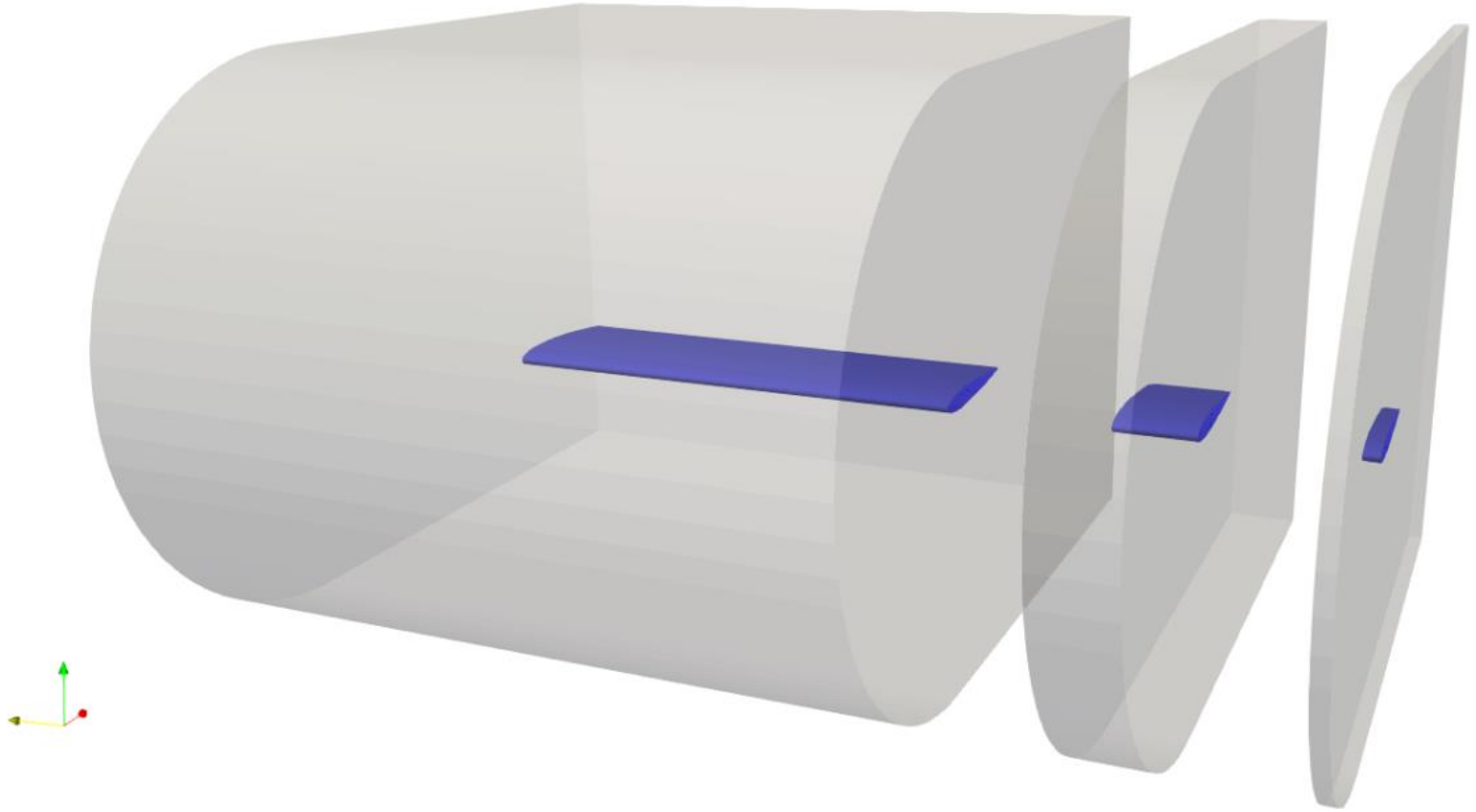
Mesh Generation with blockMesh

Structured C-Mesh for airfoils and wings

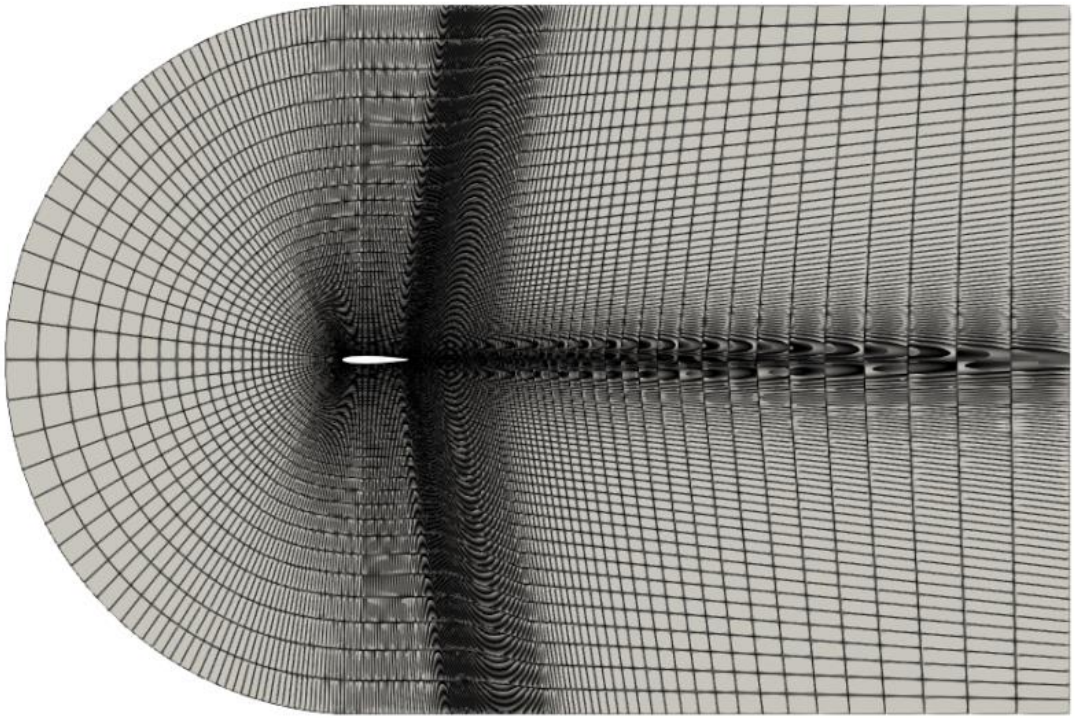


v3 (23/08/2024)

