CFD Fundamentals (5cfu) – Exercise 2

GROUP 5:

- MATTEO CIGADA (10767277, 270632)
- FEDERICO D’AGOSTINI (10788635, 281037)
- SIMONE D’INCÀ(10837331, 287102)

Contents

Introduction 3

Simulations with wall functions 3

k-Epsilon 3

Boundary conditions 3

Setup of the simulation 5

Post Processing 10

Spalart Allmaras 13

Boundary conditions 13

Setup of the simulation 13

Post Processing 16

Simulations without wall functions (SA) 19

Boundary conditions 19

Setup of the simulation 19

Post Processing 21

Analysis and comparison 24

KPI 24

Flow over the profile 24

Velocity distribution 26

Introduction

2D incompressible flow around a NACA 4412 airfoil with an angle of attack of 0◦ was analyzed. The fluid flow is ambient air (ν = 15 × 10–6 m2/s, ρ= 1.2 kg/m3). The following simulations have been performed:

- Case A: k-Epsilon
- Case B: SA with wall functions
- Case C: SA without wall functions

The simulations were carried out using OpenFOAM 11, with simpleFoam as the solver. ParaView was used for post-processing.

Simulations with wall functions

k-Epsilon

Boundary conditions

The boundary conditions have been imposed according to the rules that guarantee the Well-Posedness of the problem

- **Pressure $\boldsymbol{p}$** $[m^2/s^2]$:

Inlet	zeroGradient
Outlet	fixedValue (83333.33)
Profile	zeroGradient
Front and back	empty
Top and bottom	symmetry

  - The value at the outlet was provided by the assignment

- **Velocity $\boldsymbol{U}$** $[m/s]$:

Inlet	fixedValue (20)
Outlet	zeroGradient
Profile	noSlip
Front and back	empty
Top and bottom	symmetry

- The value at the inlet was provided by the assignment

- **Turbulent kinetic energy κ** $[m^2/s^2]$:

Inlet	turbulentIntensityKineticEnergyInlet
Outlet	zeroGradient
Profile	kqRWallFunction
Front and back	empty
Top and bottom	symmetry

  - The value at the inlet is computed directly by OpenFoam starting from the value of the turbulence intensity ($I = \frac{u'}{U} = 1\%$) provided by the assignment
  - On the profile, the use of wall functions is imposed

- **Dissipation rate ε** $[m^2/s^3]$:

Inlet	turbulentMixingLengthDissipationRateInlet
Outlet	zeroGradient
Profile	epsilonWallFunction
Front and back	empty
Top and bottom	symmetry

  - The value at the inlet is computed directly by OpenFoam starting from the value of the mixing length ($l_T = \frac{1}{10}c = 0.01m$) provided by the assignment
  - On the profile, the use of wall functions is imposed

- **Turbulent kinematic viscosity (*nut*) ν_T** $[m^2/s]$:

Inlet	calculated
Outlet	zeroGradient
Profile	nutkWallFunction;
Front and back	empty
Top and bottom	symmetry

- The value at the inlet is computed directly by OpenFoam during the simulation itself, it is not an imposed value
- On the profile, the use of wall functions is imposed

Setup of the simulation

Mesh

The mesh used is C shaped and the domain has been divided into 5 blocks (Figure 1). The mesh boundaries were placed sufficiently distant from the airfoil in every direction to make sure the flow is undisturbed at the domain limits. Since the simulation is 2D, only 1 cell has been placed along the z direction.

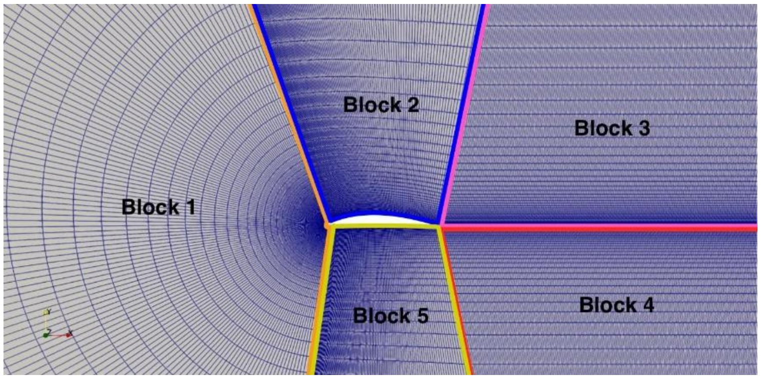
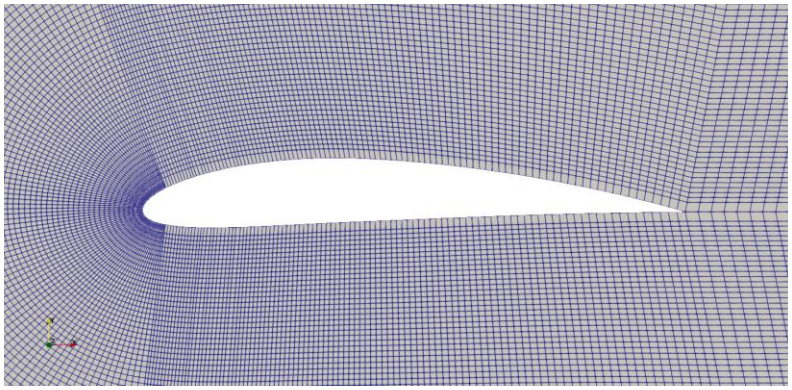


Figure 1

The three driving parameters considered when designing the mesh are the non-orthogonality on the airfoil, the y+ of the first cell and the aspect ratio.

- The non orthogonality on the profile is controlled by adjusting the position of the vertices that define the blocks.
- The y+ of the first cell around the airfoil is controlled by tuning the percentage of cells close to the airfoil and modifying the parameters in the simpleGrading. Since wall functions were used in this simulation, the first cell has been defined to make y+ greater than 30.
  (y+ estimators available on internet have been used as starting point)
- The aspect ratio is controlled by adjusting the number of cells along the longitudinal direction, the goal is to keep the aspect ratio as low as possible. High values of aspect ratio lead to an increased number of iterations to reach convergence.

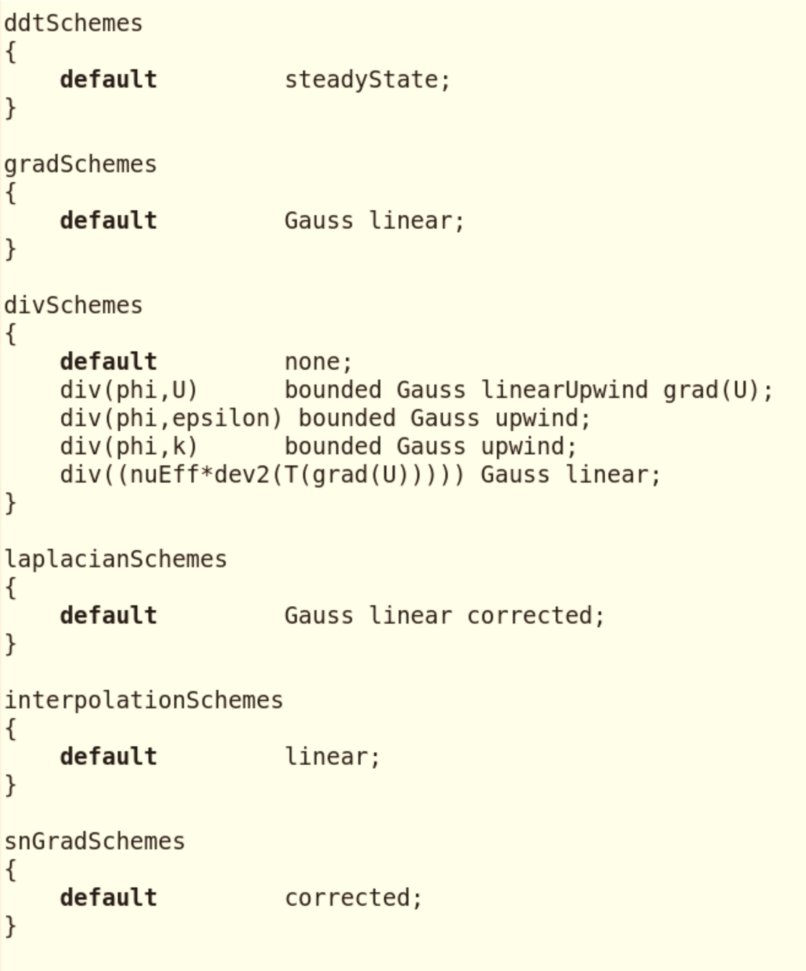
The mesh reported here is the most refined one and, along with it, also the checkMesh results are presented.



