fakultät für mathematik

technische universität dortmund

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

Transport problems

Scalar conservation law

$$\frac{\partial u}{\partial t} + \nabla \cdot \mathbf{f}(u) = 0$$

Transport problems

Scalar conservation law

$$\frac{\partial u}{\partial t} + \nabla \cdot \mathbf{f}(u) = 0$$

- Convection-diffusion equation

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

Scalar conservation law

$$\frac{\partial u}{\partial t} + \nabla \cdot \mathbf{f}(u) = 0$$

Galerkin FEM

$$M_C \frac{\mathrm{d}u}{\mathrm{d}t} = Ku$$

- Convection-diffusion equation

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

Transport problems

Scalar conservation law

$$\frac{\partial u}{\partial t} + \nabla \cdot \mathbf{f}(u) = 0$$

Low-order scheme

$$M_L \frac{\mathrm{d}u}{\mathrm{d}t} = Ku + Du = Lu$$

- Convection-diffusion equation

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

Transport problems

Scalar conservation law

$$\frac{\partial u}{\partial t} + \nabla \cdot \mathbf{f}(u) = 0$$

High-resolution scheme

$$M_L \frac{\mathrm{d}u}{\mathrm{d}t} = Lu + \bar{f}(u)$$

- Convection-diffusion equation

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

Transport problems

Scalar conservation law

$$\frac{\partial u}{\partial t} + \nabla \cdot \mathbf{f}(u) = 0$$

High-resolution scheme

$$M_L \frac{\mathrm{d}u}{\mathrm{d}t} = Lu + \bar{f}(u)$$

- Convection-diffusion equation

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

- Compressible Euler equations

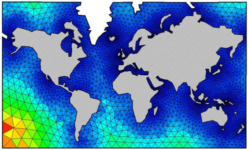
$$\frac{\partial}{\partial t}\begin{pmatrix} \rho \\ \rho\mathbf{v} \\ \rho E \end{pmatrix} + \nabla \cdot \begin{pmatrix} \rho\mathbf{v} \\ \rho\mathbf{v}\otimes\mathbf{v} + pI \\ (\rho E + p)\mathbf{v} \end{pmatrix} = 0$$

Algebraic flux correction $\hookrightarrow$ *talks by D. Kuzmin, M. Gurris*

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

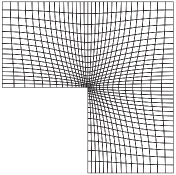
Unstructured meshes

- 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?

technische universität
dortmund

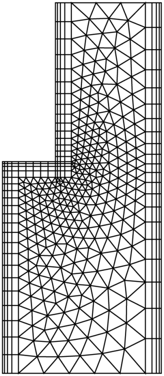
Unstructured meshes

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

AFC schemes can handle hybrid meshes

Structured grids

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



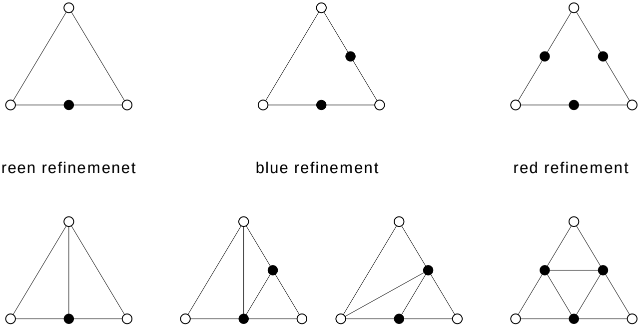
Design goals for h-adaptation

- 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 *R.E. Bank, A.H. Sherman, A. Weiser*

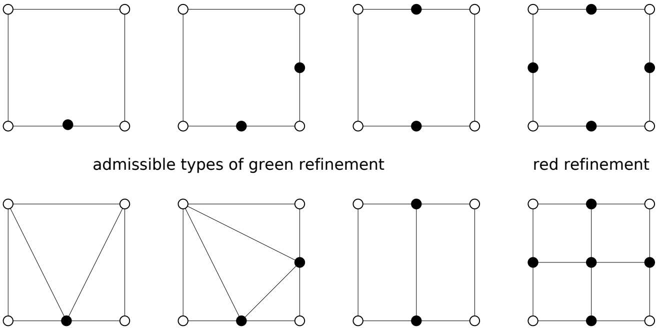
1. subdivide marked elements regularly (red refinement)
2. eliminate ‘hanging nodes’ by transition cells (green refinement)



The red-green strategy revisited

Refinement algorithm in 2D — *R.E. Bank, A.H. Sherman, A. Weiser*

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



Mesh genealogy

technische universität dortmund

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

$$\mathcal{E}_m = \{\Omega_k \,:\, k = 1, \dots, N_{\mathrm{E}}\} \quad \text{and} \quad \mathcal{V}_m = \{\mathrm{v}_i \,:\, i = 1, \dots, N_{\mathrm{V}}\}$$

Mesh genealogy

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

$$\mathcal{E}_m = \{\Omega_k \, : \, k = 1, \ldots, N_\mathrm{E}\} \quad \text{and} \quad \mathcal{V}_m = \{\mathrm{v}_i \, : \, i = 1, \ldots, N_\mathrm{V}\}$$

