Algebraic flux correction and its application to convection-dominated flow

Matthias Möller

matthias.moeller@math.uni-dortmund.de

Institute of Applied Mathematics (LS III)
University of Dortmund, Germany

8th Hirschegg Workshop on Conservation Laws
9-15 September 2007, Hirschegg

Overview

1. Algebraic flux correction
   - Motivation
   - Design principles
   - Flux limiting
2. Nonlinear solution algorithm
   - Choice of preconditioners
   - Construction of Jacobians
   - Numerical examples
3. Recent work
   - Application to compressible flows
   - Grid adaptivity
4. Conclusions

Motivation

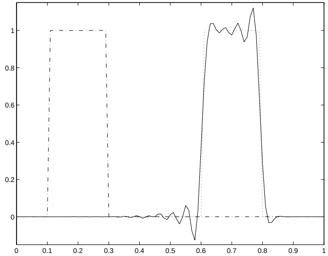
Scalar transport equation

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

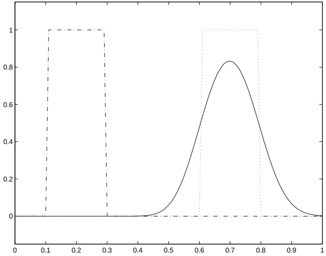
Standard Galerkin FEM

$$M_c \frac{\mathrm{d}u}{\mathrm{d}t} + Ku = 0$$

High-order methods: *wiggles*



Low-order methods: *smearing*



Remedy: **nonlinear** combination of high- and low-order discretizations.

Algebraic flux correction, *Kuz05a*

High-order scheme $M_c \frac{\mathrm{d}u}{\mathrm{d}t} + Ku = 0$ $K = \{k_{ij}\}, \exists k_{ij} > 0, j \neq i$

Apply artificial diffusion $D = \{d_{ij}\}$, $d_{ij} = -\max\{k_{ij}, 0, k_{ji}\} = d_{ji}$

Low-order scheme $M_l \frac{\mathrm{d}u}{\mathrm{d}t} + Lu = 0$ $L = K + D$, $l_{ij} \leq 0, \forall j \neq i$

Decompose the residual difference into internodal fluxes

$$r_i^H - r_i^L = \sum_{j \neq i} f_{ij}, \quad f_{ij} = \left[-m_{ij}\frac{\mathrm{d}}{\mathrm{d}t} + d_{ij}\right](u_j - u_i) = -f_{ji}$$

Nonlinear high-resolution scheme containing limited antidiffusion

$$M_l \frac{\mathrm{d}u}{\mathrm{d}t} + Lu = f, \quad f_i = \sum_{j \neq i} \alpha_{ij} f_{ij}, \quad 0 \leq \alpha_{ij} \leq 1$$

Algebraic flux correction, *Kuz05a*

High-order scheme $M_c \frac{\mathrm{d}u}{\mathrm{d}t} + Ku = 0$ $K = \{k_{ij}\}, \exists k_{ij} > 0, j \neq i$

Apply artificial diffusion $D = \{d_{ij}\}$, $d_{ij} = -\max\{k_{ij}, 0, k_{ji}\} = d_{ji}$

Low-order scheme $M_l \frac{\mathrm{d}u}{\mathrm{d}t} + Lu = 0$ $L = K + D$, $l_{ij} \leq 0, \forall j \neq i$

Decompose the residual difference into internodal fluxes

$$r_i^H - r_i^L = \sum_{j \neq i} f_{ij}, \quad f_{ij} = \left[-m_{ij}\frac{\mathrm{d}}{\mathrm{d}t} + d_{ij}\right](u_j - u_i) = -f_{ji}$$

Nonlinear high-resolution scheme containing limited antidiffusion

$$M_l \frac{\mathrm{d}u}{\mathrm{d}t} + Lu = f, \quad f_i = \sum_{j \neq i} \alpha_{ij} f_{ij}, \quad 0 \leq \alpha_{ij} \leq 1$$

Algebraic flux correction, *Kuz05a*

High-order scheme $M_c \frac{\mathrm{d}u}{\mathrm{d}t} + Ku = 0$ $\quad K = \{k_{ij}\}, \exists k_{ij} > 0, j \neq i$

Apply artificial diffusion $D = \{d_{ij}\}$, $\quad d_{ij} = -\max\{k_{ij}, 0, k_{ji}\} = d_{ji}$

Low-order scheme $M_l \frac{\mathrm{d}u}{\mathrm{d}t} + Lu = 0$ $\quad L = K + D,\ l_{ij} \leq 0, \forall j \neq i$

Decompose the residual difference into internodal fluxes

$$r_i^H - r_i^L = \sum_{j \neq i} f_{ij}, \quad f_{ij} = \left[-m_{ij} \frac{\mathrm{d}}{\mathrm{d}t} + d_{ij}\right](u_j - u_i) = -f_{ji}$$

Nonlinear high-resolution scheme containing limited antidiffusion

$$M_l \frac{\mathrm{d}u}{\mathrm{d}t} + Lu = f, \quad f_i = \sum_{j \neq i} \alpha_{ij} f_{ij}, \quad 0 \leq \alpha_{ij} \leq 1$$

Algebraic flux correction, *Kuz05a*

High-order scheme $M_c \frac{\mathrm{d}u}{\mathrm{d}t} + Ku = 0$ $K = \{k_{ij}\}, \exists k_{ij} > 0, j \neq i$

Apply artificial diffusion $D = \{d_{ij}\}$, $d_{ij} = -\max\{k_{ij}, 0, k_{ji}\} = d_{ji}$

Low-order scheme $M_l \frac{\mathrm{d}u}{\mathrm{d}t} + Lu = 0$ $L = K + D,\ l_{ij} \leq 0, \forall j \neq i$

Decompose the residual difference into internodal fluxes

$$r_i^H - r_i^L = \sum_{j \neq i} f_{ij}, \quad f_{ij} = \left[-m_{ij} \frac{\mathrm{d}}{\mathrm{d}t} + d_{ij}\right](u_j - u_i) = -f_{ji}$$

Nonlinear high-resolution scheme containing limited antidiffusion

$$M_l \frac{\mathrm{d}u}{\mathrm{d}t} + Lu = f, \quad f_i = \sum_{j \neq i} \alpha_{ij} f_{ij}, \quad 0 \leq \alpha_{ij} \leq 1$$