Number of cells	176625
Skewness (max)	1,27
Non orthogonality (max/mean)	22,41° / 6,79°
Aspect ratio	9,50

Figure 2

Domain discretization schemes (fvSchemes)



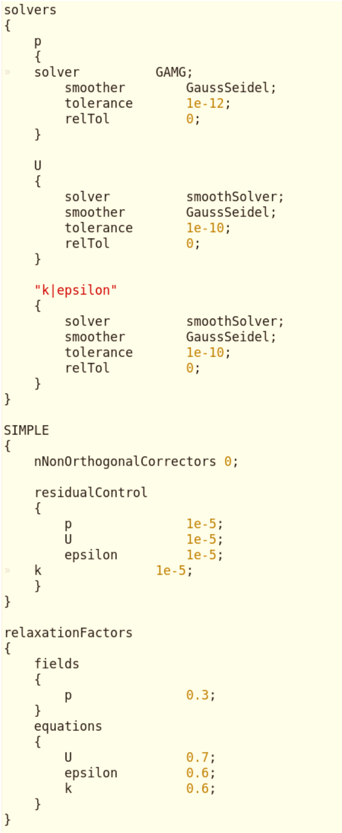
```
ddtSchemes
{
    default         steadyState;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default         none;
    div(phi,U)      bounded Gauss linearUpwind grad(U);
    div(phi,epsilon) bounded Gauss upwind;
    div(phi,k)      bounded Gauss upwind;
    div((nuEff*dev2(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}
```

Figure 3

- **Time marching term**: given that the simulation operates under steady-state conditions, the time derivative terms in the equation are set to zero.
- **Gradient term**: a second order accuracy scheme is being used. Since no unnatural behaviours/oscillations were observed in the results, there was no need to use a gradient limiter

- **Convective flux term for U**: secondo order scheme for convective flux of U were employed to have good convergence speed:
- **Convective flux terms for ε and k**: first-order accuracy schemes were employed because, when combined with the second-order scheme for the convective flux of U, they reduced the number of iterations required for convergence. Higher-order schemes were tested, they often led to diverging simulations or an unacceptably high number of iterations. This behavior might be attributed to oscillations that can arise with second-order schemes.
- **Diffusive flux term**: a second order accuracy scheme is being used. The mesh has low non orthogonality, however, using a scheme designed for perfectly orthogonal meshes didn't seem a good option. For this reason, an explicit correction was introduced; since it didn't introduce unboundedness in the results, there was no need to limit this correction
- The **interpolation scheme** is set to linear (default option) and for the same reasons described for the diffusive term, an explicit correction was introduced also for the surface normal gradient scheme

Resolution schemes (fvSolution)



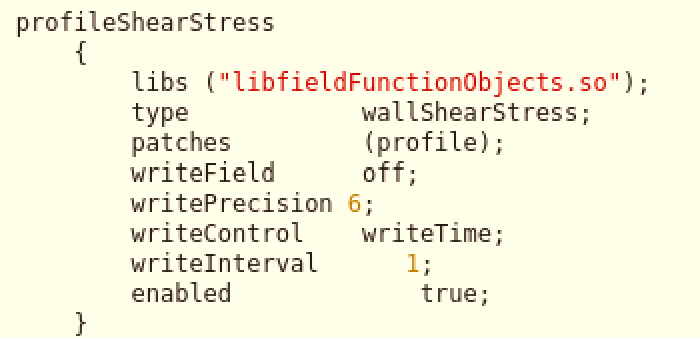
```
solvers
{
    p
    {
    solver          GAMG;
        smoother        GaussSeidel;
        tolerance       1e-12;
        relTol          0;
    }

    U
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        tolerance       1e-10;
        relTol          0;
    }

    "k|epsilon"
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        tolerance       1e-10;
        relTol          0;
    }
}

SIMPLE
{
    nNonOrthogonalCorrectors 0;

    residualControl
    {
        p               1e-5;
        U               1e-5;
        epsilon         1e-5;
    k               1e-5;
    }
}

relaxationFactors
{
    fields
    {
        p               0.3;
    }
    equations
    {
        U               0.7;
        epsilon         0.6;
        k               0.6;
    }
}
```

Figure 4

- A geometric algebraic multi-grid method has been used for the pressure equation-related resolution matrix with Gauss Seidel as smoother. Since the pressure correction step equation is purely diffusive, the resolution matrix is expected to be symmetric; for this reason, a "preconditioned conjugate gradient" method with incomplete Cholesky preconditioner is expected to be a valid alternative. This option has been tested, and no significant differences have been observed neither in the results or in the number of iterations needed for convergence.
- A smooth solver is a commonly used option for the velocity equation-related resolution matrix. Gauss Seidel method is used because it is faster compared to other options like Jacobi.
- The number of non-orthogonal correctors has been set to 0 because, considering the low non-orthogonality, there is no need to introduce additional correction steps.
- The relaxation factors have been tuned to minimize oscillations in the residuals and to reduce as much as possible the number of iterations needed to reach convergence.
- Considering the turbulence model that is being used, an algorithm to solve the linear system associated with the $k-\varepsilon$ equations is needed. Preconditioned solver was used (Gauss Seidel).

Computation of the KPI

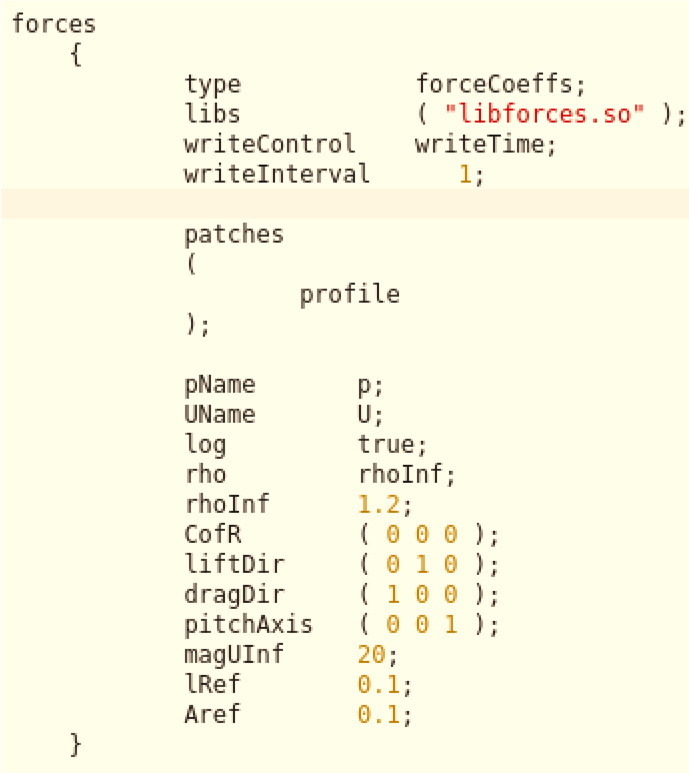
- The **shear stresses on the profile** are directly computed by OpenFoam through a function that provides at each time step saved the x,y,z components of the shear stress vector. Differently from the previous exercise, the output file is not directly read by a Matlab script but it is managed through Paraview and saved in a .csv file together with other useful data like pressure and velocity.



```
profileShearStress
    {
        libs ("libfieldFunctionObjects.so");
        type            wallShearStress;
        patches         (profile);
        writeField      off;
        writePrecision 6;
        writeControl    writeTime;
        writeInterval      1;
        enabled             true;
    }
```