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

$$g(\mathrm{v}_i) := \begin{cases} 0 & \text{if} \quad \mathrm{v}_i \in \mathcal{V}_0 \\ \max\limits_{\mathrm{v}_j \in \Gamma_{kl}} g(\mathrm{v}_j) + 1 & \text{if} \quad \mathrm{v}_i \in \Gamma_{kl} := \bar{\Omega}_k \cap \bar{\Omega}_l \\ \max\limits_{\mathrm{v}_j \in \partial\Omega_k} g(\mathrm{v}_j) + 1 & \text{if} \quad \mathrm{v}_i \in \Omega_k \setminus \partial\Omega_k \end{cases}$$

Mesh genealogy

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

$$\mathcal{E}_m = \{\Omega_k \;:\; k = 1, \ldots, N_{\mathrm{E}}\} \quad \text{and} \quad \mathcal{V}_m = \{\mathrm{v}_i \;:\; i = 1, \ldots, N_{\mathrm{V}}\}$$

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

$$g(\mathrm{v}_i) := \begin{cases} 0 & \text{if} \quad \mathrm{v}_i \in \mathcal{V}_0 \\ \max\limits_{\mathrm{v}_j \in \Gamma_{kl}} g(\mathrm{v}_j) + 1 & \text{if} \quad \mathrm{v}_i \in \Gamma_{kl} := \bar{\Omega}_k \cap \bar{\Omega}_l \\ \max\limits_{\mathrm{v}_j \in \partial\Omega_k} g(\mathrm{v}_j) + 1 & \text{if} \quad \mathrm{v}_i \in \Omega_k \setminus \partial\Omega_k \end{cases}$$

Mesh genealogy

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

$$\mathcal{E}_m = \{\Omega_k \,:\, k = 1, \ldots, N_{\mathrm{E}}\} \quad \text{and} \quad \mathcal{V}_m = \{\mathrm{v}_i \,:\, i = 1, \ldots, N_{\mathrm{V}}\}$$

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

$$g(\mathrm{v}_i) := \begin{cases} 0 & \text{if} \quad \mathrm{v}_i \in \mathcal{V}_0 \\ \max\limits_{\mathrm{v}_j \in \Gamma_{kl}} g(\mathrm{v}_j) + 1 & \text{if} \quad \mathrm{v}_i \in \Gamma_{kl} := \bar{\Omega}_k \cap \bar{\Omega}_l \\ \max\limits_{\mathrm{v}_j \in \partial\Omega_k} g(\mathrm{v}_j) + 1 & \text{if} \quad \mathrm{v}_i \in \Omega_k \setminus \partial\Omega_k \end{cases}$$

Mesh genealogy

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

$$\mathcal{E}_m = \{\Omega_k \,:\, k = 1, \ldots, N_{\mathrm{E}}\} \quad \text{and} \quad \mathcal{V}_m = \{\mathrm{v}_i \,:\, i = 1, \ldots, N_{\mathrm{V}}\}$$

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

$$g(\mathrm{v}_i) := \begin{cases} 0 & \text{if} \quad \mathrm{v}_i \in \mathcal{V}_0 \\ \max\limits_{\mathrm{v}_j \in \Gamma_{kl}} g(\mathrm{v}_j) + 1 & \text{if} \quad \mathrm{v}_i \in \Gamma_{kl} := \bar{\Omega}_k \cap \bar{\Omega}_l \\ \max\limits_{\mathrm{v}_j \in \partial\Omega_k} g(\mathrm{v}_j) + 1 & \text{if} \quad \mathrm{v}_i \in \Omega_k \setminus \partial\Omega_k \end{cases}$$

Mesh genealogy

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

$\mathcal{E}_m = \{\Omega_k \,:\, k = 1, \ldots, N_{\mathrm{E}}\}$ and $\mathcal{V}_m = \{\mathrm{v}_i \,:\, i = 1, \ldots, N_{\mathrm{V}}\}$

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

$$g(\mathrm{v}_i) := \begin{cases} 0 & \text{if} \quad \mathrm{v}_i \in \mathcal{V}_0 \\ \max\limits_{\mathrm{v}_j \in \Gamma_{kl}} g(\mathrm{v}_j) + 1 & \text{if} \quad \mathrm{v}_i \in \Gamma_{kl} := \bar{\Omega}_k \cap \bar{\Omega}_l \\ \max\limits_{\mathrm{v}_j \in \partial\Omega_k} g(\mathrm{v}_j) + 1 & \text{if} \quad \mathrm{v}_i \in \Omega_k \setminus \partial\Omega_k \end{cases}$$

- 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

technische universität
dortmund

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

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 $d(\mathrm{v}_i) := g(\mathrm{v}_i),\ \forall \mathrm{v}_i \in \mathcal{V}_m \quad \Rightarrow \quad d(\mathrm{v}_i) = 0,\ \forall \mathrm{v}_i \in \mathcal{V}_0$

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 $d(\mathrm{v}_i) := g(\mathrm{v}_i),\ \forall \mathrm{v}_i \in \mathcal{V}_m \quad \Rightarrow \quad d(\mathrm{v}_i) = 0,\ \forall \mathrm{v}_i \in \mathcal{V}_0$
2. vertex $\mathrm{v}_i \in \mathcal{V}_m$ is locked, i.e. $d(\mathrm{v}_i) := -|d(\mathrm{v}_i)|$ 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

technische universität dortmund

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 $d(\mathrm{v}_i) := g(\mathrm{v}_i),\ \forall \mathrm{v}_i \in \mathcal{V}_m \quad \Rightarrow \quad d(\mathrm{v}_i) = 0,\ \forall \mathrm{v}_i \in \mathcal{V}_0$
2. vertex $\mathrm{v}_i \in \mathcal{V}_m$ is locked, i.e. $d(\mathrm{v}_i) := -|d(\mathrm{v}_i)|$ 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 $g(\mathrm{v}_i) < g(\mathrm{v}_j)$ for some $\mathrm{v}_j \in \mathcal{V}_m$

