

Towards Aerodynamic Simulations in Julia with Trixi.jl

JuliaCon 2024

Daniel Doebring¹

in collaboration with

Arpit Babbar², Simon Candelaresi³, Jesse Chan⁴,
Ahmad Peyvan⁵, and Andrew Winters⁶

¹ACoM, RWTH Aachen University ²TIFR-CAM, Bangalore ³HLRS, Stuttgart ⁴RCMOR, Rice University

⁵Applied Mathematics, Brown University ⁶TIMA, Linköping University



First Things First

Acknowledgements:

- **Arpit Babbar:**
Drag, Lift, Pressure and Friction Coefficients
- **Simon Candelaresi:**
Restarting simulations
- **Jesse Chan & Ahmad Peyvan:**
Parabolic adaptive mesh refinement (AMR), particularly mortars
- **Andrew Winters:**
Boundary identification for Gmsh generated meshes



First Things First

Acknowledgements:

- **Arpit Babbar:**
Drag, Lift, Pressure and Friction Coefficients
- **Simon Candelaresi:**
Restarting simulations
- **Jesse Chan & Ahmad Peyvan:**
Parabolic adaptive mesh refinement (AMR), particularly mortars
- **Andrew Winters:**
Boundary identification for Gmsh generated meshes

Reproducibility:

- Public repository on GitHub:

→ https://github.com/trixi-framework/talk-2024-juliacon_aerodynamics



Outline

① Scope of the Talk

② Case Study: NACA 4412 Airfoil

③ Extensions

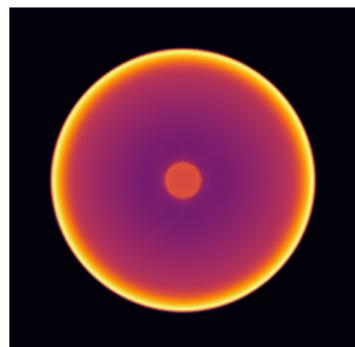
④ Conclusion & Outlook



About Trixi.jl

Trixi.jl → <https://github.com/trixi-framework/Trixi.jl>

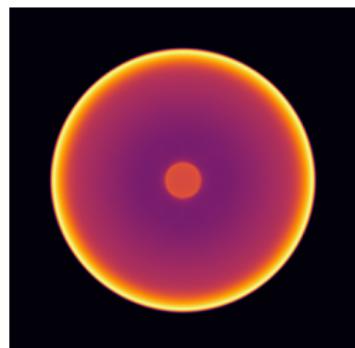
- is a Julia package for simulation of *compressible* flows (PDEs)



About Trixi.jl

Trixi.jl → <https://github.com/trixi-framework/Trixi.jl>

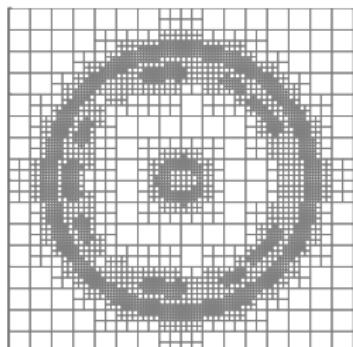
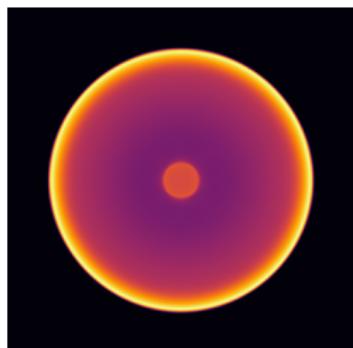
- is a Julia package for simulation of *compressible* flows (PDEs)
- implements the Discontinuous Galerkin (DG) method



About Trixi.jl

Trixi.jl → <https://github.com/trixi-framework/Trixi.jl>

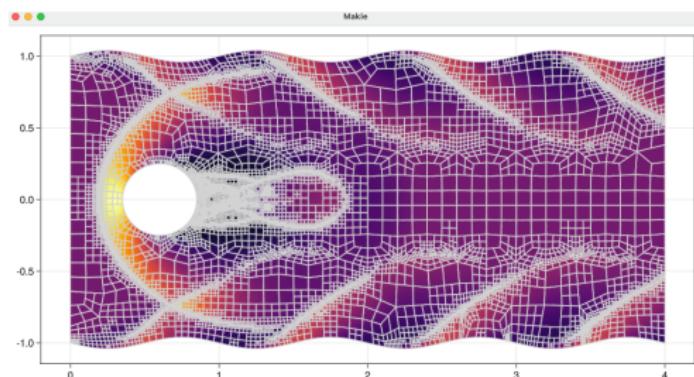
- is a Julia package for simulation of *compressible* flows (PDEs)
- implements the Discontinuous Galerkin (DG) method
- is a **grid-based** solver with adaptive mesh refinement (AMR)



Picking up where we left off

Andrew's talk at JuliaCon '22:

- HOHQMesh
- Inviscid (Hyperbolic) problems



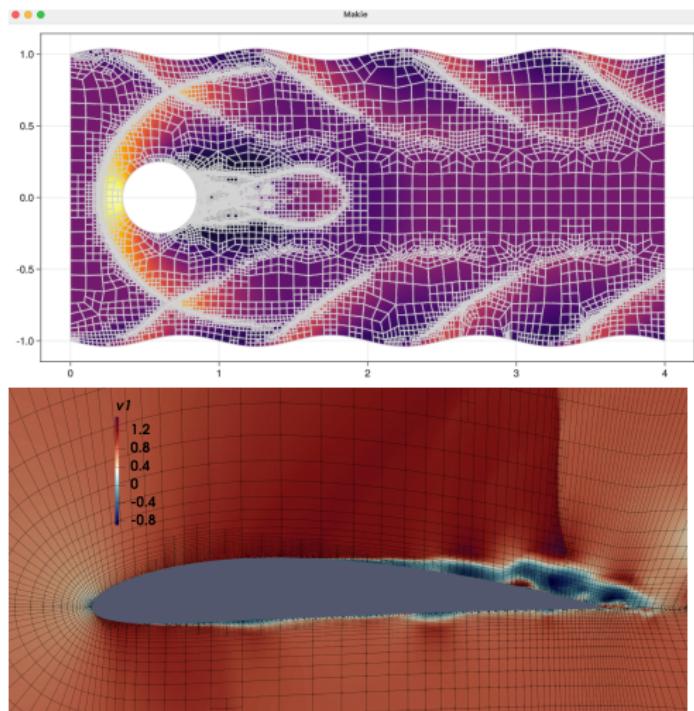
Picking up where we left off

Andrew's talk at JuliaCon '22:

- HOHQMesh
- Inviscid (Hyperbolic) problems

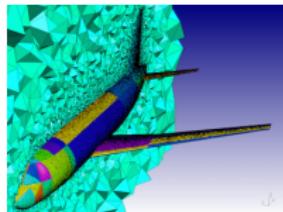
Today:

- Usage of existing meshes
- Viscous (Hyperbolic-Parabolic) problems



Motivation and Objectives

Mesh generation is a non-trivial task, especially for "true", i.e., non-extruded 3D meshes!



Picture taken from
<https://gmsh.info/>



^a<https://github.com/trixi-framework/P4est.jl>

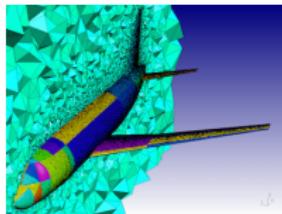
^b<https://github.com/cburstedde/p4est>

Motivation and Objectives

Mesh generation is a non-trivial task, especially for "true", i.e., non-extruded 3D meshes!

Goals: Be able to use existing

- Meshes in standardized format (`.inp`)
- Tools for mesh generation ⇒ Gmsh



Picture taken from
<https://gmsh.info/>



^a<https://github.com/trixi-framework/P4est.jl>

^b<https://github.com/cburstedde/p4est>

Motivation and Objectives

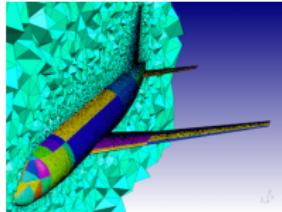
Mesh generation is a non-trivial task, especially for "true", i.e., non-extruded 3D meshes!

Goals: Be able to use existing

- Meshes in standardized format (`.inp`)
- Tools for mesh generation ⇒ Gmsh

While **maintaining features** like

- Adaptive mesh refinement (AMR)
- Parallelization
- Visualization
- Restarting



Picture taken from
<https://gmsh.info/>



^a<https://github.com/trixi-framework/P4est.jl>

^b<https://github.com/cburstedde/p4est>

Motivation and Objectives

Mesh generation is a non-trivial task, especially for "true", i.e., non-extruded 3D meshes!

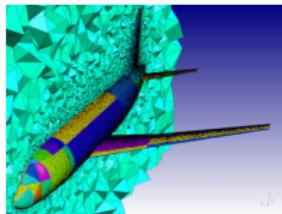
Goals: Be able to use existing

- Meshes in standardized format (`.inp`)
- Tools for mesh generation ⇒ Gmsh

While **maintaining features** like

- Adaptive mesh refinement (AMR) ✓
- Parallelization ✓
- Visualization ✓
- Restarting ✓

⇒ Use `P4est.jla`, wrapper around `p4estb` library



Picture taken from
<https://gmsh.info/>



^a<https://github.com/trixi-framework/P4est.jl>

^b<https://github.com/cburstedde/p4est>

NASA Turbulence Modeling Resource (TMR)

Plenty high-quality meshes for

- Airfoils
- Jets
- Channels
- ...

available online

→ <https://turbmodels.larc.nasa.gov/index.html>



Langley Research Center

Turbulence Modeling Resource



NASA Turbulence Modeling Resource (TMR)

Plenty high-quality meshes for

- Airfoils
- Jets
- Channels
- ...

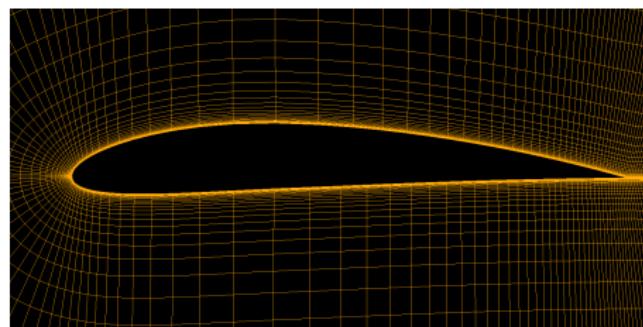
available online

→ <https://turbmodels.larc.nasa.gov/index.html>



Langley Research Center

Turbulence Modeling Resource



Here:

Focus on **2D NACA 4412 airfoil**

→ 14336 Quadrilateral elements



Outline

① Scope of the Talk

② Case Study: NACA 4412 Airfoil

③ Extensions

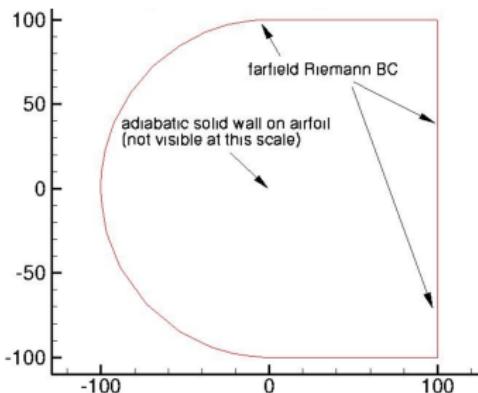
④ Conclusion & Outlook



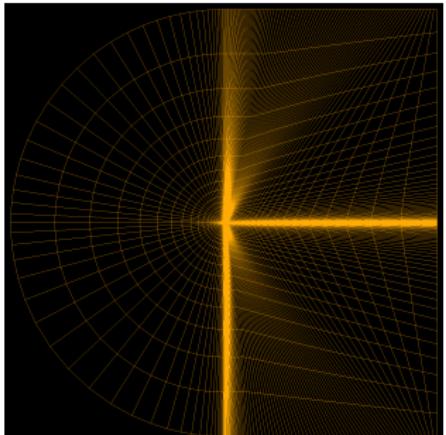
Case Study: NACA 4412 Airfoil

Mesh import workflow

- ① Download 3D mesh & boundary information file



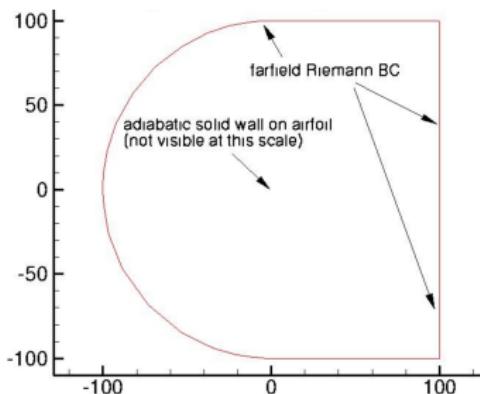
Picture taken from https://turbmodels.larc.nasa.gov/naca4412sep_val.html



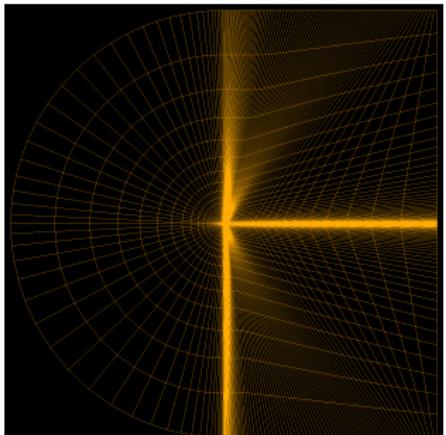
Case Study: NACA 4412 Airfoil

Mesh import workflow

- ① Download 3D mesh & boundary information file
- ② Convert .p3d to .msh via p3d2gmsh.py
→ <https://github.com/mrklein/p3d2gmsh>



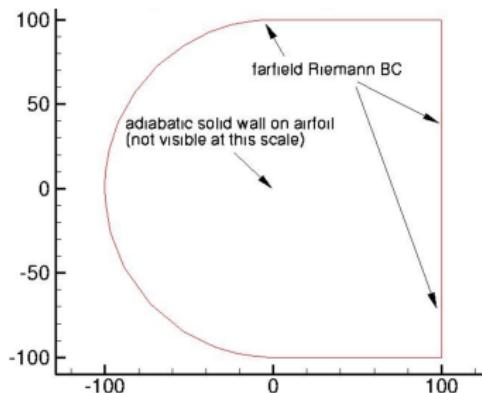
Picture taken from https://turbmodels.larc.nasa.gov/naca4412sep_val.html



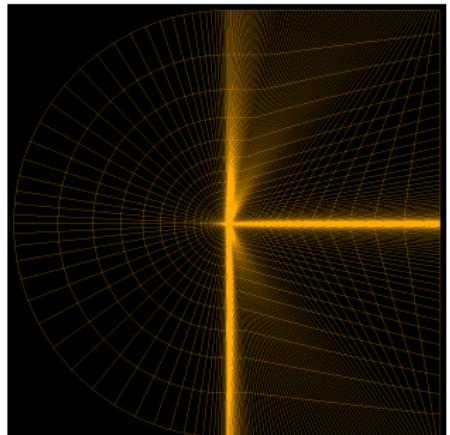
Case Study: NACA 4412 Airfoil

Mesh import workflow

- ① Download 3D mesh & boundary information file
- ② Convert .p3d to .msh via p3d2gmsh.py
→ <https://github.com/mrklein/p3d2gmsh>
- ③ Geometric ID assignment
→ AssignGeoIDs_SwapCoords.jl



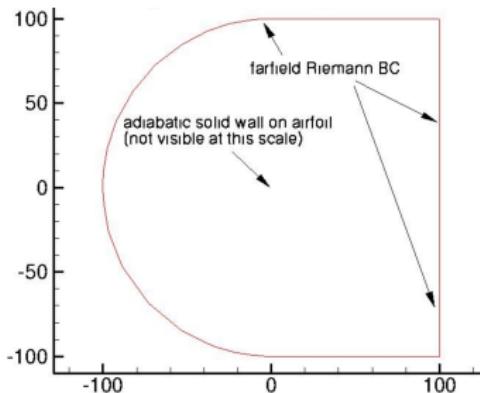
Picture taken from https://turbmodels.larc.nasa.gov/naca4412sep_val.html