Flux limiting

Nonlinear high-resolution scheme *(lumped-mass version)*

$$M_l \frac{\mathrm{d}u}{\mathrm{d}t} + Lu = f, \quad f_i = \sum_{j \neq i} \alpha_{ij} f_{ij}, \quad f_{ij} = d_{ij}(u_j - u_i) = -f_{ji}$$

$\alpha_{ij} \equiv 0 \;\Rightarrow\; f = 0 \quad \Rightarrow \quad M_l \frac{\mathrm{d}u}{\mathrm{d}t} + Lu = 0$ low-order scheme

$\alpha_{ij} \equiv 1 \;\Rightarrow\; f = Du \;\Rightarrow \quad M_l \frac{\mathrm{d}u}{\mathrm{d}t} + Ku = 0$ high-order scheme

Goal: Determine α_{ij} as close to 1 as possible such that

$$(Lu - f)_i = \sum_{j \neq i} q_{ij}(u_j - u_i), \quad q_{ij} \leq 0 \;\forall j \neq i$$

Multidimensional flux correction

1. Compute the sums of positive/negative antidiffusive fluxes

$$P_i^+ = \sum_{j \neq i} \max\{0, f_{ij}\}, \qquad P_i^- = \sum_{j \neq i} \min\{0, f_{ij}\}$$

2. Define the corresponding upper/lower bounds

$$Q_i^+ = \sum_{j \neq i} q_{ij} \max\{0, u_j - u_i\}, \qquad Q_i^- = \sum_{j \neq i} q_{ij} \min\{0, u_j - u_i\}$$

3. Evaluate the nodal correction factors for positive/negative fluxes

$$R_i^+ = \min\{1, Q_i^+/P_i^+\}, \qquad R_i^- = \min\{1, Q_i^-/P_i^-\}$$

4. Apply correction factors to the raw antidiffusive fluxes

$$f_i = \sum_{j \neq i} \alpha_{ij} f_{ij} \qquad a_{ij} = \begin{cases} \min\{R_i^+, R_j^-\}, & \text{if } f_{ij} \geq 0 \\ \min\{R_i^-, R_j^+\}, & \text{otherwise} \end{cases}$$

Multidimensional flux correction

1. Compute the sums of positive/negative antidiffusive fluxes

$$P_i^+ = \sum_{j\neq i} \max\{0, f_{ij}\}, \qquad P_i^- = \sum_{j\neq i} \min\{0, f_{ij}\}$$

2. Define the corresponding upper/lower bounds

$$Q_i^+ = \sum_{j\neq i} q_{ij} \max\{0, u_j - u_i\}, \qquad Q_i^- = \sum_{j\neq i} q_{ij} \min\{0, u_j - u_i\}$$

3. Evaluate the nodal correction factors for positive/negative fluxes

$$R_i^+ = \min\{1, Q_i^+/P_i^+\}, \qquad R_i^- = \min\{1, Q_i^-/P_i^-\}$$

4. Apply correction factors to the raw antidiffusive fluxes

$$f_i = \sum_{j\neq i} \alpha_{ij} f_{ij} \qquad a_{ij} = \begin{cases} \min\{R_i^+, R_j^-\}, & \text{if } f_{ij} \geq 0 \\ \min\{R_i^-, R_j^+\}, & \text{otherwise} \end{cases}$$

Multidimensional flux correction

1. Compute the sums of positive/negative antidiffusive fluxes

$$P_i^+ = \sum_{j \neq i} \max\{0, f_{ij}\}, \qquad P_i^- = \sum_{j \neq i} \min\{0, f_{ij}\}$$

2. Define the corresponding upper/lower bounds

$$Q_i^+ = \sum_{j \neq i} q_{ij} \max\{0, u_j - u_i\}, \qquad Q_i^- = \sum_{j \neq i} q_{ij} \min\{0, u_j - u_i\}$$

3. Evaluate the nodal correction factors for positive/negative fluxes

$$R_i^+ = \min\{1, Q_i^+/P_i^+\}, \qquad R_i^- = \min\{1, Q_i^-/P_i^-\}$$

4. Apply correction factors to the raw antidiffusive fluxes

$$f_i = \sum_{j \neq i} \alpha_{ij} f_{ij} \qquad a_{ij} = \begin{cases} \min\{R_i^+, R_j^-\}, & \text{if } f_{ij} \geq 0 \\ \min\{R_i^-, R_j^+\}, & \text{otherwise} \end{cases}$$

Multidimensional flux correction

1. Compute the sums of positive/negative antidiffusive fluxes

$$P_i^+ = \sum_{j\neq i} \max\{0, f_{ij}\}, \qquad P_i^- = \sum_{j\neq i} \min\{0, f_{ij}\}$$

2. Define the corresponding upper/lower bounds

$$Q_i^+ = \sum_{j\neq i} q_{ij} \max\{0, u_j - u_i\}, \qquad Q_i^- = \sum_{j\neq i} q_{ij} \min\{0, u_j - u_i\}$$

3. Evaluate the nodal correction factors for positive/negative fluxes

$$R_i^+ = \min\{1, Q_i^+/P_i^+\}, \qquad R_i^- = \min\{1, Q_i^-/P_i^-\}$$

4. Apply correction factors to the raw antidiffusive fluxes

$$f_i = \sum_{j\neq i} \alpha_{ij} f_{ij} \qquad a_{ij} = \begin{cases} \min\{R_i^+, R_j^-\}, & \text{if } f_{ij} \geq 0 \\ \min\{R_i^-, R_j^+\}, & \text{otherwise} \end{cases}$$

Nonlinear solution algorithm

Nonlinear algebraic system for $0 < \theta \leq 1$ $\quad N(u) := Lu - f$

$$M_l \frac{u^{n+1} - u^n}{\Delta t} + \theta N(u^{n+1}) + (1-\theta) N(u^n) = 0$$

Fixed-point iteration $\quad u^{(0)} := u^n$

$$u^{(m+1)} = u^{(m)} - C^{-1} F(u^{(m)}), \qquad m = 0, 1, \ldots$$

until $\quad \|F(u^{(m+1)})\| \leq \mathrm{tol} \quad$ or $\quad \|F(u^{(m+1)})\| \leq \mathrm{tol}\|F(u^n)\|.$

