# Compressible Navier-Stokes (Euler) Solver based on Deal.II Library

Lei Qiao

Northwestern Polytechnical University

Xi'an, China

Texas A&M University

College Station, Texas

Fifth deal.II Users and Developers Workshop

Texas A&M University, College Station, TX, USA

August 5, 2015

# Outline

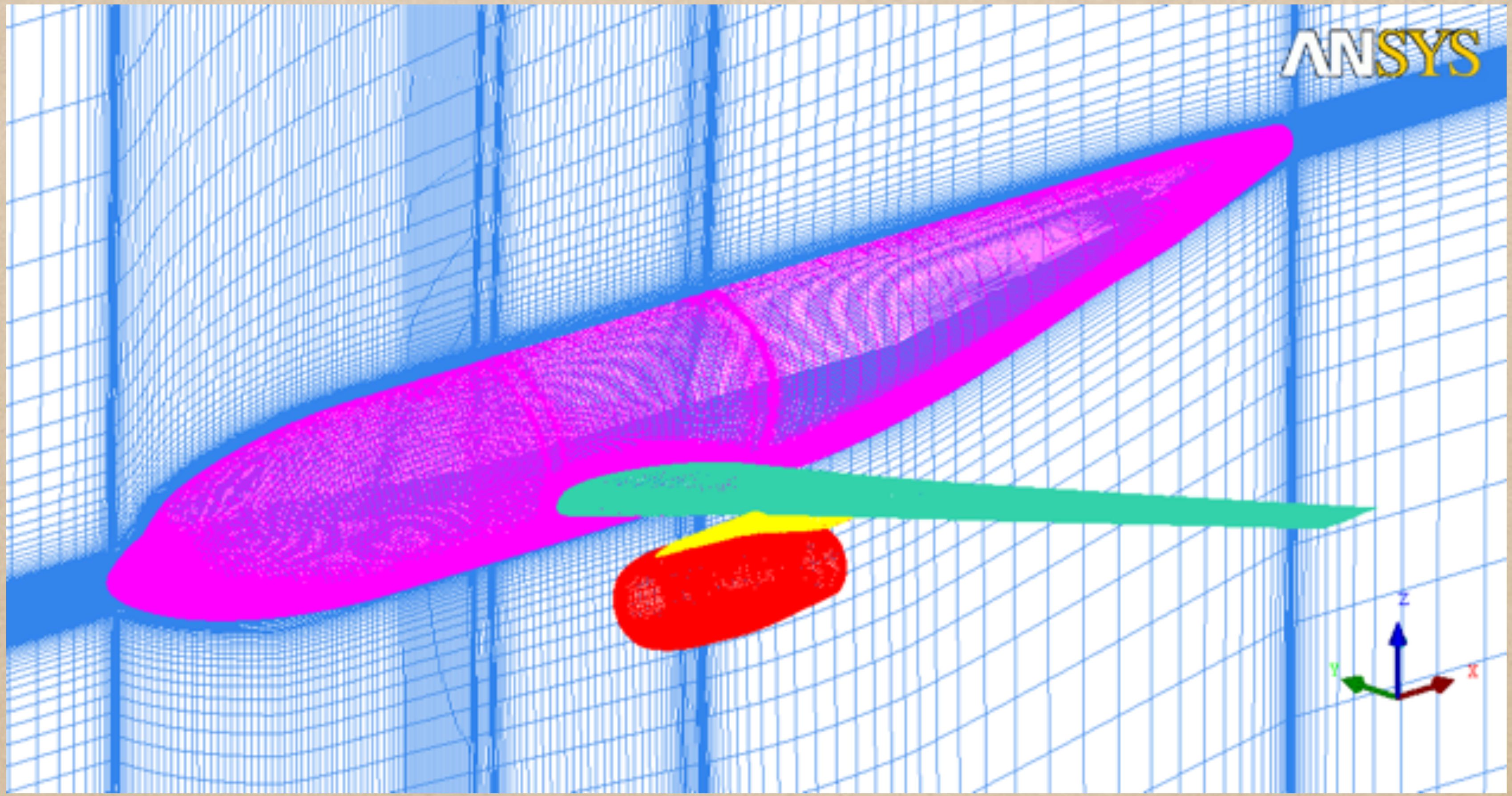
Motivation and Goal

Current work

Conclusion and TODOs

# Motivation

This is what I typically need to compute



# The pain point

Mesh generation:

- Time consuming
- Boring
- Experience depending

Solution:

- Mesh adaptation — let solver tell what is good mesh

# Deal.II the library

- Mesh adaptation with tree data structure
- FEM discretization
- Parallelization and excellent scalability