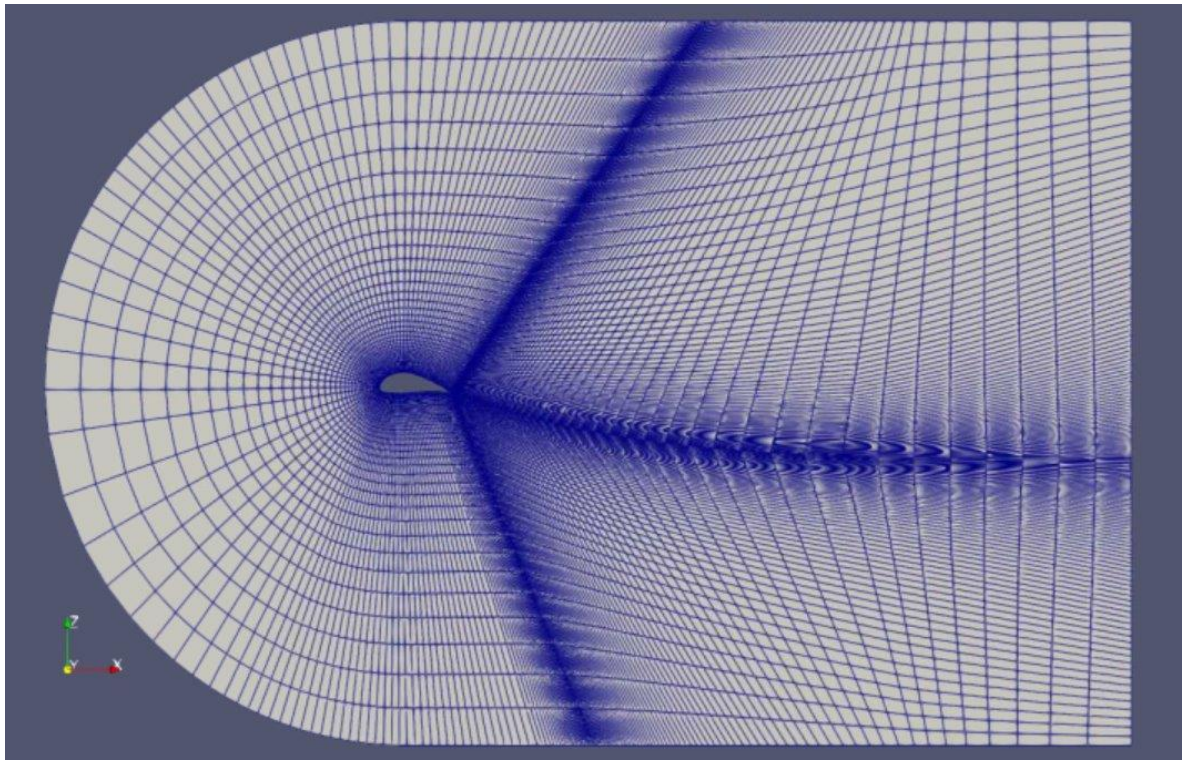
Objectives

- Generation of **structured C-type** grids for airfoils and wings
- Mesh generation using **blockMesh** utility of OpenFOAM
- **Inputs**: Airfoil coordinates file, wing shape file, and block structure
- **Outputs**: blockMesh input files incl. vertices, blocks, edges, BCs…