Figure 5

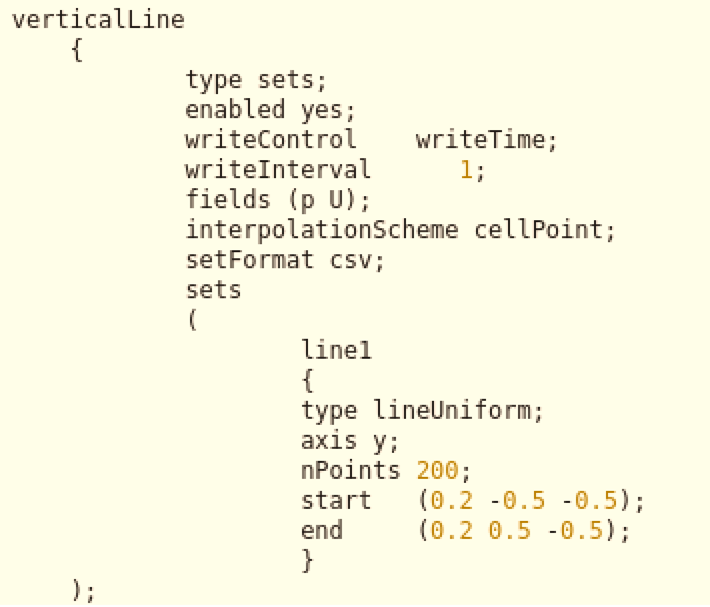
- **Lift and Drag Coefficient** are computed directly by OpenFoam by means of the following function, that directly computes Cl, Cd and Cm starting from assigned values of a reference length and area (reference area = chord*width of the mesh in z direction)



```
forces
    {
            type            forceCoeffs;
            libs            ( "libforces.so" );
            writeControl    writeTime;
            writeInterval      1;

            patches
            (
                    profile
            );

            pName       p;
            UName       U;
            log         true;
            rho         rhoInf;
            rhoInf      1.2;
            CofR        ( 0 0 0 );
            liftDir     ( 0 1 0 );
            dragDir     ( 1 0 0 );
            pitchAxis   ( 0 0 1 );
            magUInf     20;
            lRef        0.1;
            Aref        0.1;
    }
```

Figure 6

- **Velocity distribution one chord downstream of trailing edge** is computed through a multi-step procedure. The following function extrapolates pressure and velocity data from an a set of points along a line that passes through assigned starting and ending points. At the end of each simulation, the values of velocity along the line are read and plot by means of a Matlab script



```
verticalLine
    {
            type sets;
            enabled yes;
            writeControl    writeTime;
            writeInterval      1;
            fields (p U);
            interpolationScheme cellPoint;
            setFormat csv;
            sets
            (
                    line1
                    {
                    type lineUniform;
                    axis y;
                    nPoints 200;
                    start   (0.2 -0.5 -0.5);
                    end     (0.2 0.5 -0.5);
                    }
    );
```

Figure 7

- The **static and total pressure distribution** have been extracted from Paraview as anticipated. The static pressure and velocity magnitude values at various points along the profile were exported to a .csv file together with the wall shear stress magnitude; this file is subsequently read by a Matlab script.

Post Processing

Results

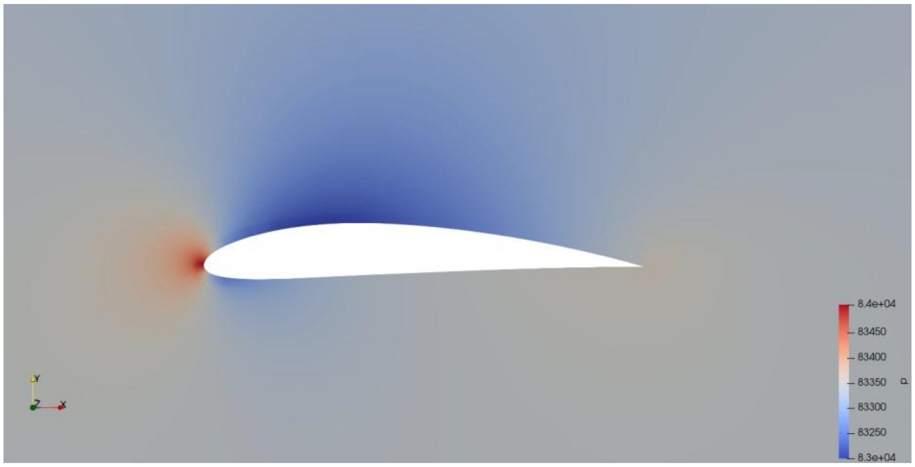


Figure 8

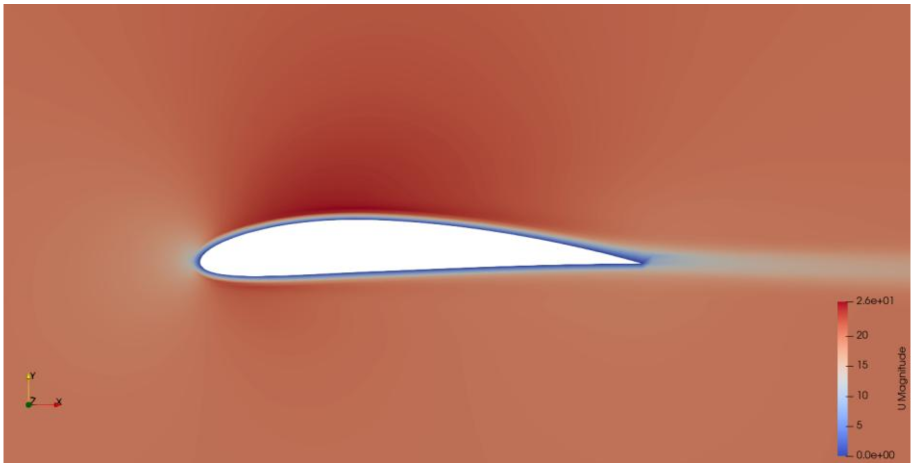


Figure 9

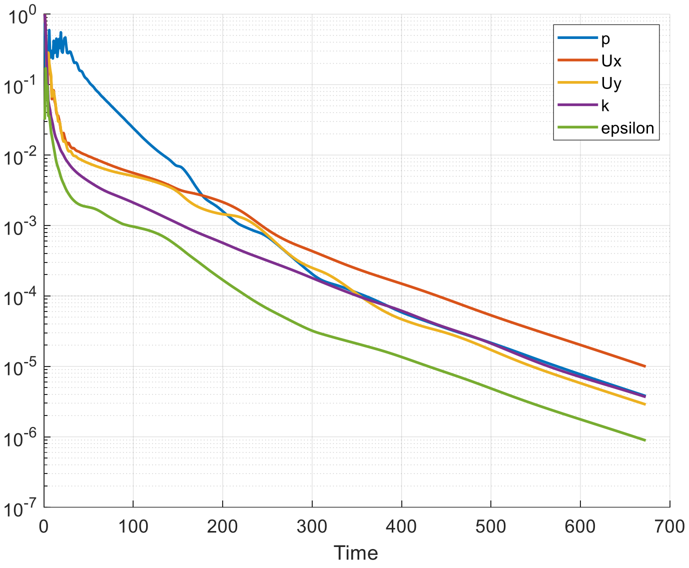


Figure 10

Convergence is considered reached when residuals are reduced by five order of magnitude.