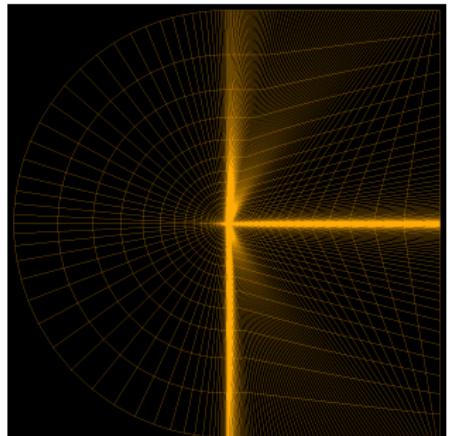
Case Study: NACA 4412 Airfoil

Mesh import workflow

- ① Download 3D mesh & boundary information file
- ② Convert .p3d to .msh via p3d2gmsh.py
→ <https://github.com/mrklein/p3d2gmsh>
- ③ Geometric ID assignment
→ AssignGeoIDs_SwapCoords.jl
- ④ Export .msh via Gmsh to .inp



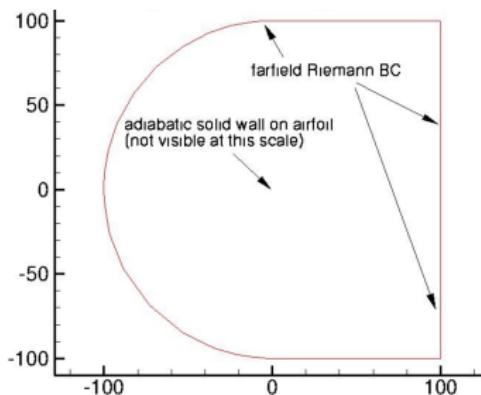
Picture taken from https://turbmodels.larc.nasa.gov/naca4412sep_val.html



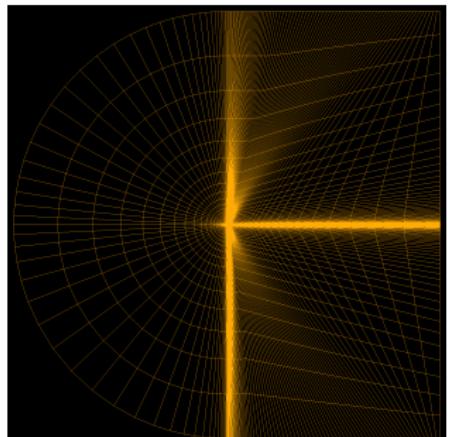
Case Study: NACA 4412 Airfoil

Mesh import workflow

- ① Download 3D mesh & boundary information file
- ② Convert .p3d to .msh via p3d2gmsh.py
→ <https://github.com/mrklein/p3d2gmsh>
- ③ Geometric ID assignment
→ AssignGeoIDs_SwapCoords.jl
- ④ Export .msh via Gmsh to .inp
- ⑤ Truncate 3D to 2D in .inp format
→ Abaqus3Dto2D.jl



Picture taken from https://turbmodels.larc.nasa.gov/naca4412sep_val.html



Case Study: NACA 4412 Airfoil

→ Ready for Trixi.jl:

- Mesh

```
using Trixi

# 225x65 C-grid
mesh_file = "NACA4412_2_2D_unique.inp"
boundary_symbols = [ # Farfield (Riemann-invariant) boundaries
                     :b2_symmetry_y_strong, :b4_farfield_riem,
                     :b5_farfield_riem, :b7_farfield_riem,
                     :b8_to_stitch_a,
                     # Airfoil
                     :b6_viscous_solid]

k = 3 # Local polynomial degree
mesh = P4estMesh{2}(mesh_file, polydeg = k,
                    boundary_symbols = boundary_symbols)
```



Case Study: NACA 4412 Airfoil

Inviscid transonic flow

- Equations & Solver

```
gamma = 1.4 # Ratio of specific heats
equations = CompressibleEulerEquations2D(gamma)
```

Case Study: NACA 4412 Airfoil

Inviscid transonic flow

- Equations & Solver

```
gamma = 1.4 # Ratio of specific heats
equations = CompressibleEulerEquations2D(gamma)

basis = LobattoLegendreBasis(k)
shock_indicator = IndicatorHennemannGassner(equations,
                                              basis,
                                              variable = density_pressure)
volume_integral = VolumeIntegralShockCapturingHG(shock_indicator;
                                                 volume_flux_dg = flux_chandrashekar,
                                                 volume_flux_fv = flux_hllc)

# About  $1.15 \cdot 10^6$  degrees of freedom
solver = DGSEM(polydeg = k,
                surface_flux = flux_hllc,
                # Use split-form DG
                volume_integral = volume_integral)
```



Case Study: NACA 4412 Airfoil

Inviscid transonic flow

- Initial Condition

$$Ma_{\infty} = 0.8, \alpha = 1.25^\circ$$

```
@inline function initial_condition_mach08_flow(x, t, equations)
    # Non-dimensionalized (for gas with  $\gamma = 1.4$ )
    rho_freestream = 1.4
    v1 = 0.7998096216639273    #  $0.8 * \cos(1.25^\circ)$ 
    v2 = 0.017451908027648896 #  $0.8 * \sin(1.25^\circ)$ 
    p_freestream = 1.0

    prim = SVector(rho_freestream, v1, v2, p_freestream)
    return prim2cons(prim, equations)
end

initial_condition = initial_condition_mach08_flow
```



Case Study: NACA 4412 Airfoil

Inviscid transonic flow

- Boundary Conditions & Semidiscretization

```
bc_free_stream = BoundaryConditionDirichlet(initial_condition)

bc = Dict(# Farfield boundaries
            :b2_symmetry_y_strong => bc_free_stream,
            :b4_farfield_riem => bc_free_stream,
            :b5_farfield_riem => bc_free_stream,
            :b7_farfield_riem => bc_free_stream,
            :b8_to_stitch_a => bc_free_stream,
            # Airfoil
            :b6_viscous_solid => boundary_condition_slip_wall)
```