Examples

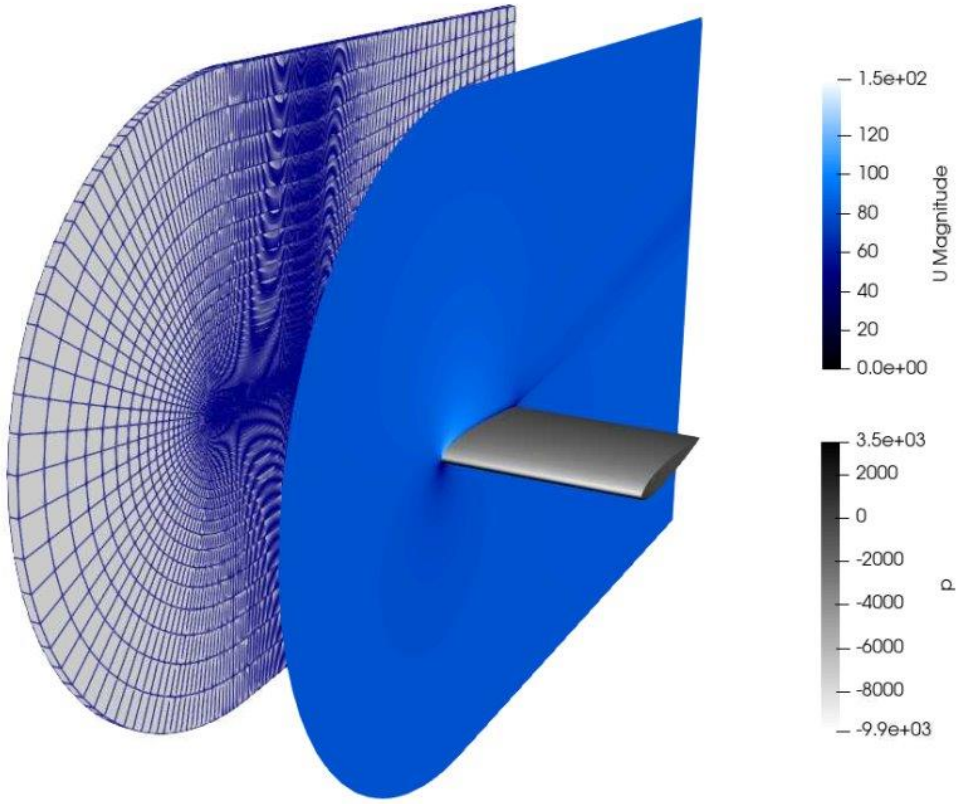


2D grid of NACA 0012 airfoil

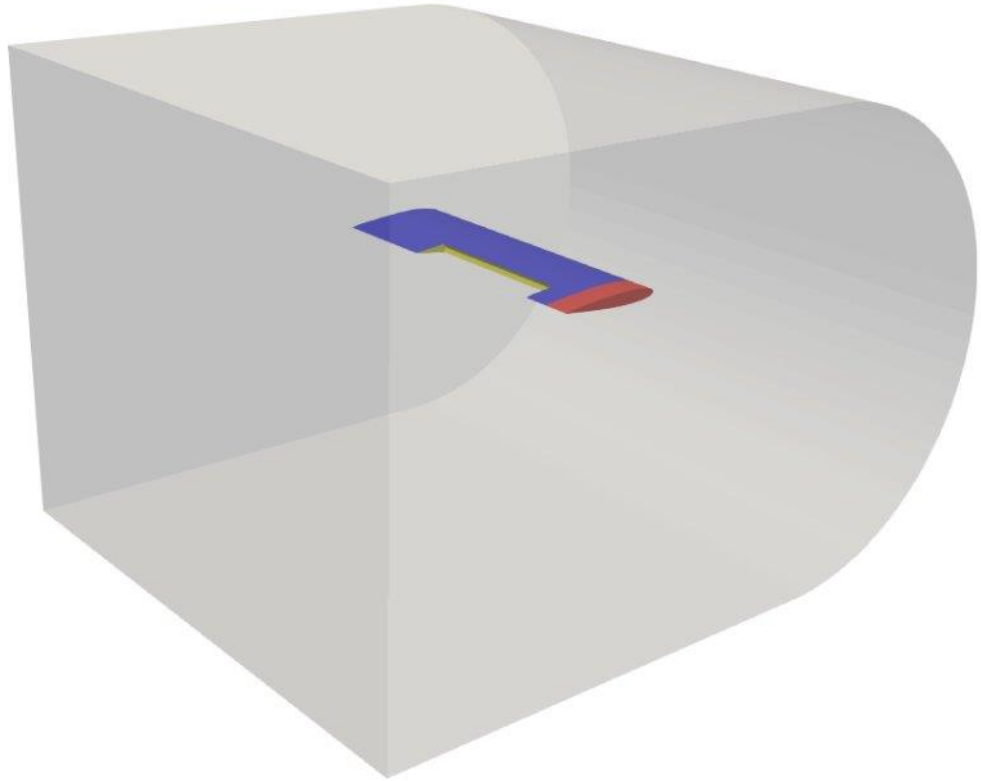


2D grid of MegAWES Mrev-v2 airfoil

Examples



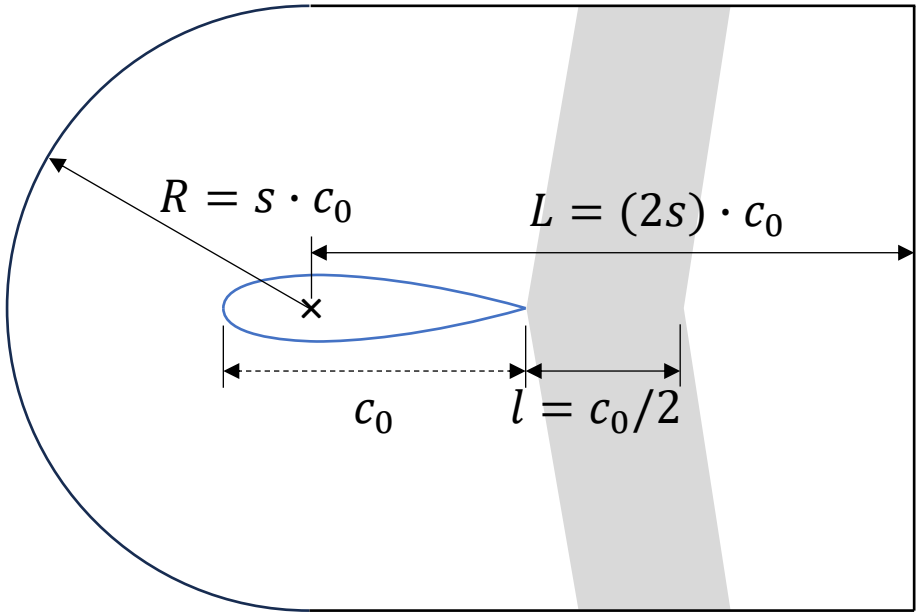
Half-wing with tip flow extension



Half-wing with aileron cut-off

Mesh structure

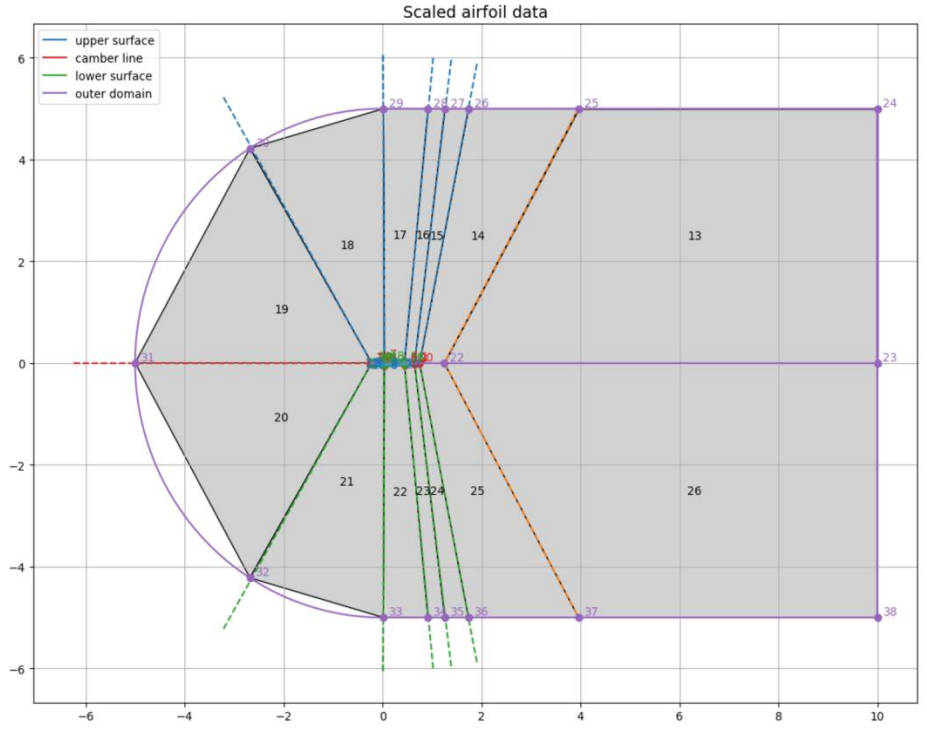
- **Structured C-type** grid (origin at COG)
- Default dimension **extents**:
  - C-shape of radius $R = s \cdot c_0$
  - Refinement zone $l = c_0/2$
  - Wake zone of length $L = (2s) \cdot c_0$
- Domain **scaled** by value s



Refinement zone is not scaled with $\boldsymbol{s}$

Mesh structure (2D layer structure)

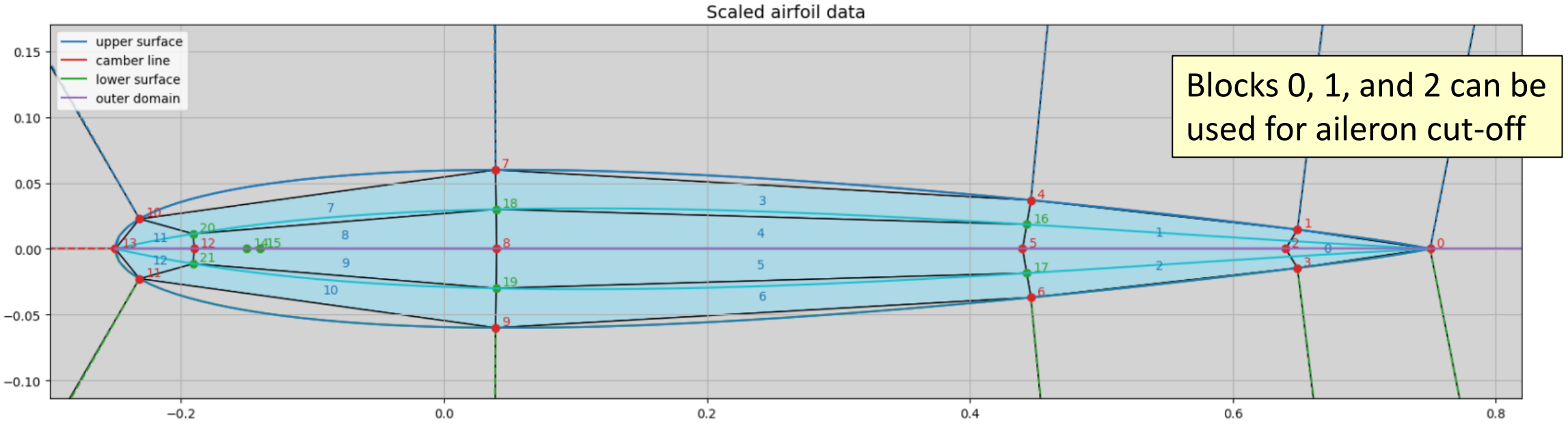
- **Inside** and **outside** of airfoil is meshed
- Mesh structure controlled by
  - 13 user-defined control points on airfoil
  - 8 computed control points on airfoil
  - 17 user-defined control points in domain
  - 13 interior blocks and 14 exterior blocks



Structure control (airfoil **inside**)

- 14 user-defined control points (0-13) on airfoil
- 8 interpolated control points (14-21) on airfoil

} 13 interior blocks (0-12)



