Czech Technical University in Prague
Faculty of Mechanical Engineering
Department of Aerospace Engineering

CFD Methodology of Formula Student Race Car External Aerodynamics

by

TOMÁŠ KREJČÍ

A bachelor's thesis submitted to
the Faculty of Mechanical Engineering, Czech Technical University in Prague,
in partial fulfilment of the requirements for the degree of Bachelor.

Bachelor degree study programme: Theoretical Fundamentals of Mechanical Engineering

Prague, May 2024

Supervisor:
Ing. Jiří Teichman
Department of Aerospace Engineering
Faculty of Mechanical Engineering
Czech Technical University in Prague
Technická 4
160 00 Prague 6
Czech Republic

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Krejčí Tomáš** Personal ID number: **502696**

Faculty / Institute: **Faculty of Mechanical Engineering**

Department / Institute: **Department of Aerospace Engineering**

Study program: **Theoretical Fundamentals of Mechanical Engineering**

Branch of study: **No Special Fields of Study**

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

CFD Methodology of Formula Student Race Car External Aerodynamics

Bachelor’s thesis title in Czech:

CFD metodologie externí aerodynamiky závodního vozu Formula Student

Guidelines:

1) Define CFD simulations requirements for the aerodynamic design of the Formula Student car
2) Develop methodology for computational mesh creation for the Ansys Fluent solver
3) Develop methodology for solver settings and postprocessing using Ansys Fluent

Bibliography / sources:

According to instructions of supervisor

Name and workplace of bachelor’s thesis supervisor:

Ing. Jiří Teichman Department of Aerospace Engineering FME

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **26.04.2024** Deadline for bachelor thesis submission: **24.05.2024**

Assignment valid until: _____

Ing. Jiří Teichman	Ing. Milan Dvořák, Ph.D.	doc. Ing. Miroslav Španiel, CSc.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student’s signature

Acknowledgements

I would like to express my gratitude to my bachelor's thesis supervisor, Ing. Jiří Teichman. He has been very supportive of all my problems and questions. Thanks to him I gained basic knowledge about CFD and its use for FS race car CFD simulation.

I would like to thank Bc. Jan Plesník, eForce teammate with whom I was able to discuss many CFD problems as well.

Big thank goes to the university team eForce Prague Formula and all its members who were more like a second family to me. I was pleased to spend most of my time during bachelor studies with so inspirational, motivated, and smart students as well as honoured by getting an opportunity to lead such an amazing team for two seasons as a team captain. Without this team, my studies would be completely different.

Other people worth mentioning are Ondrej Guth, Josef Hlaváč, Tomáš Zahradnický, Daniel Sýkora and Pavel Tvrdík - creators of this amazing Latex template who were so kind to share this template in Latex library so that other students might use it.

It would be also fair to mention and thank to my "artificial" consultant ChatGPT, who did not write a single sentence in this work but was extremely helpful when searching for sources of information and suggesting codes for Latex and Matlab.

Finally, my greatest thanks go to my family, parents especially, for their infinite care and support throughout my bachelor studies ...

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

. . . . . .
Tomáš Krejčí
In Prague, 24. May 2024

Abstract

This bachelor's thesis aims to set a new CFD methodology for Formula Student team eForce Prague Formula. The goal is to recognize the CFD requirements of the aerodynamics group and develop a CFD simulation setup according to these requirements. Various setups within the CFD software, both meshing and solver, were compared, and the best option was chosen. Based on the comparisons, a new CFD method was created and fully automatized.

Keywords:
CFD, Formula Student, meshing, solver, aerodynamics

Abstrakt

Tato bakalářská páce se zabývá návrhem nové CFD metodiky pro Formula Student tým eForce Prague Formula. Cílem je stanovení požadavků aerodynamické skupiny na CFD simulaci, a následné nastavení této simulace dle těchto požadavků. Při nastavování simulace bylo porovnáno několik způsobů nastavení jak v meshingu, tak solveru. Na základě těchto porovnání byla zvolena optimální metodika.

Klíčová slova:
CFD, Formula Student, meshing, solver, aerodynamika

Contents

Abbreviations

Miscellaneous Abbreviations

CFD	Computational fluid dynamics
FS	Formula Student
CTU	Czech Technical University
FEE	Faculty of Electrical Engineering
CAD	Computer aided design
BOI	Body of influence
CoP	Centre of pressure
AoA	Angle of attack
GR	Growth ratio
2D	Two dimensional
3D	Three dimensional
MRF	Moving reference frame
DNS	Direct numerical solution
LES	Large eddy simulation
RANS	Reynolds-averaged Navier-Stokes
SST	Shear Stress Transport
GEKO	Generalized k-omega
GPU	Graphics processing unit
CPU	Central processing unit
RAM	Random-access memory
GUI	Graphical user interface
ID	Identifier
PC	Personal computer
GB	Gigabyte
SIMPLE	Semi-Implicit Method for Pressure-Linked Equations
IEEE	Institute of Electrical and Electronical Engineers

Mathematical Terminology

Cl	[1]	Coefficient of lift
Cd	[1]	Coefficient of drag
$y+$	[1]	Non dimensional distance from wall
Re	[1]	Reynolds number
ρ	[kg/m^3]	Density
u	[m/s]	Velocity
L	[m]	Characteristic dimension for Re
μ	[Pa.s]	Dynamic viscosity
δ	[m]	Height of boundary layer
x	[m]	Characteristic dimension for boundary layer thickness
π		Ludolph‘s number
S_i		Source term
C_0	[1]	User-defined empirical coefficient
C_1	[1]	User-defined empirical coefficient
v	[m/s]	Velocity

Chapter 1

Introduction

A famous racing car driver Enzo Ferrari once allegedly said: "Aerodynamics are for people who can't build engines." Nowadays we can certainly tell how wrong his quota was, as shown in Figure 1.1 Since the beginning of motorcar racing teams and engineers have been trying to improve racecar aerodynamics in order to gain performance advantage over their competitors and achieve faster lap times of their cars. The beginnings of these developments were not easy for race car engineers since they could only rely on physics knowledge and driver's feedback through trial and error method. The real breakthrough came in the 20th century when the first CFD methods were developed and as their development continued, CFD software became a crucial part of every professional race car development.

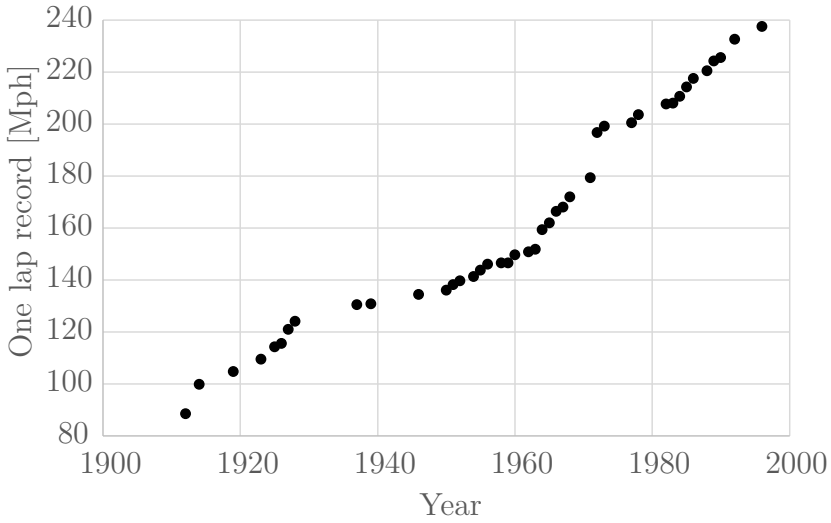

Figure 1.1: Evolution of One-Lap Qualification Record at Indianapolis 500. Note the biggest step over one year, in 1972, when the traditional front and rear wings were implemented for the first time.[1]

1.1 Formula Student

To understand why this thesis began and what this thesis aims for, the reader should be confronted with a brief introduction to Formula Student competition. The first FS competition was held in 1981 in USA from where it expanded all over the world. Now (2024), there are 25 independent competitions in 5 continents. Some competitions share the same organizer and the same set of rules. All European teams (except Great Britain) follow the same rules released by the biggest and most prestigious FS competition in the world, Formula Student Germany. This student program aims to broaden university students‘ knowledge, both theoretical and practical. Any university can establish team which has to design and build race car according to the current rules. Then the team competes against other teams at mentioned competitions. Czech Technical University in Prague has its own FS team, eForce Prague Formula, which competes in an electric category with over 13 years of experience.

For more information about the competition visit the official site of Formula Student Germany (www.formulastudent.de) and for more information about the team eForce Prague Formula visit (www.eforce.cvut.cz).

1.2 Motivation

Until the 2024 FS season, there were two teams at CTU in Prague, eForce FEE Prague Formula competing in electric/driverless category and CTU Cartech, a few years older team competing in the combustion/hybrid category. As the new regulations and automotive trends brought an end to combustion FS cars, the teams decided to merge for season 2024. This merge brought many new challenges to the new team, especially because of different design approaches in previously separated teams. These different approaches were visible also in the aerodynamics department with the most significant difference - the approach to CFD. Team CTU Cartech used Star CCM+ whereas eForce FEE Prague Formula used Ansys Fluent software. All user interface experience and preferences of a single team‘s CFD developers put aside, both software are the peak of what CFD industry offers. When both software are used correctly and identically, there should not be any significant difference in result. Some simulations may differ in computational time, but this also depends on the specific use of the software.

However many differences were also examined in various stages of the simulation setup such as surface mesh, volumetric mesh with prism layers, and choice of turbulence model. This thesis should compare different approaches and establish a new methodology for eForce Prague Formula team according to these observations because the simulation methodology for season 2024 was based on Ing. Martin Ševčík masters thesis [2] even though the simulation setup was already outdated. Moreover, the goal is to try completely new methods such as GPU solver which can save more than 80 % of computational time.

1.3 Requirements

The aerodynamics group of eForce Prague Formula team is divided into two main development groups. The first one is focused on CFD development whereas the second (and larger) one is focused on aerodynamic package development in terms of CAD and manufacturing. This group uses CFD results for a better understanding of their design changes in CAD but the in-depth functionality and setup of the CFD software stays hidden to the group members, mainly because of the specialization one needs to operate with CFD software. The aerodynamics CFD workflow is visualized in Figure 1.2.

In conclusion, CFD software that allows users to upload new CAD model and obtain all necessary results automatically is needed. So-called one-click simulation, however, brings another problem: a possible lack of knowledge transfer among CFD engineers, especially when a high level of automation is implemented. This problem can be avoided by creating step-by-step tutorials for team novices.

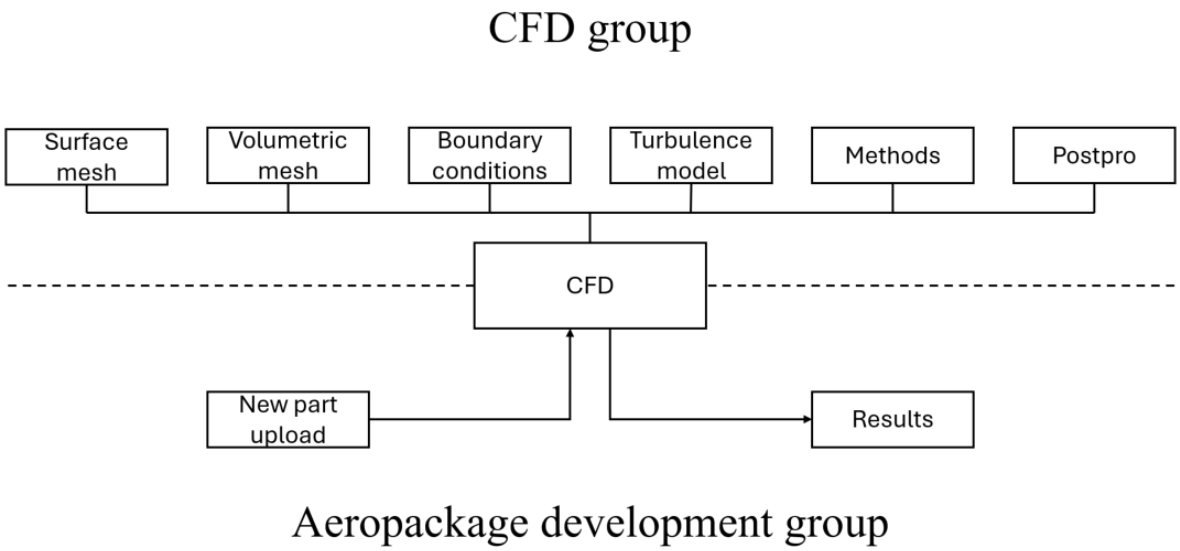

Figure 1.2: Scheme of aerodynamics group CFD workflow

1.4 Thesis Structure

The thesis is organized into 8 chapters as follows:

1. *Introduction*: Describes the motivation behind this thesis and recognizes requirements of eForce Prague Formula FS team.
2. *CAD Model for CFD*: This chapter discusses various approaches to CAD modelling and its impact on CFD setup.
3. *Mesh*: In this section mesh refinement and boundary layer capture are the main topics.
4. *Boundary Conditions*: In Chapter 4, boundary conditions for axial fan and cooling are compared.
5. *Turbulence models*: Chapter 5 discusses one of the key aspects of this thesis. It compares an already outdated turbulence model for FS race car simulation with a newly proposed and adjusted model.
6. *CFD Solver Computation*: Compares various approaches to solution methodology in terms of numerical accuracy, convergence criterion, and computational time. Also, compares two different approaches in terms of computer hardware usage.
7. *Final Simualtion*: Summarizes the results of previous comparisons and defines a new CFD methodology for FS team eForce Prague Formula.
8. *Conclusion*: Brings a conclusion to this thesis and proposes further research of CFD methodology within eForce Prague Formula team.

Chapter 2

CAD Model for CFD

First, we need to discuss the very beginning of every CFD simulation which is CAD model. However, since this thesis is not about CAD modeling we will discuss this topic very briefly and mention only some important aspects.

2.1 Simplification

CAD model of Formula Student race car is very complex, containing every component of the car, from large parts such as monocoque to the smallest ones such as bolts, brackets, and many others. This model would be overly detailed for whole car simulation, resulting in problems with meshing and long computational time. Because of that, a simplified CAD model is created. This model contains detailed parts crucial for influencing the airflow around the car. Other parts such as previously mentioned bolts, brackets, etc. are replaced with simple objects like cylinders or polyhedrons. Figure 2.1 shows the difference between the standard front axle assembly and the CFD-optimized CAD model of the same assembly.

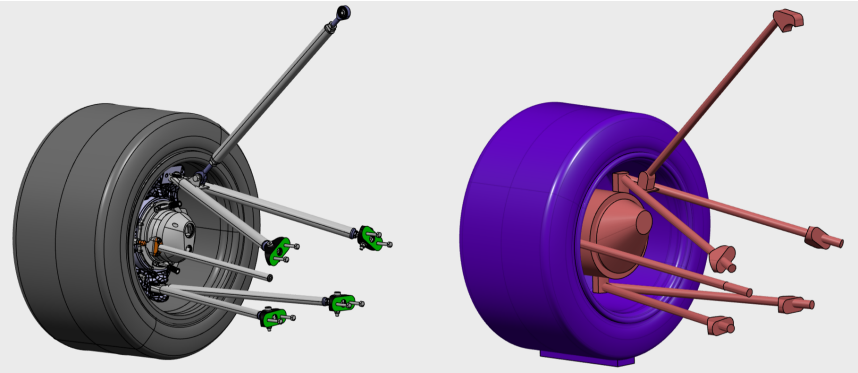

Figure 2.1: Difference between standard CAD geometry (left) and simplified geometry (right)

2.2 Tyre Contact Patch

Contact patch between the tyre and road is a crucial area for race car underbody aerodynamic performance since the rotating wheel creates inwash and outwash vortices which influence airflow toward the rest of the car. Every tyre has a slightly different contact patch depending on load, compound, and wear. Also, the tyre height which changes with tyre load is important for open-wheeled cars regarding side wing and rear wing aerodynamical performance.