Case Study: NACA 4412 Airfoil

Inviscid transonic flow

- Boundary Conditions & Semidiscretization

Case Study: NACA 4412 Airfoil

Inviscid transonic flow

- Boundary Conditions & Semidiscretization

```
bc_free_stream = BoundaryConditionDirichlet(initial_condition)

bc = Dict(# Farfield boundaries
            :b2_symmetry_y_strong => bc_free_stream,
            :b4_farfield_riem => bc_free_stream,
            :b5_farfield_riem => bc_free_stream,
            :b7_farfield_riem => bc_free_stream,
            :b8_to_stitch_a => bc_free_stream,
            # Airfoil
            :b6_viscous_solid => boundary_condition_slip_wall)

semi = SemidiscretizationHyperbolic(mesh, equations,
                                       initial_condition, solver;
                                       boundary_conditions = bc)

tspan = (0.0, 10.0)
ode = semidiscretize(semi, tspan) # Compatible with OrdinaryDiffEq.jl
```

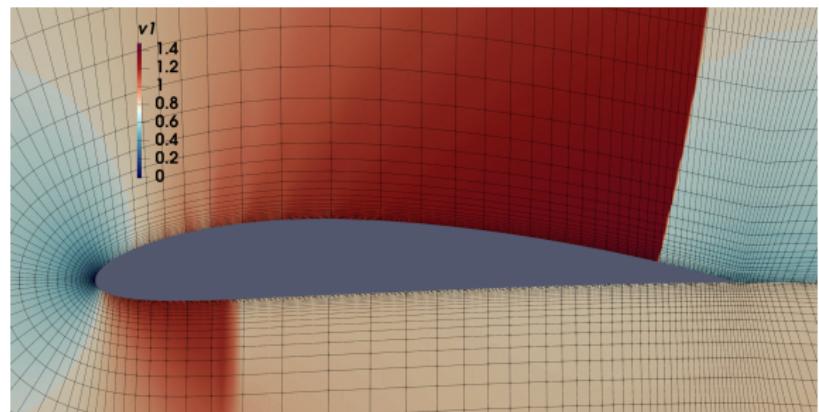


Case Study: NACA 4412 Airfoil

Inviscid transonic flow

$$t_0 = 10.0 \rightarrow t_f = 10.5$$

- NASA TMR mesh

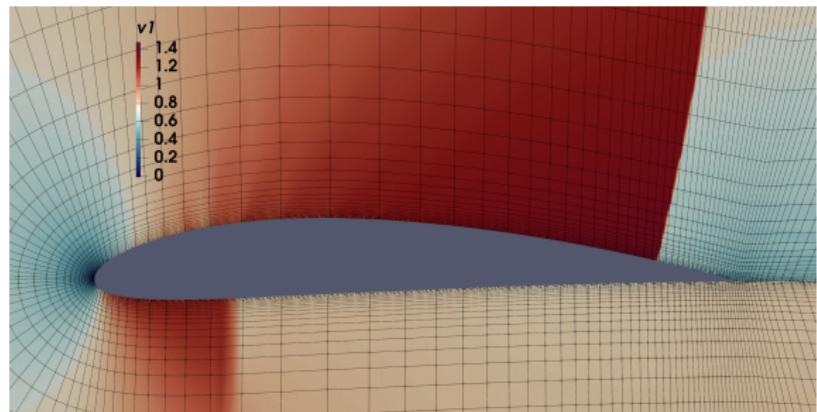


Case Study: NACA 4412 Airfoil

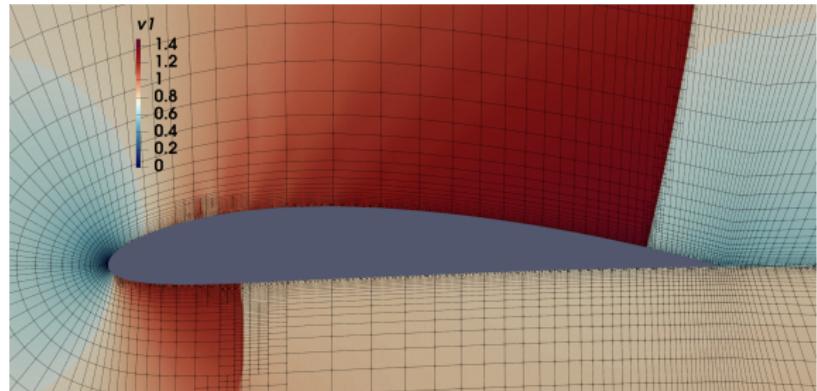
Inviscid transonic flow

$$t_0 = 10.0 \rightarrow t_f = 10.5$$

- NASA TMR mesh



- NASA TMR mesh
+ 3-Level AMR



Case Study: NACA 4412 Airfoil

Viscous transonic flow

- Viscous Equations & Boundary Conditions

Case Study: NACA 4412 Airfoil

Viscous transonic flow

- Viscous Equations & Boundary Conditions

Case Study: NACA 4412 Airfoil

Viscous transonic flow

- Viscous Equations & Boundary Conditions

```
prandtl_number() = 0.72 #  $Pr = \mu c_p / \lambda$ 
mu() = 1.12e-7 # Constant dynamic viscosity  $\mu \Rightarrow Re = 10^7$ 
equations_parabolic = CompressibleNavierStokesDiffusion2D(equations, # Euler equations
                                                               mu = mu(),
                                                               Prandtl = prandtl_number())

velocity_airfoil = NoSlip((x, t, equations) -> SVector(0.0, 0.0))
heat_airfoil = Adiabatic((x, t, equations) -> 0.0)

bc_airfoil = BoundaryConditionNavierStokesWall(velocity_airfoil,
                                                heat_airfoil)

bc_parabolic = Dict(:b2_symmetry_y_strong => bc_free_stream,
                     #: Other free-stream BCs =#
                     :b6_viscous_solid => bc_airfoil)
```