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 $d(\mathrm{v}_i) := g(\mathrm{v}_i),\ \forall \mathrm{v}_i \in \mathcal{V}_m \quad \Rightarrow \quad d(\mathrm{v}_i) = 0,\ \forall \mathrm{v}_i \in \mathcal{V}_0$
2. vertex $\mathrm{v}_i \in \mathcal{V}_m$ is locked, i.e. $d(\mathrm{v}_i) := -|d(\mathrm{v}_i)|$ 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 $g(\mathrm{v}_i) < g(\mathrm{v}_j)$ for some $\mathrm{v}_j \in \mathcal{V}_m$
3. vertices are locked to preclude the creation of blue elements

Mesh re-coarsening

technische universität dortmund

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 $d(\mathrm{v}_i) := g(\mathrm{v}_i),\ \forall \mathrm{v}_i \in \mathcal{V}_m \quad \Rightarrow \quad d(\mathrm{v}_i) = 0,\ \forall \mathrm{v}_i \in \mathcal{V}_0$
2. vertex $\mathrm{v}_i \in \mathcal{V}_m$ is locked, i.e. $d(\mathrm{v}_i) := -|d(\mathrm{v}_i)|$ 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 $g(\mathrm{v}_i) < g(\mathrm{v}_j)$ for some $\mathrm{v}_j \in \mathcal{V}_m$
3. vertices are locked to preclude the creation of blue elements

Result: Vertex v_i is locked if $d(\mathrm{v}_i) \leq 0$; otherwise it can be deleted.

Mesh re-coarsening

technische universität dortmund

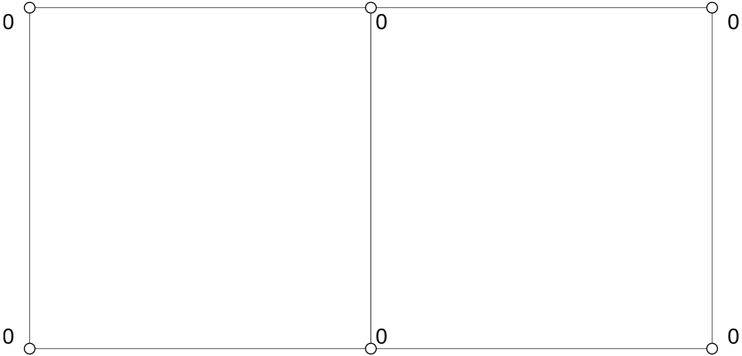
Re-coarsening algorithms (vertex-based approach)

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

1. initialize $d(\mathrm{v}_i) := g(\mathrm{v}_i),\ \forall \mathrm{v}_i \in \mathcal{V}_m \quad \Rightarrow \quad d(\mathrm{v}_i) = 0,\ \forall \mathrm{v}_i \in \mathcal{V}_0$
2. vertex $\mathrm{v}_i \in \mathcal{V}_m$ is locked, i.e. $d(\mathrm{v}_i) := -|d(\mathrm{v}_i)|$ 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 $g(\mathrm{v}_i) < g(\mathrm{v}_j)$ for some $\mathrm{v}_j \in \mathcal{V}_m$
3. vertices are locked to preclude the creation of blue elements

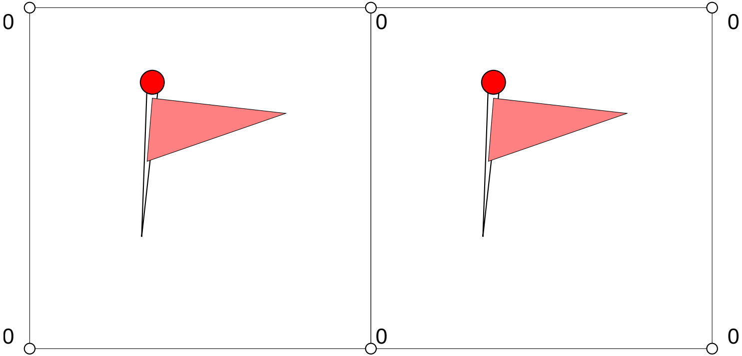
Result: Vertex v_i is locked if $d(\mathrm{v}_i) \leq 0$; otherwise it can be deleted.
All vertices of the initial mesh are locked by construction!