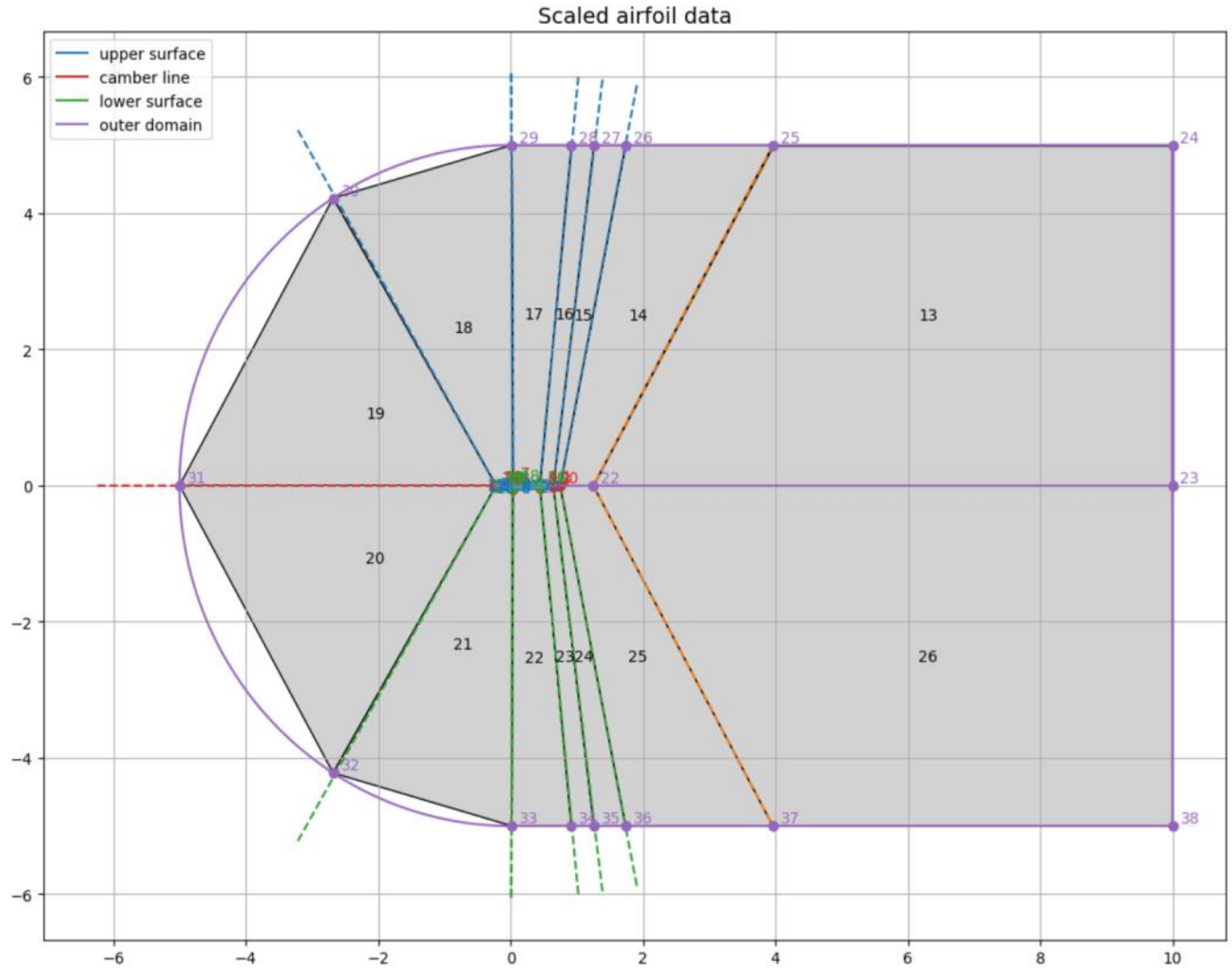
Structure control (airfoil **outside**)

– 17 user-defined control

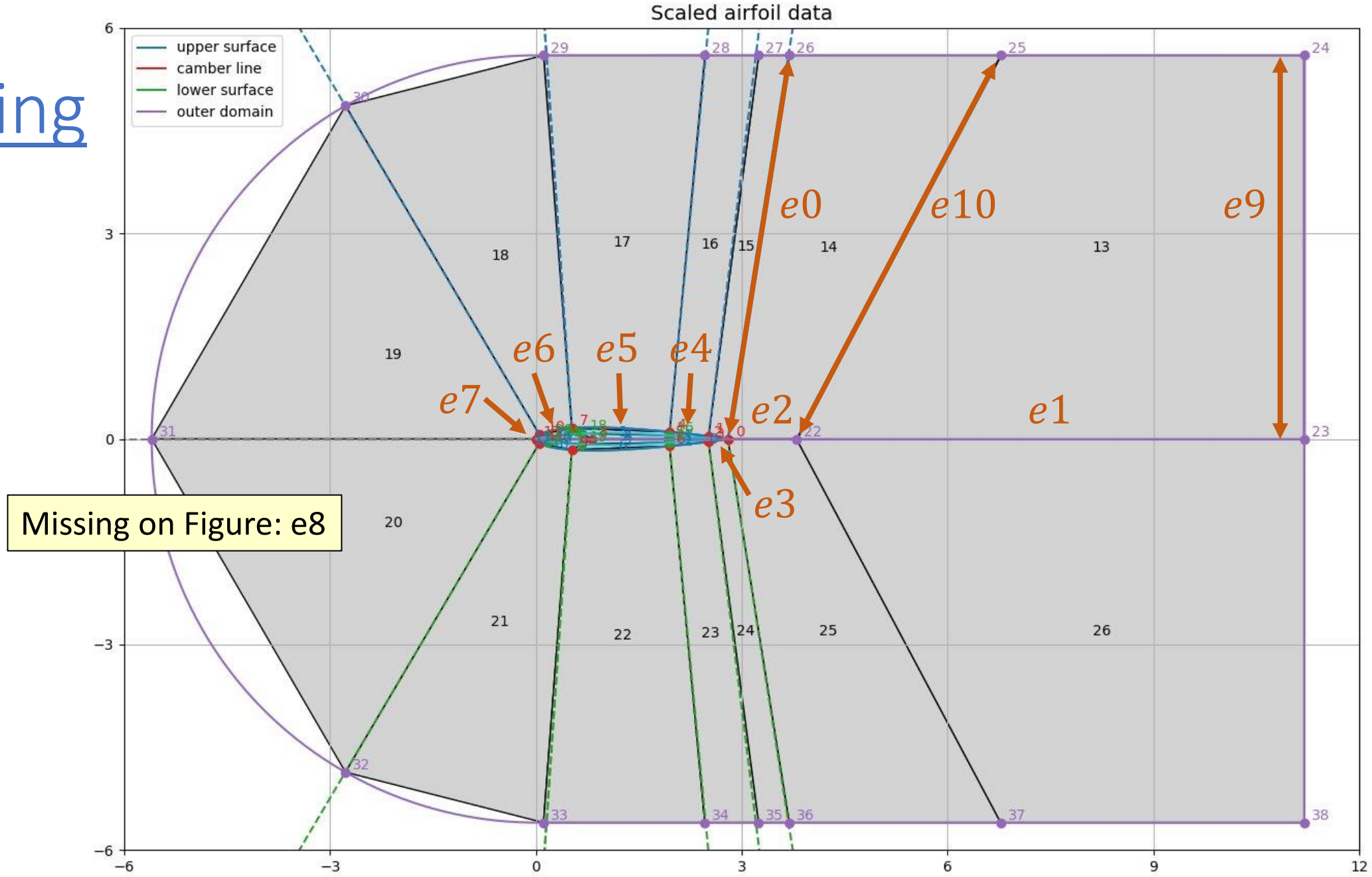
points (22-38) in domain

⟶ 14 exterior blocks (13-26)

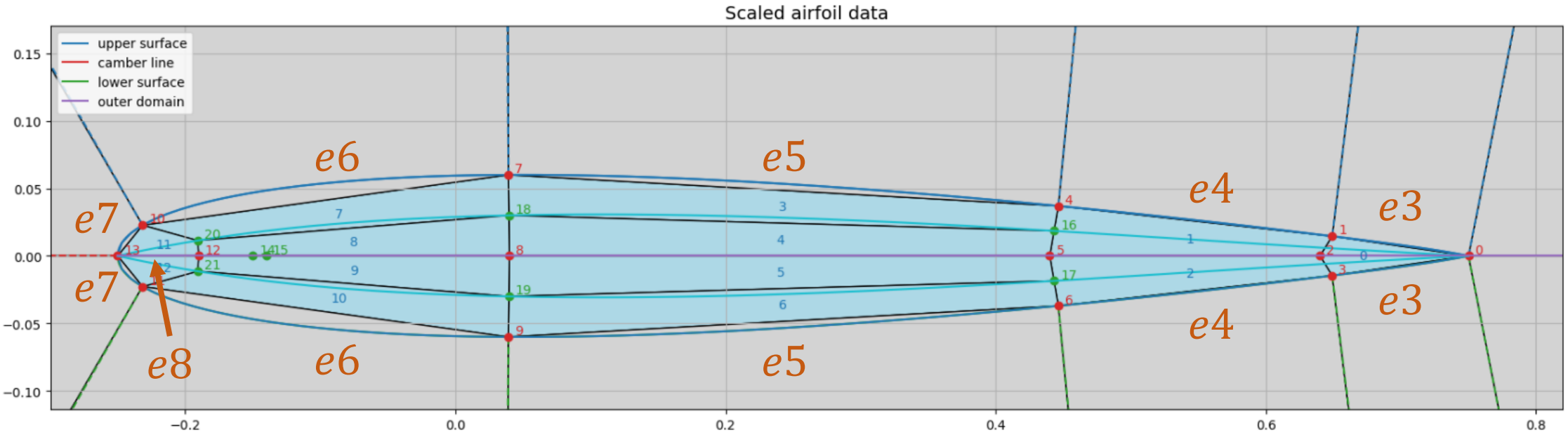
Default values of control points 0-38 are provided for different C-mesh radii for both NACA0012 and Mrev-v2 airfoils



Grading


Scaled airfoil data
upper surface
camber line
lower surface
outer domain
e0
e1
e2
e3
e4
e5
e6
e7
e9
e10
Missing on Figure: e8

Grading



To ensure a smooth interface between blocks 1 and blocks 3 and 4, respectively 2 with 5 and 6, the resolution of edge e8 should be half of edge e3. Ideally, their gradings should also match.