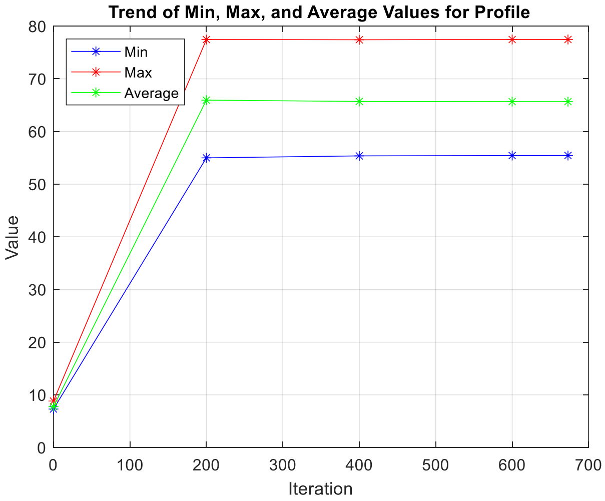


Figure 11

In this case, with wall function applied: $y^+ = 50 \div 500$

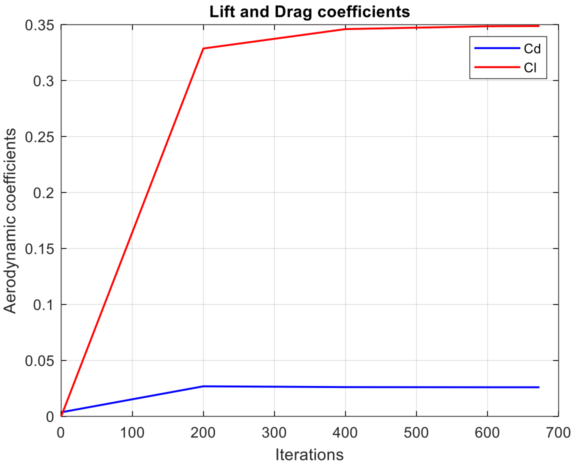


Figure 12

Convergence of the resolution is also guaranteed by the KPI convergence to a stable value.

Grid independence study

Mesh						**it**	**KPI**			
Iteration	sk	non-hort		aspect-ratio	n celle		**CD**	**CL**	delta_CD	delta_CL
		max	avg							
Mesh00	1,16	20,93	6,78	7,05	7800,00	150,00	**0,0180**	**0,3709**		
Mesh01	1,26	21,40	6,79	4,77	17550,00	185,00	**0,0216**	**0,3720**	-16,4%	-0,3%
Mesh02	1,26	21,66	6,79	4,34	35000,00	290,00	**0,0231**	**0,3592**	-6,7%	3,6%
Mesh03	1,26	21,93	6,79	6,43	78600,00	442,00	**0,0248**	**0,3549**	-6,8%	1,2%
Mesh04	1,27	22,21	6,79	9,50	176625,00	673,00	**0,0260**	**0,3488**	-4,5%	1,7%

Figure 13

Spalart Allmaras

Boundary conditions

The boundary conditions for pressure and velocity haven't changed, therefore, only the ones for the SA equation (*nuTilda*) are being discussed:

- Dynamic turbulent viscosity (*nuTilda*) μ_T $[m^2/s]$:

Inlet	fixedValue (0.00134)
Outlet	zeroGradient
Profile	fixedValue (0)
Front and back	empty
Top and bottom	symmetry

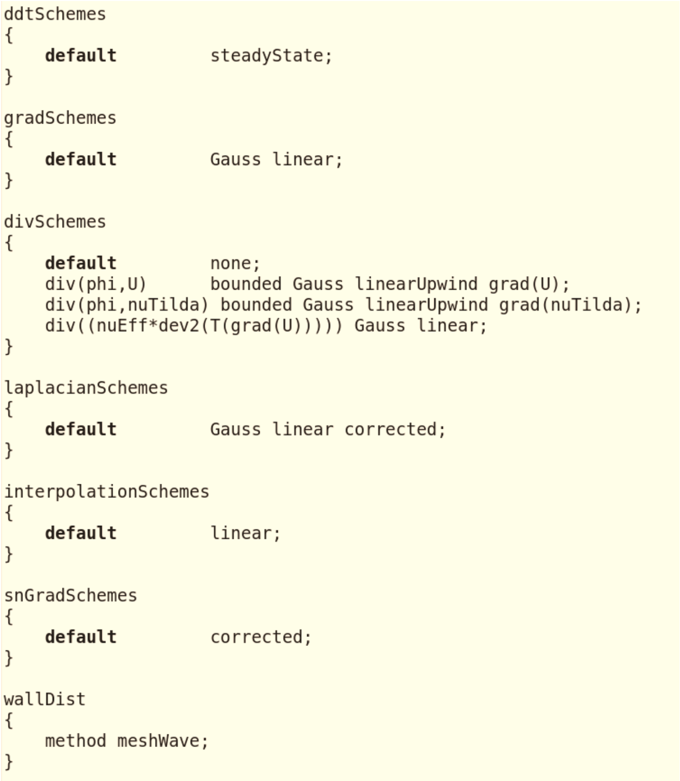
  - The value at the inlet has been assigned starting from previous estimations of the turbulent kinetic energy and the dissipation rate

Setup of the simulation

Mesh

The mesh used for this simulation is the same one used for the $\kappa - \varepsilon$ simulation. The reason for this is that wall functions are still used and the y+ that must be guaranteed is the same

Domain discretization schemes (fvSchemes)



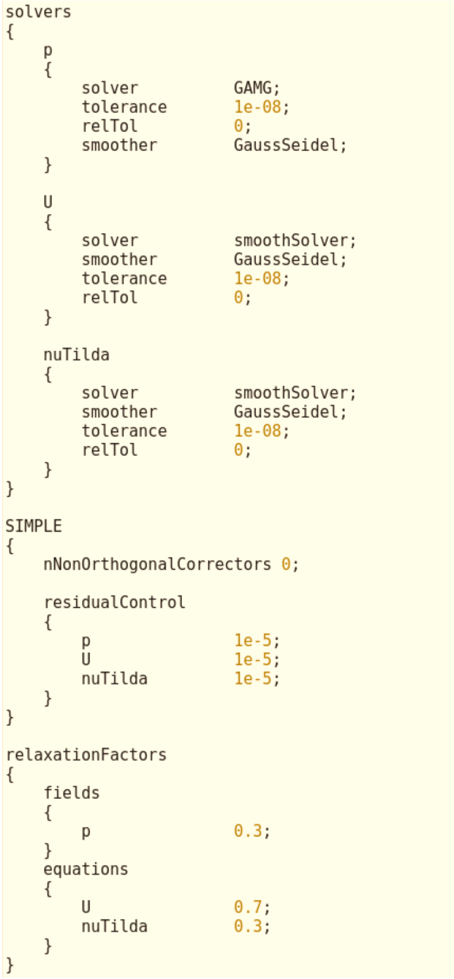
```
ddtSchemes
{
    default         steadyState;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default         none;
    div(phi,U)      bounded Gauss linearUpwind grad(U);
    div(phi,nuTilda) bounded Gauss linearUpwind grad(nuTilda);
    div((nuEff*dev2(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

wallDist
{
    method meshWave;
}
```

Figure 14

The only difference from the previous case is inside the Convective Flux section, where it is specified the scheme for $\tilde{\nu}$ for this model.