Refinement algorithm: initial mesh

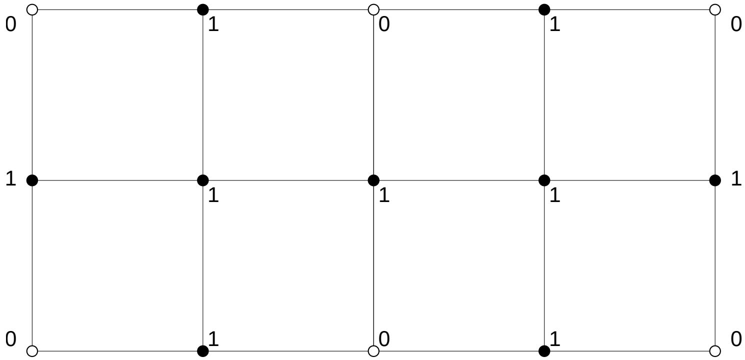


Step-by-step illustration

Refinement algorithm: mark elements for regular refinement

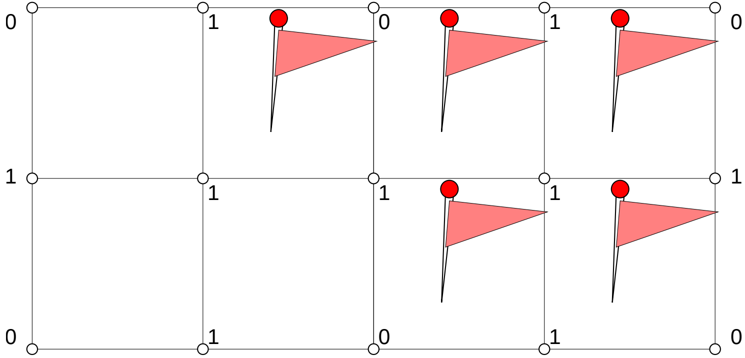


Refinement algorithm: perform regular refinement



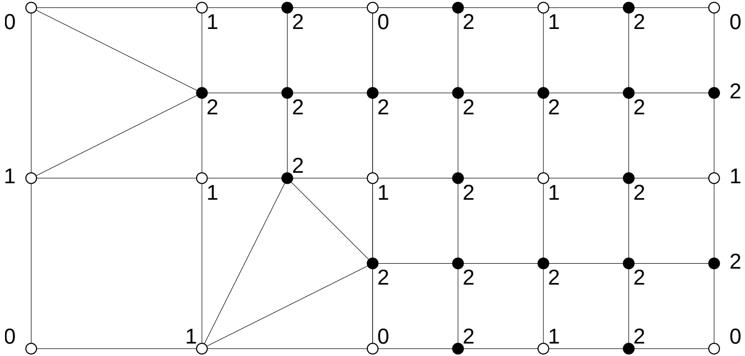
Step-by-step illustration

Refinement algorithm: mark elements for regular refinement



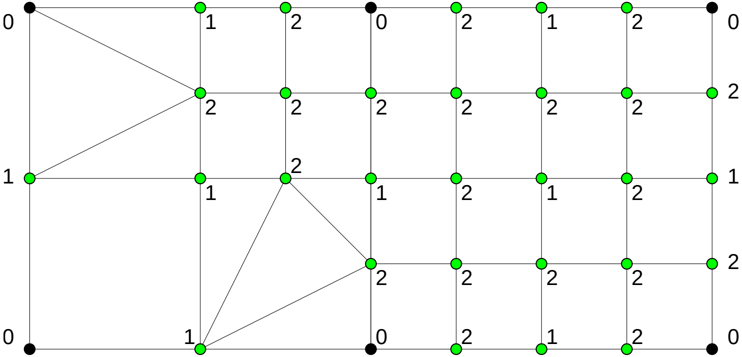
Step-by-step illustration

Refinement algorithm: perform regular refinement + transition cells

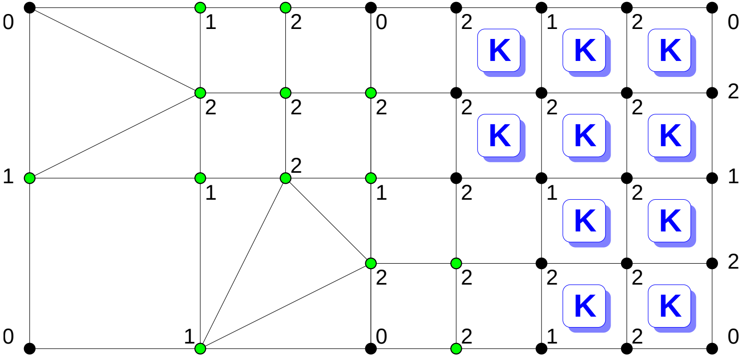


Step-by-step illustration

Re-coarsening algorithm: vertices from initial mesh are locked



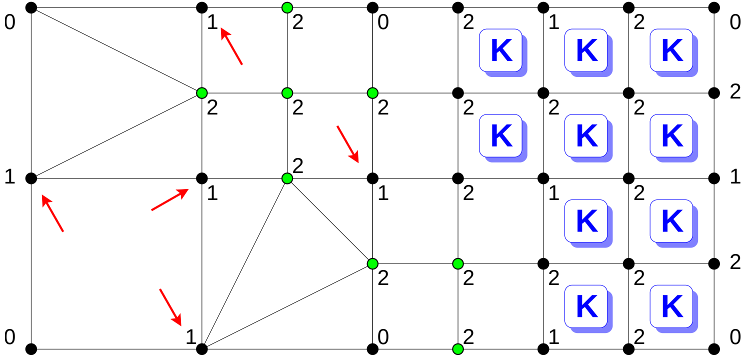
Re-coarsening algorithm: keep cells and lock connected vertices