The utility “checkMesh” outputs the “wallDistance”, ie. the distance from the wall to the cell center(!), hence “wallDistance”= $\Delta y/2$

Default meshes

See file: /meshGeneration/default_files/default_cmesh_parameters.py

– Meshes generated for $Re_c = 6\text{M}$

– Tuned for OpenFOAM requirements

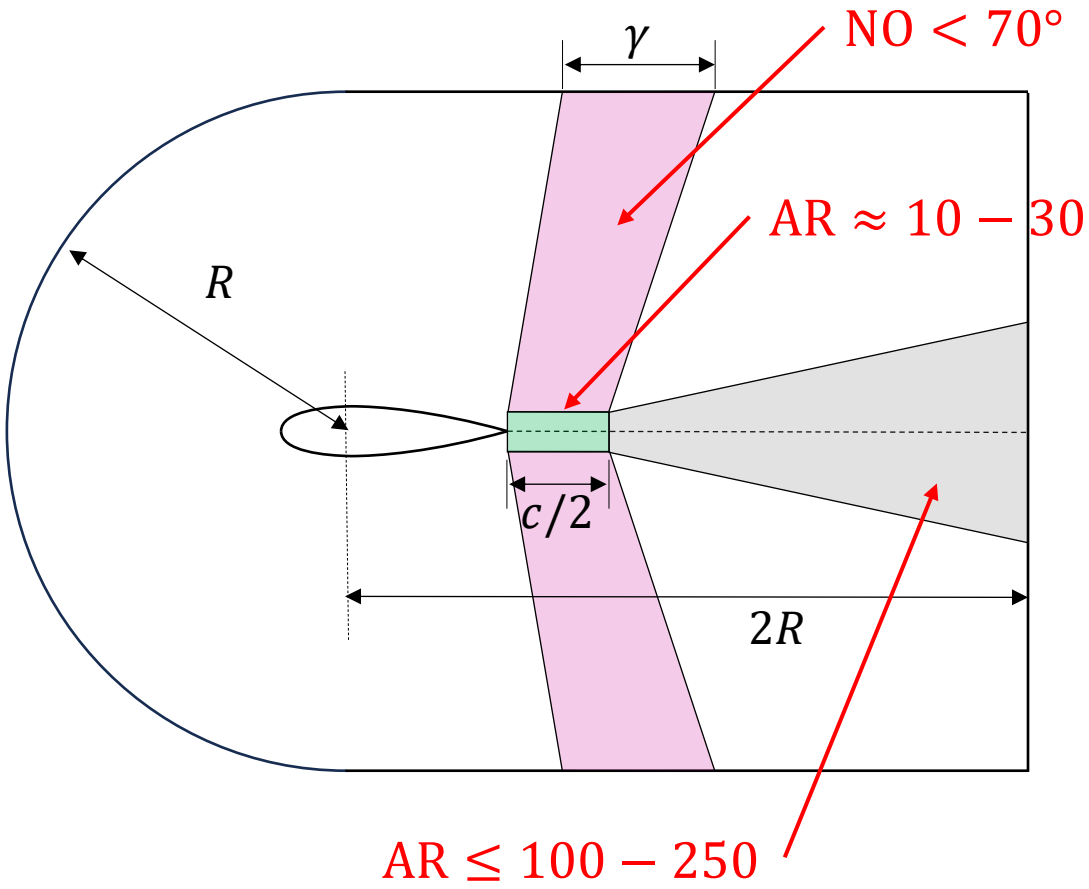
  – Non-orthogonality, Aspect ratio…

– Refinement for $y^+ - AR$ combinations

  – FINE: $y^+ \approx 1, \Delta y \approx 1 \cdot 10^{-5}$

  – MEDIUM: $y^+ \approx 50, \Delta y \approx 5 \cdot 10^{-4}$

  – COARSE: $y^+ \approx 500, \Delta y \approx 5 \cdot 10^{-3}$



Mesh creation using the class “Mesh”

See files: /examples/generate_mesh.py and /meshGeneration/mesh_class.py

```
# ---------------------- Create OpenFOAM mesh ---------------------- #
'''
Create OpenFOAM mesh
'''

# Create mesh
mesh = Mesh(casename, files, params, flags)

# Build mesh
mesh.initialize_mesh()
mesh.generate_default()
mesh.generate_vertices()
mesh.generate_blocks()
mesh.generate_edges()
mesh.generate_interfaces()
mesh.generate_boundaries()
if PLOT:
    plt.show()
```

Step-by-step mesh creation
Easy for debugging

```
# Initialize mesh
# Generate mesh.default
# Generate mesh.vertices
# Generate mesh.blocks
# Generate mesh.edges
# Generate mesh.interfaces
# Generate mesh.boundaries
```

Default edge meshing

If no default grading parameters are available, the parameters of the COARSE mesh of the NACA0012 airfoil with $R = 10c$ are chosen

See file: /meshGeneration/default_files/default_cmesh_parameters.py

```
# ------------------------------------------------#
# NACA0012 airfoil
if (airfoil_name == "NACA0012"):

    # C-mesh R=5c
    if (mesh_scale == 5):

        # Fine level
        if (mesh_level == "FINE"):

            # C-mesh with 005c (gamma_c0 = 2.3, A = 1000, B = 50)
            n0 = 465
            g0 = 9.8e3
            g9 = g0/1000.
            g10= g0/50.

            # n0 n1 n2 n3 n4 n5 n6 n7 n8 n9 n10
            edge_resolution = [n0] + [64, 256]  + [96, 32, 40, 64, 64]        + [8]  + [n0] + [n0]
            edge_grading1 =   [g0] + [70., 70.] + [30., 2.5, 1.1, 0.125, 0.1] + [1.] + [g9] + [g10]
            edge_grading2 =   [g0] + [15., 3.]  + [1., 4., 1., 5., 4.]        + [1.] + [g9] + [g10]
```

Airfoil name from [“NACA0012”, “Mrev-v2”]

Mesh scale R/c from [20,10, 5, 3, 1.5, 0.75]

Refinement level from [“FINE”, “MEDIUM”, “COARSE”]