Moreover, a tyre contact patch is an area with a sharp angle between the tyre and the road but large geometry simplification can not be done because of the reasons mentioned in the paragraph above. Various approaches to this problem exist. The most common solutions are:

- Creating a solid block underneath the tyre to increase the contact patch angle
- Refining mesh underneath the tyre

Both mentioned approaches were simulated and compared to decide which solution is the best for FS race car application. All simulation settings were kept the same for the sake of valid CFD comparison.

2.2.1 Ideal Geometry With Solid Block

This approach should be the easiest for both modeling and meshing. Two different sizes were compared - a smaller one (v1) and a bigger one (v2). Dimensions of both blocks are shown in Figure 2.2.

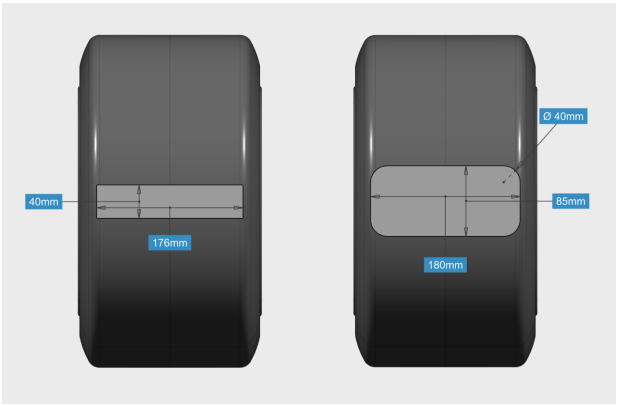

Figure 2.2: Difference between a smaller block v1 (left) and a bigger block v2 (right)

The big block has proved to be the easiest and quickest solution. Having over 1 million cells less (please note that because of the bad cells orthogonal quality, v1 mesh had to be locally improved by BOI, for definition of BOI see Section 3.2) and no problem during

volumetric meshing thanks to the lack of sharp radiuses create from big block robust solution that does not require large mesh, which might be suitable for teams with lack of computational power or inexperience with CFD, see Table 2.1. Table 2.2 shows a comparison of aerodynamic performance. Especially the difference in lift is more than 9 %.

Figure 2.3 shows drag over iterations and Figure 2.4 shows lift over iterations. In both figures, there is a visible difference in convergence and final result which supports the idea that the big block solution is more robust for meshing and solver but lacks real-life accuracy.

Table 2.1: Mesh size comparison

	Number of cells
Big Block	2 611 708
Small Block	3 683 233

Table 2.2: Values of drag and lift, last 300 iterations average

	Drag [N]	**Lift [N]**
Big Block	8.16	-10.04
Small Block	7.93	-9.12

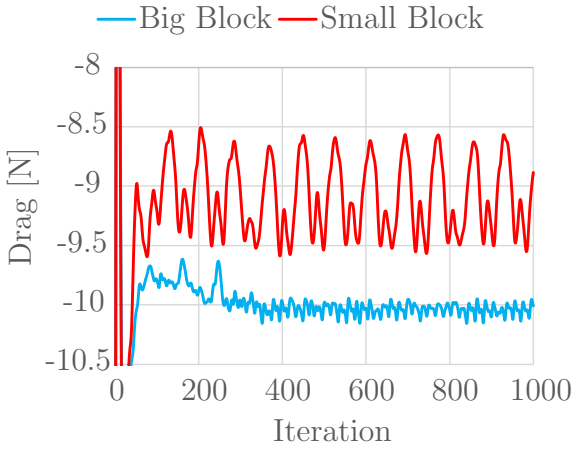

Figure 2.3: Big Block vs Small Block drag report

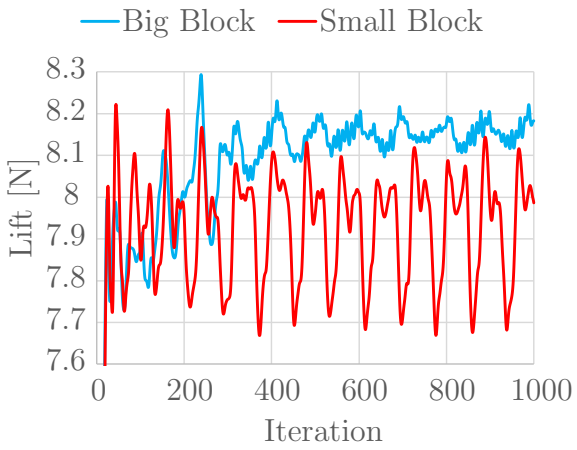

Figure 2.4: Big Block vs Small Block lift report

2.2.2 Comparison of Ideal Geometry and Small Solid Block

Now that we have compared different sizes of solid block geometry and decided that a smaller block is a better solution for our case, let's compare the smaller block with absolutely ideal geometry of the tyre. All settings of simulation were kept the same and both tyres required additional BOI around the area of contact patch. Because every CFD mesh is limited by its minimum size defined in scoped sizing, even ideal geometry tyre can not stay intact. After a wrap was created around the ideal geometry tyre, a contact patch was created by the limits of scoped sizing as shown in Figure 2.5. In the end, the size of the contact patch of ideal tyre was almost identical to the size of a small solid block contact patch. Refinement of ideal tyre scoped sizing might be done but the volumetric mesh would be too great for whole car simulation then. As we mentioned the almost-identical contact patch dimension in both cases, there is no surprise the results were identical as well, see Table 2.3, Table 2.4, Figure 2.6 and Figure 2.7.

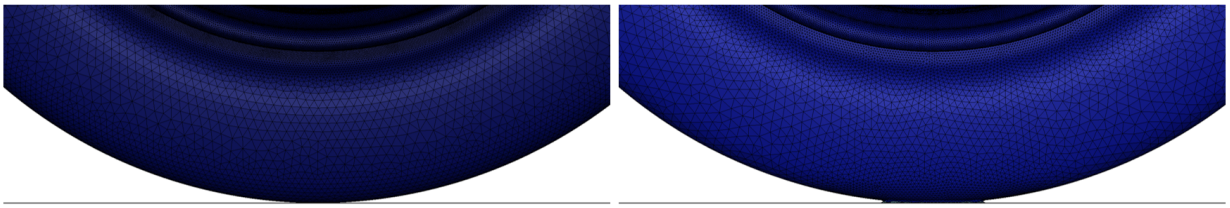

Figure 2.5: Ideal geometry before mesh (left) and ideal geometry after mesh (right)

Table 2.3: Mesh size comparison

	Number of cells
Ideal	3 669 357
Small Block	3 683 233

Table 2.4: Values of drag and lift, last 300 iterations average

	Drag [N]	**Lift [N]**
Ideal	7.89	-9.07
Small Block	7.93	-9.12

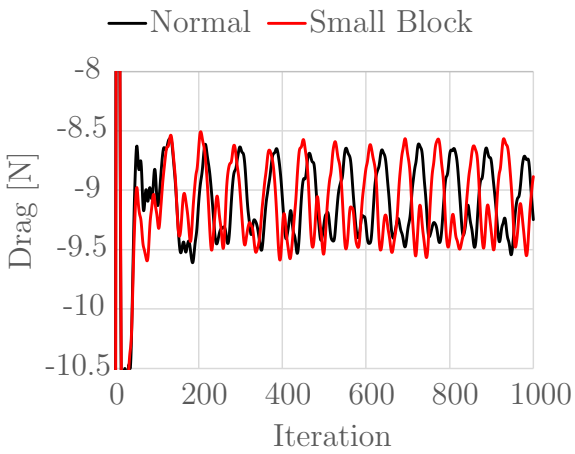

Figure 2.6: Ideal Geometry vs Small Block drag report

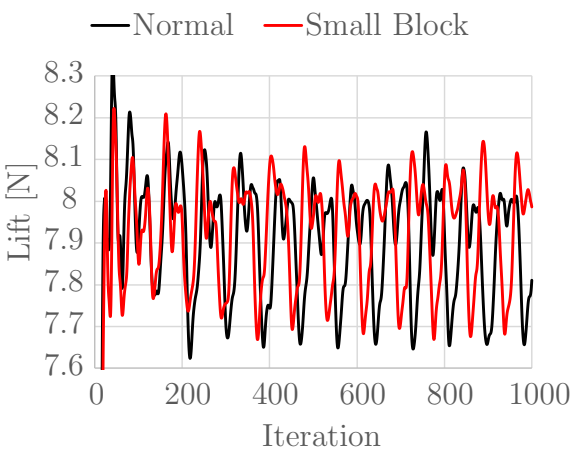

Figure 2.7: Ideal geometry vs Small Block lift report

To summarize this research let's compare all three variants. The tyre contact patch can be simplified by creating solid block geometry to refine the sharp angle between the tyre and the ground but we have to be very careful when choosing the dimensions of such a block since it can drastically influence the result of our simulation. When choosing between small block simplification and an ideal geometry approach, both simulations show the same results so the decision depends on overall simulation setup and CFD software use. For example, students from former FS team CTU Cartech used small block geometry because they had problems with wrap generation around the sharp angles whereas students from eForce FEE Prague Formula used a direct approach with ideal tyre geometry since their Fluent workflow had no problem with such angles. The final pictures of this subsection show a comparison of all three variants in terms of total pressure distribution at 2 mm above ground (Figure 2.8). It is visible how different geometry approaches can change not only the final result of the tyre but also the airflow towards other crucial aerodynamic elements like the side wing and underbody.

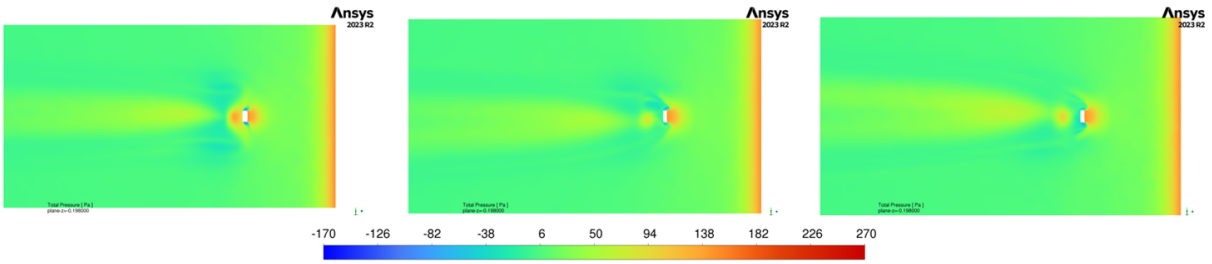

Figure 2.8: Total pressure field from left side: Big Block, Small Block, Ideal geometry

2.2.3 Tyre Radius Regarding Tyre Load

In the previous subsection, we discussed the influence of tyre contact patch on the overall simulation result. We supposed ideal tyre geometry however, the geometry of the tyre is not always ideal in real race conditions. As more load is applied to the tyre, the geometry changes as well. According to the data processed by eForce Prague Formula suspension department, the tyre radius can change by ∼5 mm when a load of 300 N is applied to a single wheel. Even higher load is possible when FS race car reaches its top speed during acceleration for example. This tyre behaviour is not a problem of CFD methodology so no comparison will be done in this work however, it is an interesting case which should be examined by eForce Prague Formula aerodynamics department in the following seasons. The detailed relationship between tyre loaded radius and normal load is shown in Figure 2.9.

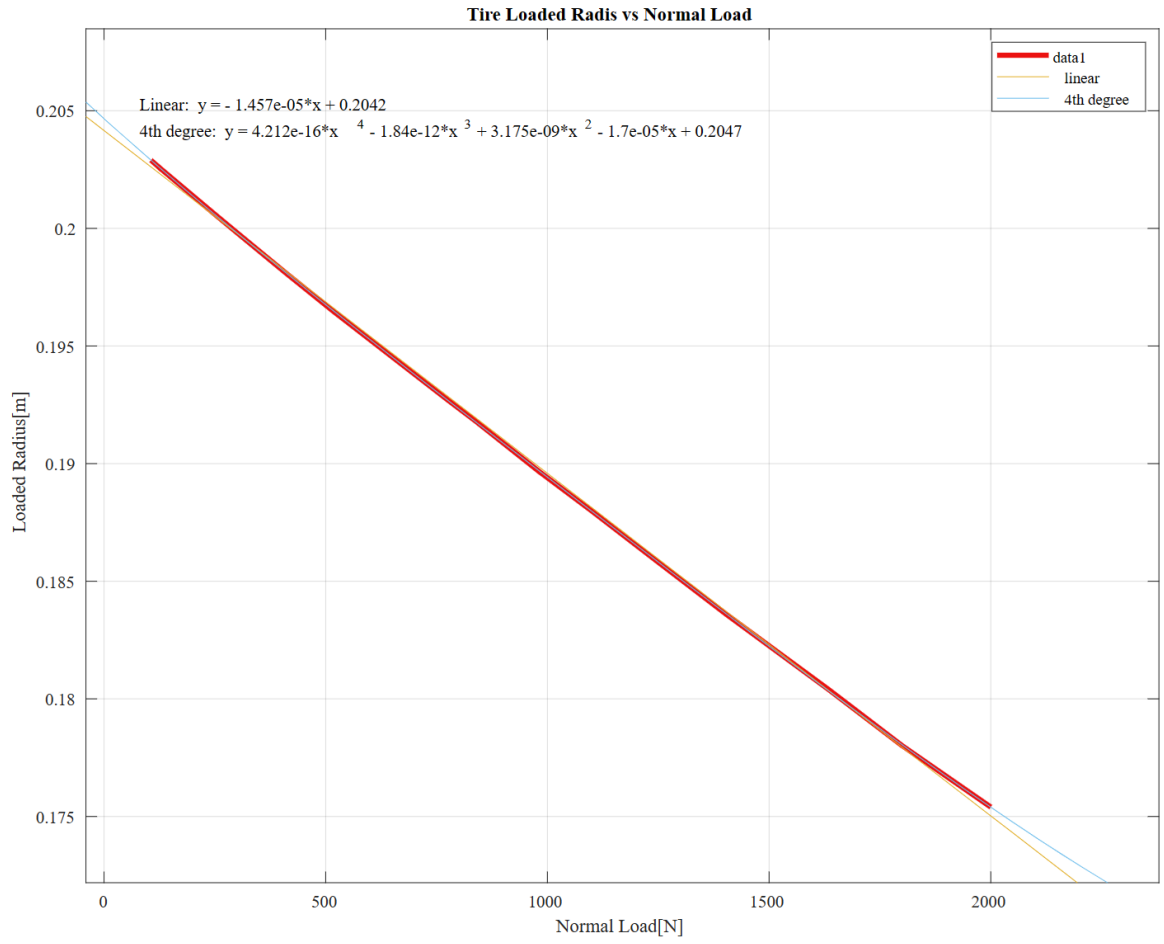

Figure 2.9: Relationship between tyre loaded radius and normal load

2.3 Trailing Edge Geometry

Airfoils can have various shapes of trailing edges. Most common airfoils used for FS race car have sharp trailing edges in the design phase. However, since all wings are hand-crafted in a university workshop, the final geometry tends to be slightly different. Especially the trailing edge becomes blunt after lamination, cutout, and final assembly. The thickness of such trailing edge was measured to be approximately 1 mm. Further CFD research was done to estimate the effect of this change in terms of wing performance. A conventional Selig 1223 airfoil (commonly used airfoil amongst many FS teams) was used to create a wing cascade from three identical airfoils, similar to wings used in Formula Student.

