# A Robust Semi-Implicit Finite Element Scheme for Nonlinear Hyperbolic Systems

**M. Gurris, D. Kuzmin, S. Turek**

**Institute of Applied Mathematics**

**Dortmund University of Technology**

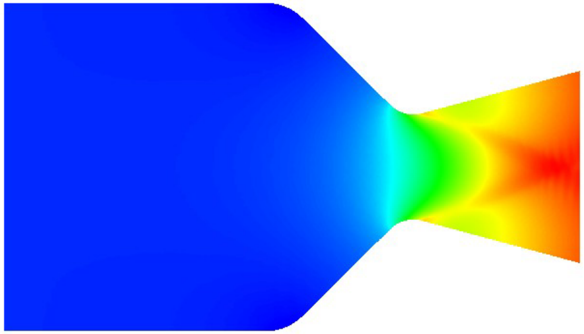
***marcel.gurris@math.tu-dortmund.de***

# Contents & Aims

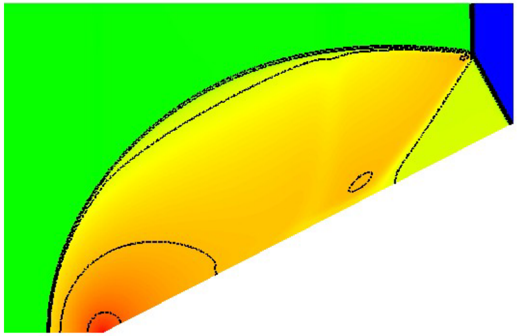
**Aims**

- Derivation of a robust semi-implicit FE scheme for stationary nonlinear hyperbolic systems
- Parameter-free Newton-like method
- Application to the Euler equations
- Particle-laden gas flows

**Euler Equations: JPL Nozzle Flow**



**Euler Equations: Compression Corner**



**Contents**

- Semi-implicit pseudo time stepping and relation to Newton‘s method
- FEM discretization of the Euler equations
- Weak boundary conditons
- Construction of the preconditioner
- Numerical results

# Definition of a Hyperbolic System

**Conservative form:**

$$\partial_t U + \nabla \cdot F(U) = \partial_t U + \partial_x F^{(x)} + \partial_y F^{(y)} = 0$$

**Hyperbolicity:**

The PDE-system is called hyperbolic, if the flux jacobians are diagonalizable with real eigenvalues

$$\frac{\partial F^{(x)}(U)}{\partial U} = R^{(x)-1} \Lambda^{(x)} R^{(x)} \qquad \frac{\partial F^{(y)}(U)}{\partial U} = R^{(y)-1} \Lambda^{(y)} R^{(y)}$$

**Euler equations:**

$$\partial_t \begin{bmatrix} \rho \\ \rho \vec{u} \\ \rho E \end{bmatrix} + \nabla \cdot \begin{bmatrix} \rho \vec{u} \\ \rho \vec{u} \otimes \vec{u} + \mathrm{I} P \\ \vec{u}(\rho E + P) \end{bmatrix} = 0$$

# Implicit Solver & Newton's Method

**Backward Euler scheme:**

$$M_L \frac{U^{n+1} - U^n}{\Delta t} = F^{n+1}$$

**Taylor Linearization:**

$$F^{n+1} = F^n + \left(\frac{\partial F}{\partial U}\right)^n \left(U^{n+1} - U^n\right) + O\left(\left\| U^{n+1} - U^n \right\|^2\right)$$

**Semi-implicit scheme:**

$$\left[\frac{M_L}{\Delta t} - \left(\frac{\partial F}{\partial U}\right)^n\right]\left(U^{n+1} - U^n\right) = F^n$$

**Newton's method:**

$$-\left(\frac{\partial F}{\partial U}\right)^n \left(U^{n+1} - U^n\right) = F^n$$

- Newton's method corresponds to an infinite CFL number
- Second-order convergence if F is differentiable