Case Study: NACA 4412 Airfoil

Viscous transonic flow

- Semidiscretization

```
#  
semi = SemidiscretizationHyperbolicParabolic(mesh, (equations, equations_parabolic),  
                                              initial_condition, solver;  
                                              boundary_conditions = (bc, # ↓  
                                              bc_parabolic))
```

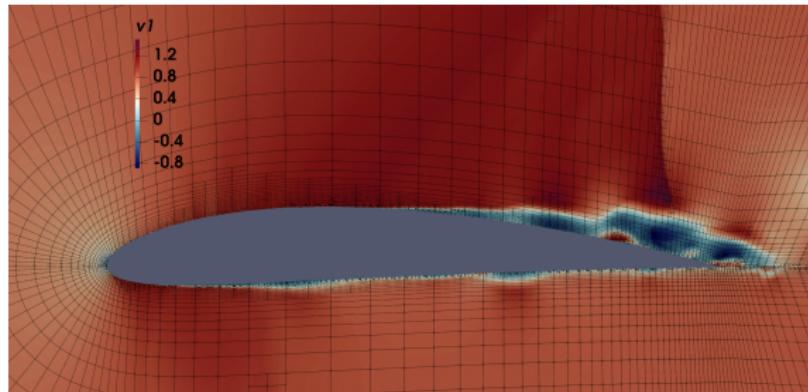


Case Study: NACA 4412 Airfoil

Viscous transonic flow

- Semidiscretization

```
#  
semi = SemidiscretizationHyperbolicParabolic(mesh, (equations, equations_parabolic),  
                                              initial_condition, solver;  
                                              boundary_conditions = (bc, # ↓  
                                              bc_parabolic))
```

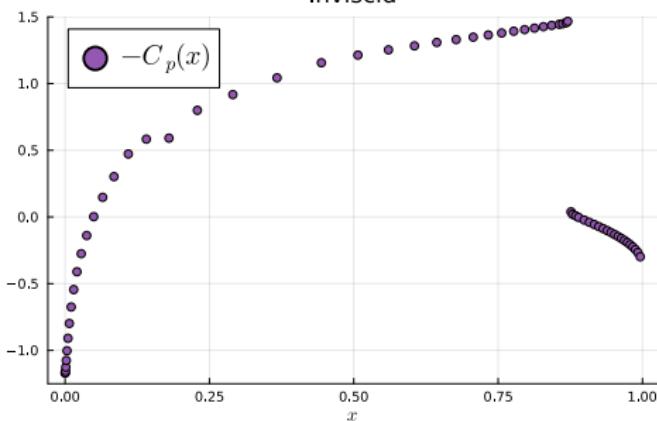


Case Study: NACA 4412 Airfoil

Pressure Coefficients

$$C_p(x) = \frac{p(x) - p_\infty}{2\rho_\infty L_\infty u_\infty^2}$$

Inviscid

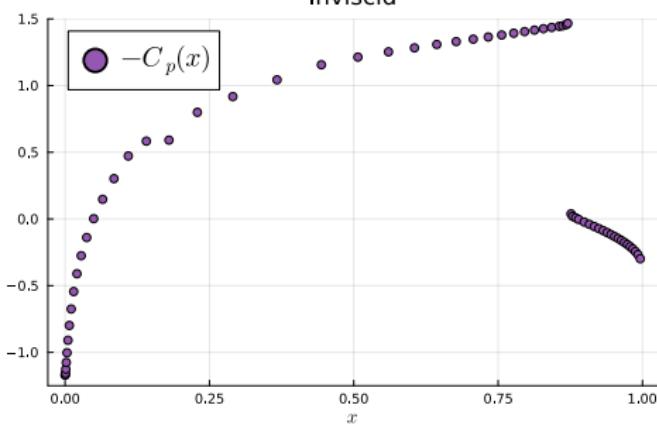


Case Study: NACA 4412 Airfoil

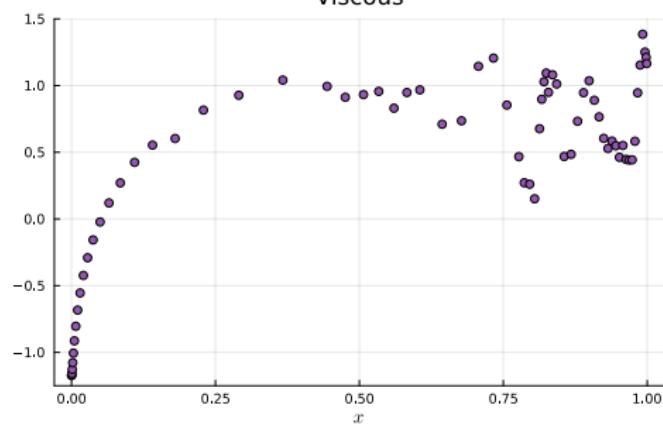
Pressure Coefficients

$$C_p(x) = \frac{p(x) - p_\infty}{2\rho_\infty L_\infty u_\infty^2}$$

Inviscid



Viscous



Outline

① Scope of the Talk

② Case Study: NACA 4412 Airfoil

③ Extensions

④ Conclusion & Outlook



Case Study: SD7003 Airfoil

Viscous laminar (incompressible) flow

- $\text{Re} = 10^4, \text{Ma} = 0.2$
- $\text{Pr} = 0.72, \gamma = 1.4$
- $\alpha = 4^\circ$