In order to receive the most reliable results, simulation settings were the following:

- Refined mesh at all trailing edges
- Poly-hexcore volumetric mesh
- 12 prisms layers with 1.5 growth ratio
- SST k-omega turbulence model
- Coupled solver with 2nd order upwind

Note that more prism layers with a smaller growth ratio might be used however, the y+ values met the selected turbulence model criteria, as well as the height of the boundary layer at single elements of the wing. Also, this simulation should go in hand with the actual simulation setup used in whole-car simulation to verify the correctness of that setup so a more refined prism layer would be uninformative.

The blunt geometry of the trailing edge has proved to be an easier and faster way for simulation setup. No special operations during wrap generation nor prism layer generation were needed. On the other hand, airfoil with an ideally sharp trailing edge needed more pre-wrap preparations in terms of edge zone scoped sizing refinement. Stair stepping during the generation of prism layers had to be adjusted also. Despite these differences in meshing, the rest of the simulation steps and setup remained the same. The difference in meshes between sharp and blunt trailing edge can be seen in Figure 2.10.

Results in Figure 2.11 show that the blunt trailing edge (real geometry on FS race car) has slightly better aerodynamic performance than the sharp trailing edge (wing in CAD model), even though, in theory, the ideal sharp edge should have better aerodynamic performance. It is important to mention that the difference is so small that we could further discuss whether the difference was made solely because of the geometry of the trailing edge or because it was caused by other aspects such as the gap between flaps. Anyway, after this comparison, we can say that there is no significant change in the results no matter which geometry of trailing edge we use. Table 2.5 shows a negligible difference in mesh size however, this might become a noticeable difference when computing whole-FS race car simulation. With every prism layer and geometry complexity added, the number of cells rapidly increases because the surface mesh at the sharp trailing edge must be refined

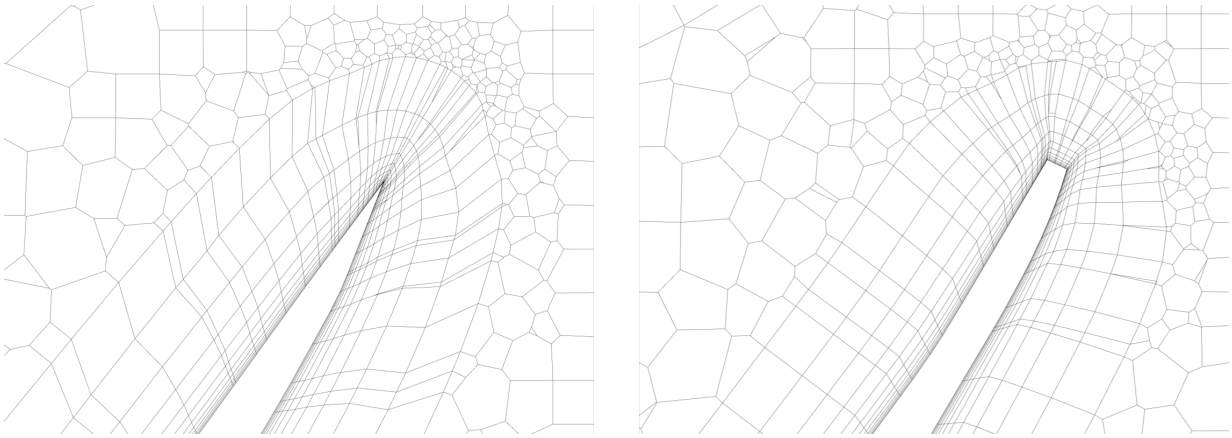

Figure 2.10: Mesh of sharp trailing edge (left) and mesh of blunt trailing edge (right)

accordingly. For the whole-car simulation, I would recommend using a blunt trailing edge as it is an easier and more robust approach for the volumetric meshing process. Moreover, the blunt trailing edge represents the actual FS race car trailing edges more accurately.

Table 2.5: Mesh size comparison

	Number of cells
Sharp	4 734 471
Blunt	4 372 344

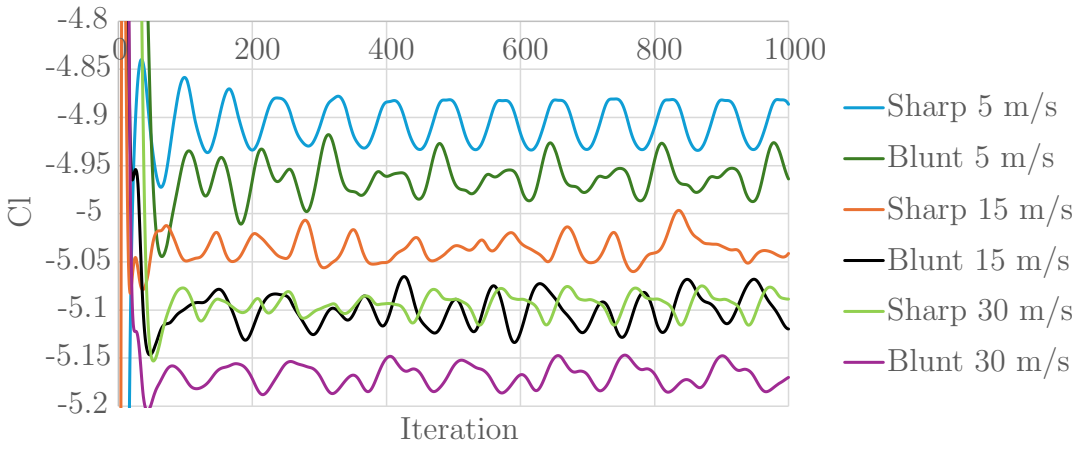

Figure 2.11: Coefficient of lift for sharp and blunt trailing edge geometries in different velocities

Chapter 3

Mesh

This chapter focuses on, probably the most important, section of the CFD simulation process. Underestimating the significance of both surface and volumetric fine mesh leads to problems with the CFD solver, especially its initialization and convergence criterion. Various applications require different mesh quality and size so many approaches exist. Bodies of influence, tunnel size, prism layers, cell shape, and many more aspects can be changed to reach the desired solver convergence and accuracy. The following sections focus on different setup approaches and their influence of CFD simulation in terms of grid size, simulation time, and result accuracy.

3.1 Testing mesh

Before we begin with our first comparison, the mesh used for this and the following simulations should be defined. A simplified model of FS race car was created, consisting of chassis, tyres, front wing, and rear wing. This model contains all parts that are important for the representation of race car - chassis with the driver, moving wheels, front wing as an aerodynamic device with the lowest ground clearance, and rear wing as an aerodynamic device with the most significant airflow separation. Other parts, such as suspension, rollover bar, side wing, and cooling would only increase the simulation time without adding extra information to this comparison. The tunnel dimensions for this mesh are 7.2x7.2x33 meters. This mesh, shown in Figure 3.1, is called "CTU.24_demo" and will be used throughout many following simulations and comparisons.

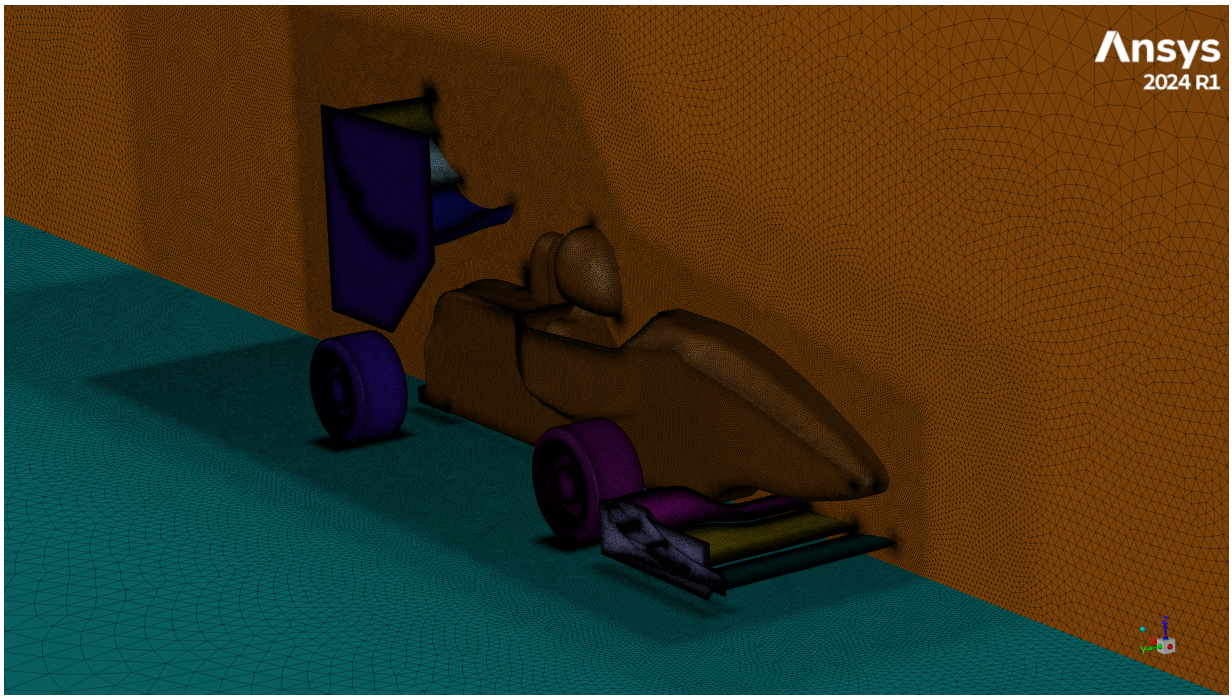

Figure 3.1: CTU_24_demo mesh

3.2 Body Of Influence

Body of Influence (=BOI) is a mesh refinement near simulated objects. During the process of meshing, cell size and cell growth ratio between the simulated object and virtual tunnel (or similar boundary area) are defined in scoped sizing control. However, the general scoped sizing selection only defines the maximum cell size and growth ratio within the simulated area and the minimal/maximal cell size on selected faces or edges. This might become a problem in the close surroundings of the simulated object.

Let's suppose FS race car simulation where we set cell size (suppose standard size from 0.5 mm to 320 mm) and growth ratio (suppose standard ratio 1.3) on the car surface and inside the tunnel. This scoped sizing will automatically generate cells of the same size around the car without any regard for pressure or velocity field. The mesh will have rapidly increased cell size in proximity to the car where many vortices are created. This might cause trouble with solution convergence. A solution to this uncertainty might be decreasing the growth ratio which would create larger amount of smaller cells. However, this approach is far from ideal because it will also create many more cells in free-flow areas, where fine mesh is not crucial for solution stability, thus increasing the computational time with no benefit to the results.

The best and most used solution is to create virtual areas around the simulated object, where large velocity and pressure gradients are expected and then set the required cell size

within this area. By doing so, a fine mesh can be generated in the specific surroundings of the car whereas larger cells can be applied to free-flow areas as shown in Figure 3.2.

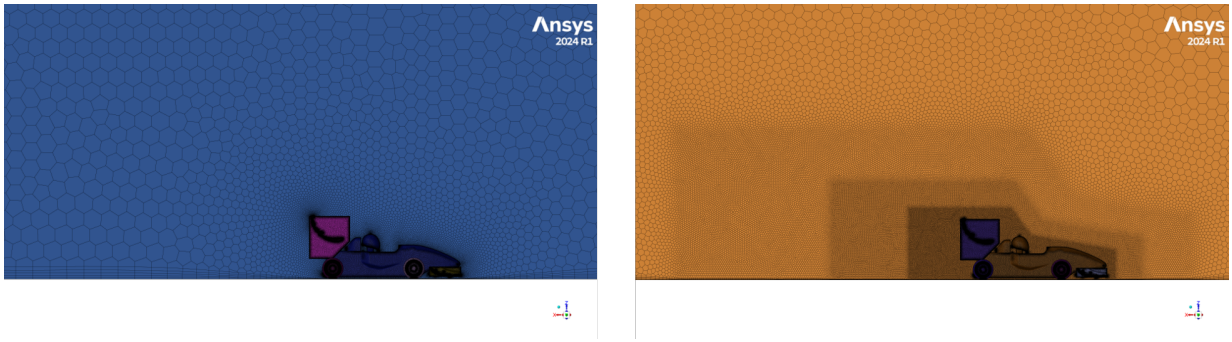

Figure 3.2: Mesh without BOI (left) and with BOI (right)

To support this theory, the following simulation was performed. Two identical models, CTU.24_demo, were created with one model wrapped without BOI and the second one with BOI. Compared parameters were the following:

- Mesh size
- Coefficient of drag - last 100 iterations average
- Coefficient of lift - last 100 iterations average
- Centre of pressure (CoP)
- Velocity in 6 selected points
- Time of numerical solution

The results are shown in Table 3.1 and Figures 3.3, 3.4, 3.5, and 3.6.