# Discretization of Spatial Derivatives

**High-order scheme:**

$$M_C \frac{dU}{dt} = KU$$

**Low-order scheme:**

$$M_L \frac{dU}{dt} = KU + DU = LU$$

**High-resolution scheme:**

$$M_L \frac{dU}{dt} = KU + DU + F^*U = K^*U$$

**Equivalent representation:**

$$M_L \frac{dU}{dt} = L^*U$$

# High-Order Scheme

**Galerkin FEM:**

$$\sum_j m_{ij} \frac{dU_j}{dt} = -\sum_j c_{ij} \cdot F_j, \quad \forall i$$

**Matrix coefficients:**

$$m_{ij} = \int_\Omega \varphi_i \varphi_j dx \qquad c_{ij} = \int_\Omega \varphi_i \nabla \varphi_j dx$$

**Lumped-mass Galerkin scheme:**

$$m_i = \sum_j m_{ij} \quad \Rightarrow \quad m_i \frac{dU_i}{dt} = -\sum_j c_{ij} \cdot F_j, \quad \forall i$$

$$F(U) = A(U)U \quad \Rightarrow \quad M_L \frac{dU}{dt} = KU$$

# Low-Order Scheme: Edge-Based Decomposition

**Characteristic LED criterion:**

$$m_i \frac{dU_i}{dt} = \sum_{j \neq i} L_{ij} \left(U_j - U_i\right)$$

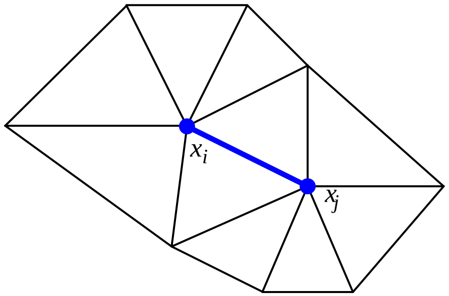
is local extremum diminishing (LED), if $L_{ij}$ are positive semi-definit for $j \neq i$

$$\begin{aligned}(KU)_i &= -\sum_j c_{ij} \cdot F_j \\ &= -\sum_{j \neq i} c_{ij} \cdot \left(F_j - F_i\right)\end{aligned}$$

$$\begin{aligned}(KU)_i &= -\sum_{j \neq i} c_{ij}^{(x)} \left(F_j^{(x)} - F_i^{(x)}\right) + c_{ij}^{(y)} \left(F_j^{(y)} - F_i^{(y)}\right) \\ &= -\sum_{j \neq i} \left(c_{ij}^{(x)} A_{ij}^{(x)} + c_{ij}^{(y)} A_{ij}^{(y)}\right)\left(U_j - U_i\right)\end{aligned}$$

**Roe linearization**

$$F_j^{(x,y)} - F_i^{(x,y)} = A_{ij}^{(x,y)} \left(U_j - U_i\right)$$



# Design of the Artificial Diffusion Operator

**Requirements:**

- Block symmetry, zero row and column sums
- Enough dissipation to enforce the LED criterion

**Local diffusion operator:** $D_{loc} = \begin{pmatrix} -D_{ij} & D_{ij} \\ D_{ij} & -D_{ij} \end{pmatrix}, \qquad D_{ij} = R_{ij} |\Lambda_{ij}| R_{ij}^{-1}$

**Diffusive fluxes:** $f_{ij}^{diff} = D_{ij}\left(U_j - U_i\right), \qquad f_{ji}^{diff} = -f_{ij}^{diff}$

**Low-order operator:** $\boxed{L_{ij} = K_{ij} + D_{ij} = R_{ij}\Lambda_{ij} R_{ij}^{-1} + R_{ij}|\Lambda_{ij}| R_{ij}^{-1}}$

# Flux Limiting & Characteristic Variables

**Characteristic variables:**

$$\frac{\partial U}{\partial t} + A^d \frac{\partial U}{\partial x^d} = 0 \quad \xrightarrow[A^d = R \Lambda R^{-1}]{W = R^{-1} U} \quad \frac{\partial W_k}{\partial t} + \lambda_k^d \frac{\partial W_k}{\partial x^d} = 0$$

Compute characteristic variables for each space dimension separately

- Decoupling of the linearized equations along the edge ij
- Characteristic flux limiter of TVD type (Kuzmin 2007)

$$\Delta W_{ij} = R_{ij}^{-1}\left(U_j - U_i\right) \quad \Longrightarrow \quad F_{ij}^* = R_{ij} \Delta \hat{W}_{ij} \qquad F_{ji}^* = -F_{ij}^*$$