Also here, a second order scheme is used to achieve a good convergence speed.

Resolution schemes (fvSolution)



```
solvers
{
    p
    {
        solver          GAMG;
        tolerance       1e-08;
        relTol          0;
        smoother        GaussSeidel;
    }

    U
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        tolerance       1e-08;
        relTol          0;
    }

    nuTilda
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        tolerance       1e-08;
        relTol          0;
    }
}

SIMPLE
{
    nNonOrthogonalCorrectors 0;

    residualControl
    {
        p               1e-5;
        U               1e-5;
        nuTilda         1e-5;
    }
}

relaxationFactors
{
    fields
    {
        p               0.3;
    }
    equations
    {
        U               0.7;
        nuTilda         0.3;
    }
}
```

Figure 15

Also here, as for fvSchemes, the only difference from previous case is in the specification of the solver for $\tilde{\nu}$ quantity, where a smoothSolver with Gauss Seidle method was used.

Computation of the KPI

The KPIs that need to be computed haven’t changed, therefore the procedure followed is the same for previous case.

Post Processing

Results

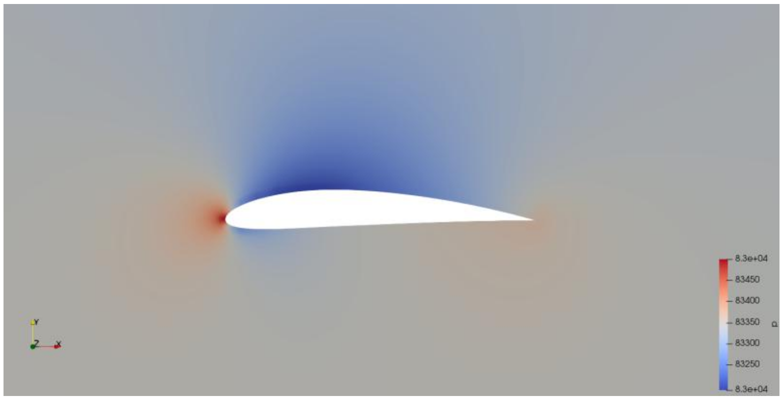


Figure 16

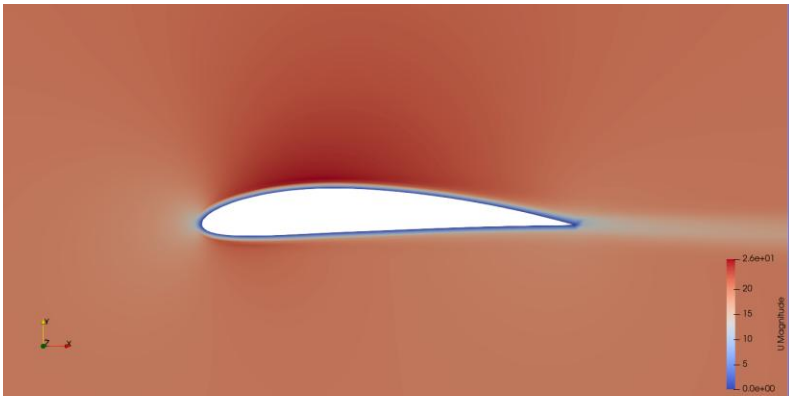


Figure 17

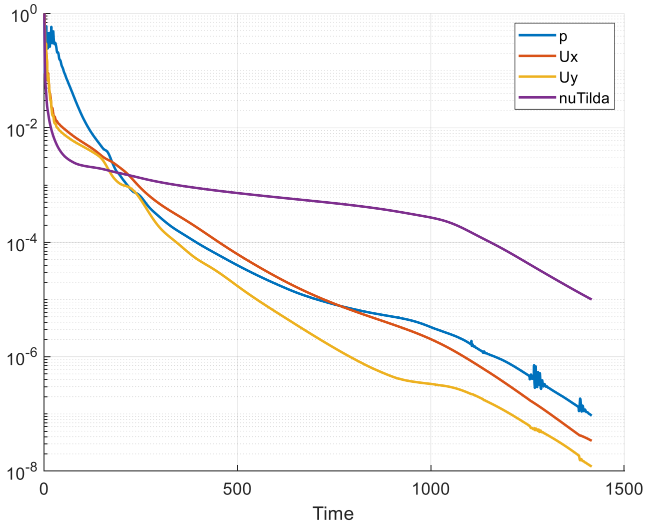


Figure 18

Convergence is considered reached when residuals are reduced by five order of magnitude.

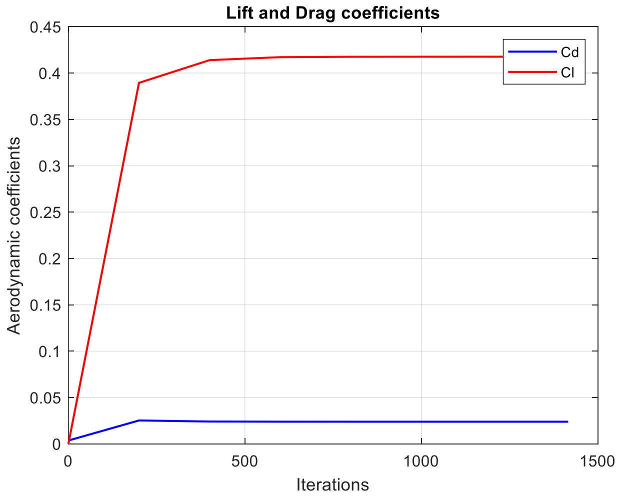


Figure 19

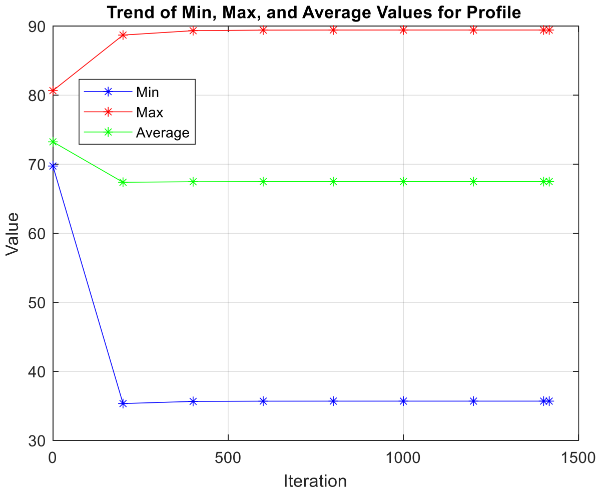
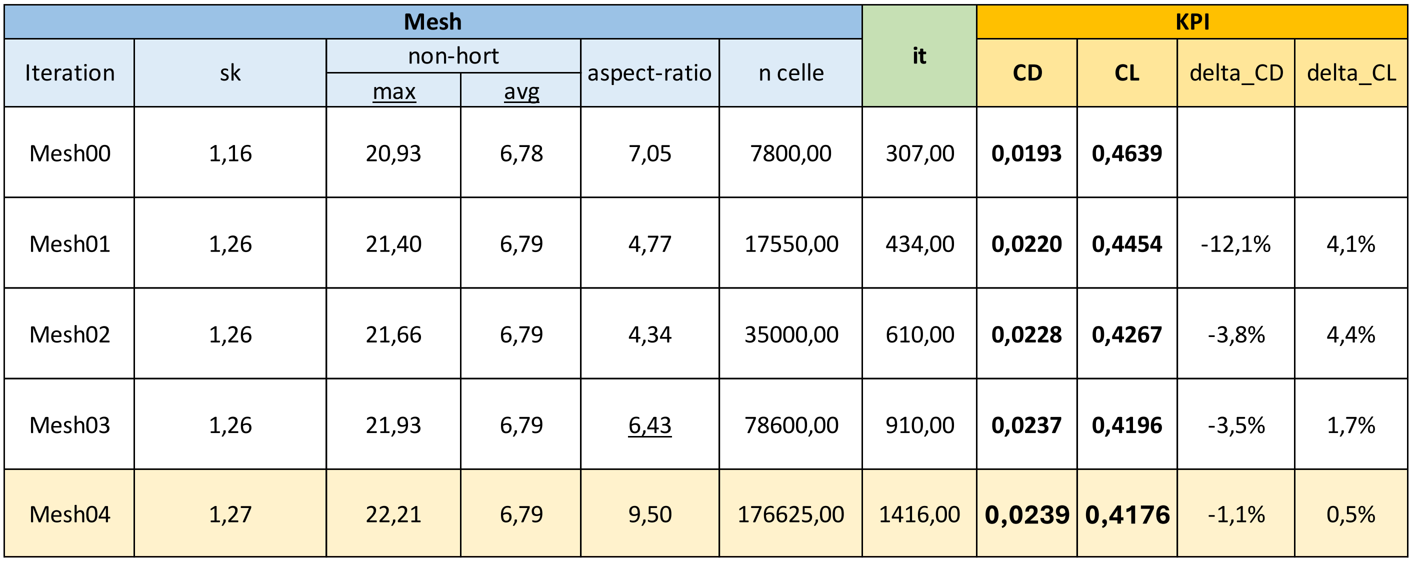