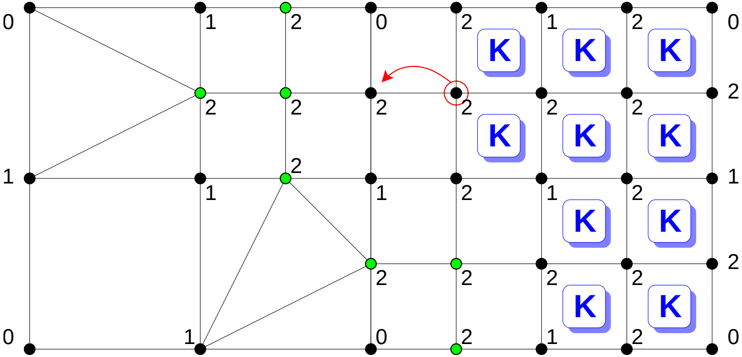
Step-by-step illustration

technische universität
dortmund

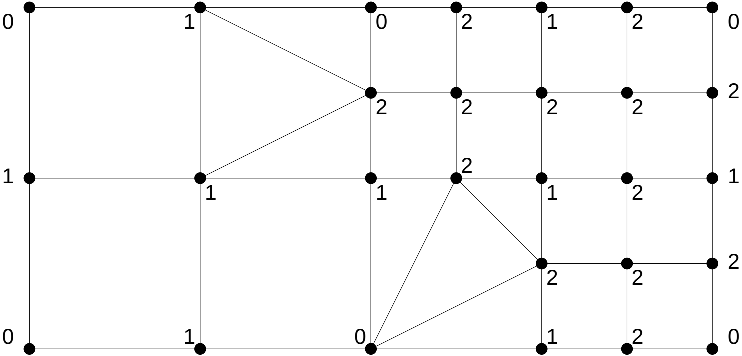
Re-coarsening algorithm: lock vertices if there are younger neighbors



Re-coarsening algorithm: lock vertices to preclude blue elements



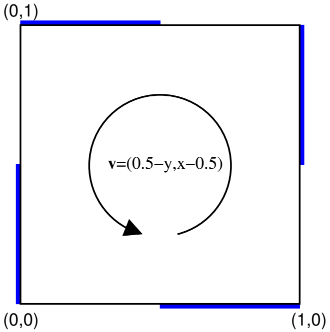
Re-coarsening algorithm: remove vertices and update elements



Solid body rotation

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

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



domain and velocity

- dynamic mesh adaptation
  - every 5 time steps
  - protective layers
- approximate $\nabla u \approx \mathbf{g}(\nabla u_h)$

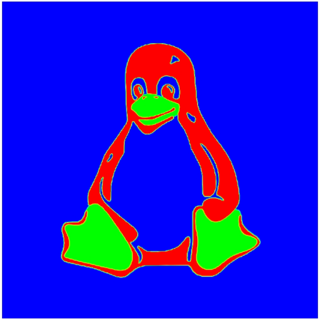
$$\|\nabla u - \nabla u_h\|^2_\Omega \approx \sum_k \eta_k$$

where $\eta_k = \|\mathbf{g} - \nabla u_h\|^2_{\Omega_k}$

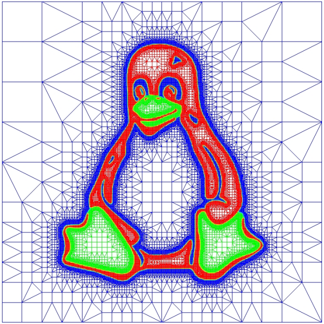
Solid body rotation

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

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



initial/exact solution



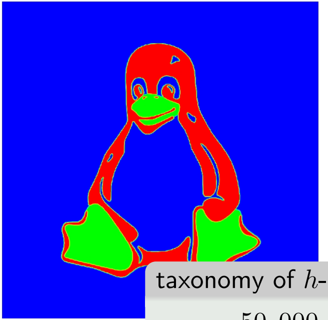
$1/512 \leq h \leq 1/8$

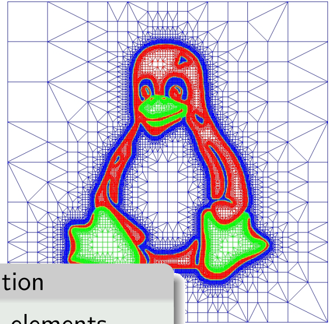
Solid body rotation

technische universität dortmund

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

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





initial/exa

$\leq 1/8$

taxonomy of h-adaptation

$\sim 50,000$ P_1/Q_1 elements

vs.

>1 million Q_1 elements

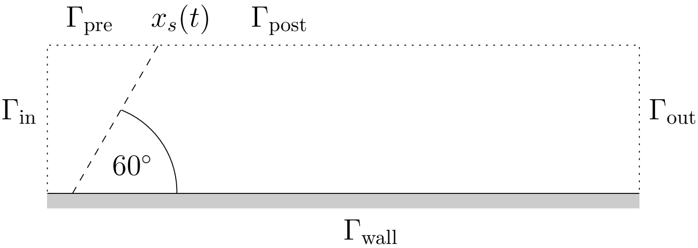
Double Mach reflection

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

$$\begin{bmatrix} \rho_{\text{pre}} \\ u_{\text{pre}} \\ v_{\text{pre}} \\ p_{\text{pre}} \end{bmatrix} = \begin{bmatrix} 8.0 \\ 8.25\cos(30^\circ) \\ -8.25\sin(30^\circ) \\ 116.5 \end{bmatrix} \qquad \begin{bmatrix} \rho_{\text{post}} \\ u_{\text{post}} \\ v_{\text{post}} \\ p_{\text{post}} \end{bmatrix} = \begin{bmatrix} 1.4 \\ 0.0 \\ 0.0 \\ 1.0 \end{bmatrix}$$

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

$$\Gamma_{\text{pre}} = \{x < x_s(t), y = 1\}, \qquad \Gamma_{\text{post}} = \{x \geq x_s(t), y = 1\}$$



Goal-oriented error estimation

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

Goal-oriented error estimation

technische universität dortmund

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

- $$a(w,u) = \int_\Omega w \nabla \cdot (\mathbf{v}u)\,\mathrm{d}\mathbf{x} + \int_\Omega \nabla w \cdot (d\nabla u)\,\mathrm{d}\mathbf{x}$$