**High-resolution scheme:** $\left(K^* U\right)_i = \sum_{j \neq i} \left(K_{ij} + D_{ij}\right)\left(U_j - U_i\right) - F_{ij}^*$

# The Interior Flux Jacobian

**Finite differences:** $$\frac{\partial F}{\partial U} \approx \frac{F(U+\vec{\varepsilon}) - F(U)}{|\vec{\varepsilon}|}$$ **or** $$\frac{\partial F}{\partial U} \approx \frac{F(U+\vec{\varepsilon}) - F(U-\vec{\varepsilon})}{2|\vec{\varepsilon}|}$$

- A convenient way to ´differentiate´ F
- Convergence behavior depends on the choice of $\vec{\varepsilon}$

**Analytic derivation:**

- F must be differentiable
- Complicated algebra and programming
- Deterioration of matrix properties

**Edge-based approximate Jacobian:**

- No free parameter
- No additional fill-in
- Increased robustness due to improved matrix properties
- Extension to a limited version is possible

$$F_{ij} \overset{\text{low}}{=} c_{ij} \cdot F_j + |c_{ij}||\mathbf{A}_{ij}|(U_j - U_i)$$

$$\frac{\partial F_{ij}}{\partial U_j}^{(x,y)} \approx c_{ij}^{(x,y)} \frac{\partial F_{ij}}{\partial U_j}^{(x,y)} + |c_{ij}||\mathbf{A}_{ij}^{(x,y)}|$$

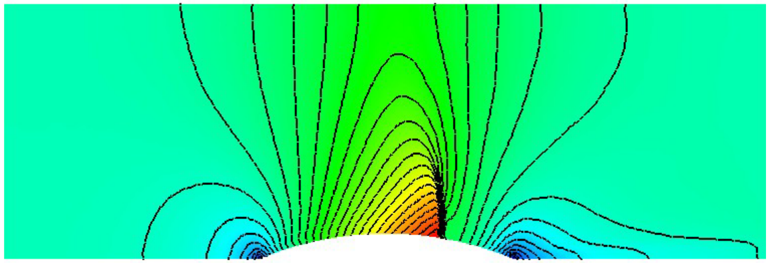
# Weak Boundary Conditions

**Integration by parts:**

$$\sum_j \int_\Omega \varphi_i \varphi_j dx \frac{dU_j}{dt} = \sum_j \int_\Omega \varphi_j \nabla \varphi_i dx \cdot F_j - \sum_j \int_{\partial\Omega} \varphi_i \varphi_j n ds \cdot F_j, \quad \forall i$$

- Improved convergence rates and robustness in steady computations
- Exclusively the boundary integrals are affected by the boundary conditions
- Fully implicit boundary treatment

**GAMM Channel**



$$\sum_j \int_{\partial\Omega} \varphi_i \varphi_j n ds \cdot F_j$$

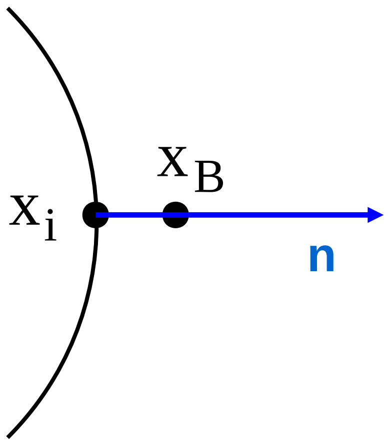
→ Evaluate the boundary flux using the solution of a Riemann problem →

$$\sum_j \int_{\partial\Omega} \varphi_i \varphi_j n ds \cdot \tilde{F}_j$$

# The Boundary Flux

**Evaluation of the boundary integral:**

- Ghost nodes
- Edgewise evaluation




**Roe flux:**

$$F_{iB} = \frac{F_i + F_B}{2} - \frac{1}{2}|A_{iB}|(U_B - U_i)$$

- A popular approximate Riemann solver
- $U_B$ is defined in terms of the Riemann invariants
- Boundary values are prescribed for the incoming Riemann invariants