Figure 20

In this case, with wall function applied: $y^+ = 50 \div 500$

Grid independence study



Mesh						it	KPI			
Iteration	sk	non-hort		aspect-ratio	n celle		**CD**	**CL**	delta_CD	delta_CL
		max	avg							
Mesh00	1,16	20,93	6,78	7,05	7800,00	307,00	**0,0193**	**0,4639**		
Mesh01	1,26	21,40	6,79	4,77	17550,00	434,00	**0,0220**	**0,4454**	-12,1%	4,1%
Mesh02	1,26	21,66	6,79	4,34	35000,00	610,00	**0,0228**	**0,4267**	-3,8%	4,4%
Mesh03	1,26	21,93	6,79	6,43	78600,00	910,00	**0,0237**	**0,4196**	-3,5%	1,7%
Mesh04	1,27	22,21	6,79	9,50	176625,00	1416,00	**0,0239**	**0,4176**	-1,1%	0,5%

Figure 21

Simulations without wall functions (SA)

Boundary conditions

Considering the goal of this simulation, the boundary conditions for the kinematic turbulent viscosity *nut* have been changed to impose the simulation not to use the wall functions. On the other hand, the boundary conditions for other quantities haven't changed.

- Turbulent kinematic viscosity (*nut*) ν_T $[m^2/s]$:

Inlet	calculated
Outlet	zeroGradient
Profile	**fixedValue (0)**
Front and back	empty
Top and bottom	symmetry

Setup of the simulation

Mesh

Compared to the previous simulations, the mesh used in this case is significantly different. For instance, the cells have been redistributed to make sure that the first cell has y+ equal to 1 and 10 cells cover a distance up to y+ = 10. Once again, the number of cells in the different blocks was adjusted to have a smooth transition between the smallest and the largest cells and to keep the aspect ratio low. Once again, a snapshot of the mesh is reported below together with a detail of the boundary layer modelling and the checkMesh results:

Number of cells	90800
Skeweness (max)	0,927
Non orthogonality (max/mean)	26,87° / 7,15°
Aspect ratio	126,5

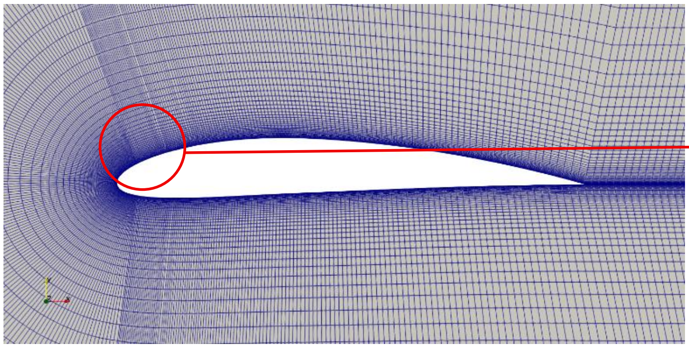


Figure 22

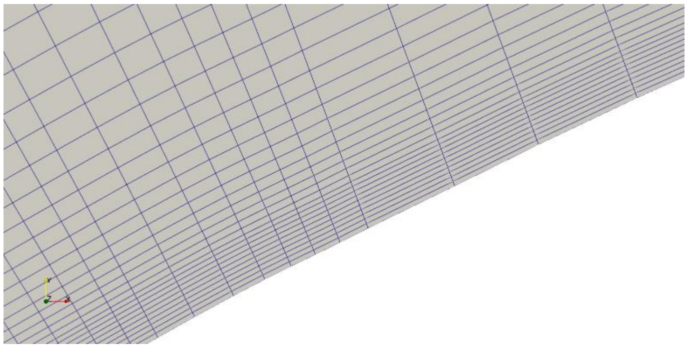
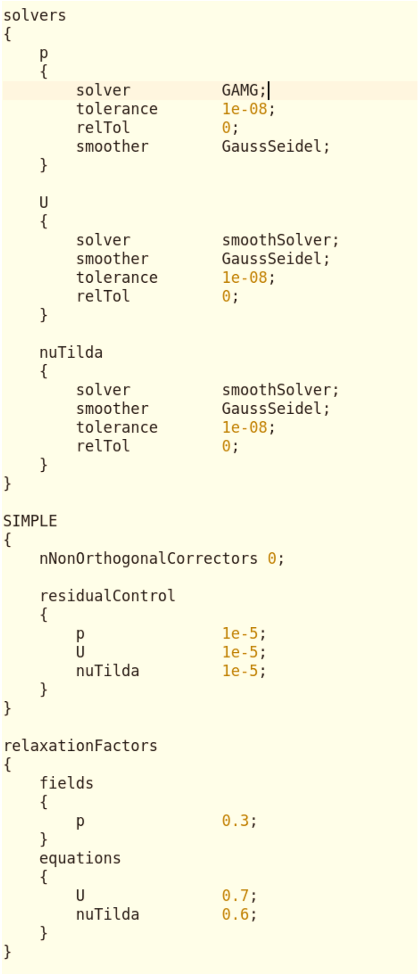


Figure 23

Domain discretization schemes (fvSchemes)

The turbulence model has not changed and in the same way, also the discretization schemes haven't changed.

Resolution schemes (fvSolution)



```
solvers
{
    p
    {
        solver          GAMG;
        tolerance       1e-08;
        relTol          0;
        smoother        GaussSeidel;
    }

    U
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        tolerance       1e-08;
        relTol          0;
    }

    nuTilda
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        tolerance       1e-08;
        relTol          0;
    }
}

SIMPLE
{
    nNonOrthogonalCorrectors 0;

    residualControl
    {
        p               1e-5;
        U               1e-5;
        nuTilda         1e-5;
    }
}

relaxationFactors
{
    fields
    {
        p               0.3;
    }
    equations
    {
        U               0.7;
        nuTilda         0.6;
    }
}
```

Figure 24

From the previous case the only thing that was changed is the value of the relaxation factor of $\tilde{\nu}$ quantity, that was slightly increased (from 0.3 to 0.6) to speed up the convergence of this quantity and reduce the overall iterations to reach the convergence.