# Goal

A N-S solver  based on deal.II that:

- Starts from initial value on coarse mesh
- Converges to solution on a reasonable fine mesh adaptively
- With High-order capability

# Current work

Governing Equation

Solving technique

Solver implementation

Solver verification

# Governing Equation

Compressible Navier-Stokes Equation

$$\frac{\partial \mathbf{Q}(\mathbf{w})}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{w}) = \mathbf{S}(\mathbf{w})$$

Use primitive variables as working var.

$$\mathbf{w} = (u_j, \rho, p)^T$$

# Governing Equation

The Flux

$$\mathbf{F} = \mathbf{F}_c - \mathbf{F}_v$$

$$\mathbf{F}_c(\mathbf{w}) = \begin{pmatrix} \rho\mathbf{u} \otimes \mathbf{u} + \mathbf{I}p \\ \rho\mathbf{u} \\ (E+p)\mathbf{u} \end{pmatrix} = \begin{pmatrix} \rho u_i u_j + \delta_{ij} p \\ \rho u_i \\ (E+p)u_i \end{pmatrix}$$

# Governing Equation

The Flux

$$\mathbf{F} = \mathbf{F}_c - \mathbf{F}_v$$

$$\mathbf{F}_v(\mathbf{w}) = \frac{1}{Re^*_\infty} \left( \begin{array}{c} \tau_{ij} \\ 0 \\ \tau_{ij} u_i + \kappa \frac{\partial T}{\partial x_j} \end{array} \right)$$

With

$$T = \frac{p}{\gamma R \rho}$$

$$\tau_{ij} = \mu(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}) + \lambda \delta_{ij} \frac{\partial u_k}{\partial x_k}$$

# Governing Equation

Weak form:

On domain $\Omega$ , Find $\mathbf{w} \in \mathcal{S}$ that satisfy

$$\int_\Omega v_l \frac{\partial Q_l(\mathbf{w})}{\partial t} + \int_\Omega v_l \frac{\partial F_{l,i}(\mathbf{w})}{\partial x_i} = \int_\Omega v_l S_l(\mathbf{w})$$

$$\forall \mathbf{v} = v_l \in \mathcal{T}$$

$$\mathbf{w} = (u_j, \rho, p)^T$$

# Governing Equation

Weak form:

Integrate by part to get boundary flux

$$\int_\Omega v_l \frac{\partial Q_l(\mathbf{w})}{\partial t} + \int_{\partial\Omega} v_l F_{l,i}(\mathbf{w}) n_i$$
$$- \int_\Omega \frac{\partial v_l}{\partial x_i} F_{l,i}(\mathbf{w}) = \int_\Omega v_l S_l(\mathbf{w})$$

# Governing Equation

Treat boundary conditions and discontinuities on hanging node with this boundary flux

Compute numerical flux with Roe scheme[1]

# Governing Equation

## Boundary conditions:

Far field: Riemann invariant

Slip wall: Non-penetration

$$\mathbf{u} \cdot \mathbf{n} = 0$$

$$\rho_{out} = \rho_{in}$$

$$p_{out} = p_{in}$$

# Solving technique

BDF-1 (Implicit Euler) time integration

Newton linearization

Direct linear solver from Trilinos*

* Iterative solver doesn't stable enough, only used for scalability test

For steady run, time step size is determined according to norm of Newton update[2]