Table 3.1: no BOI and BOI comparison

Mesh refinement	**no BOI**	**BOI**
Mesh size	18 mil	23 mil
Coefficient of drag (last 100 it. avg.)	-214 772	1.58
Coefficient of lift (last 100 it. avg.)	-152 797	-3.56
Centre of pressure (% rear axle)	n/a	54.2
Time of numerical solution	35 min	65 min

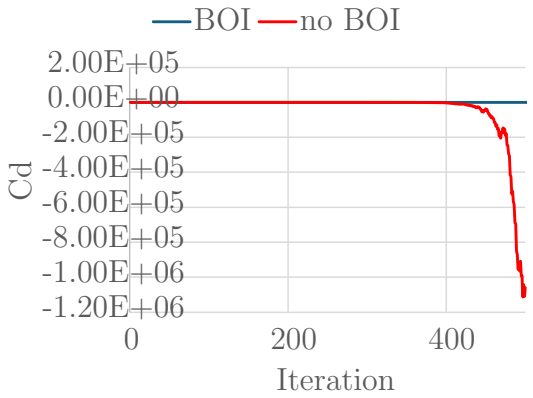

Figure 3.3: Mesh refinement drag report comparison

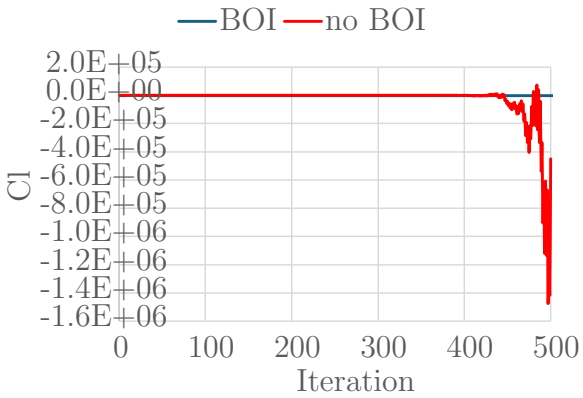

Figure 3.4: Mesh refinement lift report comparison

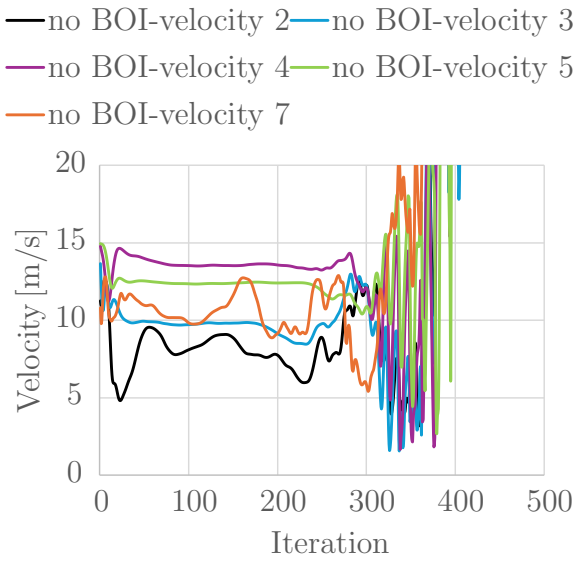

Figure 3.5: no BOI Velocity Report

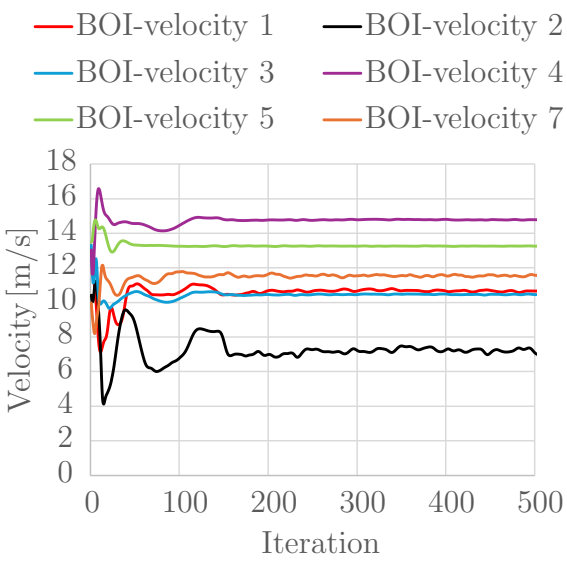

Figure 3.6: BOI Velocity Report

From the results in Table 3.1, Cd and Cl reports in Figures 3.3 and 3.4, and velocity reports in Figures 3.5 and 3.6 is obvious how much difference local mesh refinement can make. The simulation without any mesh refinement had 5 million cells less and its computational time was almost 50 % faster, however, the price in terms of solution stability was unacceptable. The refined mesh solution convergence was within 4 % whereas the unrefined mesh proved to be extremely unstable after 400 iterations. Such instability might

be caused by a single cell that could not, potentially, cause any significant disturbance in many other simulations without mesh refinement. However, this represents a huge risk for serial CFD automatization within the aerodynamic group development so BOI should never be neglected. Nevertheless, further research on this topic should be done by eForce Prague Formula aerodynamic group because decreasing the BOI to its minimum, while maintaining sufficient solution stability, might significantly save computational time.

3.3 Prism Layer

This section aims to explain the basic theory behind boundary layer capture in CFD and why it is so important to the overall simulation result. Before we continue, some values and terms should be explained.

3.3.1 Reynolds Number

Reynolds number is a non-dimensional number that can be defined as a ratio of inertial and viscous forces. It is one of the key numbers for aerodynamics because it allows to compare various geometries regardless of their dimensions and flow velocity. This is crucial for example when a wind tunnel test is performed. To save the cost of wind tunnel testing, and because some real-life objects might be too big for conventional wind tunnels, smaller models are tested in wind tunnel. To ensure wind tunnel results are valid with reality, the Reynolds number must be preserved:

$$Re = \frac{\rho \cdot u \cdot L}{\mu} \tag{3.1}$$

where ρ is density of the fluid, u is velocity of the fluid, L is the characteristic dimension, and μ is dynamic viscosity of the fluid.

3.3.2 Boundary Layer

The boundary layer is a region near a wall (or any surface in general) where the velocity of airflow increases from $u = 0$ at the wall to $u = u_\infty$, where u_∞ is freestream velocity. Figure 3.7 shows the boundary layer evolvement along a flat plane where δ is the height of a boundary layer. If we consider an undisturbed flow in front of the flat plate, the initial boundary layer will be laminar, becoming turbulent afterwards. The change between the laminar and turbulent boundary layer depends on the Reynolds number but it is important to mention that this change does not happen when the Reynolds number reaches a certain value but a certain range. The laminar boundary layer has a lower surface friction resulting in lower friction drag however, for race car applications turbulent boundary layer is usually preferred as the airflow separation is delayed because of the momentum transfer normal to the direction of average speed.[1]

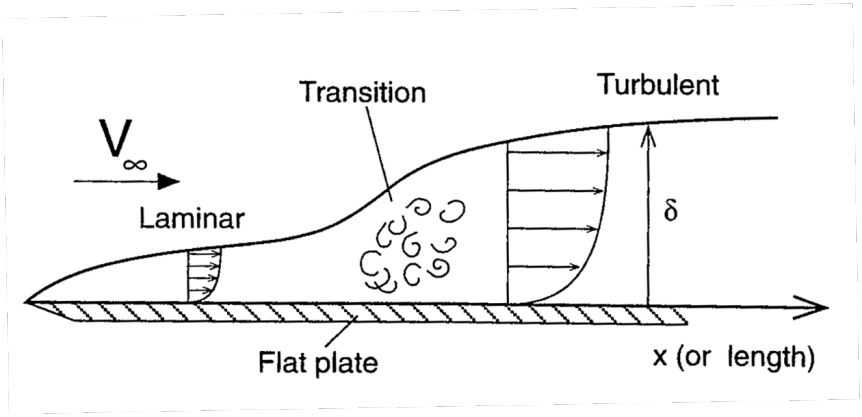

Figure 3.7: Boundary layer scheme[1]

3.3.3 Capture of Boundary Layer in CFD

To capture a boundary layer in CFD, a prism layer is used for the discretization of the velocity gradient within the boundary layer. There are various shapes of prism layer cells as well as their number, minimal height, and growth ratio. All these parameters can be changed in order to achieve a solid capture of the boundary layer. The following parameters should not be neglected:

The **height of the prism layer** should be the same or higher than the expected thickness of the turbulent boundary layer. The height can be expressed as:

$$\delta = 0.37 \cdot \left(\frac{x}{Re_x^{1/5}} \right) \tag{3.2}$$

where δ in turbulent boundary layer thickness, x is characteristic dimension, and Re_x is Reynolds number based on the characteristic dimension x.

The **height of the first prism** should comply with the selected turbulence model and its criterion for $y+$ dimensionless value according to the law of the wall pictured in Figure 3.8. This figure shows three regions of the wall law. In viscous sublayer $y+ \in < 0, 5)$ the velocity gradient is computed directly from the velocity values at each prism layer whereas in the logarithmic-law region $y+ \in (30, 300\rangle$, the height of the first prism layer is greater than the thickness of the boundary layer itself so the velocity gradient is computed using a mathematical formula. The range of $y+ \in \langle 5, 30\rangle$, the so-called buffer layer, is to be avoided since both of the previously mentioned approaches are inaccurate in this region. From the facts presented in this paragraph, the logarithmic-law approach might seem like the best option since we need only one or a few prisms to capture the velocity gradient

instead of the viscous sublayer approach however, the logarithmic-law approach proved to be inaccurate when complex geometries with adverse pressure gradients are simulated which is exactly the case of FS race car. [3]

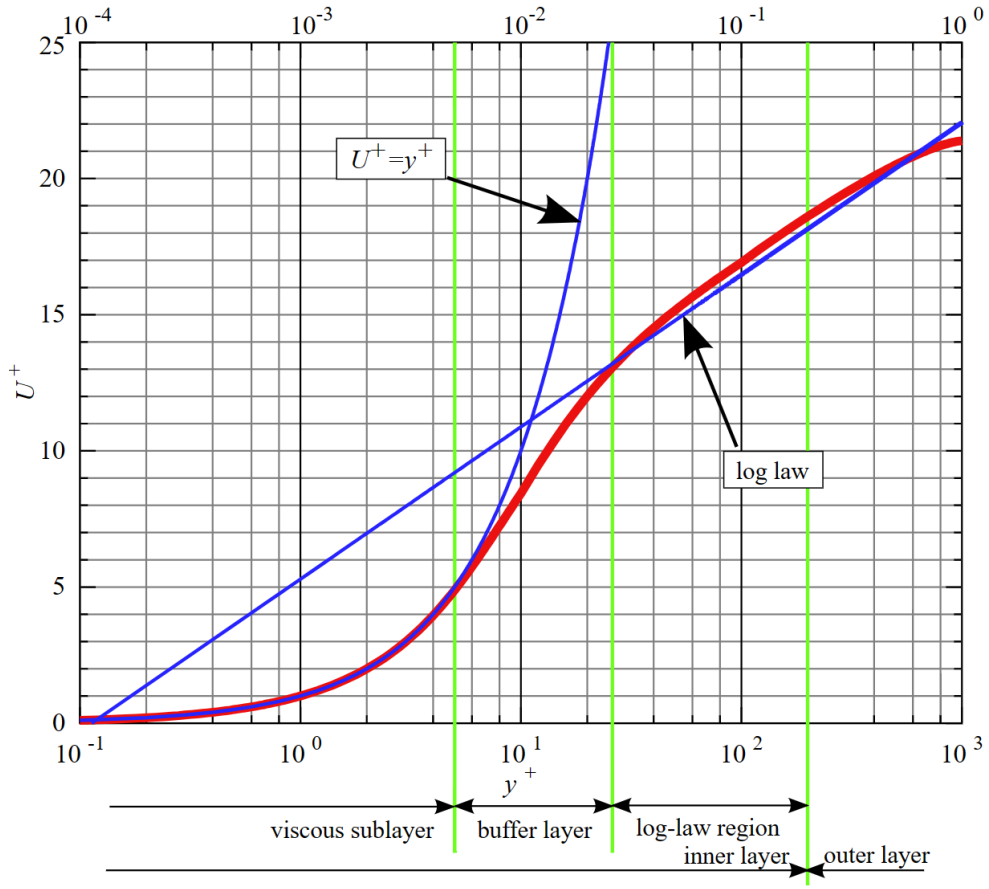

Figure 3.8: Law of the wall [4]

The **growth ratio (=GR)** of each prism should be as low as possible to ensure the most accurate velocity gradient coverage, especially near the wall where the gradient is the greatest. The optimal growth ratio should be recommended by every software developer. According to Ansys Fluent, the optimal growth ratio is 1.2. [3]

Minimal amount of prisms in the prism layer should be recommended by the software developer as well. Again, according to Ansys, the minimal amount of layers should be at least 10 for general simulation purposes and at least 15 layers for complex geometries such as FS race car. [5]

3.3.4 Theory Verification in CFD

To support the previously mentioned theory and recommendations, three simulations were done. For the first two simulations, GOE 226 airfoil was used in 3D CFD simulation. To observe various prism layer setup behaviours in different airflow conditions, two angles of attack (=AoA) were chosen according to Cl/Alpha plot for $Re = 500000$, as shown in Figure 3.9. The angle of attack 0° provides stable airflow around the airfoil whereas airflow separation happens when AoA 15° is reached. To preserve Reynolds number, the chord of the airfoil was set to 0.5 m and freestream velocity to 14.2 m/s. The wingspan was 1.2 m with an endplate (0.575x0.95x0.006 m) at each end to avoid unfavourable vortices.

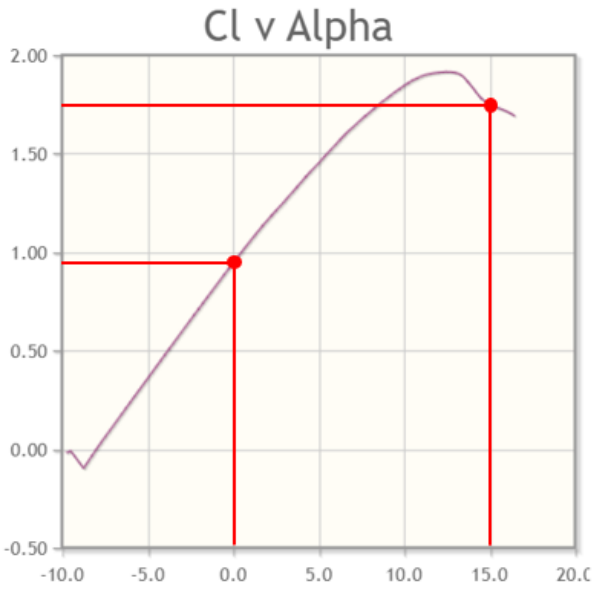

Figure 3.9: GOE 226 Cl/Alpha plot for $Re = 500000$ [6]

For both angles of attack, four different prism layer setups were compared. The first prism height was set to 0.025 mm to reach the desired $y+ \in < 0,5)$ value, and the total height of the prism layer was computed to 13.538 mm according to equation 3.2. The Low_GR means the lowest growth ratio while reaching the lowest height of the overall prisms layer. The Mid_GR was a compromise between the growth ratio and total prism layer height, with a higher growth ratio than recommended and a lower total prism layer thickness than computed. The High_GR version has the highest growth ratio however, it complies with the required total thickness. The last setup is to set a benchmark for other versions as the growth ratio was kept 1.2 and also the minimal total thickness was reached, resulting in a total number of 26 prism layers. Table 3.2 shows specific values of parameters for each setup. It is important to mention that the airfoil had race car orientation so the result of the coefficient of lift will be negative.