Computation of the KPI

The KPIs that need to be computed haven't changed, therefore the procedure followed is the same for previous case.

Post Processing

Results

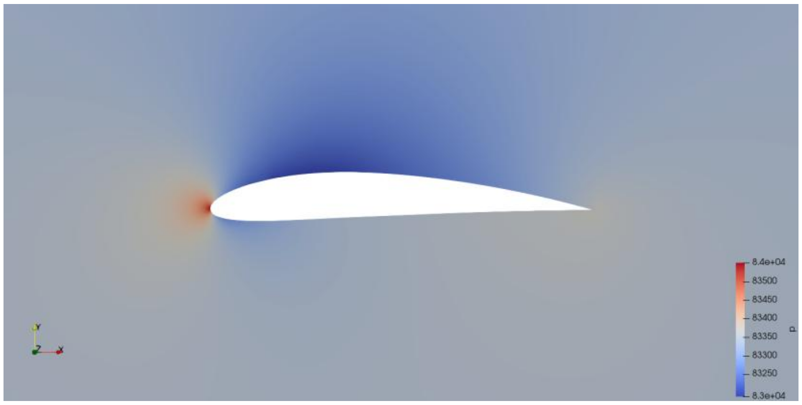


Figure 25

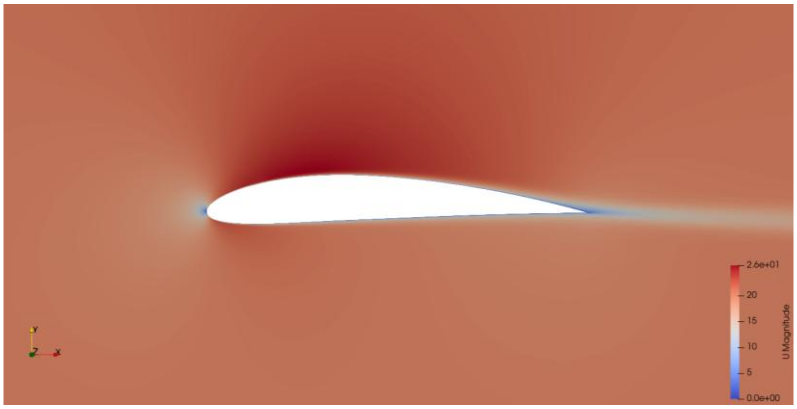


Figure 26

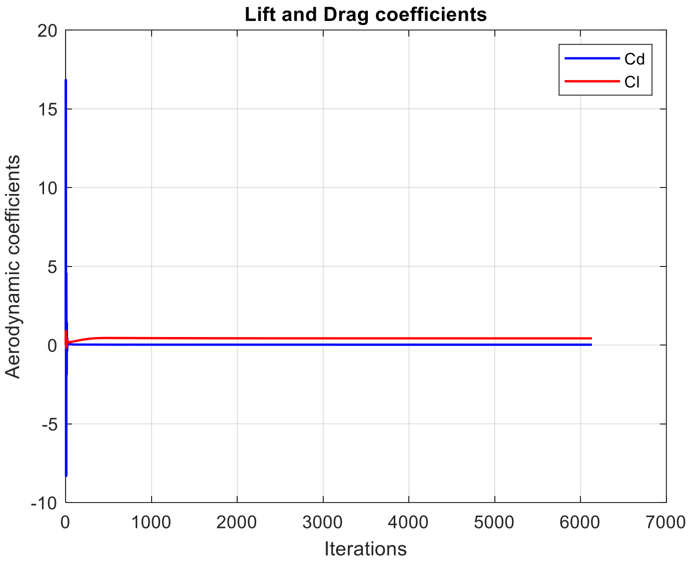


Figure 27

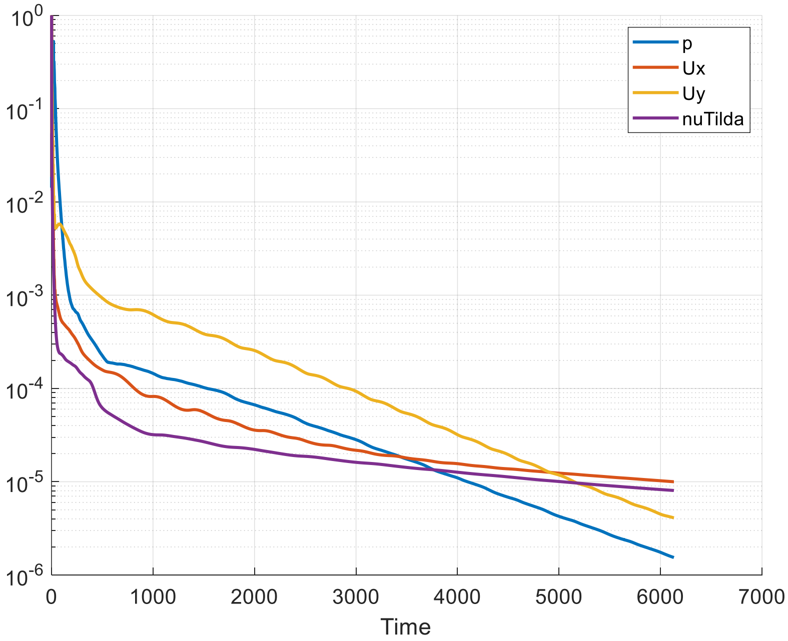


Figure 28

Convergence is considered reached when residuals are reduced by five orders of magnitude.

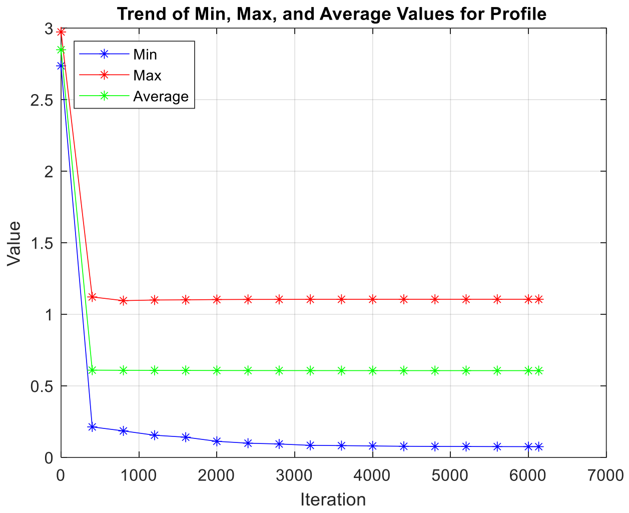


Figure 29

In this case, with wall function applied

- $y^+ < 1$
- Ten cells within $y^+ = 10$

Analysis and comparison

KPI

KPI obtained	C_L	C_D	E
Case A (k-Eps + WF)	0.348803	0.025983	13.424
Case B (SA + WF)	0.417591	0.023917	17.460
Case C (SA without WF)	0.425263	0.022393	18.991

k-EPS + WF: since the model used is not tuned for this specific application, results are not considered reliable.

Spalart-Allmaras: analyzing the differences between case B and C:

- The variation on the **lift coefficient** between the 3 cases is minimal as it is mainly influenced by the large-scale flow, which is only slightly influenced by the choice of boundary layer resolution method.
- On the other hand, the **drag coefficient**, being influenced by the near wall resolution, is affected by the different methods chosen. Friction coefficient is mainly influenced by the gradient between the boundary layer.