Practical implementation

$$\begin{aligned} C\Delta u^{(m+1)} &= F(u^{(m)}) && m = 0, 1, \ldots \\ u^{(m+1)} &= u^{(m)} - \Delta u^{(m+1)} && u^{(0)} = u^n \end{aligned}$$

Nonlinear solution algorithm

Nonlinear algebraic system for $0 < \theta \leq 1$ $\quad N(u) := Lu - f$

$$F(u^{n+1}) := M_l \frac{u^{n+1} - u^n}{\Delta t} + \theta N(u^{n+1}) + (1-\theta) N(u^n) = 0$$

Fixed-point iteration $\quad u^{(0)} := u^n$

$$u^{(m+1)} = u^{(m)} - C^{-1} F(u^{(m)}), \qquad m = 0, 1, \ldots$$

until $\quad \|F(u^{(m+1)})\| \leq \mathrm{tol} \quad$ or $\quad \|F(u^{(m+1)})\| \leq \mathrm{tol} \|F(u^n)\|.$

Practical implementation

$$\begin{aligned} C \Delta u^{(m+1)} &= F(u^{(m)}) & m &= 0, 1, \ldots \\ u^{(m+1)} &= u^{(m)} - \Delta u^{(m+1)} & u^{(0)} &= u^n \end{aligned}$$

Choice of preconditioner

Linearized system $C\Delta u^{(m+1)} = F(u^{(m)})$

- Diagonal mass matrix $C = M_l/\Delta t$
  - cheap, but only usable for very small time steps

- Low-order operator $C = M_l/\Delta t + \theta\, L$
  - fixed-point defect correction scheme may converge slowly
  - no update needed if the velocity field remains unchanged

- Jacobian operator $C = M_l/\Delta t + \theta \frac{\partial N(u)}{\partial u}$
  - nonlinear operator $N(u)$ is constructed discretely
  - flux limiter may not be globally differentiable

Idea: approximate by divided differences and see what happens

Jacobian matrix, *Moe07a*

Nodal approximation of Jacobian $J = \{j_{ik}\}$

$$\mathcal{D}_k[N_i] := \frac{N_i(u + he_k) - N_i(u - he_k)}{2h} \quad \Rightarrow \quad j_{ik} = \mathcal{D}_k[N_i] + \mathcal{O}(h^2)$$

$$\begin{aligned}
\mathcal{D}_k[N_i] &= \mathcal{D}_k[(Lu - f)_i] = \mathcal{D}_k[(Ku + Du - f)_i] \\
&= \mathcal{D}_k[\sum_j k_{ij} u_j + \sum_{j \neq i} (1 - \alpha_{ij}) f_{ij}] \\
&= \widehat{k_{ik}} + \sum_j \mathcal{D}_k[k_{ij}] u_j + \sum_{j \neq i} \mathcal{D}_k[(1 - \alpha_{ij}) f_{ij}]
\end{aligned}$$

Averaged coefficient $\quad \widehat{k_{ik}} := \dfrac{k_{ik}(u + he_k) + k_{ik}(u - he_k)}{2}$

Jacobian matrix, *Moe07a*

Nodal approximation of Jacobian $J = \{j_{ik}\}$

$$\mathcal{D}_k[N_i] := \frac{N_i(u + he_k) - N_i(u - he_k)}{2h} \quad \Rightarrow \quad j_{ik} = \mathcal{D}_k[N_i] + \mathcal{O}(h^2)$$

$$\begin{aligned}
\mathcal{D}_k[N_i] &= \mathcal{D}_k[(Lu - f)_i] = \mathcal{D}_k[(Ku + Du - f)_i] \\
&= \mathcal{D}_k[\sum_j k_{ij} u_j + \sum_{j \neq i} (1 - \alpha_{ij}) f_{ij}] \\
&= \widehat{k_{ik}} + \sum_j \mathcal{D}_k[k_{ij}] u_j + \sum_{j \neq i} \mathcal{D}_k[(1 - \alpha_{ij}) f_{ij}]
\end{aligned}$$

Averaged coefficient $\widehat{k_{ik}} := \frac{k_{ik}(u + he_k) + k_{ik}(u - he_k)}{2}$

Jacobian matrix, *Moe07a*

Nodal approximation of Jacobian $J = \{j_{ik}\}$

$$\mathcal{D}_k[N_i] := \frac{N_i(u + he_k) - N_i(u - he_k)}{2h} \quad \Rightarrow \quad j_{ik} = \mathcal{D}_k[N_i] + \mathcal{O}(h^2)$$

$$\begin{aligned}
\mathcal{D}_k[N_i] &= \mathcal{D}_k[(Lu - f)_i] = \mathcal{D}_k[(Ku + Du - f)_i] \\
&= \mathcal{D}_k[\sum_j k_{ij} u_j + \sum_{j \neq i} (1 - \alpha_{ij}) f_{ij}] \\
&= \widehat{k_{ik}} + \sum_j \mathcal{D}_k[k_{ij}] u_j + \sum_{j \neq i} \mathcal{D}_k[(1 - \alpha_{ij}) f_{ij}]
\end{aligned}$$

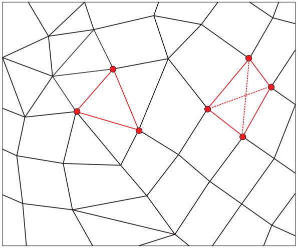
Averaged coefficient $\quad \widehat{k_{ik}} := \dfrac{k_{ik}(u + he_k) + k_{ik}(u - he_k)}{2}$

Sparsity pattern, *Moe07a*

Jacobian coefficients $j_{ik} = \widehat{k_{ik}} + \sum_j \mathcal{D}_k[k_{ij}]u_j + \sum_{j\neq i} \mathcal{D}_k[(1-\alpha_{ij})f_{ij}]$

There exists an edge ij iff the FE basis functions have overlapping supports

$$G := \langle K \rangle \quad g_{ii} = 1,\ g_{ij} = 1 \Leftrightarrow \exists ij$$



Structure of the Jacobian $\quad Z := \langle J \rangle$

$$z_{kk} \neq 0, \qquad z_{ik} \neq 0 \Leftrightarrow \exists ik$$

$$z_{jk} \neq 0 \Leftrightarrow \exists ik \wedge \exists ij$$

Remarks

- Same structure is used for edge-oriented stabilization techniques
- Sparsity pattern can be generated by multiplication: $Z = G^2$