**Eigenvalues:**

$$v_n - c, \quad v_n, \quad v_n, \quad v_n + c$$

# The Boundary Flux Jacobian

- The ghost state depends on the imposed boundary condition and on the interior state (subsonic or wall boundary)

**Ghost state:**

$$U_B = U_B(U_i, BC)$$

**Roe flux:**

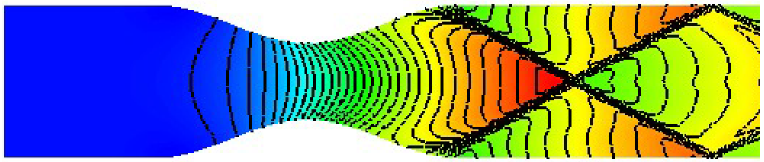
$$F_{iB} = \frac{F_i + F_B}{2} - \frac{1}{2}|A_{iB}|(U_B - U_i)$$

**Approximation:**

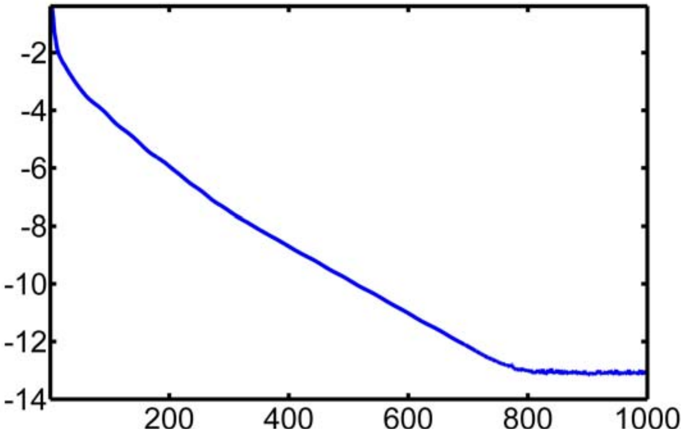
$$\frac{\partial F_{iB}}{\partial U_i} \approx \frac{1}{2}\left[\frac{\partial F(U_i)}{\partial U_i} + |A_{iB}| + \left[\frac{\partial F(U_B)}{\partial U_B} + |A_{iB}|\right]\frac{\partial U_B}{\partial U_i}\right]$$