# Solver implementation

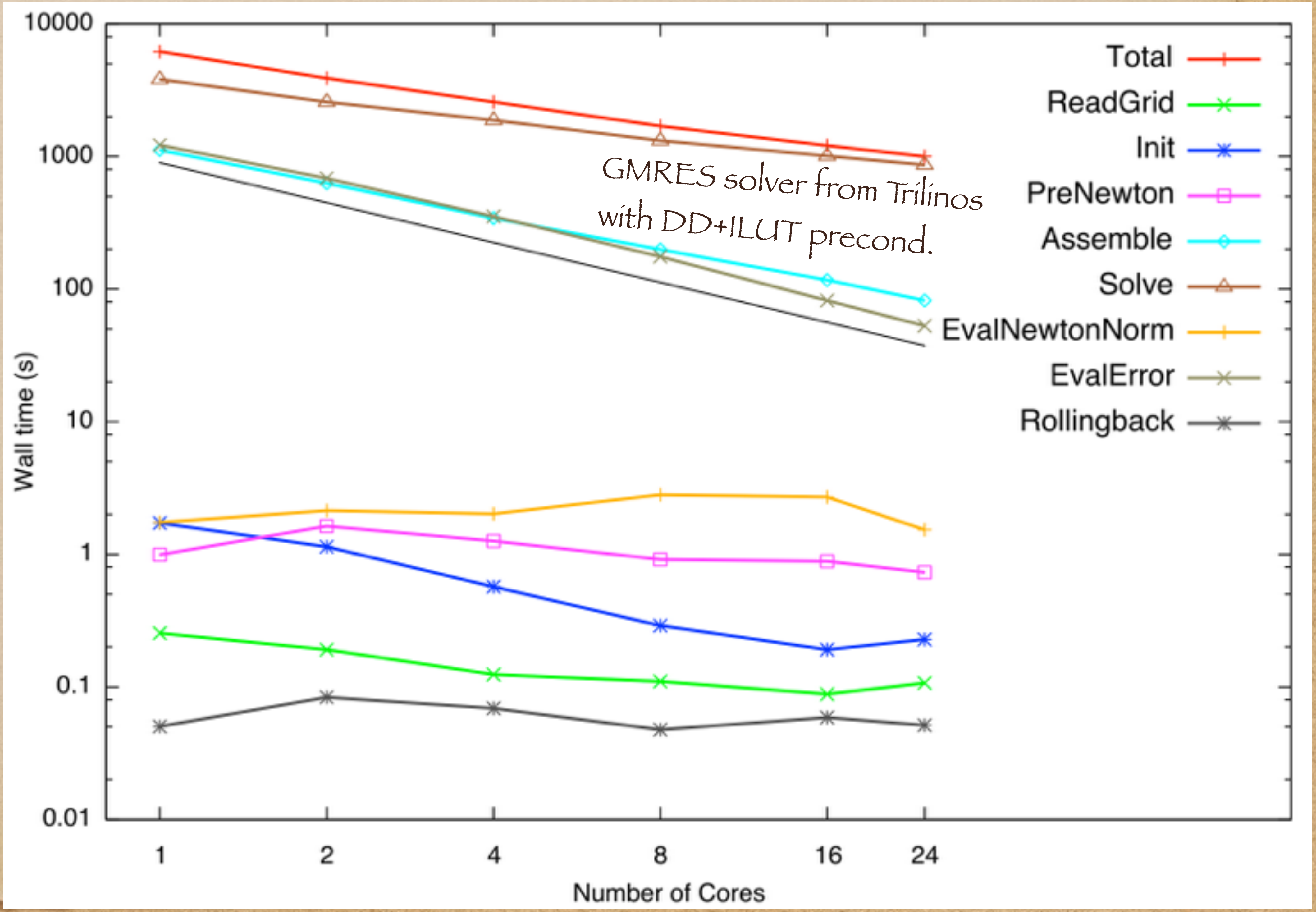
- Starts from step-33 of deal.II tutorial
  - Change working vars. from conservative ones to primitive ones
  - viscous flux
  - Riemann boundary condition
  - Roe flux[2] to replace the too viscous Lax flux
  - Parallelize: learned from step-40
- Version control with Git
- CMake project
- Regression test suit

# Solver verification

- Scalability
- Manufactured solution[3]
  - Subsonic
  - Supersonic
- Adaptive simulation over circle
- Flow over foil NACA2412

# Scalability

22,288 cells
91,312 DoFs



# Solver verification

- Manufactured solution[3]

$$\mathbf{w} = \mathbf{J}(x,y) = \begin{cases} u(x,y) = u_0 + u_x \sin(a_{ux}\pi x) + u_y \cos(a_{uy}\pi y) + u_{xy}\cos(a_{uxy}\pi xy) \\ v(x,y) = v_0 + v_x \cos(a_{vx}\pi x) + v_y \sin(a_{vy}\pi y) + v_{xy}\cos(a_{vxy}\pi xy) \\ \rho(x,y) = \rho_0 + \rho_x \sin(a_{\rho x}\pi x) + \rho_y \cos(a_{\rho y}\pi y) + \rho_{xy}\cos(a_{\rho xy}\pi xy) \\ p(x,y) = p_0 + p_x \cos(a_{px}\pi x) + p_y \sin(a_{py}\pi y) + p_{xy}\sin(a_{pxy}\pi xy) \end{cases}$$

$$\mathbf{J}_h(x,y)$$

Solver

$$\frac{\partial \mathbf{Q}(\mathbf{w})}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{w}) = \mathbf{S}(\mathbf{w})$$

$$\mathbf{S}(x,y) = \frac{\partial \mathbf{Q}(\mathbf{J})}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{J})$$