Sparsity pattern, *Moe07a*

Jacobian coefficients $j_{ik} = \widehat{k_{ik}} + \sum_j \mathcal{D}_k[k_{ij}]u_j + \sum_{j\neq i} \mathcal{D}_k[(1-\alpha_{ij})f_{ij}]$

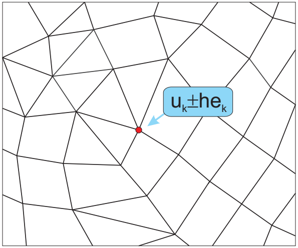
There exists an edge ij iff the FE basis functions have overlapping supports

$$G := \langle K \rangle \quad g_{ii} = 1,\ g_{ij} = 1 \Leftrightarrow \exists ij$$

Structure of the Jacobian $\quad Z := \langle J \rangle$

$$z_{kk} \neq 0, \quad z_{ik} \neq 0 \Leftrightarrow \exists ik$$

$$z_{jk} \neq 0 \Leftrightarrow \exists ik \wedge \exists ij$$



Remarks

- Same structure is used for edge-oriented stabilization techniques
- Sparsity pattern can be generated by multiplication: $Z = G^2$

Sparsity pattern, *Moe07a*

Jacobian coefficients $j_{ik} = \widehat{k_{ik}} + \sum_j \mathcal{D}_k[k_{ij}]u_j + \sum_{j \neq i} \mathcal{D}_k[(1-\alpha_{ij})f_{ij}]$

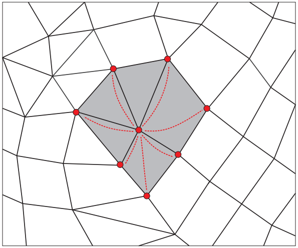
There exists an edge ij iff the FE basis functions have overlapping supports

$$G := \langle K \rangle \quad g_{ii} = 1,\ g_{ij} = 1 \Leftrightarrow \exists ij$$

Structure of the Jacobian $\quad Z := \langle J \rangle$

$$z_{kk} \neq 0, \quad z_{ik} \neq 0 \Leftrightarrow \exists ik$$

$$z_{jk} \neq 0 \Leftrightarrow \exists ik \wedge \exists ij$$



Remarks

- Same structure is used for edge-oriented stabilization techniques
- Sparsity pattern can be generated by multiplication: $Z = G^2$

Sparsity pattern, *Moe07a*

Jacobian coefficients $j_{ik} = \widehat{k_{ik}} + \sum_j \mathcal{D}_k[k_{ij}]u_j + \sum_{j \neq i} \mathcal{D}_k[(1 - \alpha_{ij})f_{ij}]$

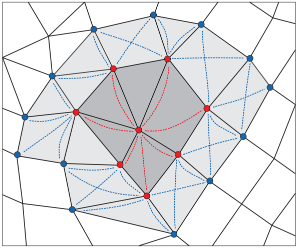
There exists an edge ij iff the FE basis functions have overlapping supports

$$G := \langle K \rangle \quad g_{ii} = 1,\ g_{ij} = 1 \Leftrightarrow \exists ij$$

Structure of the Jacobian $\quad Z := \langle J \rangle$

$$z_{kk} \neq 0, \quad z_{ik} \neq 0 \Leftrightarrow \exists ik$$

$$z_{jk} \neq 0 \Leftrightarrow \exists ik \wedge \exists ij$$



Remarks

- Same structure is used for edge-oriented stabilization techniques
- Sparsity pattern can be generated by multiplication: $Z = G^2$

Sparsity pattern, *Moe07a*

Jacobian coefficients $j_{ik} = \widehat{k_{ik}} + \sum_j \mathcal{D}_k[k_{ij}]u_j + \sum_{j\neq i} \mathcal{D}_k[(1-\alpha_{ij})f_{ij}]$

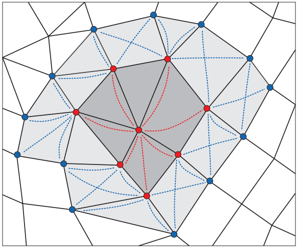
There exists an edge ij iff the FE basis functions have overlapping supports

$$G := \langle K \rangle \quad g_{ii} = 1,\ g_{ij} = 1 \Leftrightarrow \exists ij$$

Structure of the Jacobian $\quad Z := \langle J \rangle$

$$z_{kk} \neq 0, \quad z_{ik} \neq 0 \Leftrightarrow \exists ik$$

$$z_{jk} \neq 0 \Leftrightarrow \exists ik \wedge \exists ij$$



Remarks

- Same structure is used for edge-oriented stabilization techniques
- Sparsity pattern can be generated by multiplication: $Z = G^2$

Example: *Stationary convection-diffusion*

Convection-diffusion equation

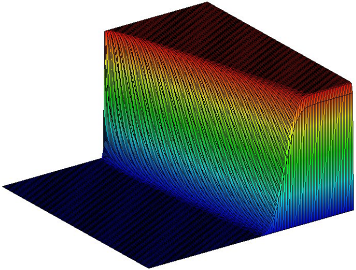
$$\mathbf{v} \cdot \nabla u - d\Delta u = 0$$

$$\mathbf{v} = (\cos 10^\circ, \sin 10^\circ)$$

Initial guess in $\Omega = (0,1)^2$

$$u_u(x,y) = \begin{cases} 1 - x & \text{if } y \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

FEM-TVD: $d = 10^{-3}$



128×128 Q_1-elements

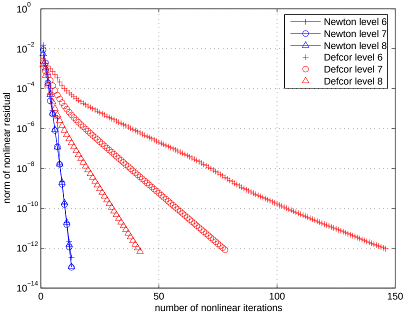
Boundary conditions

$$u(x,0) = 0, \qquad \frac{\partial u}{\partial y}(x,1) = 0, \qquad u(0,y) = \begin{cases} 1 & \text{if } y \geq 0.5, \\ 0 & \text{otherwise.} \end{cases}$$

$$u(1,y) = 0,$$

Example: $\partial_\tau u + \mathbf{v} \cdot \nabla u - d\Delta u = 0$

$$d = 10^{-3}, \quad \Delta\tau = 1.0, \quad \text{BiCGSTAB+ILU}, \quad \mathrm{rel}_{lin} \leq 10^{-3}$$