Table 3.2: GOE 226 prism layer setup

	1st prisms height [mm]	**GR**	**Layers**	**Total height [mm]**
26 prisms	0.025	1.2	26	14.18
Low_GR	0.025	1.2	15	1.8
Mid_GR	0.025	1.3	15	4.182
High_GR	0.025	1.45	15	14.57

Compared parameters were the following:

- Coefficient of drag - last 100 iterations average
- Coefficient of lift - last 100 iterations average
- Difference [%] from the benchmark simulation with 26 prisms
- Simulation stability in terms of Cl and Cd

After 500 iterations, the following results were obtained for AoA 0°:

Table 3.3: GOE 226 with AoA 0° prisms setup comparison

	Cd (last 100 it. avg.)	**Cl (last 100 it. avg.)**	**Difference [%] (Cd/Cl)**
26 prisms	0.039	-0.564	0/0
Low_GR	0.035	-0,521	10/8
Mid_GR	0.039	-0.569	0/1
High_GR	0.038	-0.563	3/0.2

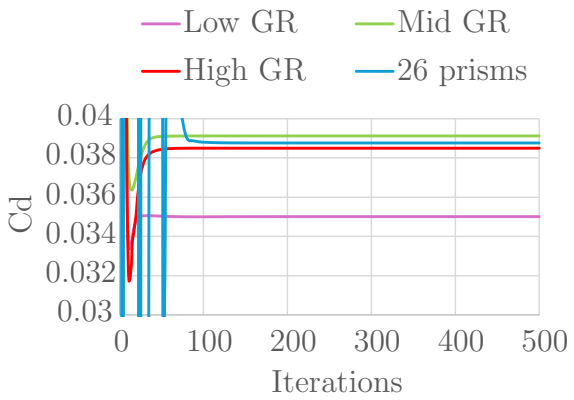

Figure 3.10: Stability of drag report comparison

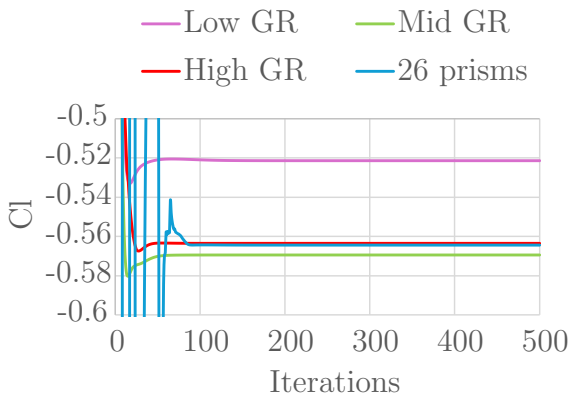

Figure 3.11: Stability of lift report comparison

From the results shown in Table 3.3, Figure 3.10, and Figure 3.11, is visible that all setups provide a stable solution. Both Mid_GR and High_GR are very close to the benchmark whereas the Low_GR is 8-10 % of the benchmark version. However, CFD results of FS race car are expressed in values with two decimal places instead of three so if we were to use this criterion then even the Low_GR version would be 0 % of the benchmark in terms of Cd and 7 % in terms of Cl. Let us compare the AoA 15° to find out if the same trends apply when the airflow is separated from the airfoil.

Compared parameters were the same and after 500 iterations, the following results were obtained for AoA 15°:

Table 3.4: GOE 226 with AoA 15° prisms setup comparison

	Cd (last 100 it. avg.)	**Cl (last 100 it. avg.)**	**Difference [%] (Cd/Cl)**
26 prisms	0.205	-1.562	0/0
Low_GR	0.211	-1.630	3/4
Mid_GR	0.211	-1.600	3/2.5
High_GR	0.204	-1.544	0.5/1.2

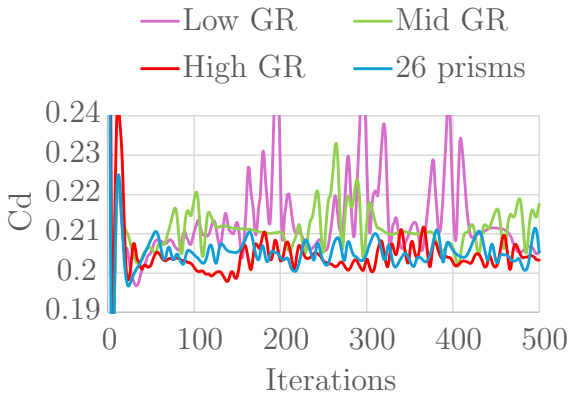

Figure 3.12: Stability of drag report comparison

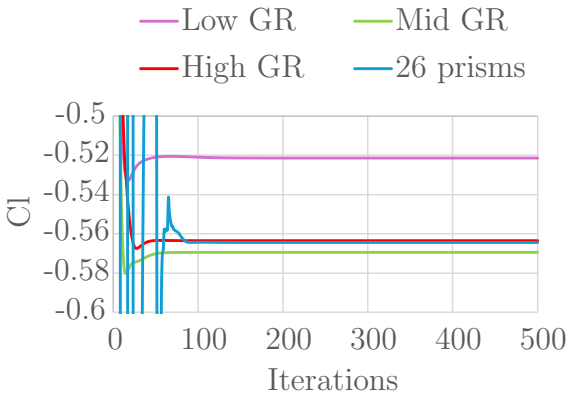

Figure 3.13: Stability of lift report comparison

The results in Table 3.4 show that the greater the overall thickness of the prism layer was, the more similar the result was to the benchmark simulation with 26 prisms. This supports the theory that the prism layer is crucial for capturing the boundary layer and its velocity gradient so the overall thickness should not be neglected even at the expense of a higher growth ratio. Figures 3.12 and 3.13 also show that the lower prism layer thickness was, compared to the estimated boundary layer thickness, the solver faced greater

instability as part of the boundary layer was computed by standard polyhedra cells. A comparison of the velocity field and prism layers is shown in Figures 3.14 and 3.15.

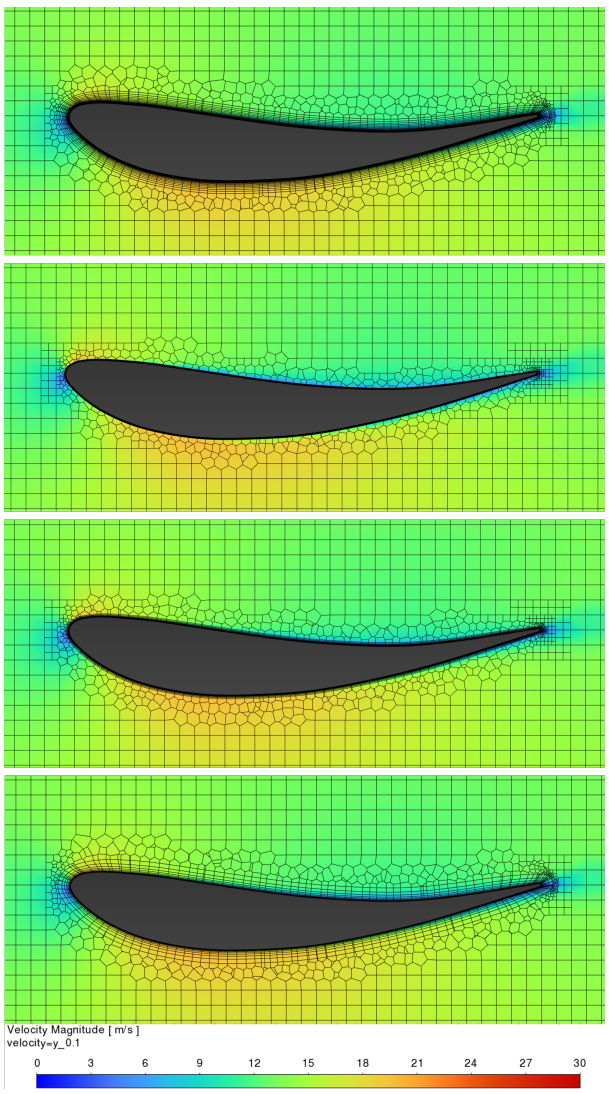

Figure 3.14: Comparison of prism layers for GOE 226 AoA 0°. Descending from the top: 26 prisms, Low_GR, Mid_GR, High_GR.

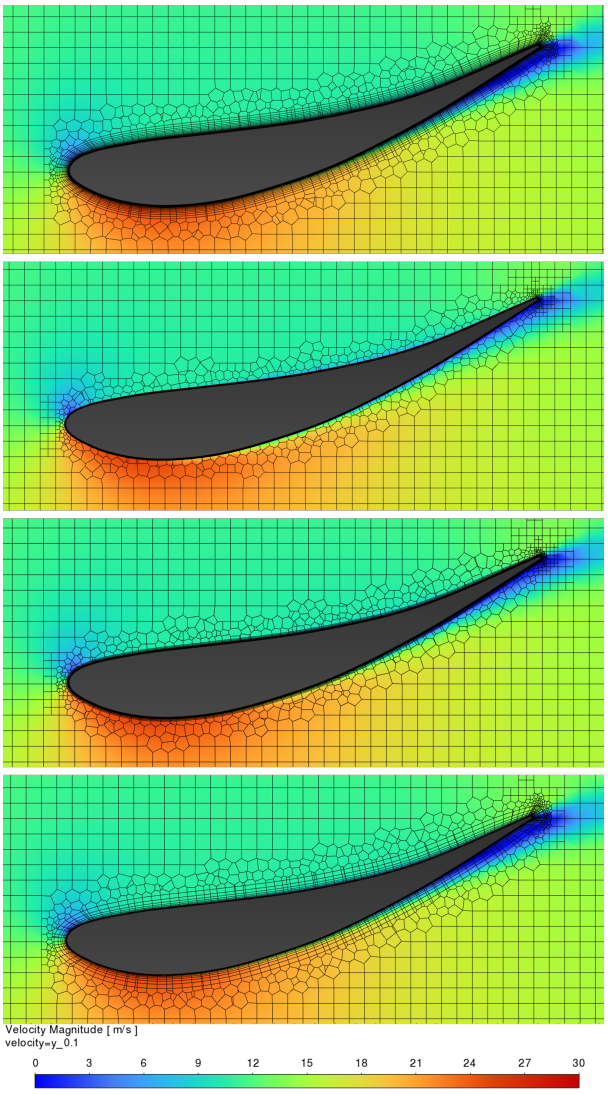

Figure 3.15: Comparison of prism layers for GOE 226 AoA 15°. Descending from the top: 26 prisms, Low_GR, Mid_GR, High_GR.

Finally, to conclude this research about the prism layer and its setup, a real-case scenario was simulated. A rear wing from the development stages of FS race car named CTU.24 was used. The setup, as shown in Table 3.5, was the same as in the case of GOE 226 airfoil except one more version was simulated. The version is called "Optimal" and it applies various growth ratios and total prism layer thickness on specific parts of the wing according to its dimensions so that smaller elements of the rear wing can have a lower growth

ratio while fulfilling the requirements of the boundary layer thickness, thus optimizing the relation between growth ratio and prism layer thickness. Detailed "Optimal" setup parameters are shown in Table 3.7.

Table 3.5: FS race car rear wing prism layer setup

	1st prisms height [mm]	**GR**	**Layers**	**Total height [mm]**
26 prisms	0.025	1.2	26	14.18
Low_GR	0.025	1.2	15	1.8
Mid_GR	0.025	1.3	15	4.182
High_GR	0.025	1.45	15	14.57
Optimal	0.025	1.3 - 1.44	15	4.182 - 13.431

Table 3.6: Optimal prism layer setup

	1st prisms height [mm]	**GR**	**Layers**	**Total height [mm]**
Large Flap	0.025	1.44	15	13.431
Middle Flap	0.025	1.35	15	6.368
Small Flap	0.025	1.3	15	4.182
Endplate	0.025	1.44	15	13.431

Compared parameters were the same as in the case of GOE 226 airfoil and after 500 iterations, the following results were obtained:

Table 3.7: GOE 226 with AoA 0° prisms setup comparison

	Cd (last 100 it. avg.)	**Cl (last 100 it. avg.)**	**Difference [%] (Cd/Cl)**
26 prisms	2.384	-5.424	0/0
Low_GR	2.364	-5.409	1/0.3
Mid_GR	2.309	-5.292	3/2.4
High_GR	2.339	-5.333	2/1.7
Optimal	2.374	-5.412	0.4/0.2

Table 3.7 shows surprising results when the lowest growth ratio version is nearly similar to the benchmark value of 26 prisms. Also, the Optimal version proved to be highly accurate. Because the low-growth version proved to be less accurate when airflow was attached in GOE 226 comparison (and the rear wing is the most extreme version where airflow separation usually happens on FS race car), the "Optimal" setup seems to be the most suitable one for FS race car application. However, according to Figures 3.16 and 3.17, Mid_GR and High_GR provided the best solver stability so for FS race car simulation

I would recommend using the "Optimal" setup where airflow separation is expected and High_GR setup where airflow tends to stay attached.

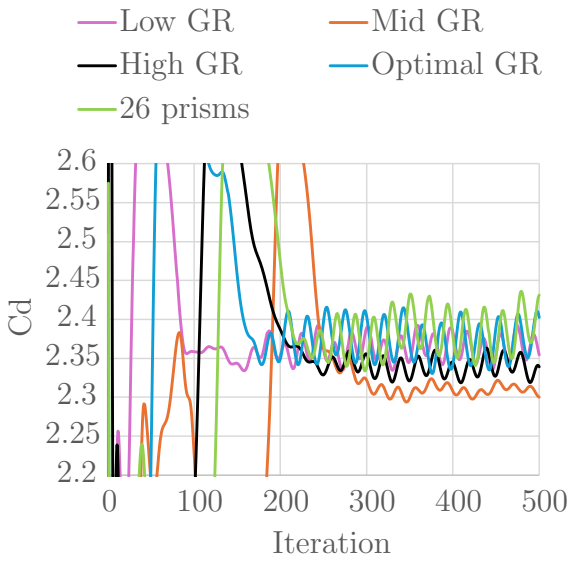

Figure 3.16: Stability of drag report comparison

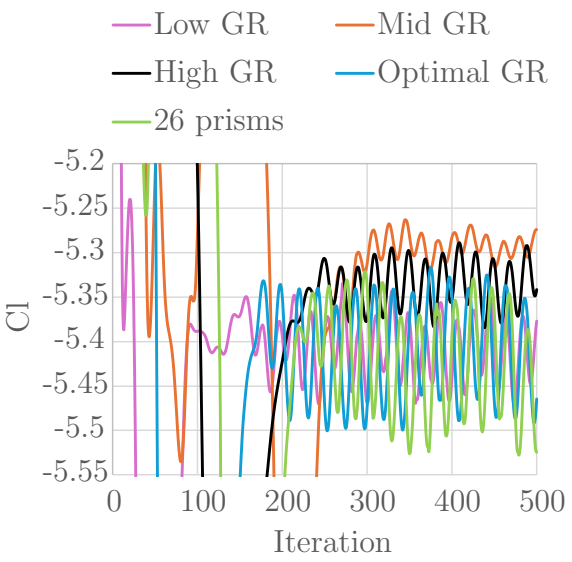

Figure 3.17: Stability of lift report comparison

Chapter 4

Boundary Conditions

Boundary conditions are an essential part of every CFD simulation setup. Some of these conditions are part of almost every possible fluid-flow simulation, such as velocity/pressure inlet/outlet. Every FS race car simulation also has boundary conditions applied to the tunnel floor (moving wall in translational motion) and tyres (also moving wall but in rotational motion which adds tangential velocity to the cell on the tyre surface). These mentioned boundary conditions are not anything new to Formula Student community and every team uses them. However, more advanced boundary conditions that have result on the overall aerodynamic performance of the FS race car exist. This chapter is about these boundary conditions, their setup and comparison.

4.1 Axial Fan on FS Race Car

Every FS race car regardless of its powertrain needs a cooling system. Most of FS internal combustion engines require cooling of the engine and turbocharger. Electric vehicles have similar requirements for cooling - motors, motor controllers, and more advanced ones, autonomous units as well. These cooling systems usually need 1-2 fans per radiator however, since these fans were never significantly powerful (some teams use even PC cooling fans), its correct setup was never a big issue. This was all changed by Formula Student Germany rules 2020v1 when a powered ground effect was no longer restricted. Some teams started experimenting with actively sucked air from the underbody of their cars. Team eForce Prague Formula has decided to apply this innovation to its new car for season 2024 by adding two Schubeler DS-86-AXI HDS 7.5 kW fans behind side wings. Since this is a really powerful axial fan, various approaches for fan boundary conditions in CFD should be compared.