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 $z \in H^1_0(\Omega)$

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

Goal-oriented error estimation

technische universität dortmund

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

- $a(w,u) = \int_\Omega w \nabla \cdot (\mathbf{v}u)\,\mathrm{d}\mathbf{x} + \int_\Omega \nabla w \cdot (d\nabla u)\,\mathrm{d}\mathbf{x}$

Primal problem: find $u \in H_b^1(\Omega)$

$$a(w,u) = (w,f) \quad \forall w \in H_0^1(\Omega)$$

Dual problem: find $z \in H_0^1(\Omega)$

$$a(z,w) = j(w) \quad \forall w \in H_0^1(\Omega)$$

Error representation: $u \approx \bar{u} = \sum_j \bar{u}_j \varphi_j$

$$j(u - \bar{u}) = (z,f) - a(z,\bar{u}) = \rho(z,\bar{u})$$

Goal-oriented error estimation

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

- $$a(w,u) = \int_\Omega w \nabla \cdot (\mathbf{v}u)\,\mathrm{d}\mathbf{x} + \int_\Omega \nabla w \cdot (d\nabla u)\,\mathrm{d}\mathbf{x}$$

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

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

- $$a(w,u) = \int_\Omega w\nabla \cdot (\mathbf{v}u)\,\mathrm{d}\mathbf{x} + \int_\Omega \nabla w \cdot (d\nabla u)\,\mathrm{d}\mathbf{x}$$

Primal problem: find $u \in H_b^1(\Omega)$

$$a(w,u) = (w,f) \quad \forall w \in H_0^1(\Omega)$$

Dual problem: find $z \in H_0^1(\Omega)$

$$a(z,w) = j(w) \quad \forall w \in H_0^1(\Omega)$$

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

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

- $a(w,u) = \int_\Omega w\nabla \cdot (\mathbf{v}u)\,\mathrm{d}\mathbf{x} + \int_\Omega \nabla w \cdot (d\nabla u)\,\mathrm{d}\mathbf{x}$

Primal problem: find $u \in H_b^1(\Omega)$

$$a(w,u) = (w,f) \quad \forall w \in H_0^1(\Omega)$$

Dual problem: find $z \in H_0^1(\Omega)$

$$a(z,w) = j(w) \quad \forall w \in H_0^1(\Omega)$$

Error representation: $u \approx \bar{u} = \sum_j \bar{u}_j \varphi_j, \quad z \approx \bar{z} = \sum_i \bar{z}_i \varphi_i$

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

Dual weighted residual error needs to be estimated ($\rho(z-\bar{z},\bar{u})$)

Galerkin orthogonality error can be computed ($\rho(\bar{z},\bar{u})$)

Error splitting

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

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

Error splitting

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

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

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

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

Error splitting

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

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

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

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

Computable DWR error

$$\rho(\hat{z} - \bar{z}, \bar{u}) = \int_\Omega (\hat{z} - \bar{z})(f - \nabla \cdot (\mathbf{v}\bar{u} - d\,\mathbf{g}(\bar{u})))\,\mathrm{d}\mathbf{x} \qquad \textit{residual error}$$

$$+ \; d \int_\Omega \nabla(\hat{z} - \bar{z}) \cdot (\mathbf{g}(\bar{u}) - \nabla \bar{u})\,\mathrm{d}\mathbf{x} \qquad \textit{diffusive flux error}$$

Node-based error localization

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

$$|\rho(w,\bar{u})| \leq \Phi = \sum_i \Phi_i, \qquad |\rho(\bar{z},\bar{u})| \leq \Psi = \sum_i \Psi_i$$

Node-based error localization

technische universität dortmund

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

$$|\rho(w,\bar{u})| \leq \Phi = \sum_i \Phi_i, \qquad |\rho(\bar{z},\bar{u})| \leq \Psi = \sum_i \Psi_i$$

- Galerkin error $\quad \bar{z} = \sum_i \bar{z}_i \varphi_i \quad \Rightarrow \quad |\rho(\bar{z}_i\varphi_i,\bar{u})| = \Psi_i$

$$\Psi_i = \left| \int_\Omega \bar{z}_i \{ \varphi_i (f - \nabla \cdot (\mathbf{v}\bar{u})) - \nabla \varphi_i \cdot (d \nabla \bar{u}) \} \, \mathrm{d}\mathbf{x} \right|$$

Node-based error localization

technische universität dortmund

Goal-oriented estimate $j(u-\bar u) \approx \rho(w,\bar u) + \rho(\bar z,\bar u), \quad w = \hat z - \bar z$

$$|\rho(w,\bar u)| \le \Phi = \textstyle\sum_i \Phi_i, \qquad |\rho(\bar z,\bar u)| \le \Psi = \textstyle\sum_i \Psi_i$$

- Galerkin error $\bar z = \sum_i \bar z_i \varphi_i \quad \Rightarrow \quad |\rho(\bar z_i \varphi_i, \bar u)| = \Psi_i$

$$\Psi_i = \left|\int_\Omega \bar z_i \{\varphi_i (f - \nabla\cdot(\mathbf{v}\bar u)) - \nabla\varphi_i \cdot (d\nabla \bar u)\}\,\mathrm{d}\mathbf{x}\right|$$