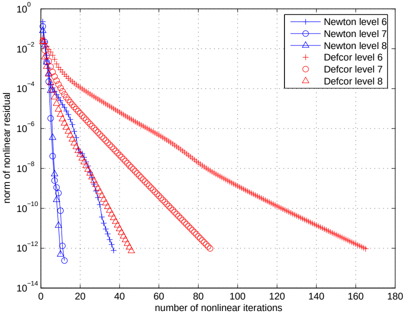
NVT	CPU	NN	NL
	Newton's method		
4225	1	13	71
16641	4	13	102
66049	23	13	189
	defect correction		
4225	3	146	1263
16641	9	78	1060
66049	35	43	1063

Newton's method *vs.* defect correction

- ▷ total CPU time reduces by factor 2-3
- ▷ reduction of nonlinear iterations by factor 4-16

Example: $\partial_\tau u + \mathbf{v} \cdot \nabla u - d\Delta u = 0$

$d = 10^{-3}, \quad \Delta\tau = 10.0, \quad$ BiCGSTAB+ILU, $\quad \mathrm{rel}_{lin} \leq 10^{-3}$



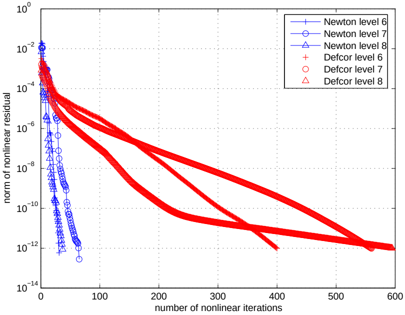
NVT	CPU	NN	NL
	Newton's method		
4225	2	37	263
16641	5	12	152
66049	19	10	182
	defect correction		
4225	4	166	1612
16641	10	86	1234
66049	38	46	1173

Newton's method *vs.* defect correction

- ▷ total CPU time reduces by factor 2-3
- ▷ reduction of nonlinear iterations by factor 4-16

Example: $\partial_\tau u + \mathbf{v} \cdot \nabla u - d\Delta u = 0$

$$d = 10^{-4}, \quad \Delta\tau = 1.0, \quad \text{BiCGSTAB+ILU}, \quad \text{rel}_{lin} \leq 10^{-3}$$



NVT	CPU	NN	NL
	Newton's method		
4225	2	32	541
16641	54	65	3514
66049	95	37	1219
	defect correction		
4225	9	400	4279
16641	63	560	8013
66049	263	595	5861

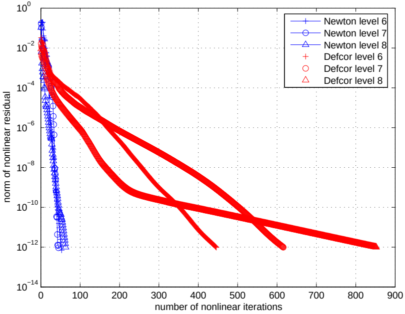
Newton's method *vs.* defect correction

▷ total CPU time reduces by factor 2-3

▷ reduction of nonlinear iterations by factor 4-16

Example: $\partial_\tau u + \mathbf{v} \cdot \nabla u - d\Delta u = 0$

$$d = 10^{-4}, \quad \Delta\tau = 10.0, \quad \text{BiCGSTAB+ILU}, \quad \mathrm{rel}_{lin} \leq 10^{-3}$$



NVT	CPU	NN	NL
	Newton's method		
4225	5	53	1144
16641	37	45	4287
66049	180	63	4147
	defect correction		
4225	10	447	5306
16641	75	616	9875
66049	356	853	7519

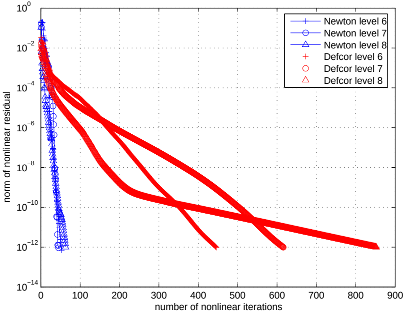
Newton's method *vs.* defect correction

▷ total CPU time reduces by factor 2-3

▷ reduction of nonlinear iterations by factor 4-16

Example: $\partial_\tau u + \mathbf{v} \cdot \nabla u - d\Delta u = 0$

$$d = 10^{-4}, \quad \Delta\tau = 10.0, \quad \text{BiCGSTAB+ILU}, \quad \mathrm{rel}_{lin} \leq 10^{-3}$$



NVT	CPU	NN	NL
	Newton's method		
4225	5	53	1144
16641	37	45	4287
66049	180	63	4147
	defect correction		
4225	10	447	5306
16641	75	616	9875
66049	356	853	7519

Newton's method *vs.* defect correction

- ▷ total CPU time reduces by factor 2-3
- ▷ reduction of nonlinear iterations by factor 4-16

Example: *Burgers' equation in space-time*

Inviscid Burgers' equation

$$\partial_x \left(\frac{u^2}{2} \right) + \partial_t u = 0$$

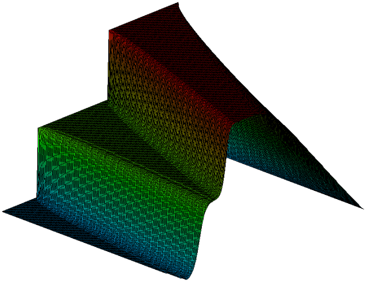
Space-time domain

$$\Omega = (0,1) \times (0,0.5)$$

Boundary conditions

$$u(x,t) = \begin{cases} 1 & \text{if } 0 \leq x < 0.4 \wedge t = 0 \\ 0.5 & \text{if } 0.4 \leq x \leq 8 \wedge t = 0 \\ 0 & \text{if } 0.8 < x \leq 1 \wedge t = 0 \\ & \text{or } x = 0 \wedge 0 \leq t \leq 0.5 \end{cases}$$