# Solver verification

- Manufactured solution[3]

$$\mathbf{w} = \mathbf{J}(x,y) = \begin{cases} u(x,y) = u_0 + u_x \sin(a_{ux}\pi x) + u_y \cos(a_{uy}\pi y) + u_{xy}\cos(a_{uxy}\pi xy) \\ v(x,y) = v_0 + v_x \cos(a_{vx}\pi x) + v_y \sin(a_{vy}\pi y) + v_{xy}\cos(a_{vxy}\pi xy) \\ \rho(x,y) = \rho_0 + \rho_x \sin(a_{\rho x}\pi x) + \rho_y \cos(a_{\rho y}\pi y) + \rho_{xy}\cos(a_{\rho xy}\pi xy) \\ p(x,y) = p_0 + p_x \cos(a_{px}\pi x) + p_y \sin(a_{py}\pi y) + p_{xy}\sin(a_{pxy}\pi xy) \end{cases}$$

Choosing different Constants to get

subsonic

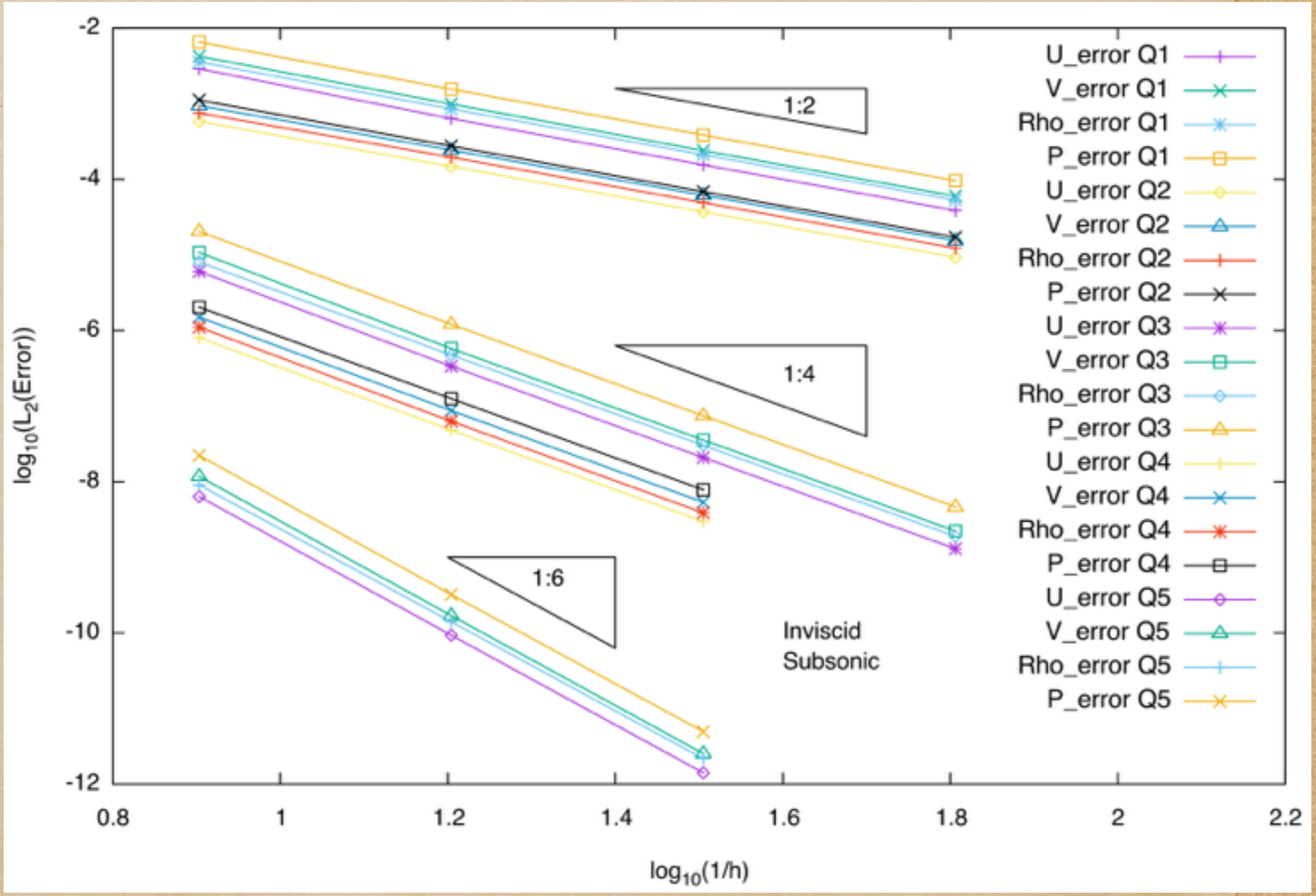
or

supersonic

case

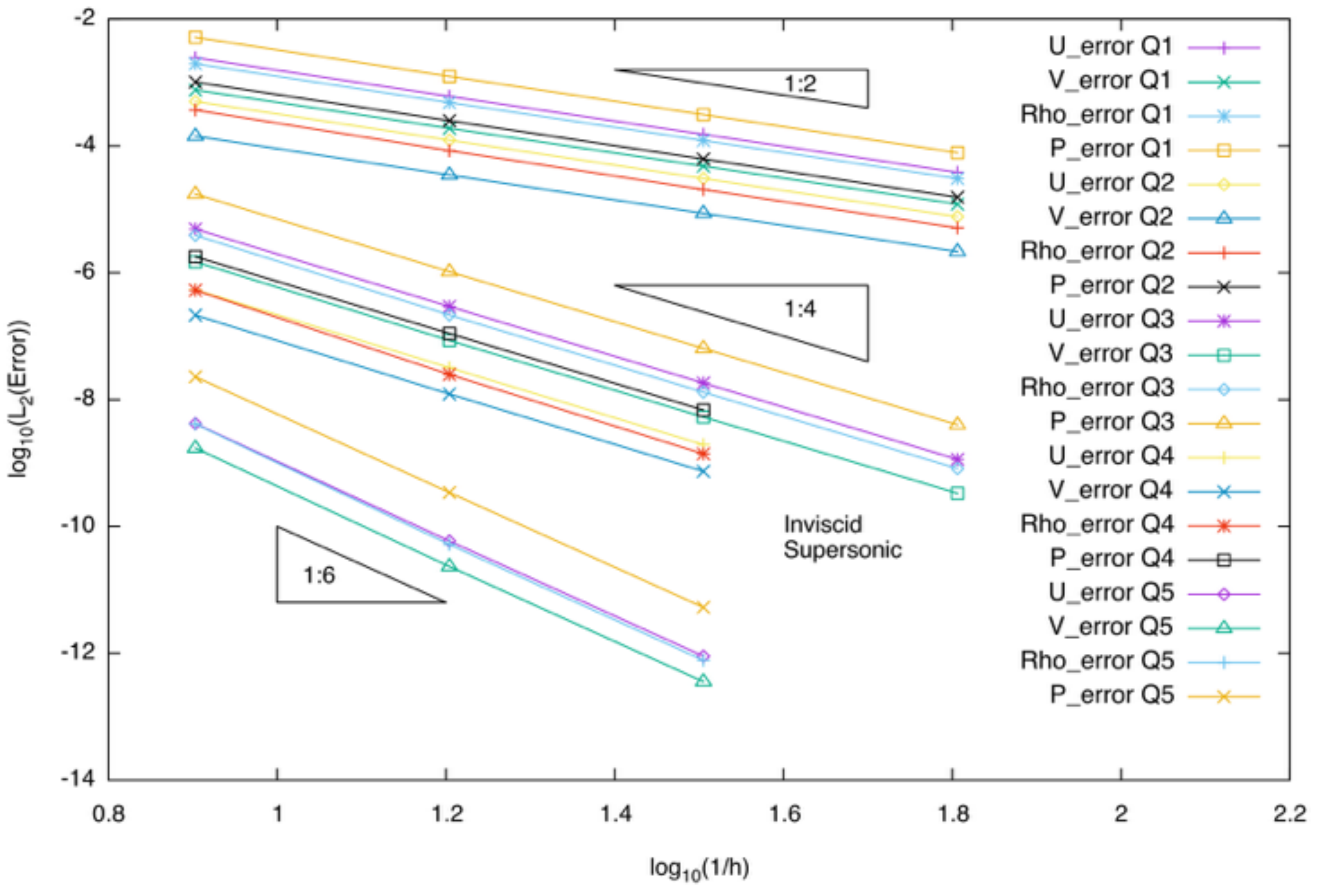
# Solver verification

- Manufactured solution
  - Subsonic