Mesh from ESM in^a

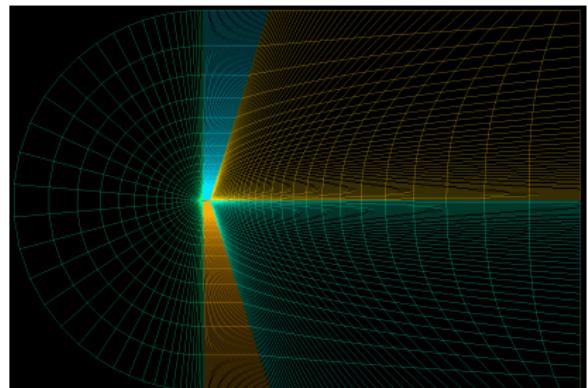
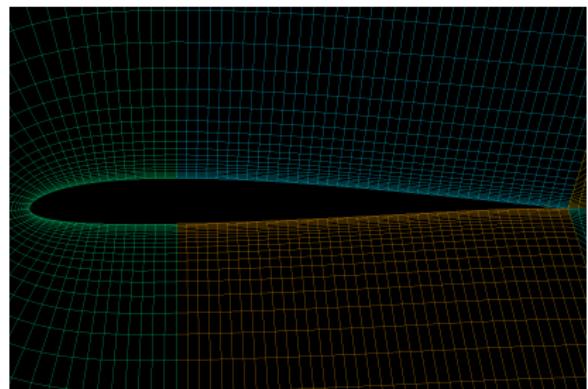
- $x_{\min} = -20, x_{\max} = 60,$
 $y_{\min} = -20, y_{\max} = 20$
- 7605 elements

Discretization:

- Space: 4th-order DGSEM
- Entropy-stable flux-differencing
- Time: 4th-order Multirate P-ERK

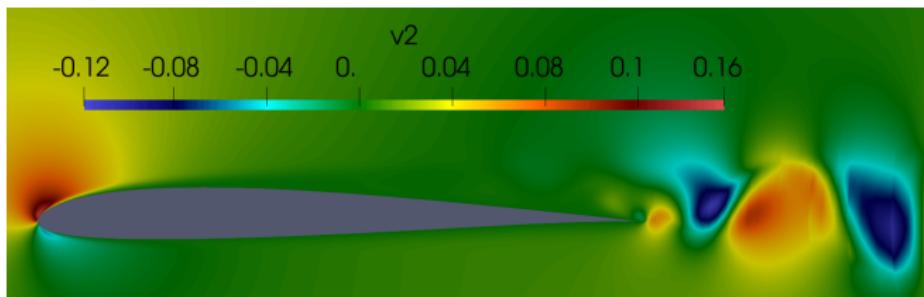
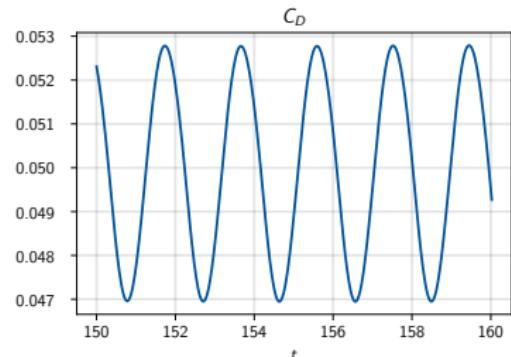
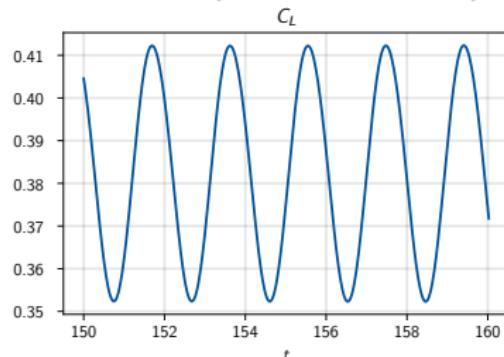


^aB. C. Vermeire, P-ERK for Stiff Systems of Equations, JCP, 2019.



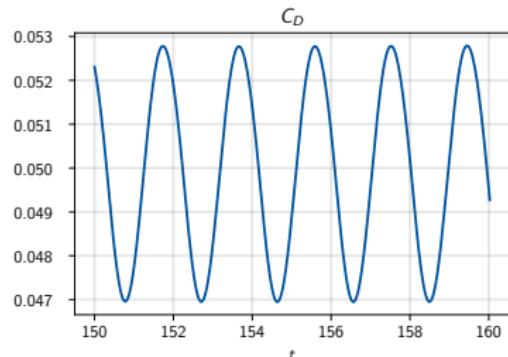
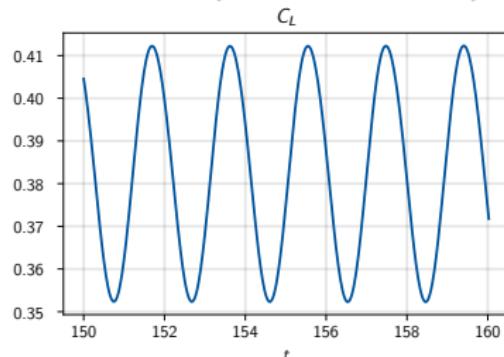
Case Study: SD7003 Airfoil

Viscous laminar (incompressible) flow



Case Study: SD7003 Airfoil

Viscous laminar (incompressible) flow



Source	$\bar{C}_L = \bar{C}_{L,p}$	$\bar{C}_D = \bar{C}_{D,p} + \bar{C}_{D,\mu}$
Trixi.jl	0.3827	0.04995
Vermeire ^a	0.3841	0.04990
Uranga et al. ^b	0.3755	0.04978
López-Morales et al. ^c	0.3719	0.04940



^aB. C. Vermeire, JCP, 2019.