4.1.1 2D fan zone

The first, easiest, and fastest approach is the 2D fan zone. A single plane is inserted between the rotor and stator, usually in the middle of the fan blades area. This plane

has an internal function that makes the plane "invisible" to the airflow but boundary conditions can still be applied to this plane. This area is then changed to 2D fan zone in the solver where the fan curve is defined. The fan curve, as shown in Figure 4.1, provides information about pressure jump dependent on the flow rate through the fan and should be available at every fan manufacturer. The swirl of the fan can be adjusted by modifying the tangential velocity profile in the setup.

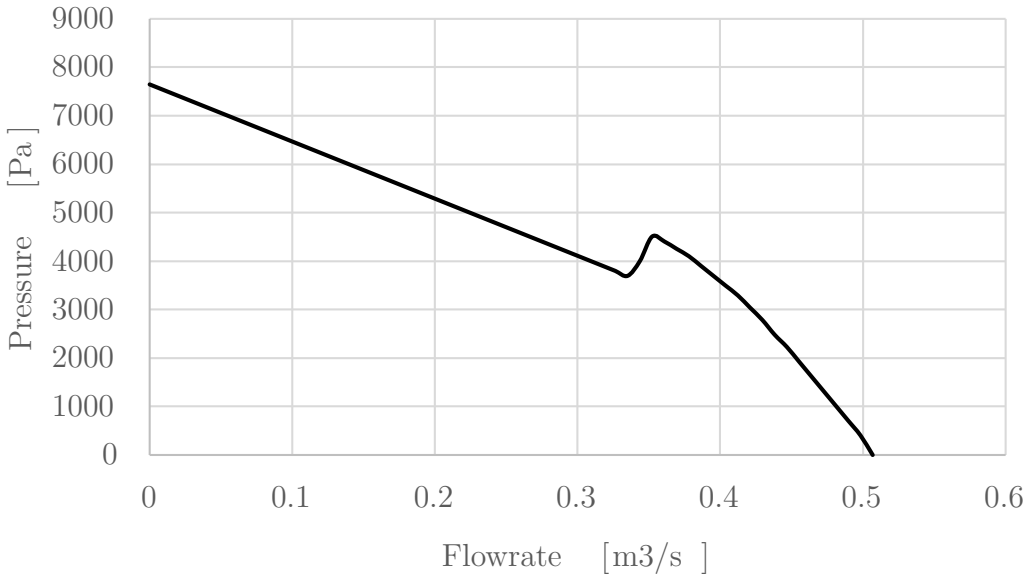

Figure 4.1: Fan curve of Schubeler DS-86-AXI HDS 7.5 kW [7]

4.1.2 3D fan zone

The 3D fan zone boundary condition works similarly to the 2D fan zone. Two planes defining the area of fan blades are inserted into the mesh and then the planes intersect the mesh into two fluid regions. One region is the standard fluid region as in case of FS race car simulation however, the second fluid region is defined as a momentum source, also known as the 3D fan zone function which simulates the effect of an axial fan by applying a distributed momentum source in a toroid-shaped fluid volume. The 3D fan zone offers some advantages over other fan functions[5]:

- 3D fan zone can simulate swirl and radial velocities whereas the 2D fan zone simulates mainly axial flow. Swirl can then be modified by defining the tangential and radial velocities.
- The 3D fan zone offers comparable results to MRF (more about MRF in the next section) without the need to model the blades, resulting in smaller mesh and faster simulation.

4.1.3 Moving Reference Frame

Moving reference frame, also known as MRF, is the most accurate method for the simulation of axial fans and all other kinds of rotating objects with non-filled geometry such as wheel rims, propellers, and turbomachinery. The meshing method is similar to the previously mentioned approaches. The area with fan blades between the rotor and stator is separated by two planes, resulting in two fluid objects. The fluid object containing fan blades is then selected as an MRF. The biggest advantage over 2D/3D fan zone conditions is that Ansys Fluent can use GPU solver for MRF whereas other momentum sources for fan boundary conditions (such as 2D and 3D fan zone) are not available for GPU solver yet. This method, however, has also many limitations such as:

- Because all blades of the fan must be finely wrapped, the resulting mesh is much larger than previously mentioned 2D/3D approaches. This is especially ineffective when MRF condition should be used in the whole FS race car CFD simulation.
- The MRF approach needs an extremely precise CAD model of the fan blades, especially in case of powerful fans like the Schubeler DS-86-AXI HDS 7.5 kW used for eForce Prague Formula CTU.24 car. Even a slight change in geometry can result in inaccurate results. Unfortunately, not many fan manufacturers are willing to share their CAD model of blades used in their fans. Figure 4.2 shows the difference between Schubeler DS-86-AXI HDS real geometry and CAD geometry that is publicly available. Notice the difference in the fan tip shape, slightly different shape of blades, and completely different stator cover behind the blades.[7]

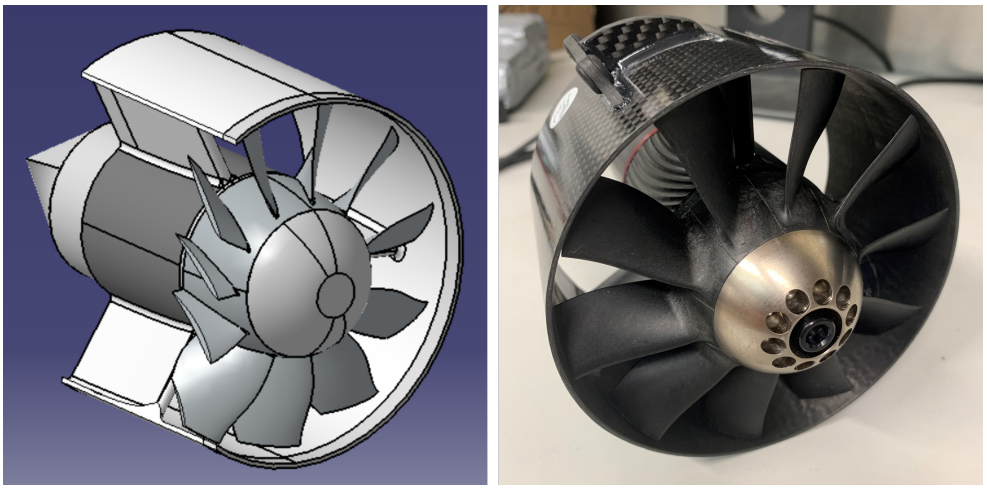

Figure 4.2: Difference between Schubeler fan available CAD geometry (left) and real fan geometry (right)

4.1.4 Fan Boundary Conditions Comparison

All three mentioned approaches were simulated and compared to find the best possible solution for FS race car simulation. Compared parameters were the following:

- Mass flow rate through the fan
- Area weighted average velocity through fan inlet
- Mesh size
- Time of numerical solution

The results in Table 4.1 show a similarity between 2D and 3D fan zones. On the other hand, the MRF boundary condition shows completely different results, presumably because of the previously mentioned differences in fan casing and blades geometry. Because of that, only 2D and 3D fan zones will be presumed as the optimal choice for FS race car CFD simulation. The 2D fan zone requires fewer cells because the mesh refinement around the fan blades area is simplified to a single plane instead of a tortoid shape, resulting in faster computational time. The mass flow rate is almost identical which is no surprise since both fan boundary conditions operated only with the axial source term defined by the fan curve.

Table 4.1: 2D fan zone, 3D fan zone, and MRF comparison

Boundary condition	**2D fan zone**	**3D fan zone**	**MRF**
Mass flow rate [kg/s]	0.546	0.554	0.707
Area weighted average [m/s]	50.787	54.575	133.789
Mesh size	4.2 mil	4.35 mil	10.1 mil
Time of numerical solution	70 min (CPU)	90 min (CPU)	17 min (GPU)

The biggest difference was in the velocity field as shown in Figure 4.3 and Table 4.1. The picture clearly shows that the 3D fan zone estimates much greater velocity behind the fan. This finding is important especially for the aerodynamic design of FS race car. The fan outlet velocity and swirl can significantly influence the behaviour of other aerodynamic parts in near proximity to the fan. Unfortunately, it is hard to conclude which of these boundary conditions is more accurate for this specific case. In general, the 3D fan zone should provide more complex and accurate results since it includes swirl and radial velocities by default however, we can not conclude the real accuracy of these boundary conditions to the true fan performance. Because of that, extensive research and data measurement should be made when FS team decides to implement such a powerful fan into its aerodynamical concept. By measuring the velocity and pressure field around the fan, the boundary conditions setup can be then adjusted to match the actual fan performance.

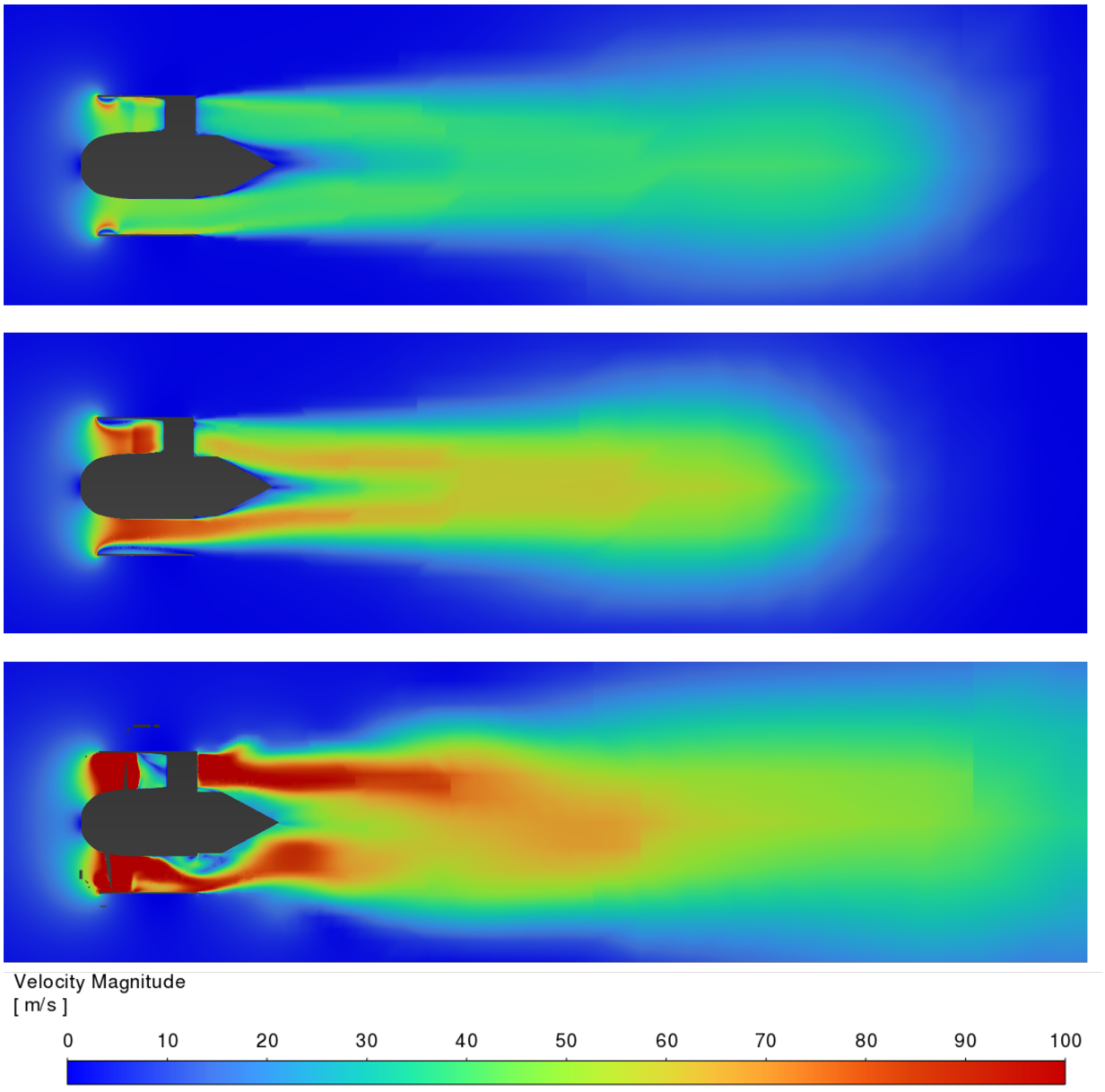

Figure 4.3: Velocity field for different fan boundary conditions. Descending from the top: 2D fan zone, 3D fan zone, and MRF

4.2 Radiator on FS Race Car

The radiator of FS race car is simulated as a porous zone. The setup is the same as in the case of the 3D fan zone - two planes marking the inlet and outlet of the radiator core are intersected with the radiator frame. Two fluid regions are then created and the fluid region

between these intersected planes is defined as a porous zone. Thanks to the research of former CTU Cartech member, Otakar Volek, team eForce Prague Formula now possesses a radiator curve which shows the pressure drop relevant to freestream velocity. Ansys Fluent allows setting the porous zone behaviour as a power law of the velocity magnitude according to equation 4.1:

$$S_i = -C_0|v|^{C_1} \tag{4.1}$$

where S_i is the source term, C_0 and C_1 are user-defined empirical coefficients. [5]

The data gathered by Otakar Volek can be fitted to the power law curve which shows the required empirical coefficients for our porous zone setup as shown in Figure 4.4. For our case $C_0 = 604.76$ and $C_1 = 1.531$ then.

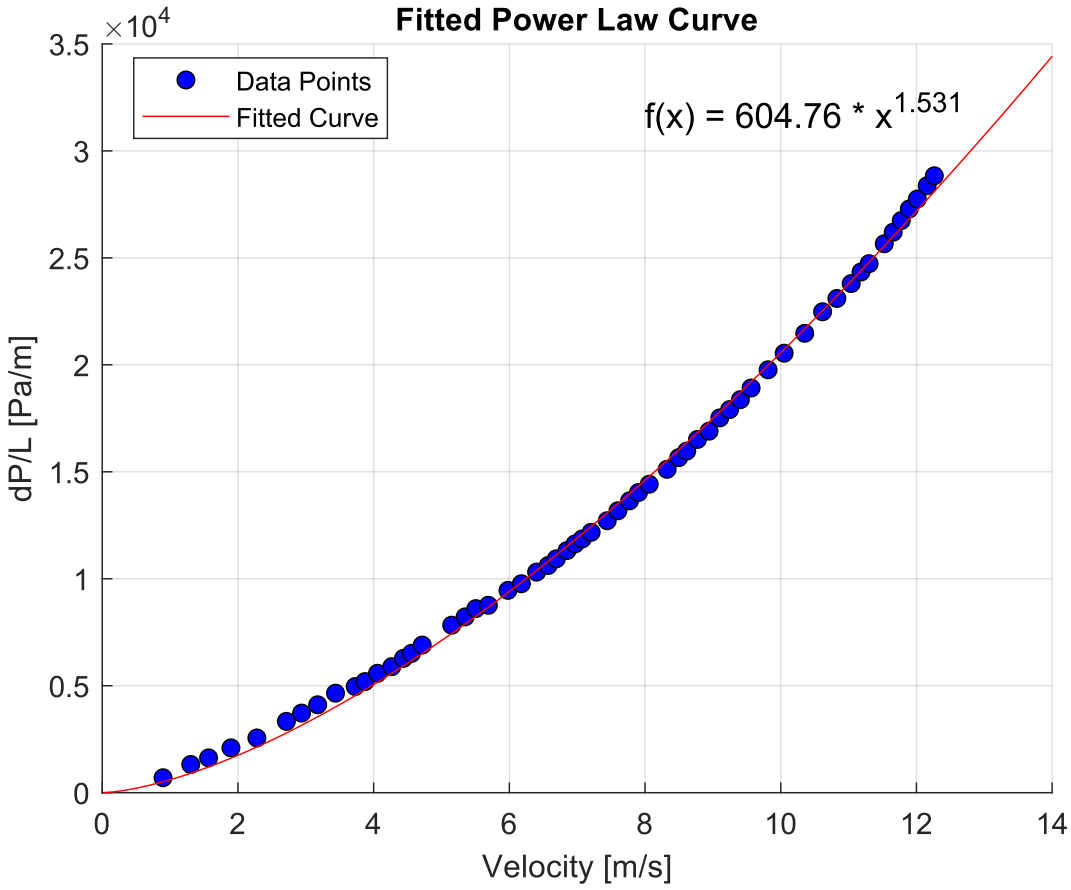

Figure 4.4: Data from radiator measurement fitted to power law curve

This section demonstrated how boundary conditions for FS race car radiator can be defined and Figure 4.5 shows how such a radiator can work in CFD simulation. Please note that this case was made only for demonstration purposes and the shape of the radiator is not the same as on a real FS race car (however the boundary conditions would remain the same in both cases).