Flow over the profile

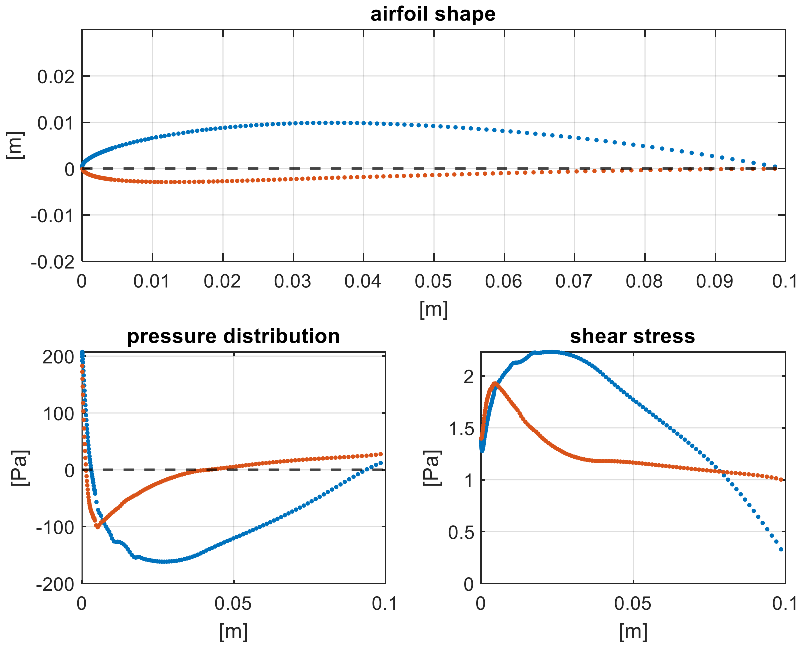


Figure 30 (case A)

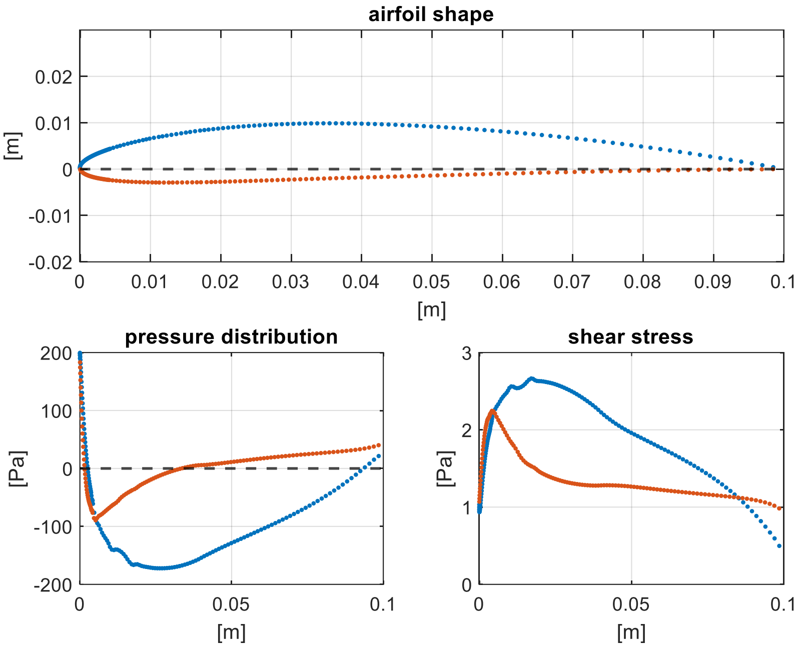


Figure 31 (case B)

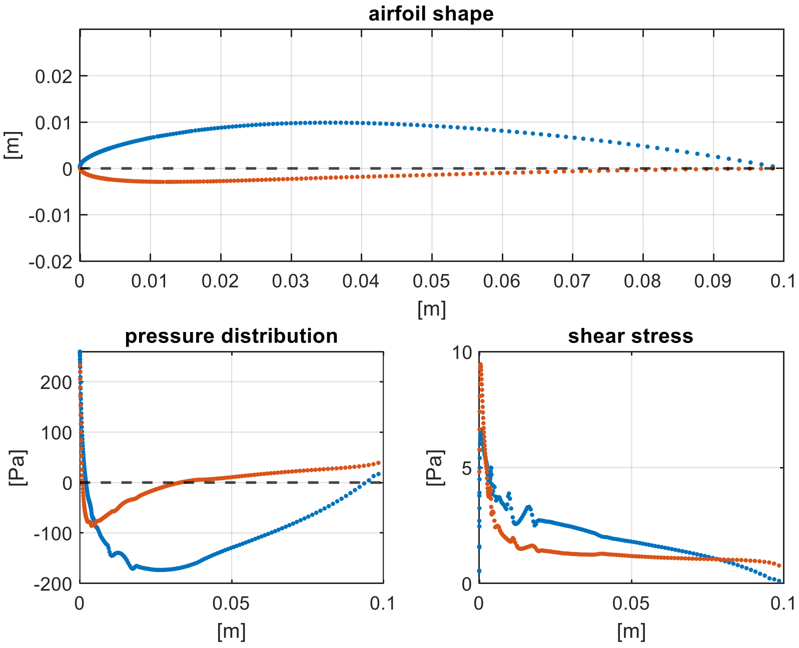


Figure 32 (case C)

Pressure coefficient distribution

(pressure values presented are subtracted by the asymptotic value)

- Trend is coherent with a typical airfoil pressure distribution

- C_P remains stable across the different simulation, as it is mainly influenced by macroscopic flow characteristics

Wall shear stress

- Results obtained with SA approach are considered more reliable than the one obtained with k-eps because the latter is generally less accurate, predicting flow behavior near stagnation points and in boundary layer development
- Case C solution, the one obtained without using wall function, solving up to the wall, is considered the most accurate representation of the wall shear stress distribution

Velocity distribution

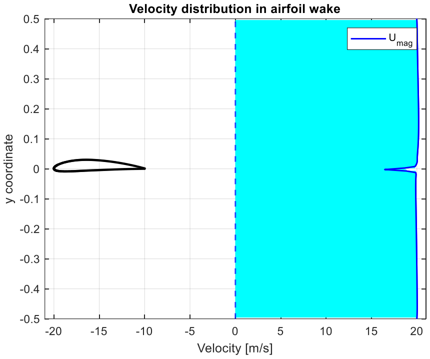


Figure 33 (case A (k-eps + WF))

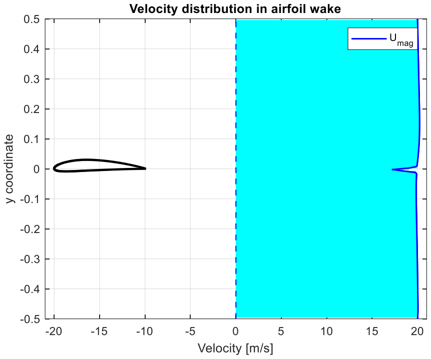


Figure 34 (case B (SA + WF))

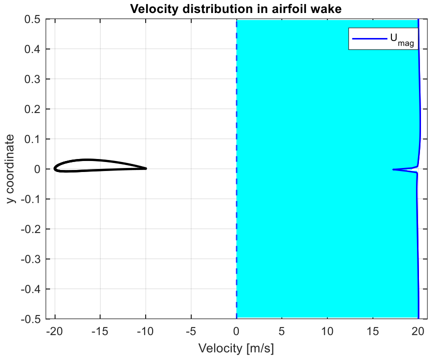


Figure 35 (case C (SA no WF))

- All the three simulations are similar, being k-eps good far from the wall this result is coherent with the theory
- The only slight difference is the asymmetry of flows over and under the wing. In the k-esp approach the velocity profile is more symmetric because of the model inability to consider the adverse pressure gradient near the wall. On the other hand, SA, which is tailored for airfoil applications, highlights more asymmetry in the flow, being the fully resolved approach the most accentuated. This is due to the better ability to capture the interactions between the wall and air.