Radial grading params

Wake grading params

Airfoil grading params

Mesh generation with blockMesh

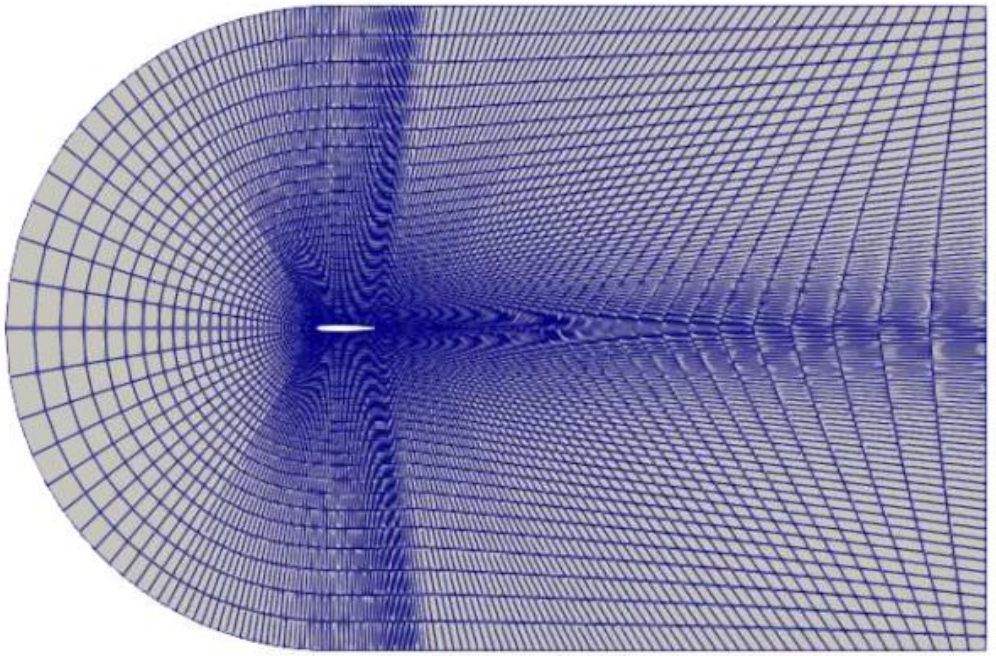
See file: /openFoamMesh/system/blockMeshDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2112                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

scale   1;

vertices
(
    #include "blockMesh_lists/list_vertices_slice0"
    #include "blockMesh_lists/list_vertices_slice1"
    #include "blockMesh_lists/list_vertices_slice2"
    #include "blockMesh_lists/list_vertices_slice3"
    #include "blockMesh_lists/list_vertices_slice4"
    #include "blockMesh_lists/list_vertices_slice5"
);

blocks
(
    #include "blockMesh_lists/list_blocks_layer0"
    #include "blockMesh_lists/list_blocks_layer1"
    #include "blockMesh_lists/list_blocks_layer2"
    #include "blockMesh_lists/list_blocks_layer3"
    #include "blockMesh_lists/list_blocks_layer4"
);
```

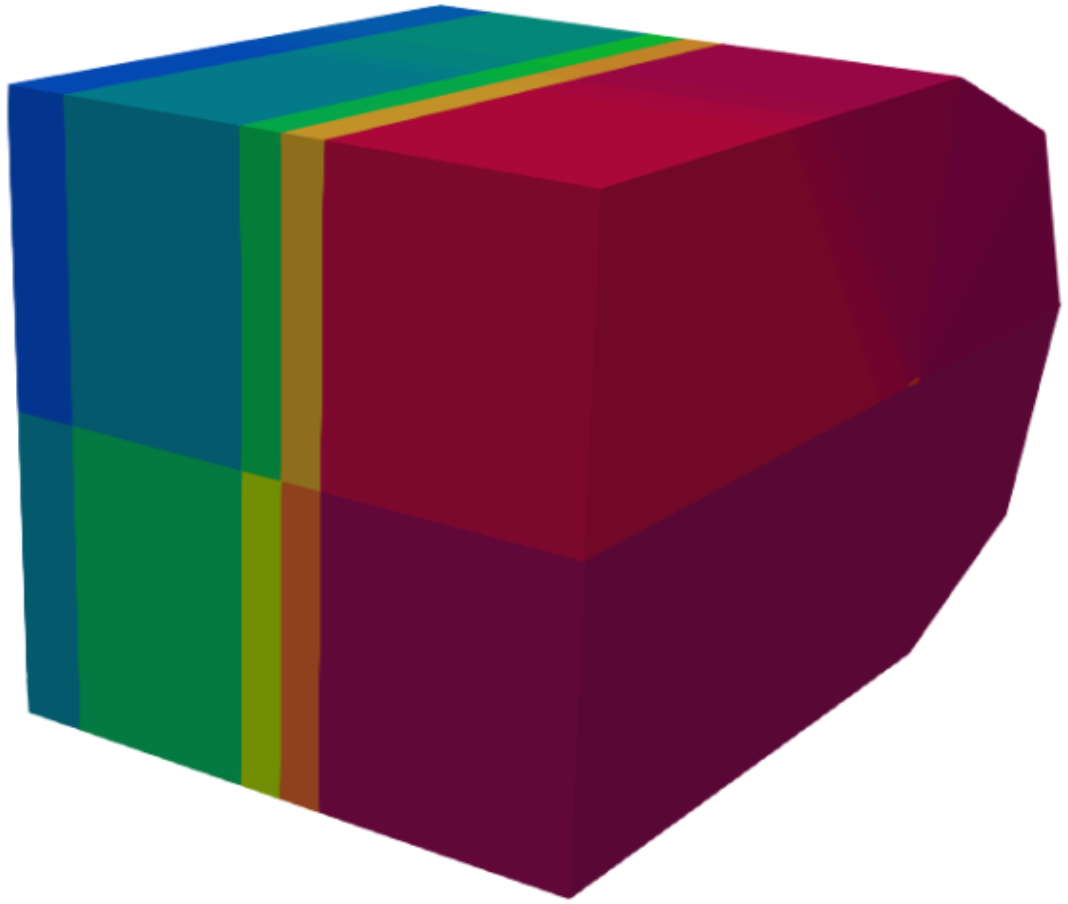
Mesh with **n** slices and **n-1** layers

```
edges
(
    #include "blockMesh_lists/list_edges_slice0"
    #include "blockMesh_lists/list_edges_slice1"
    #include "blockMesh_lists/list_edges_slice2"
    #include "blockMesh_lists/list_edges_slice3"
    #include "blockMesh_lists/list_edges_slice4"
    #include "blockMesh_lists/list_edges_slice5"
);

boundary
(
    #include "blockMesh_lists/list_boundaries_inlet"
    #include "blockMesh_lists/list_boundaries_outlet"
    #include "blockMesh_lists/list_boundaries_wingTip"
    #include "blockMesh_lists/list_boundaries_wing"
    #include "blockMesh_lists/list_boundaries_symRoot"
    #include "blockMesh_lists/list_boundaries_symTip"
    // #include "blockMesh_lists/list_boundaries_frontAndBack"
    #include "blockMesh_lists/list_boundaries_leftInterface"
    #include "blockMesh_lists/list_boundaries_rightInterface"
);

mergePatchPairs
(
    (leftInterface rightInterface)
);