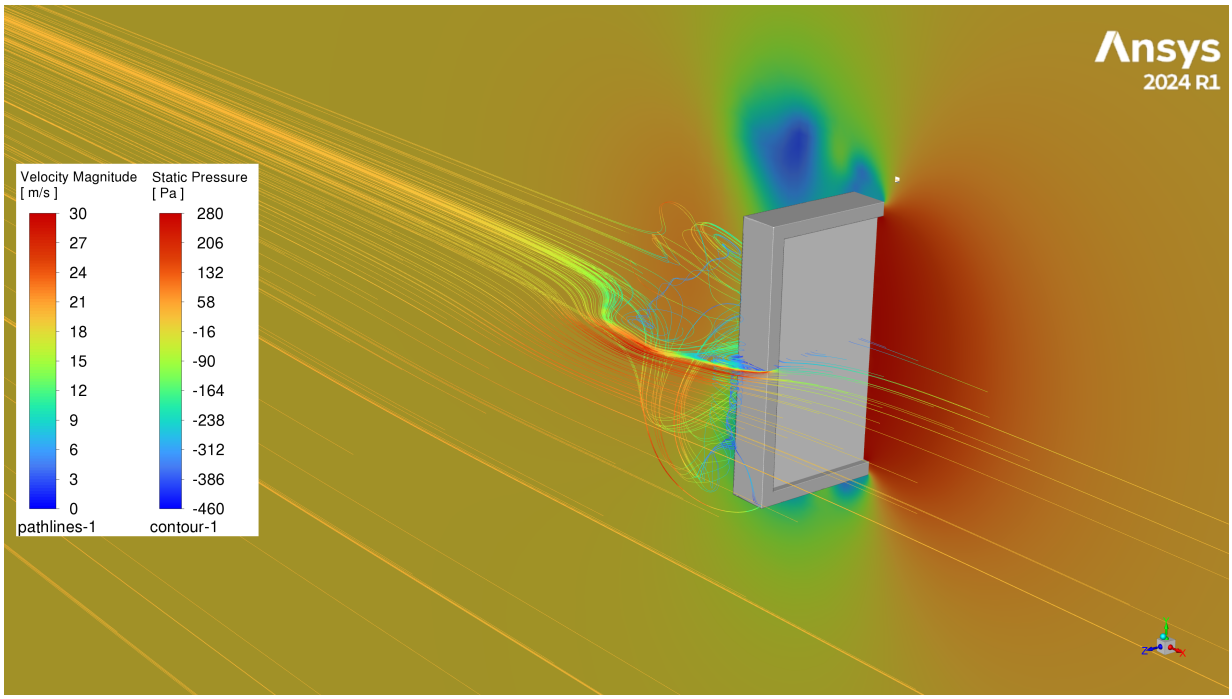

Figure 4.5: Contour of static pressure and velocity pathlines around radiator

Chapter 5

Turbulence Models

As we discussed in section 3.3, every laminar flow might change to turbulent according to various criteria such as Reynold number and complexity of the geometry. The transition between laminar and turbulent flow is solved by three approaches. The first approach is called DNS (Direct Numerical Solution) which solves fully unsteady Navier-Stokes equations however, this approach demands an extreme computational power and is only applicable to problems with simple geometry and low Reynold numbers so only academic research can be made with this approach. The second method called LES (Large Eddy Simulation) solves the largest eddies directly and the smaller ones are modeled. It is a less power-consuming method than DNS but the mesh and computational power requirements are still often beyond the possibilities of industry solutions. Because of that, the third method called RANS (Reynolds Averaged Navier-Stokes Simualtion), is commonly used for industry applications. This approach solves time-averaged flow where all eddies are modeled. Various mathematical models exist for this approach but only two of them will be mentioned and compared in this work. [3]

5.1 k-epsilon Models

It is important to imagine k-ϵ more like a family of various models than just one turbulence model. All of them are 2 equational models with one equation for turbulent kinetic energy (k), and the second one for turbulent dissipation rate (ϵ). Three varieties of the k-ϵ model exist in Ansys Fluent:

- Standard k-ϵ
- RNG k-ϵ
- Realizable k-ϵ

All these models share similar forms of transport equations however they differ in other aspects such as the method of calculating turbulent viscosity. Both RNG and Realizable

models offer a significant improvement in terms of streamline curvatures, vortices, and rotation over the Standard k-ϵ model. Anyway, all of these models show insensitivity in cases where adverse pressure gradient and boundary layer separation are present. Because of that, these models are not recommended for external flow simulations where they tend to show too optimistic results in terms of aerodynamic performance. This is exactly the case of FS race car simulation and I did not intend to mention nor anyhow compare these models with more advanced ones such as SST k-ω but to my unpleasant discovery many FS teams use the k-ϵ based models since they require less computational power, use smaller meshes, and show very stable results (not necessarily accurate). This was exactly the case of FS team eForce Prague Formula that used a Realizable k-ϵ model with enhanced wall treatment throughout the 2024 season. This method was developed by Ing. Martin Ševčík in his Masters degree thesis "Aerodynamics design methodology of Formula Student car". It is a very well-written thesis where he compared Realizable k-ϵ model with enhanced wall treatment and SST k-ω model in similar CFD software to Ansys Fluent. He also substantiated his simulations by pressure-strips validation on real FS race car. This research showed that between these two turbulent models, there was a 4% difference in the coefficient of lift and almost a 5% difference in the centre of pressure where the biggest difference in result was at the rear wing where airflow separation is expected due to the high angle of attack. This difference is also supported by Figure 5.1 where the Realizable k-ϵ model with enhanced wall treatment (V1_a, V2_a) shows approximately -0.5 better coefficient of lift than the SST k-ω model (V1_b, V2_b). Lines 100_1, 100_3, and 100_5 show the data from pressure strips. [2] [5] [8]

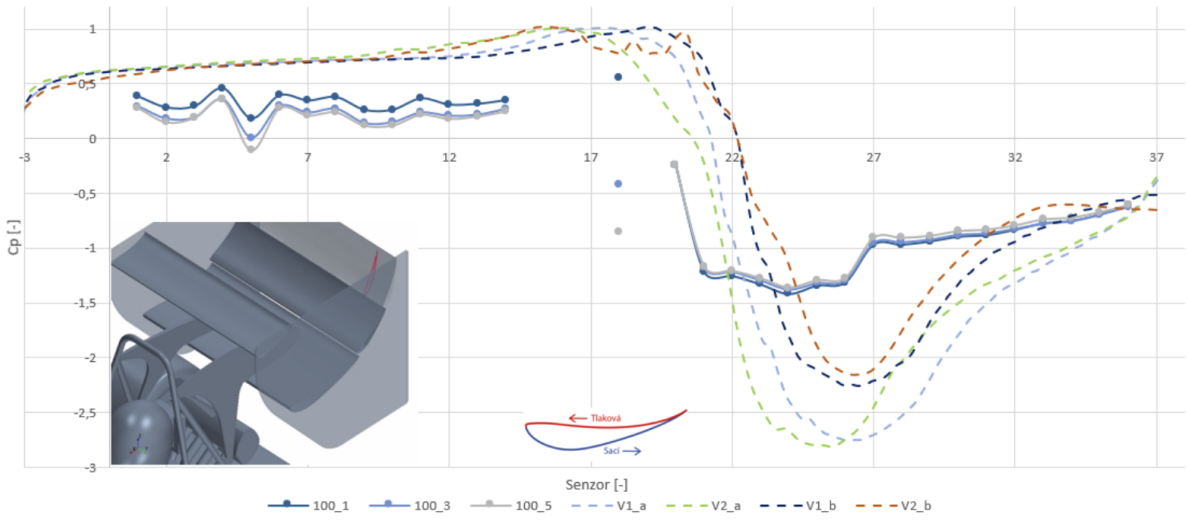

Figure 5.1: Validation of CFD simulations via pressure strips [2]

Another example of k-ϵ models inaccuracy was observed by Ing. Lukáš Pacoň in his Bachelor thesis "CFD Simualtion Validation Tests in Wind Tunnel" where he compared the k-ϵ model with the data from wind tunnel measurement of FS race car rear wing. The data from Figure 5.2 show a significant difference between the simulation (red line means downforce, green line means drag) and wind tunnel measurement (blue line means downforce, orange line means drag). X-axis shows force [N] and y-axis shows freestream velocity [m/s]. Notice the large difference in lift as the speed increases. For FS race car simulation where the average speed is considered to be 15 m/s, the difference was almost 20 %. [9]

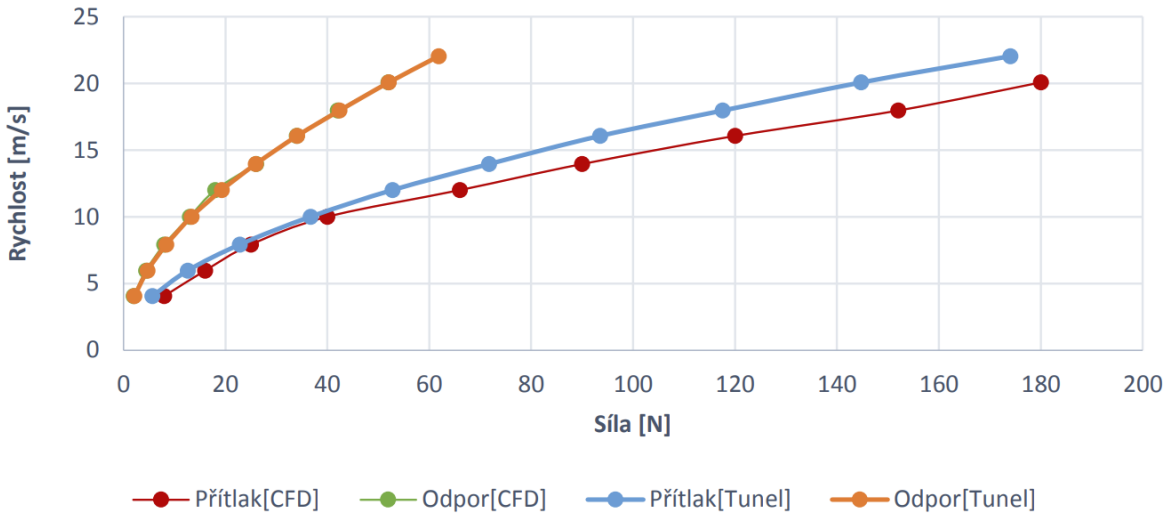

Figure 5.2: Validation of CFD simulations via wind tunnel measurement [9]

To conclude all the facts mentioned above, the Realizable k-ϵ model with enhanced wall treatment and all other k-ϵ based models are very inaccurate for needs of FS race car simulation. Even though they require less computational power, smaller and far more primitive meshes, and provide stable solutions. All these aspects might be favourable for new FS teams who are inexperienced in meshing or they do not use any aerodynamic assemblies with parts of high angle of attack or vortex generators. However, for FS team eForce Prague Formula with its computational hardware capabilities and long CFD experience, k-ϵ models should never be used.

5.2 k-omega Models

The k-ω as well as k-ϵ in not only one model. Ansys Fluent offers three k-ω based models.

- Standard k-ω
- BSL k-ω
- SST k-ω
- Generalized k-ω (GEKO)

The k-ω based models offer many advantages over the k-ϵ models, especially in terms of flow prediction in viscous sublayer. They are known to better and more accurately predict adverse pressure gradients, which in terms of aerodynamic engineering, means airflow separation and less downforce. However, the Standard k-ω model faces problems with large sensitivity in the freestream area far from the wall. This problem was solved by introducing SST k-ω turbulence model which uses ω model for near-wall turbulence modelling and ϵ modelling for freestream. For complex geometries like FS race car where many vortices and possible flow separations are expected, the SST k-ω is more suitable than k-ϵ turbulence model according to Ansys. When no experimental data are obtained from real-life testing, the default constants of each turbulence model are recommended to be preserved. When a large amount of data is collected from real-life testing, the most advanced model called GEKO can be used to further adjust the turbulence model for the specific needs of FS race car simulations. This model allows the user to further define additional constant parameters without having a negative impact on the basic calibration of the model. Choosing the GEKO model might be the next step for FS race car CFD simulation however, strong real-life data and a proper understanding of the impact of these constants on simulation results are required. [5]

5.3 Models Comparison

In the previous sections, we have mentioned all important turbulence models that can be used in FS race car applications. The two main models, Realizable k-ϵ and SST k-ω were compared in order to find the best possible solution for CFD simulations within eForce Prague Formula team. The Realizable k-ϵ model with enhanced wall treatment used by Ing. Martin Ševčík in alternative software to Ansys Fluent as mentioned in Section 5.1. and used by eForce Prague Formula during the 2024 season (in the following comparison named "2024 model") was compared with the SST k-ω turbulence model. It is important to mention that both simulations differ in meshing approach according to the needs of each turbulence model.

Compared parameters were the following:

- Coefficient of drag - last 100 iterations average
- Coefficient of lift - last 100 iterations average
- Centre of pressure (CoP)
- Mesh size
- Time of numerical solution

The results after 1000 iterations are shown in Table 5.1 below:

Table 5.1: Turbulence models comparison

Turbulence model	**"2024 model"**	**SST k-omega**
Coefficient of drag (last 100 it. avg.)	1.767	1.699
Coefficient of lift (last 100 it. avg.)	-5.04	-4.697
Centre of pressure (% rear axle)	49.5	53.3
Mesh size	20 mil	33.5 mil
Time of numerical solution	40 min (GPU)	8 hours (CPU)

Table 6.1 clearly shows a huge difference in results between both turbulence models. The "2024 model" shows a worse coefficient of drag by almost 4 %. However in the most important parameter for vehicle performance, coefficient of lift, the "2024 model" is approximately 7 % more optimistic than the SST k-ω. Moreover, the difference in centre of pressure was 3.8 % however, the fact that the centre of pressure has shifted rearwards in the case of SST k-ω is quite surprising. According to the general theory of turbulence models, the rear wing where flaps with the highest angles of attack occur should have the most adverse pressure gradient from all other car parts, therefore the SST k-ω should solve the separation more accurately, resulting in lower downforce of the rear wing that, in conclusion, results in the centre of pressure shifting toward the front of the car, not otherwise. However, the SST k-ω converged around the 200th iteration and then started oscillating with an amplitude of 2 % around the final value whereas the "2024 model" did not converge even after 1000 iterations so we might assume that even the centre of pressure shown by SST k-ω model is more accurate than the "2024 model". The "2024 model" had around 10 million cells less, resulting in much faster computational time mainly because the GPU solver which, in the case of SST k-ω, required more GPU memory than was available by PC hardware so the standard CPU solver had to be used. More about CPU and GPU solver in Section 6.3. Another important aspect of every CFD simulation is y+ value which in the case of SST k-ω was everywhere below 4.5 except suspension and rollbar tubes as shown in Figure 5.3. In the "2024 model" around 50 % the cells were in the range of $y+ \in \langle 5, 30 \rangle$ and many of them were even in areas where flow separation is expected.