Discrete upwind: 32,768 P_1-elements



$$\|u - u_h\|_1 = 1.6922e - 2$$
$$\|u - u_h\|_2 = 4.0169e - 2$$

Example: *Burgers' equation in space-time*

Inviscid Burgers' equation

$$\partial_x \left(\frac{u^2}{2} \right) + \partial_t u = 0$$

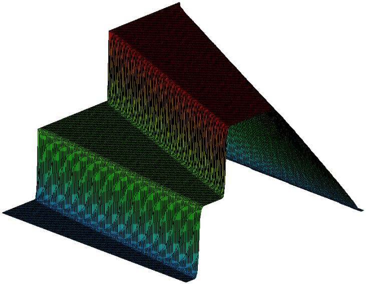
Space-time domain

$$\Omega = (0,1) \times (0,0.5)$$

Boundary conditions

$$u(x,t) = \begin{cases} 1 & \text{if } 0 \leq x < 0.4 \wedge t = 0 \\ 0.5 & \text{if } 0.4 \leq x \leq 8 \wedge t = 0 \\ 0 & \text{if } 0.8 < x \leq 1 \wedge t = 0 \\ & \text{or } x = 0 \wedge 0 \leq t \leq 0.5 \end{cases}$$

FEM-TVD: 32,768 P_1-elements



$$\|u - u_h\|_1 = 4.9211e - 3$$
$$\|u - u_h\|_2 = 1.9082e - 2$$

Example: $\partial_\tau u + \partial_x(\frac{1}{2}u^2) + \partial_t u = 0$

Discrete upwind: 1 nit./$\Delta\tau$, BiCGSTAB+ILU, $\mathrm{rel}_{lin} \leq 10^{-3}$

$\Delta\tau = 0.1$

	Newton			defect correction			error	
NVT	CPU	NN	NL	CPU	NN	NL	$\|u-u_h\|_1$	$\|u-u_h\|_2$
4225	1	25	162	1	32	172	4.86e-2	7.84e-2
16641	2	24	287	3	35	352	2.93e-2	5.63e-2
66049	15	24	518	19	41	716	1.69e-2	4.01e-2

$\Delta\tau = 1.0$

	Newton			defect correction			error	
NVT	CPU	NN	NL	CPU	NN	NL	$\|u-u_h\|_1$	$\|u-u_h\|_2$
4225	1	12	136	1	21	321	4.86e-2	7.84e-2
16641	2	11	242	4	26	661	2.93e-2	5.63e-2
66049	12	12	412	30	35	1348	1.69e-2	4.01e-2

▷ Discrete Jacobian operator yields robust Newton iteration

Example: $\partial_\tau u + \partial_x(\frac{1}{2}u^2) + \partial_t u = 0$

FEM-TVD: up to 10 nit./$\Delta\tau^n$, $\Delta\tau^n \in [0.01, 1.0]$, $\mathrm{rel}_{nonl} \leq 10^{-3}$
BiCGSTAB+ILU, $\mathrm{rel}_{lin} \leq 10^{-3}$

Newton				defect correction			
CPU	NSTEP	NN	NL	CPU	NSTEP	NN	NL
10	38	119	5817	53	476	3741	42955
73	38	167	8597	224	254	2517	46755
1295	39	303	40162	3818	1065	6152	179584

error	4225	16641	66049
$\|u - u_h\|_1$	2.29e-2	1.01e-2	4.92e-3
$\|u - u_h\|_2$	5.34e-2	3.26e-2	1.90e-2

Newton's method *vs.* defect correction

- ▷ total CPU time reduces by factor 3
- ▷ reduction of outer iterations by factor 15-30

Example: *Swirling flow, Leveque*

Swirling flow in $\Omega = (0,1)^2$

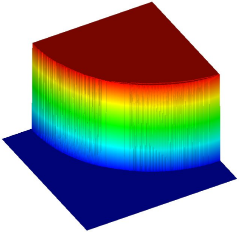
$$\frac{\partial u}{\partial t} + \nabla \cdot (\mathbf{v}u) = 0$$

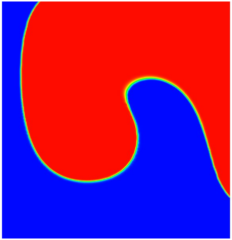
Time-dependent velocity $t \in (0, T)$

$$\begin{aligned} v_x &= \sin^2(\pi x)\sin(2\pi y)g(t) \\ v_y &= -\sin^2(\pi y)\sin(2\pi x)g(t) \\ g(t) &= \cos(\pi t/T), \qquad T = 1.5 \end{aligned}$$

Time step control: ▸ PID

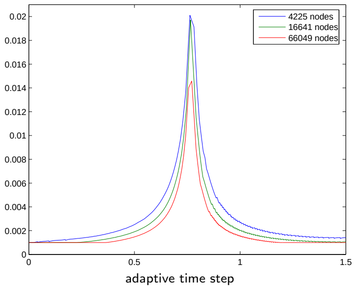
$$\Delta t \in [10^{-3}, 10^{-1}], \quad \frac{\|u^{n+1} - u^n\|}{\|u^{n+1}\|} \leq 0.5\%$$

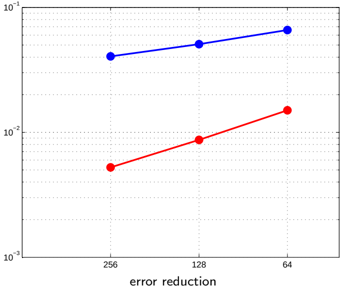




Example: *Swirling flow*, contd.

NVT	NSTEP	Newton CPU	Newton NN	Newton NL	defect correction CPU	defect correction NN	defect correction NL
4225	763 (70)	46	4474	4498	212	31898	31398
16641	977 (77)	251	5528	5556	1141	38295	38295
66049	1126 (66)	1220	5988	5990	5592	42410	42410

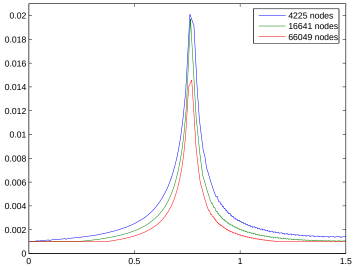


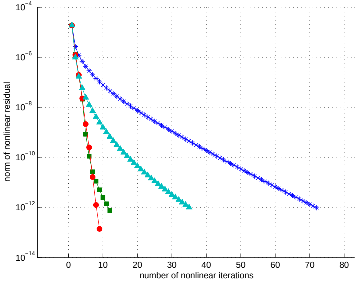


▷ CPU improved by factor 4.5; reduction of iterations by factor 7

Example: *Swirling flow*, contd.