- DWR error $\hat z = \sum_i \bar z_i \psi_i, \quad \hat z - \bar z = \sum_i w_i, \quad |\rho(w_i,\bar u)| = \Phi_i$

$$\Phi_i = \int_\Omega |w_i (f - \nabla\cdot(\mathbf{v}\bar u - d\,\mathbf{g}(\bar u)))|\,\mathrm{d}\mathbf{x} + d\int_\Omega |\nabla w_i \cdot (\mathbf{g}(\bar u) - \nabla \bar u)|\,\mathrm{d}\mathbf{x}$$

Node-based error localization

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

$$|\rho(w,\bar{u})| \leq \Phi = \sum_i \Phi_i, \qquad |\rho(\bar{z},\bar{u})| \leq \Psi = \sum_i \Psi_i$$

- Galerkin error $\bar{z} = \sum_i \bar{z}_i\varphi_i \quad \Rightarrow \quad |\rho(\bar{z}_i\varphi_i,\bar{u})| = \Psi_i$

$$\Psi_i = \left|\int_\Omega \bar{z}_i\{\varphi_i(f - \nabla\cdot(\mathbf{v}\bar{u})) - \nabla\varphi_i\cdot(d\nabla\bar{u})\}\,\mathrm{d}\mathbf{x}\right|$$

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

$$w_i = \bar{z}_i(\psi_i - \varphi_i)$$ (Schmich & Vexler, 2008)

Node-based error localization

technische universität dortmund

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

$$|\rho(w,\bar{u})| \le \Phi = \sum_i \Phi_i, \qquad |\rho(\bar{z},\bar{u})| \le \Psi = \sum_i \Psi_i$$

- Galerkin error $\quad \bar{z} = \sum_i \bar{z}_i \varphi_i \quad \Rightarrow \quad |\rho(\bar{z}_i\varphi_i, \bar{u})| = \Psi_i$

$$\Psi_i = \left| \int_\Omega \bar{z}_i \{\varphi_i (f - \nabla \cdot (\mathbf{v}\bar{u})) - \nabla \varphi_i \cdot (d\nabla \bar{u})\} \, \mathrm{d}\mathbf{x} \right|$$

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

Alternative: $\quad w_i = \varphi_i(\hat{z} - \bar{z}), \quad \sum_i \varphi_i \equiv 1$

A posteriori error estimate

Conversion to element contributions

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

A posteriori error estimate

Conversion to element contributions

$$|j(u - \bar{u})| \le \Phi + \Psi =: \eta, \qquad \eta = \sum_k \eta_k = \sum_i \Phi_i + \Psi_i$$

- Continuous error function $\xi = \sum_i \xi_i \varphi_i, \qquad \xi_i = \frac{\Phi_i + \Psi_i}{\int_\Omega \varphi_i \,\mathrm{d}\mathbf{x}}$

A posteriori error estimate

Conversion to element contributions

$$|j(u - \bar{u})| \le \Phi + \Psi =: \eta, \qquad \eta = \textstyle\sum_k \eta_k = \sum_i \Phi_i + \Psi_i$$

- Continuous error function $$\xi = \sum_i \xi_i \varphi_i, \qquad \xi_i = \frac{\Phi_i + \Psi_i}{\int_\Omega \varphi_i \, \mathrm{d}\mathbf{x}}$$
- Element contribution $$\eta_k = \int_{\Omega_k} \xi \, \mathrm{d}\mathbf{x}, \qquad \forall \Omega_k \subset \Omega$$

A posteriori error estimate

Conversion to element contributions

$$|j(u-\bar{u})| \;\leq\; \Phi + \Psi =: \eta, \qquad \eta = \textstyle\sum_k \eta_k = \sum_i \Phi_i + \Psi_i$$

- Continuous error function $\quad \xi = \sum_i \xi_i \varphi_i, \qquad \xi_i = \dfrac{\Phi_i + \Psi_i}{\int_\Omega \varphi_i \, \mathrm{d}\mathbf{x}}$
- Element contribution $\quad \eta_k = \displaystyle\int_{\Omega_k} \xi \, \mathrm{d}\mathbf{x}, \qquad \forall \Omega_k \subset \Omega$
- Effectivity index $\quad I_{\text{eff}} = \dfrac{\eta}{|j(u-\bar{u})|}$

A posteriori error estimate

Conversion to element contributions

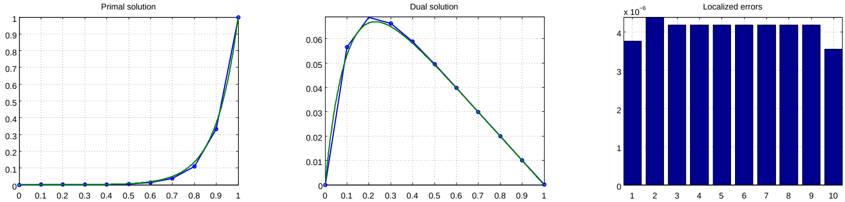
$$|j(u-\bar{u})| \le \Phi + \Psi =: \eta, \qquad \eta = \textstyle\sum_k \eta_k = \sum_i \Phi_i + \Psi_i$$

- Continuous error function $$\xi = \sum_i \xi_i \varphi_i, \qquad \xi_i = \frac{\Phi_i + \Psi_i}{\int_\Omega \varphi_i \, \mathrm{d}\mathbf{x}}$$
- Element contribution $$\eta_k = \int_{\Omega_k} \xi \, \mathrm{d}\mathbf{x}, \qquad \forall \Omega_k \subset \Omega$$
- Relative effectivity index $$I_{\mathrm{rel}} = \left| \frac{\eta - |j(u-\bar{u})|}{j(u)} \right|$$