  - convergence order



# Solver verification

- Manufactured solution
    - Supersonic
    - convergence order

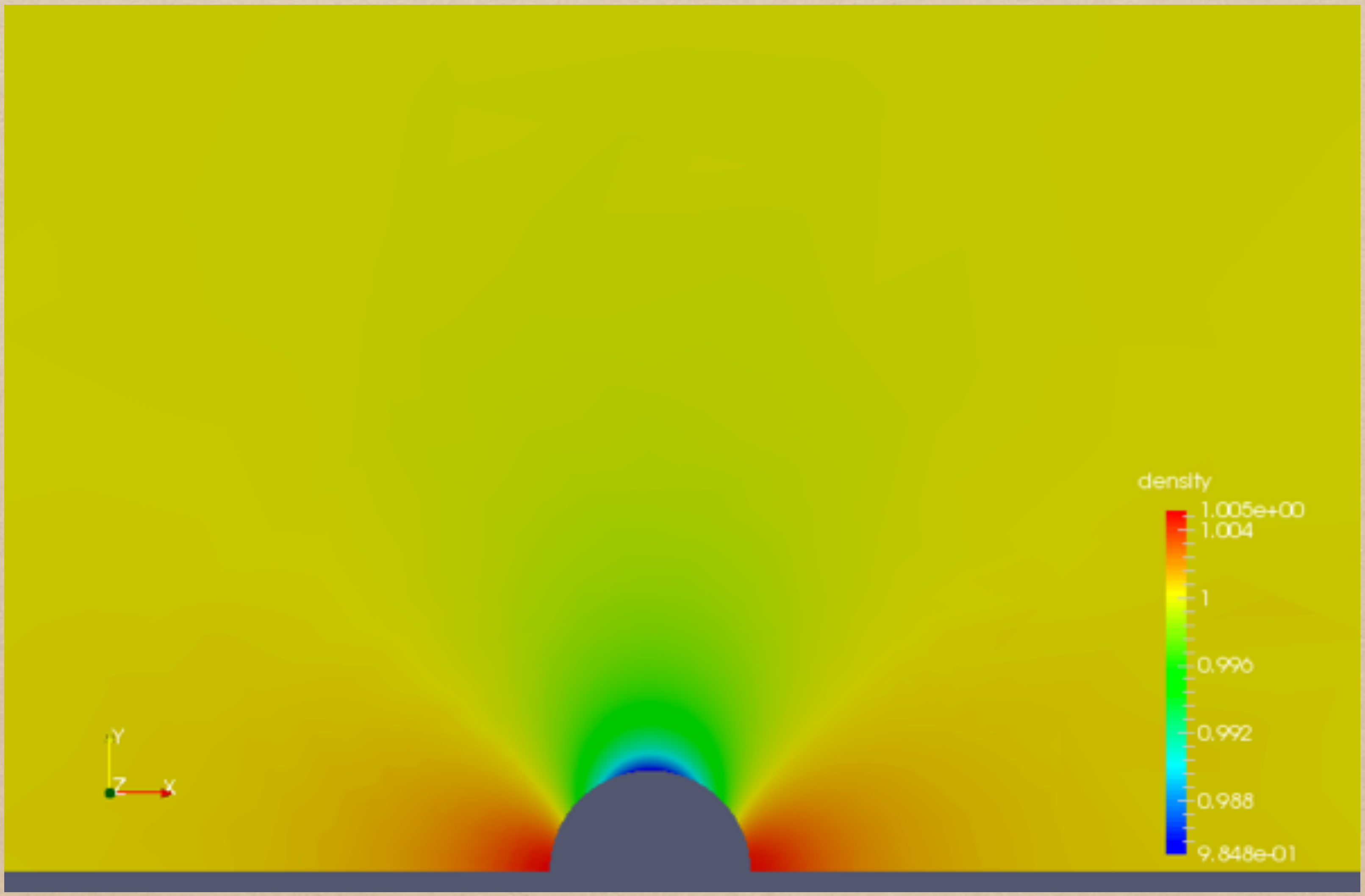


# Solver verification

- Adaptive simulation over circle
  - C1 mapping on wall boundary
  - Subsonic: Mach = 0.1
  - Almost inviscid: Cv=1e-6 just for stabilization
  - Converged Cd=0.00029 which ideal value is zero