		Newton			defect correction		
NVT	NSTEP	CPU	NN	NL	CPU	NN	NL
4225	763 (70)	46	4474	4498	212	31898	31398
16641	977 (77)	251	5528	5556	1141	38295	38295
66049	1126 (66)	1220	5988	5990	5592	42410	42410





▷ CPU improved by factor 4.5; reduction of iterations by factor 7

Outlook: *Compressible flows, Kuz05b*

Divergence Form

$$\frac{\partial U}{\partial t} + \sum_{d=1}^{3} \frac{\partial F^d}{\partial x_d} = 0$$

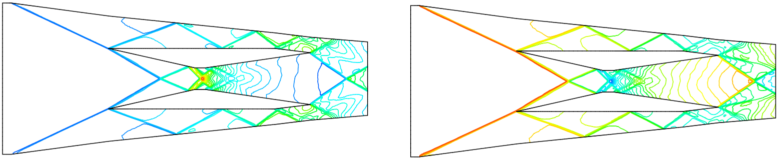
$$A^d = \frac{\partial F^d}{\partial U}$$

Quasi-linear form

$$\frac{\partial U}{\partial t} + \sum_{d=1}^{3} A^d \frac{\partial U}{\partial x_d} = 0$$

FEM discretization $\left[M_C \frac{\mathrm{dU}}{\mathrm{d}t}\right]_i = \sum_{j \neq i} \mathrm{A}_{ij}(\mathrm{U}_j - \mathrm{U}_i)$ $\qquad \mathrm{A}_{ij} = \mathrm{R}_{ij}\Lambda_{ij}\mathrm{R}_{ij}^{-1}$

Transformation to *local characteristic variables* makes it possible to perform upwinding and algebraic flux correction as in the scalar case.