^bA. Uranga et al., Int. J. Num Methods Engineering, 2011

^cM. López-Morales et al., 32nd AIAA AAC, 2014

Viscous flows: Extensions

Sutherland's law

$$\mu(T) = \mu_{\text{ref}} \left(\frac{T_{\text{ref}} + C}{T + C} \right) \left(\frac{T}{T_{\text{ref}}} \right)^{1.5}$$

```
#mu() = 1.12e-7
@inline function mu(u, equations)
    T_ref = 291.15

    R_specific_air = 287.052874
    T = R_specific_air * Trixi.temperature(u, equations) # Re-dim. temperature

    C_air = 120.0
    mu_ref_air = 1.827e-5

    return mu_ref_air * (T_ref + C_air) / (T + C_air) * (T / T_ref)^1.5
end
```

Viscous flows: Extensions

Sutherland's law

$$\mu(T) = \mu_{\text{ref}} \left(\frac{T_{\text{ref}} + C}{T + C} \right) \left(\frac{T}{T_{\text{ref}}} \right)^{1.5}$$

```
#mu() = 1.12e-7
@inline function mu(u, equations)
    T_ref = 291.15

    R_specific_air = 287.052874
    T = R_specific_air * Trixi.temperature(u, equations) # Re-dim. temperature

    C_air = 120.0
    mu_ref_air = 1.827e-5

    return mu_ref_air * (T_ref + C_air) / (T + C_air) * (T / T_ref)^1.5
end

equations_parabolic = CompressibleNavierStokesDiffusion2D(equations,
    mu = mu, # mu = mu(),
    Prandtl = prandtl_number())
```



Outline

① Scope of the Talk

② Case Study: NACA 4412 Airfoil

③ Extensions

④ Conclusion & Outlook



Conclusion & Outlook

Summary:

- Import of existing quadrilateral meshes in .inp format



Conclusion & Outlook

Summary:

- Import of existing quadrilateral meshes in .inp format
- Transonic (compressible) flow over NACA 4412 airfoil



Conclusion & Outlook

Summary:

- Import of existing quadrilateral meshes in .inp format
- Transonic (compressible) flow over NACA 4412 airfoil
 - Inviscid
 - Viscous



Conclusion & Outlook

Summary:

- Import of existing quadrilateral meshes in .inp format
- Transonic (compressible) flow over NACA 4412 airfoil
 - Inviscid
 - Viscous
- Laminar (incompressible) flow over SD7003 airfoil



Conclusion & Outlook

Summary:

- Import of existing quadrilateral meshes in .inp format
- Transonic (compressible) flow over NACA 4412 airfoil
 - Inviscid
 - Viscous
- Laminar (incompressible) flow over SD7003 airfoil

ToDo:

- Higher order (curved) quad/hex elements from Gmsh / in .inp format



Conclusion & Outlook

Summary:

- Import of existing quadrilateral meshes in .inp format
- Transonic (compressible) flow over NACA 4412 airfoil
 - Inviscid
 - Viscous
- Laminar (incompressible) flow over SD7003 airfoil

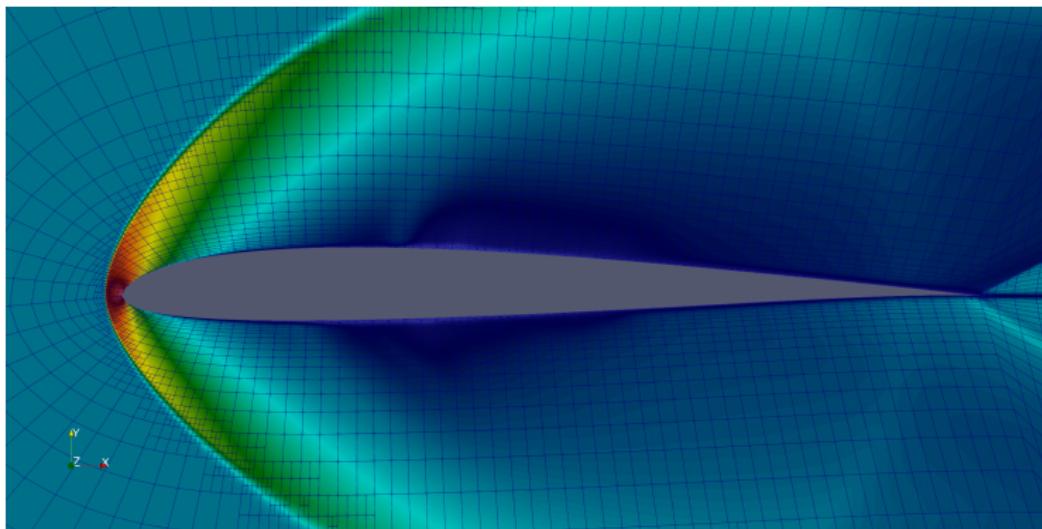
ToDo:

- Higher order (curved) quad/hex elements from Gmsh / in .inp format
- Beyond quads/hexes with T8Code.jl
→ <https://github.com/DLR-AMR/T8code.jl>



Thank you for your attention!

Questions?



Density profile for Mach 2 flow around SD7003 airfoil with AMR.