# Solver verification

- Adaptive simulation over cylinder




Converged density contour

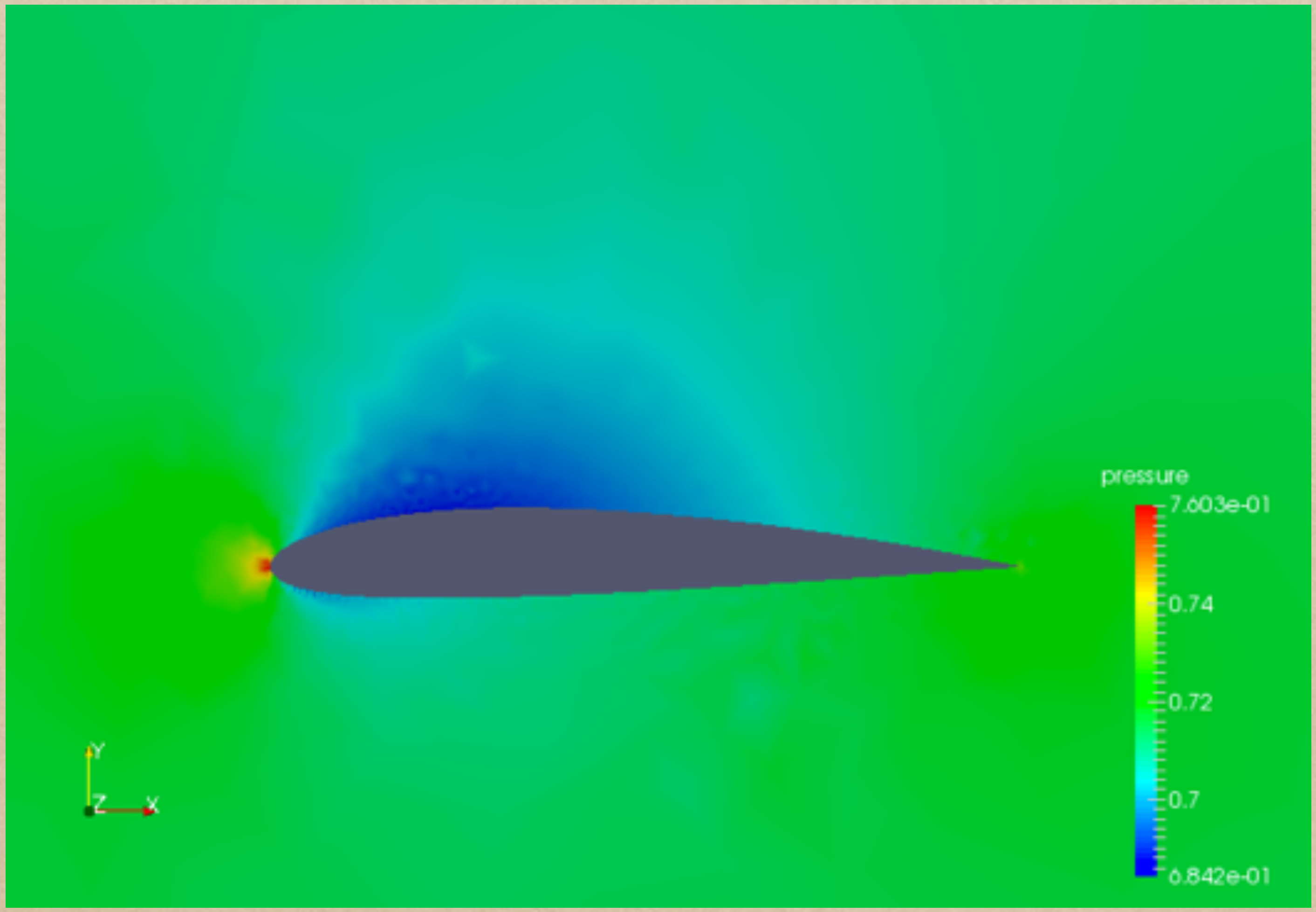
# Solver verification

- Adaptive simulation over cylinder


Converge history of density contour and mesh adaptation

# Solver verification

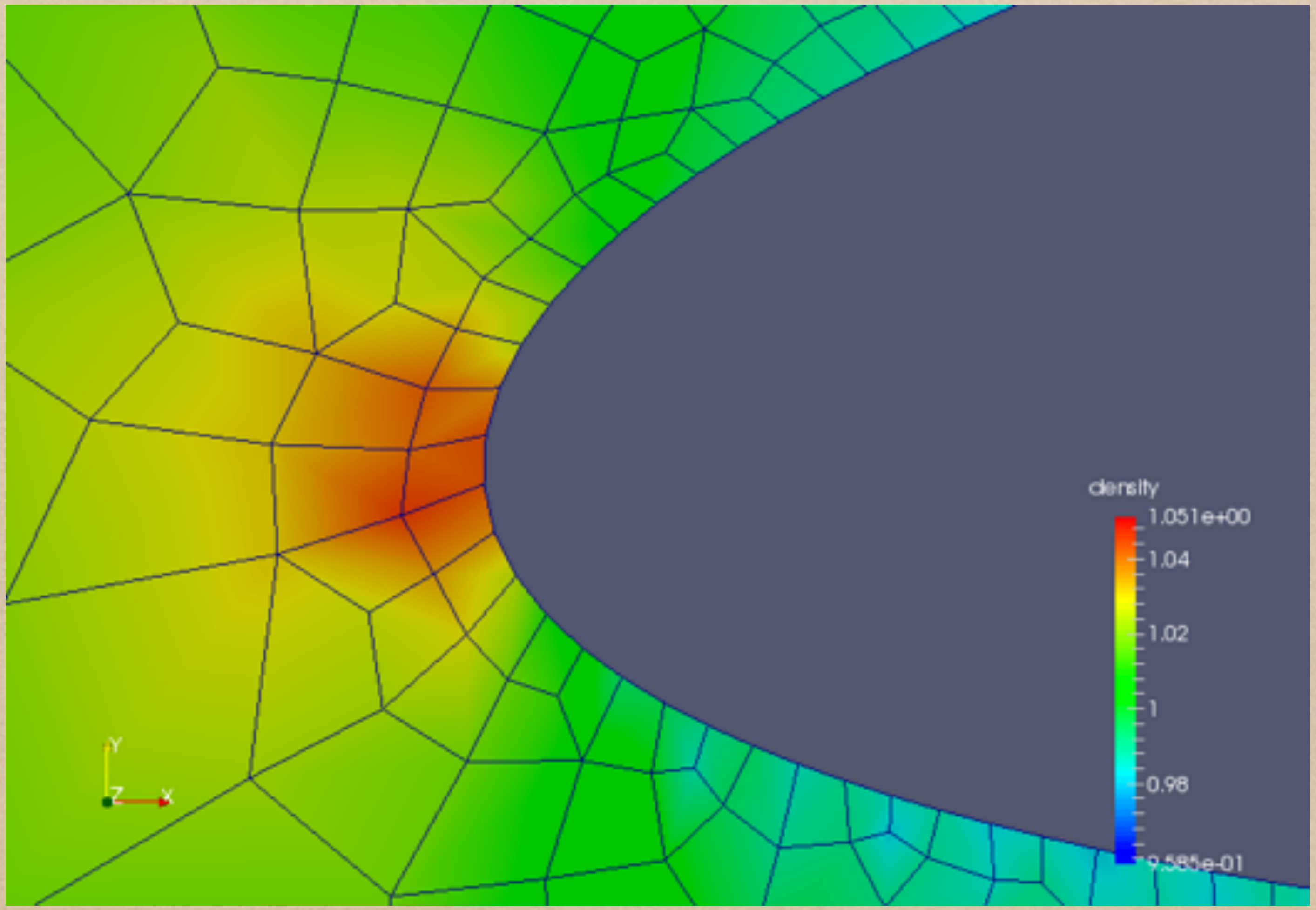
- Flow over NACA2412 foil
    - Subsonic: Mach=0.3




Converged pressure contour

# Solver verification

- Flow over NACA2412 foil
  - Subsonic: Mach=0.3




Oscillation may be caused by geometry non-smoothness

# Solver verification

- Flow over NACA2412 foil
  - Subsonic: Mach=0.3
  - Almost inviscid: Cv=3e-6 just for stabilization
  - Converged Cd=0.00025 which ideal value is zero
  - Converged Cl=0.2576, Cm=0.05235

- As a reference, xFoil [4] gives
  - Cd=-0.00076
  - Cl=0.2704
  - Cm=0.0585

# Conclusion

- A prototype NS solver based on deal.II is constructed
- The solver could run in parallel but the solver doesn't scale perfectly
- High order convergence is confirmed by MMS
- Solving process could start from very coarsen mesh and go on with mesh adaptation
- The solver can give out reasonable aerodynamics data

# ToDos

- Stable and efficient preconditioner for iterative solver (Now my hope is on MDF ordering of ILU)
- Describe wall boundary with NURBS geometry
- Non-slip boundary condition
- Anisotropic adaptation for boundary layer

# Reference


1. J. Blazek. Computational Fluid Dynamics: Principles and Applications (Second Edition). Elsevier Science, Oxford, second edition edition, 2005. ISBN 978-0-08- 044506-9.
2. J. Gatsis. Preconditioning Techniques for a Newton–Krylov Algorithm for the Compressible Navier–Stokes Equations. PhD thesis, University of Toronto, 2013.
3. C. J. Roy, C. C. Nelson, T. M. Smith, and C. C. Ober. Verification of Euler/Navier- stokes codes using the method of manufactured solutions. International Journal for Numerical Methods in Fluids, 44(6):599--620, 2004.
4. M. Drela, H. Youngren. http://web.mit.edu/drela/Public/web/xfoil/

# Thanks