Convection-diffusion in 1D

technische universität dortmund

$$\mathrm{Pe}\,\frac{\mathrm{d}u}{\mathrm{d}t} - \frac{\mathrm{d}^2 u}{\mathrm{d}x^2} = 0, \quad u(0) = 0, \quad u(1) = 1, \quad j(u) = \int_0^1 u\,\mathrm{d}x$$



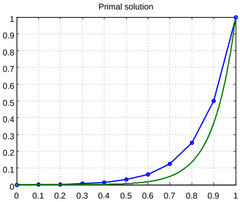
Discretization: central difference scheme, $h = 1/10$

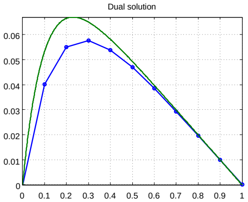
Pe	$\lvert j(u-\bar{u})\rvert$	Φ	Ψ	η	I_{rel}
1	7.67e-04	7.80e-04	4.09e-16	7.80e-04	3.05e-05
10	2.84e-05	4.10e-05	3.56e-18	4.10e-05	1.25e-04
100	–	–	–	–	–

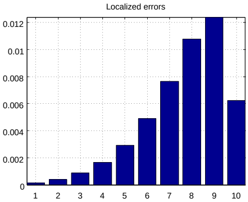
Convection-diffusion in 1D

technische universität dortmund

$$\mathrm{Pe}\,\frac{\mathrm{d}u}{\mathrm{d}t} - \frac{\mathrm{d}^2 u}{\mathrm{d}x^2} = 0, \quad u(0) = 0, \quad u(1) = 1, \quad j(u) = \int_0^1 u\,\mathrm{d}x$$







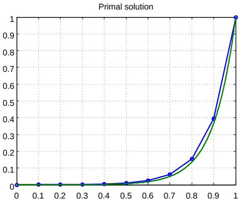
Discretization: upwind difference scheme, $h = 1/10$

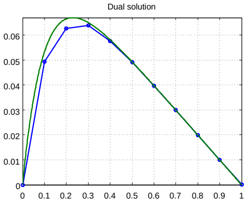
Pe	$\lvert j(u-\bar{u})\rvert$	Φ	Ψ	η	I_{rel}
1	4.52e-03	7.38e-04	3.58e-03	4.32e-03	4.79e-04
10	4.91e-02	3.06e-04	4.76e-02	4.79e-02	1.21e-02
100	5.00e-02	1.59e-09	5.00e-02	5.00e-02	1.21e-08

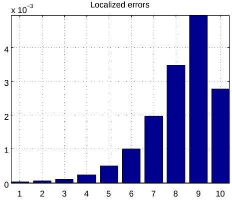
Convection-diffusion in 1D

technische universität
dortmund

$$\mathrm{Pe}\,\frac{\mathrm{d}u}{\mathrm{d}t} - \frac{\mathrm{d}^2 u}{\mathrm{d}x^2} = 0, \quad u(0) = 0, \quad u(1) = 1, \quad j(u) = \int_0^1 u\,\mathrm{d}x$$







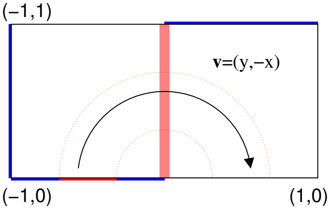
Discretization: TVD scheme, MC limiter, $h = 1/10$

Pe	$\lvert j(u-\bar{u})\rvert$	Φ	Ψ	η	I_{rel}
1	1.03e-03	7.74e-04	2.60e-04	1.03e-03	1.34e-05
10	1.51e-02	9.12e-05	1.50e-02	1.51e-02	3.81e-05
100	4.51e-02	4.23e-09	4.51e-02	4.51e-02	1.97e-07

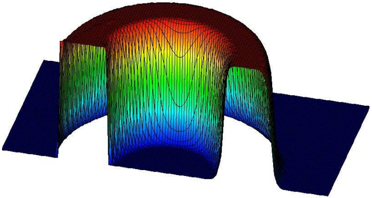
Circular convection $\nabla \cdot (\mathbf{v}u) = 0$ in $\Omega = (-1,1) \times (0,1)$

$$u(x,y) = \begin{cases} 1, & 0.35 \le r \le 0.65 \\ 0, & \text{otherwise} \end{cases} \qquad r(x,y) = \sqrt{x^2 + y^2}$$

Target functional $j(u) = \int_\omega u \, \mathrm{d}\mathbf{x}, \qquad \omega = (-0.1, 0.1) \times (0,1)$



domain and velocity



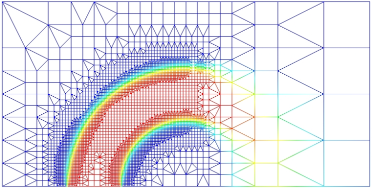
FEM-TVD, $h = 1/64$

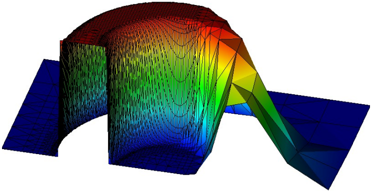
Mesh adaptation

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

$$u(x,y) = \begin{cases} 1, & 0.35 \leq r \leq 0.65 \\ 0, & \text{otherwise} \end{cases} \qquad r(x,y) = \sqrt{x^2 + y^2}$$

Target functional $j(u) = \int_\omega u \,\mathrm{d}\mathbf{x}, \qquad \omega = (-0.1, 0.1) \times (0,1)$





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