// ************************************************************************* //
```

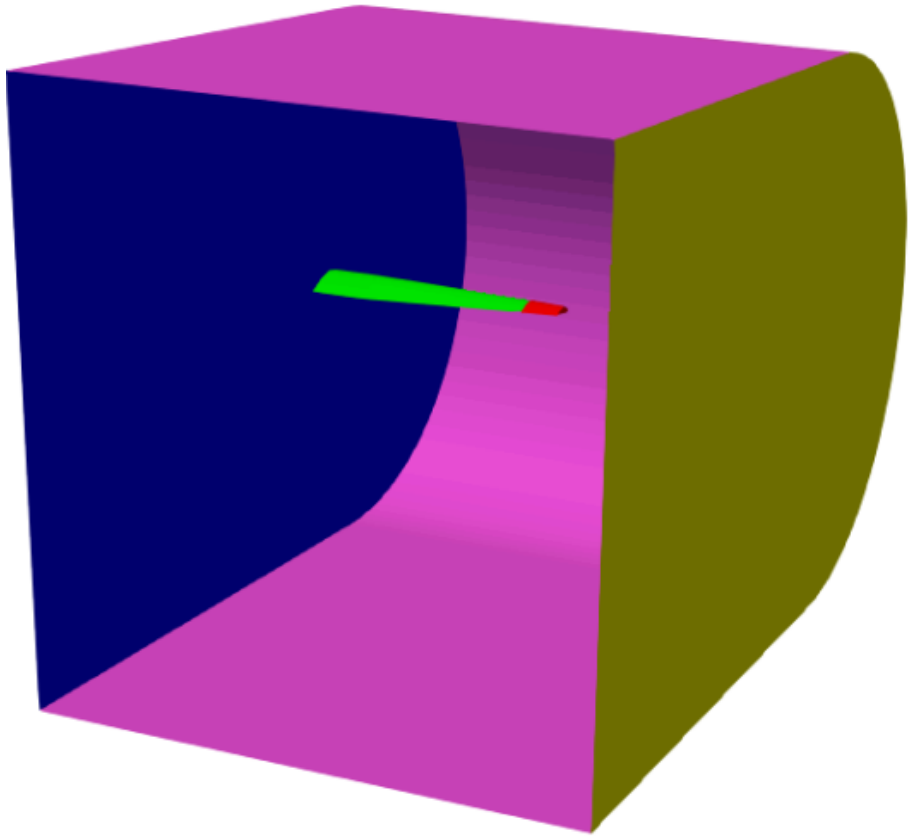
Complex block topologies: 2D airfoil



Complex block topologies: 3D wing

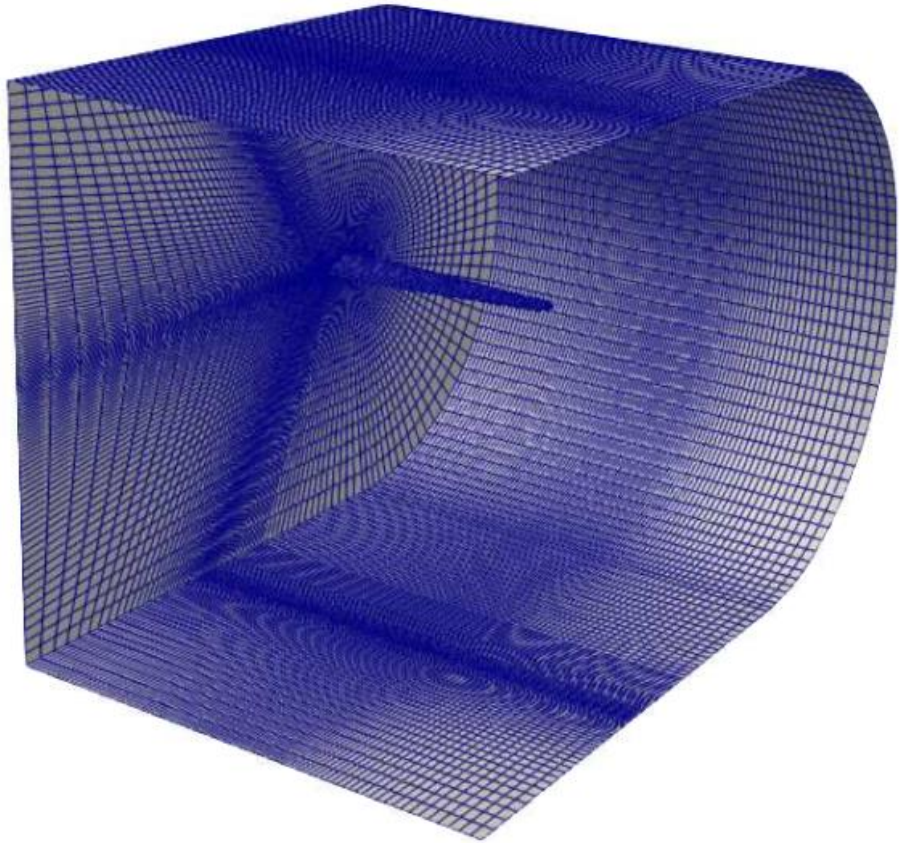


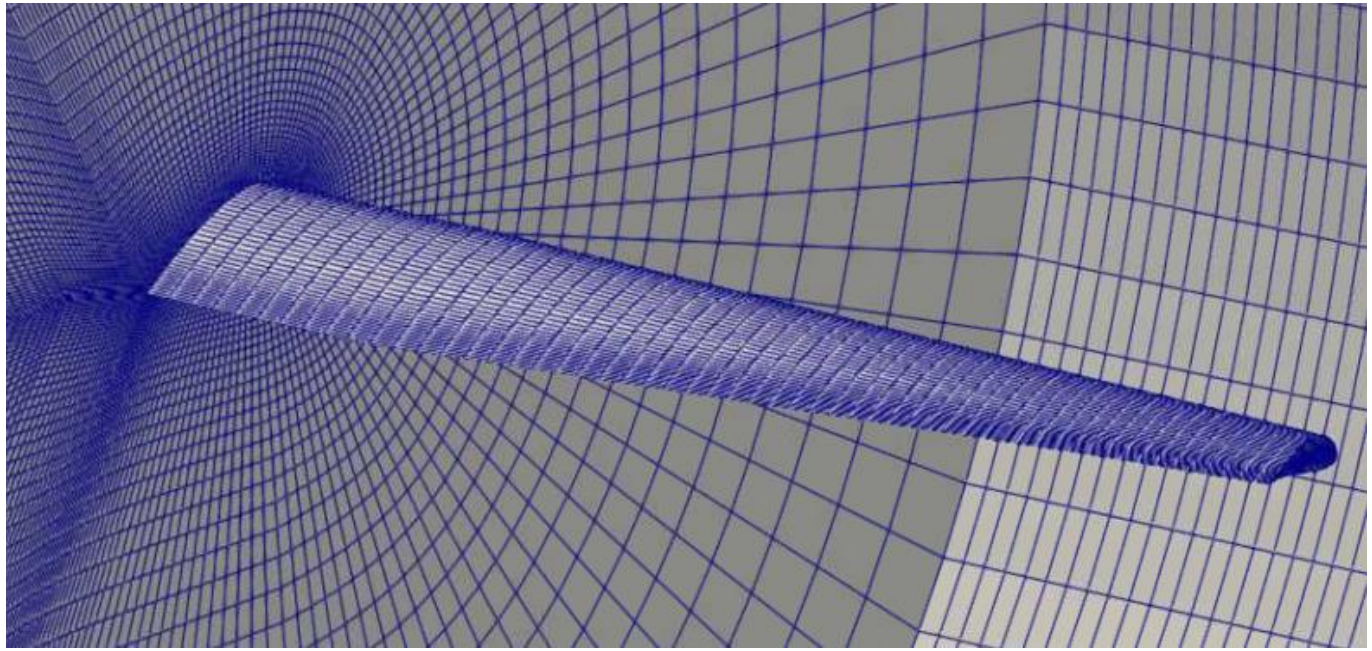
Block structure



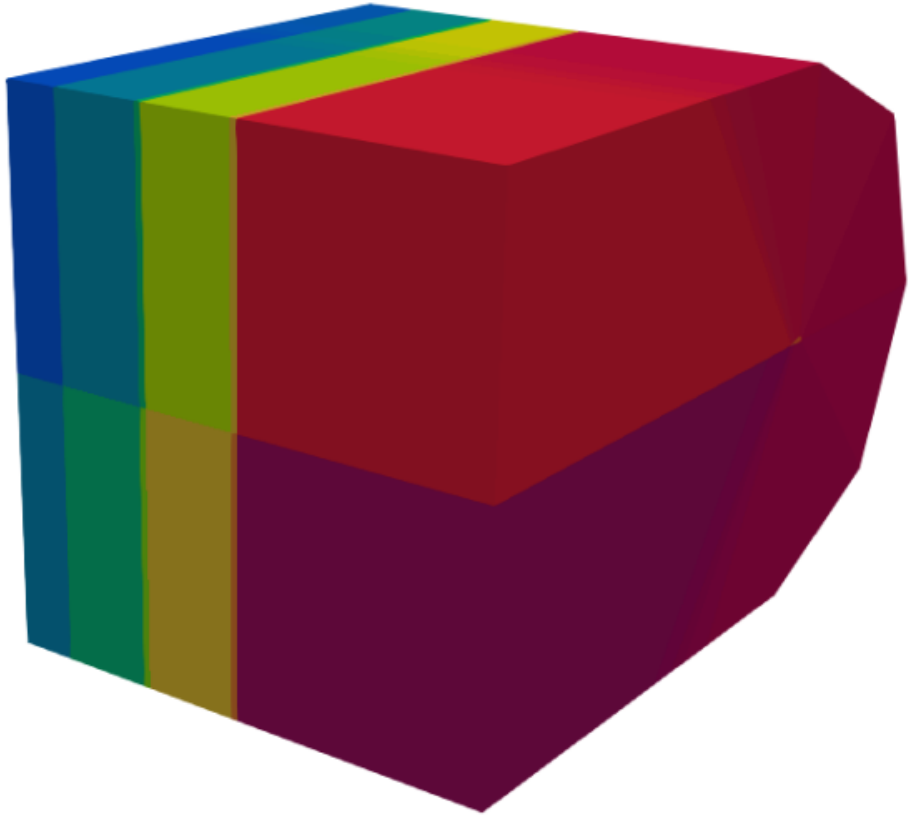
Surface boundaries

Complex block topologies: 3D wing

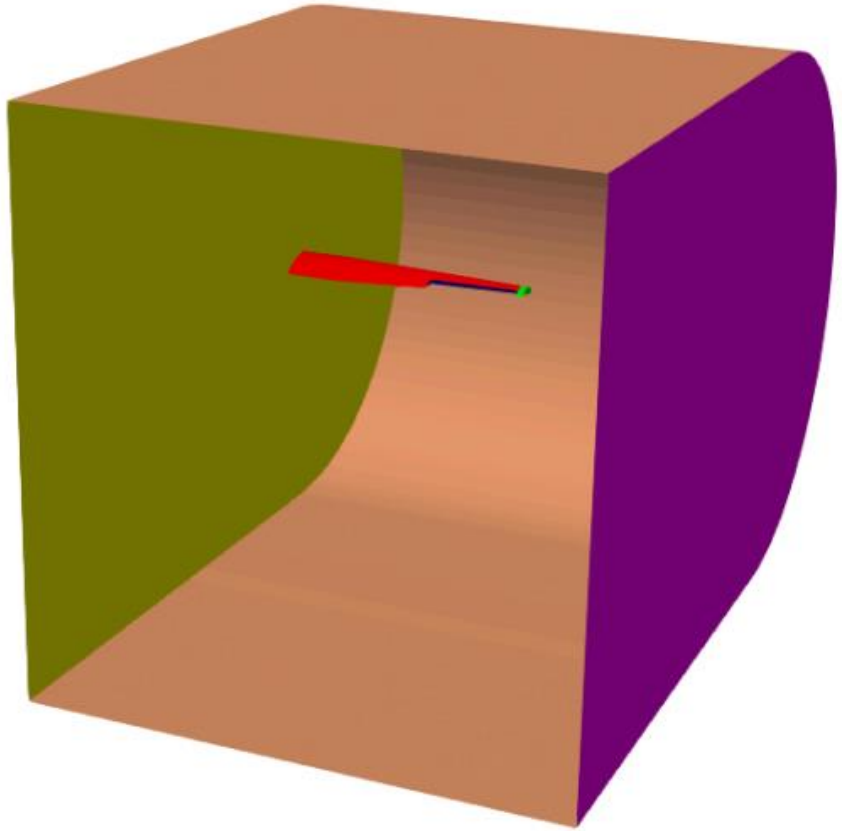




Complex block topologies: 3D wing + aileron

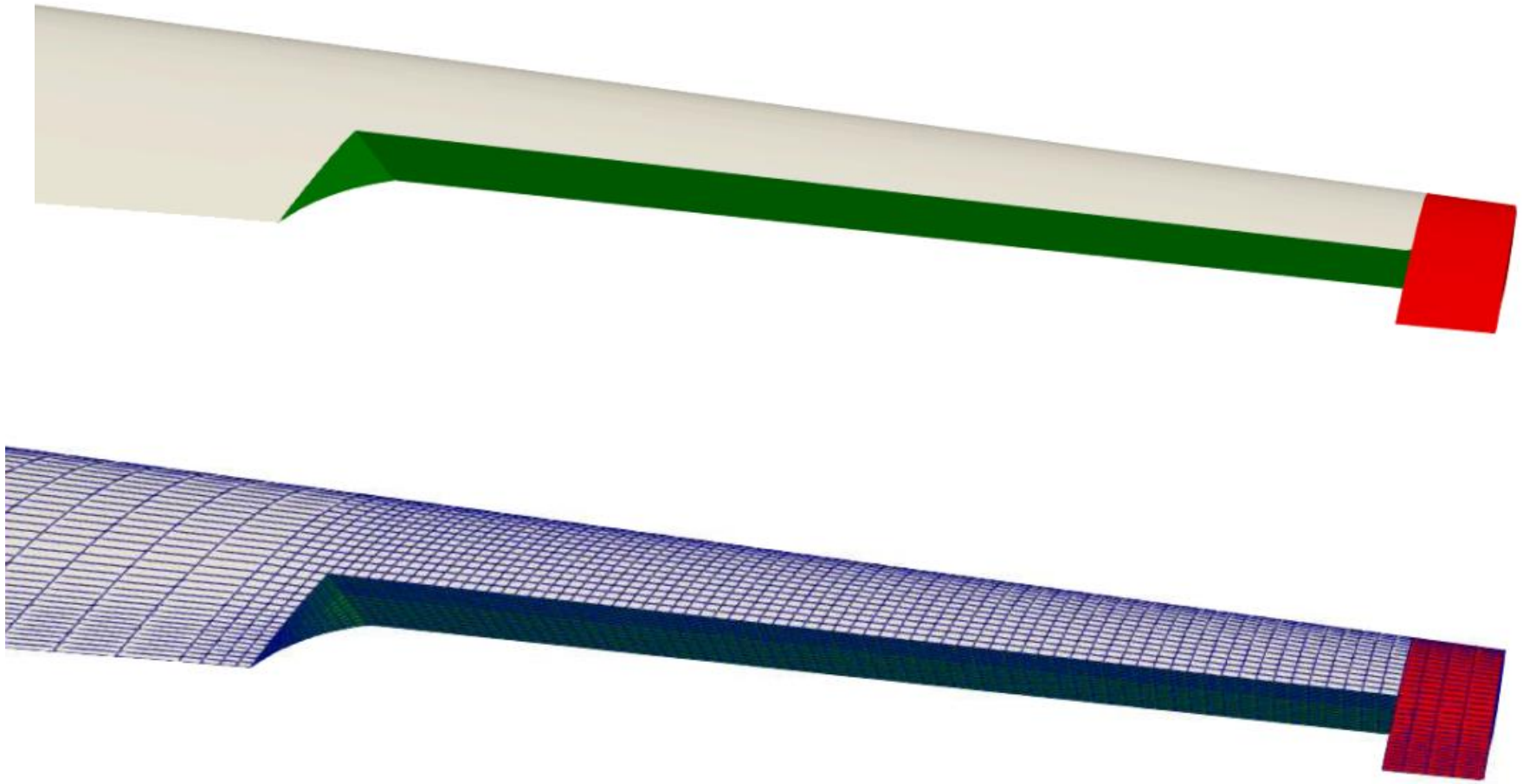


Block structure



Surface boundaries

Complex block topologies: 3D wing + aileron



Workflow on the local cluster

1. Create mesh files with PyCharm (ml purge; ml PyCharm; pycharm.sh)
   - Set up the mesh parameters in Python in *examples/mesh_generate*.py*
   - Create mesh files with Python by running *mesh_generate*.py*

2. Generate mesh with OpenFOAM (*ml purge; ml OpenFOAM/v2112-foss-2021b; source $FOAM_BASH*)
   - Copy lists of vertices, blocks, edges to openFoamMesh/system/BMLists
   - Run blockMesh-procedure with scripts ./Allclean and *./Allrun.pre*
   - You can run the commands in Allrun.pre manually if you prefer
   - Visualize the mesh in ParaView by loading file open.foam and blockFaces.vtp

3. Transform the OpenFOAM mesh to Fluent
   - Run command *foamMeshToFluent* within the folder *openFoamMesh*Load Fluent (*ml purge; ml ANSYS_CFD/2021R1*)
   - Create new folder from *fluentInterface* and adapt the boundary conditions