(u - \bar{u}) )</th>
<th>( \Phi )</th>
<th>( \Psi )</th>
<th>( \eta )</th>
<th>( I_{rel} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.03e-03</td>
<td>7.74e-04</td>
<td>2.60e-04</td>
<td>1.03e-03</td>
<td>1.34e-05</td>
</tr>
<tr>
<td>10</td>
<td>1.51e-02</td>
<td>9.12e-05</td>
<td>1.50e-02</td>
<td>1.51e-02</td>
<td>3.81e-05</td>
</tr>
<tr>
<td>100</td>
<td>4.51e-02</td>
<td>4.23e-09</td>
<td>4.51e-02</td>
<td>4.51e-02</td>
<td>1.97e-07</td>
</tr>
</tbody>
</table>
Mesh adaptation

**Circular convection**

\[ \nabla \cdot (vu) = 0 \quad \text{in} \quad \Omega = (-1, 1) \times (0, 1) \]

\[ u(x, y) = \begin{cases} 
1, & 0.35 \leq r \leq 0.65 \\
0, & \text{otherwise} 
\end{cases} \]

\[ r(x, y) = \sqrt{x^2 + y^2} \]

**Target functional**

\[ j(u) = \int_\Omega u \, dx, \quad \omega = (-0.1, 0.1) \times (0, 1) \]

**Domain and velocity**

FEM-TVD, \( h = 1/64 \)
Mesh adaptation

Circular convection \( \nabla \cdot (\mathbf{v} u) = 0 \) in \( \Omega = (-1, 1) \times (0, 1) \)

\[
 u(x, y) = \begin{cases} 
 1, & 0.35 \leq r \leq 0.65 \\
 0, & \text{otherwise}
\end{cases}
\]

\( r(x, y) = \sqrt{x^2 + y^2} \)

Target functional \( j(u) = \int_\omega u \, dx \), \( \omega = (-0.1, 0.1) \times (0, 1) \)

Goal-oriented mesh adaptation

\( j(u - \bar{u}) \approx \rho(z_h, \bar{u}) \)
Conclusions and outlook

- dynamic $h$-adaptation for unsteady flow problems
  - red-green strategy yields an adaptive mesh hierarchy
  - re-coarsening is based on the vertex locking algorithm
  - nodal generation function provides mesh information
Conclusions and outlook

- dynamic $h$-adaptation for unsteady flow problems
  - red-green strategy yields an adaptive mesh hierarchy
  - re-coarsening is based on the *vertex locking* algorithm
  - nodal generation function provides mesh information

- goal-oriented error estimation for steady flow problems
  - weighted residuals are evaluated without jump terms
  - target functional is decomposed into nodal contributions
  - Galerkin orthogonality error is used per se for adaptation
Conclusions and outlook

- dynamic $h$-adaptation for unsteady flow problems
  - red-green strategy yields an adaptive mesh hierarchy
  - re-coarsening is based on the *vertex locking* algorithm
  - nodal generation function provides mesh information

- goal-oriented error estimation for steady flow problems
  - weighted residuals are evaluated without jump terms
  - target functional is decomposed into nodal contributions
  - Galerkin orthogonality error is used per se for adaptation

- implementation of mesh adaptation procedure in 3D
- goal-oriented error estimation for unsteady flow problems
- extension to the compressible Navier-Stokes equations