# Transsonic Nozzle & GAMM Channel

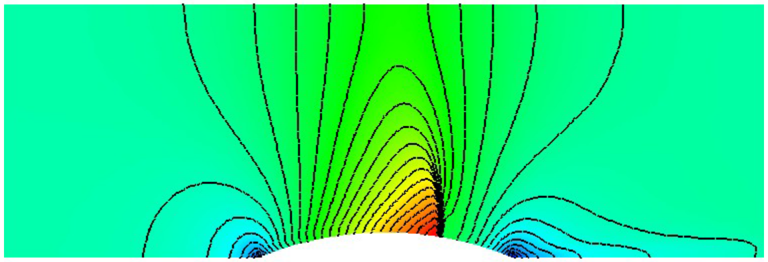
## Transsonic Nozzle M=0.8



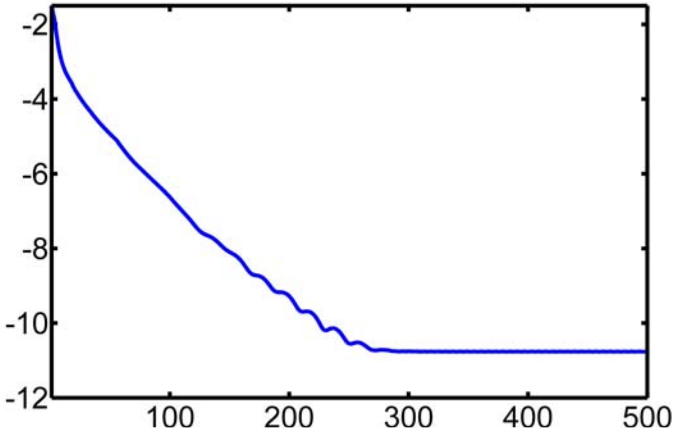
**Nonlinear Convergence**




## GAMM Channel M=0.67

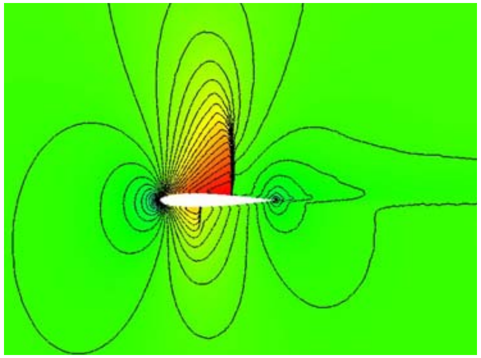


**Nonlinear Convergence**

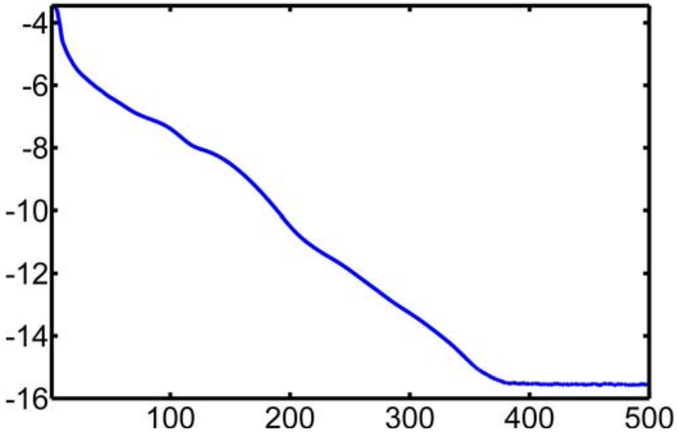


# Airfoil & Scramjet

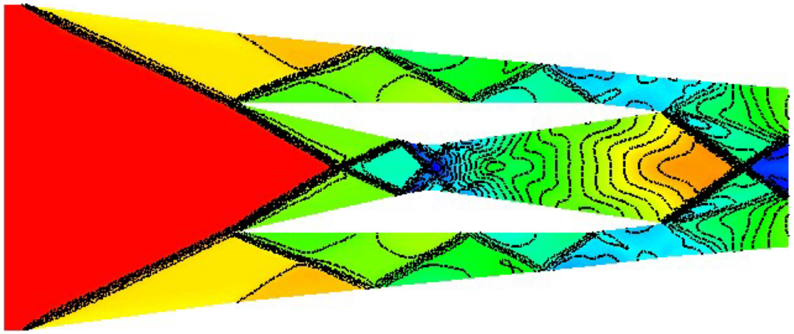
## NACA Airfoil M=0.8



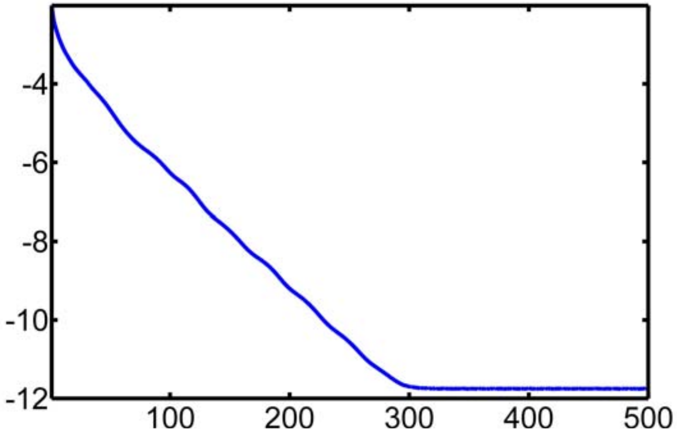
**Nonlinear Convergence**




## Scramjet M=3



**Nonlinear Convergence**



# Conclusions & Outlook

- High resolution, finite element TVD-scheme for the Euler equations
- Semi-implicit time-stepping / Newton – like solver
- No free parameters, information is inferred from the matrix entries
- Generalization to a two – fluid model is feasible and available
- Further work will focus on the development of a genuine Newton‘s method
- Adaptivity is required to reduce the computational cost (Matthias Möller)