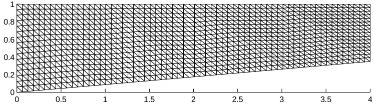


$M_\infty = 3$, $\alpha = 0^\circ$, density

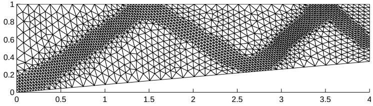
Mach number

Grid adaption, *Moe06*

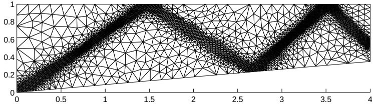
2048 triangles



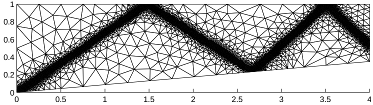
7194 triangles



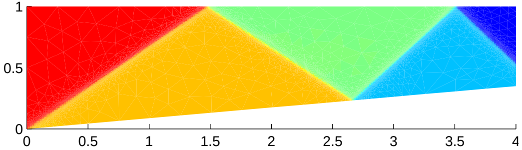
3503 triangles



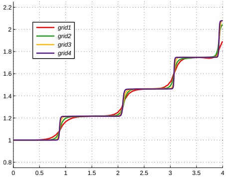
15664 triangles



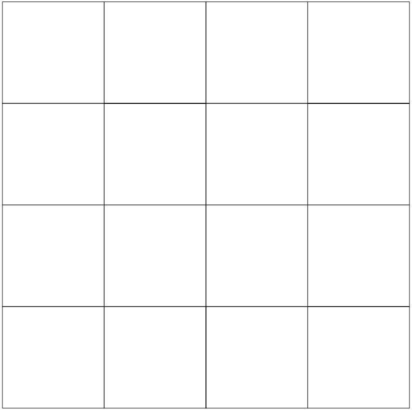
Density distribution on the final mesh



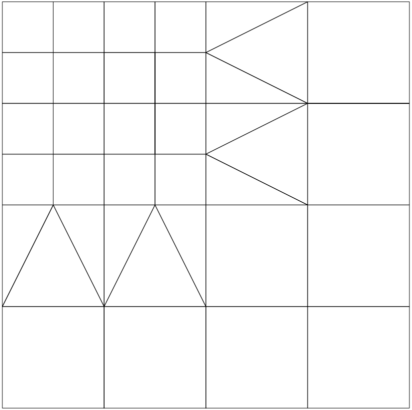
cutline at $y = 0.6$



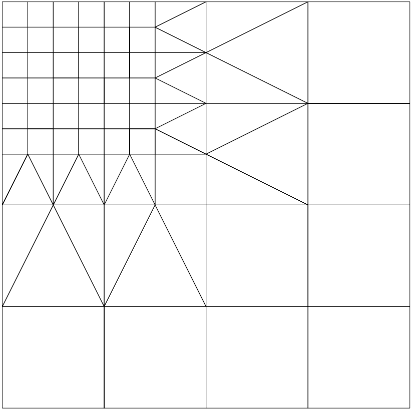
Time-dependent grid adaptivity



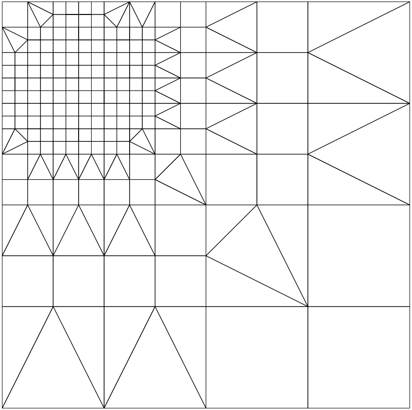
Time-dependent grid adaptivity



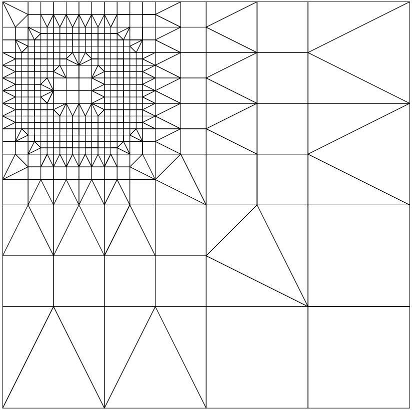
Time-dependent grid adaptivity



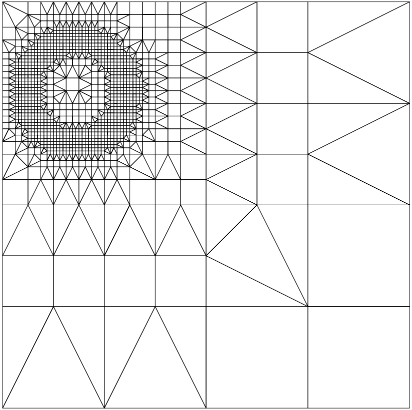
Time-dependent grid adaptivity



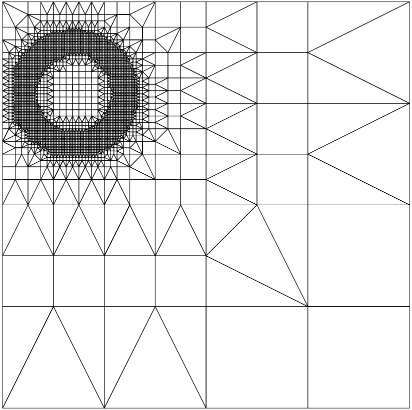
Time-dependent grid adaptivity



Time-dependent grid adaptivity



Time-dependent grid adaptivity



Conclusions and Outlook

- nonlinear high-resolution schemes can be constructed by adding artificial diffusion + limited antidiffusion
- implicit flux correction schemes can be combined with algebraic Newton methods to speed up convergence
- Jacobian matrix is approximated by divided differences and assembled efficiently edge-by-edge in a black-box fashion
- the sparsity pattern of the Jacobian needs to be extended which can be accomplished by standard matrix multiplication

<u>Further research:</u>

- compressible Euler/Navier-Stokes equations
- time-dependent grid adaptation + goal oriented error estimation

References

Kuz05a D. Kuzmin, M.M. Algebraic flux correction I. Scalar conservation laws. In: *Flux-Corrected Transport, Principles, Algorithms, and Applications*, D. Kuzmin, R. Löhner, S. Turek (eds.). Springer: Germany, 2005; 155-206

Kuz05b D. Kuzmin, M.M. Algebraic flux correction II. Compressible Euler equations. In: *Flux-Corrected Transport, Principles, Algorithms, and Applications*, D. Kuzmin, R. Löhner, S. Turek (eds.). Springer: Germany, 2005; 207-250.

Moe06 M.M., D. Kuzmin. Adaptive mesh refinement for high-resolution finite element schemes. *IJNMF* **52**, 2006; 545-569.

Moe07a M.M. Efficient solution techniques for implicit finite element schemes with flux limiters. *IJNMF* (in press).

Moe07b M.M, D. Kuzmin, D. Kourounis. Implicit FEM-FCT algorithms and discrete Newton methods for transient problems. Tech. Rep. **340**, 2007, University of Dortmund.

PID time step control

Normalize relative changes

$$e_n = \frac{\|u^n - u^{n-1}\|}{\mathrm{tol}\,\|u^n\|}$$

Reject time step if $e_n > 1$

$$\Delta t^n := \beta \Delta t^n, \quad 0 < \beta < 1$$

Compute new time step

$$\Delta t^{n+1} = \left(\frac{e_{n-1}}{e_n}\right)^{k_{\mathrm{P}}} \left(\frac{1}{e_n}\right)^{k_{\mathrm{I}}} \left(\frac{e_{n-1}^2}{e_n e_{n-2}}\right)^{k_{\mathrm{D}}} \Delta t^n$$

Parameters *(Valli, Carey, Coutinho)*

$$k_{\mathrm{P}} = 0.075, \quad k_{\mathrm{I}} = 0.175, \quad k_{\mathrm{D}} = 0.01$$

Back

Semi-implicit FCT limiter, *Part I*

- 1. Initialization $P_i^{\pm} \equiv Q_i^{\pm} \equiv R_i^{\pm} \equiv 0$
- 2.Compute positivity-preserving intermediate low-order solution

$$\tilde{u} = u^n + (1-\theta)\Delta t M_l^{-1} L u^n$$

- 3. Evaluate raw antidiffusive flux $f_{ij}^n = \Delta t d_{ij}^n (u_i^n - u_j^n)$ and update

$$P_i^{\pm} := P_i^{\pm} + \genfrac{}{}{0pt}{}{\max}{\min}\{0, f_{ij}^n\}, \quad P_j^{\pm} := P_j^{\pm} + \genfrac{}{}{0pt}{}{\max}{\min}\{0, -f_{ij}^n\}$$

- 4. Update admissible increments for both nodes

$$Q_i^{\pm} := \genfrac{}{}{0pt}{}{\max}{\min}\{Q_i^{\pm}, \tilde{u}_j - \tilde{u}_i\}, \quad Q_j^{\pm} := \genfrac{}{}{0pt}{}{\max}{\min}\{Q_j^{\pm}, \tilde{u}_i - \tilde{u}_j\}$$

- 5. Limite the raw antidiffusive fluxes f_{ij}^n

$$R_i^{\pm} := m_i Q_i^{\pm} / P_i^{\pm} \rightarrow \tilde{f}_{ij} := \begin{cases} \min\{R_i^+, R_j^-\} f_{ij}^n, & \text{if } F_{ij}^n > 0, \\ \min\{R_i^-, R_j^+\} f_{ij}^n, & \text{otherwise.} \end{cases}$$

Semi-implicit FCT limiter, *Part II*

- 1. Evaluate the target flux

$$\begin{aligned} f_{ij} &= [m_{ij} + \theta \Delta t d_{ij}^{(m)}](u_i^{(m)} - u_j^{(m)}) \\ &- [m_{ij} - (1-\theta)\Delta t d_{ij}^{n}](u_i^{n} - u_j^{n}) \end{aligned}$$

- 2.Constrain each flux by means of $\widetilde{f}_{ij}$

$$f_{ij}^* = \begin{cases} \min\{f_{ij}, \max\{0, \widetilde{f}_{ij}\}\}, & \text{if } f_{ij} > 0, \\ \max\{f_{ij}, \min\{0, \widetilde{f}_{ij}\}\}, & \text{otherwise.} \end{cases}$$

- 3. Insert the limited antidiffusive flux into the right-hand side

$$\widetilde{b}_i^{(m)} := \widetilde{b}_i^{(m)} + f_{ij}^*, \quad \widetilde{b}_j^{(m)} := \widetilde{b}_j^{(m)} - f_{ij}^*$$

Back