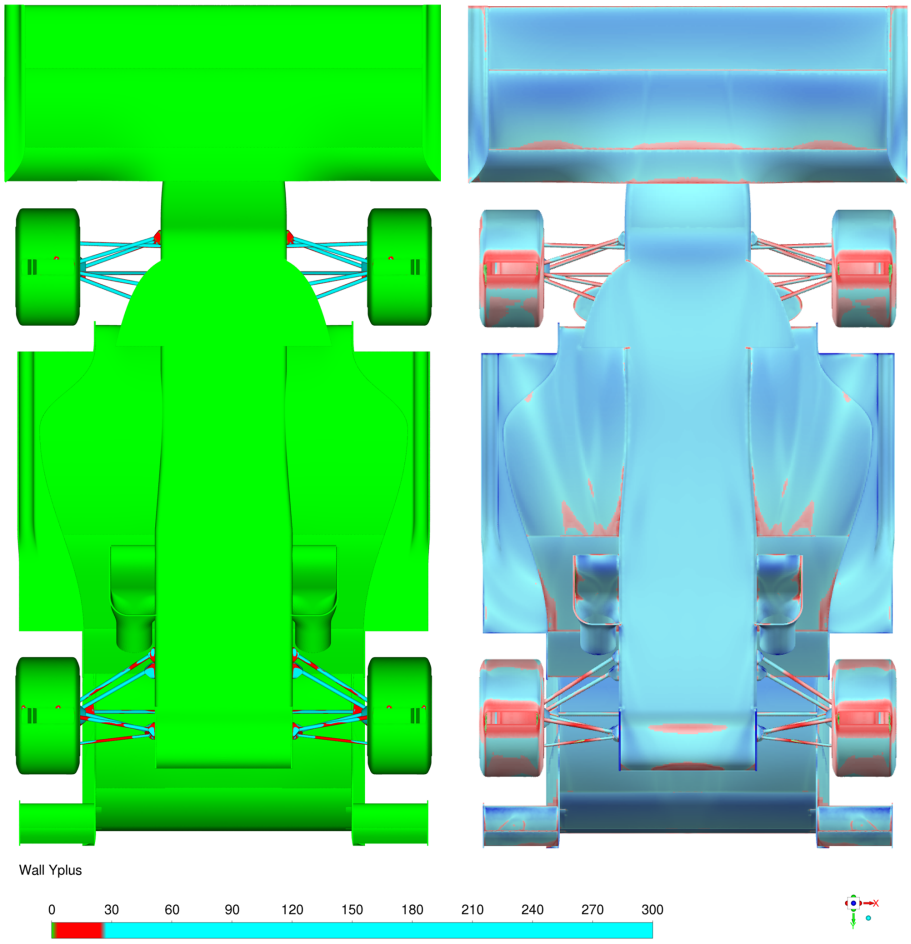

Figure 5.3: Comparison between y+ values in SST k-ω (left) and "2024 model" (right) simulations. The contours show green for $y+ \in < 0, 5)$, red for $y+ \in \langle 5, 30 \rangle$, and blue color for $y+ \in (30, 300 \rangle$

The difference in coefficient of lift is shown in Figure 5.4. Notice how the "2024 model" si more optimistic in pressure coefficient at all parts of the car. An especially large difference is visible on the first flap of the front wing as well as the centre part of the whole front wing. Pressure contours on the side wing show a lower pressure coefficient throughout the whole surface of the lower skin except of the edge of the side wing endplate where large vortex is created. This further shows how the "2024 model" is less sensitive to airflow separation from the surface and is also worse in terms of vortices prediction. Another difference worth mentioning is the middle part of the rear wing where the SST k-ω shows much greater airflow separation caused by the driver's helmet and autonomous systems camera on the roll bar.

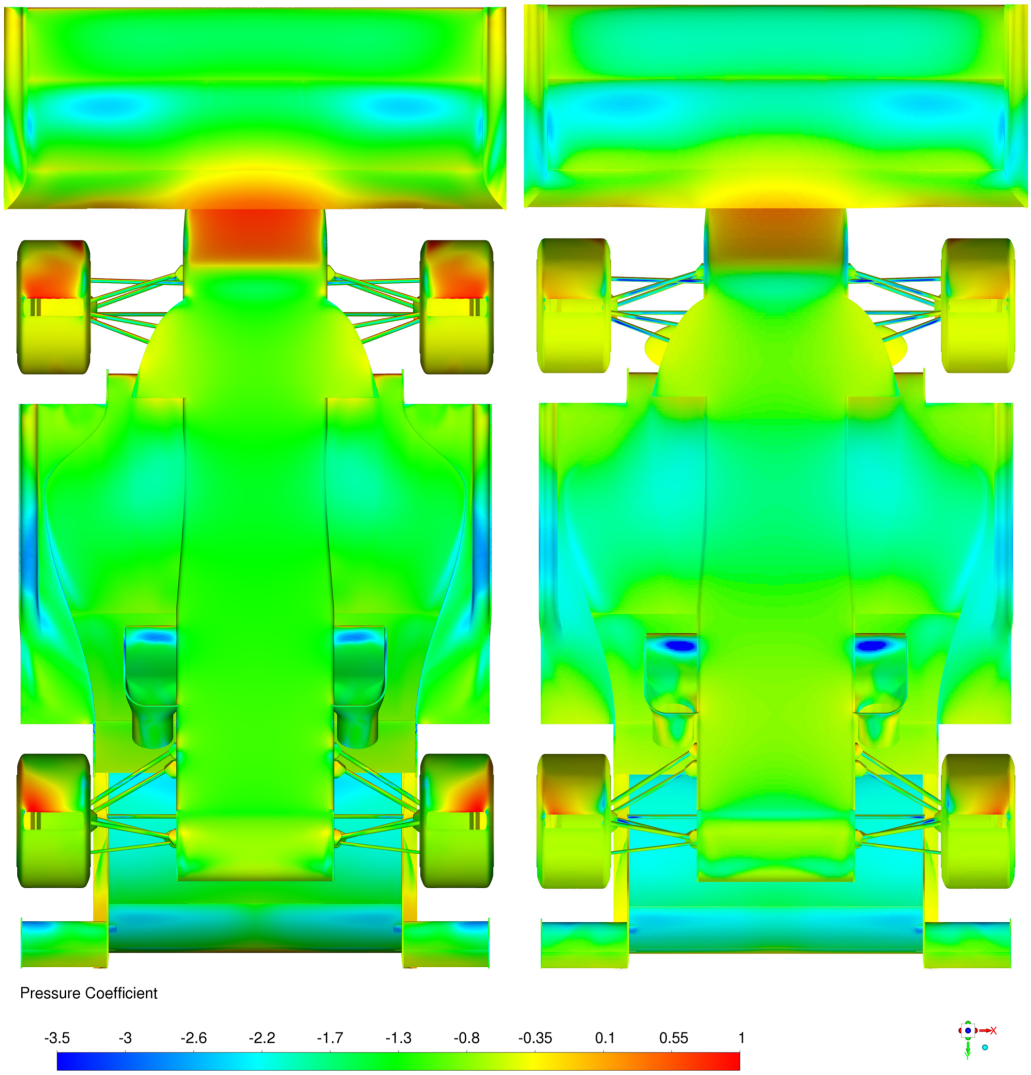

Figure 5.4: Comparison between pressure coefficient in SST k-ω (left) and "2024 model" (right) simulations

This comparison proves that only SST k-ω or more advanced turbulence models should be used for FS race car CFD simulation even though a more advanced meshing approach, larger mesh size, more hardware capabilities, and more computational time are required. In case of lower hardware capabilities, simpler bodywork geometry with a precise turbulence model is a better option than a simpler turbulence model in exchange for complex geometry with multiple-element wings and vortex generators on a car. Remember that the quality of results should always stand before the quantity!

Chapter 6

CFD Solver Computation

Now that we finished the simulation setup, let's discuss the importance of the solver computation setup. Even though this topic might seem like a different category of CFD simulation, it is tightly connected to the mesh and solver setup. Different meshes with various boundary conditions, especially large and complex ones, might require different solver approaches. Also, there should be differences in terms of computational time, stability, accuracy, and hardware requirements regarding the selected approach of the solver computation. This chapter focuses on two main aspects of CFD solver computation - Simple/Coupled scheme and comparison of GPU/CPU solver.

6.1 Single and Double Precision

Every numerical simulation needs some standard in terms of number representation to ensure accurate results throughout all different simulations and computational software. According to IEEE 754 Standard, two main approaches to number format exist. These numbers are called floating-points. Computer software uses binary digits (also known as bits) instead of decimal places and there are two main approaches to this floating-point representation, single-precision and double-precision. These approaches differ in bits used to represent actual numbers. For better understanding let's assume an irrational number like π=3.1415926535...Using for example π=3.1415926 during calculation will result in a more accurate result than using only π=3.14. The single-precision computing uses 32 bits to represent the actual number, 8 bits are used for the exponent, and 23 bits for mantissa (to represent the fractional part), whereas double-precision uses 64 bits, 11 bits used for the exponent and 52 bits for the mantissa. In general, the single-precision method is faster in terms of computational time but lacks the accuracy of the double-precision method, which is also preferable for complex simulations. [10]

To find out how this theory could affect the result of FS race car simulation, simple research was performed. Both simulations were computed with exactly identical model (CTU.24_demo), mesh, and other settings. The computation length was set to 2000 iterations to observe possible differences in convergence and overall stability. The solution

Method was set to SIMPLE (more about solution methods in Section 6.2) because of the GPU solver memory requirements. Compared parameters were the following:

- Coefficient of drag - last 100 iterations average
- Coefficient of lift - last 100 iterations average
- Solver stability
- Centre of pressure (CoP)
- Time of numerical solution
- Solver stability
- GPU memory usage
- RAM usage

The results are shown in Table 6.1 and Figures 6.1, and 6.2 below.

Table 6.1: Sigle and double precision comparison

Precision	**Single**	**Double**
Coefficient of drag (last 100 it. avg.)	1.64	1.64
Coefficient of lift (last 100 it. avg.)	-3.67	-3.64
Centre of pressure (% rear axle)	54.9	54.8
Time of numerical solution	33 min	33 min
GPU memory usage	28.6 GB	42.7 GB
RAM usage	97 GB	111 GB

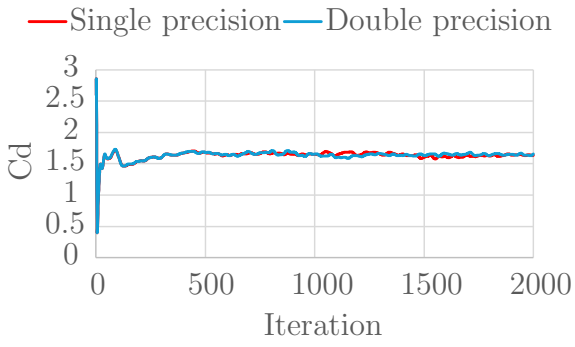

Figure 6.1: Stability of drag report comparison

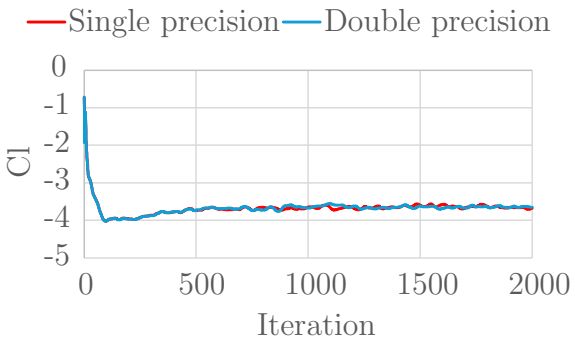

Figure 6.2: Stability of lift report comparison

From the values summarized above we can assume that there is no significant difference in the result between single and double precision. The oscillation of the most important

parameter of FS race car, lift, was within 1 % in both cases. The difference in the centre of pressure was also 0.1 %. A very noticeable difference is however in hardware requirements, where a single precision solver used ≈ 33 % less GPU memory and ≈ 13 % less RAM. This knowledge will be used in the following section, where various solution methods will be compared.

6.2 Solver Methods

This section compares solver methods. There are two types of solvers. Density-based solver which solves equations for continuity, momentum, and energy in vector form. This solver is used when a strong coupling or interdependence exists between density, energy, and momentum. Some exemplary use cases are for example high-speed flow with combustion, hypersonic flow, or shock interactions. The second type of solver, pressure-based, takes momentum and pressure as the primary variables. It is widely used since this solver can be applied to most engineering applications where incompressible or low-speed compressible flows are assumed. For FS race car simulation, a pressure-based solver is used. This solver also offers two types of algorithms for pressure-velocity coupling. Coupling methods are numerical algorithm that uses a combination of continuity and momentum equations to derive an equation for pressure (or pressure correction) when using a pressure-based solver. In this section, we will compare the most common methods used in FS - segregated (also known as SIMPLE) and coupled method. The main difference between these two algorithms is that SIMPLE solves the momentum equation and pressure-correction equation sequentially, whereas the coupled algorithm solves the momentum and pressure equations simultaneously. Because of that, the convergence of SIMPLE method is usually slower but 1.5-2x less memory is required. Both solver methods should have theoretically the same accuracy if both reach the same convergence criterion. [11]

To compare these two methods, model CTU.24_demo was used. Solver precision was set to single, as we demonstrated no influence on the result in the previous Section 6.1 and GPU solver was used (GPU solver will be discussed in the next Section 6.3). The computation length was set to 2000 iterations to observe possible difference in convergence and overall stability.

Compared parameters were the following:

- Coefficient of drag - last 100 iterations average
- Coefficient of lift - last 100 iterations average
- Centre of pressure (CoP)
- Solver stability
- Time of numerical solution
- Convergence speed
- GPU memory usage

- RAM usage

The results are shown in Table 6.2 and Figures 6.3, 6.4 below.

Table 6.2: SIMPLE and Coupled method comparison

Solver method	**SIMPLE**	**Coupled**
Coefficient of drag (last 100 it. avg.)	1.64	1.59
Coefficient of lift (last 100 it. avg.)	-3.67	-3.57
Centre of pressure (% rear axle)	54.9	54.3
Time of numerical solution	33 min	135 min
GPU memory usage	28.6 GB	69.3 GB
RAM usage	97 GB	99 GB

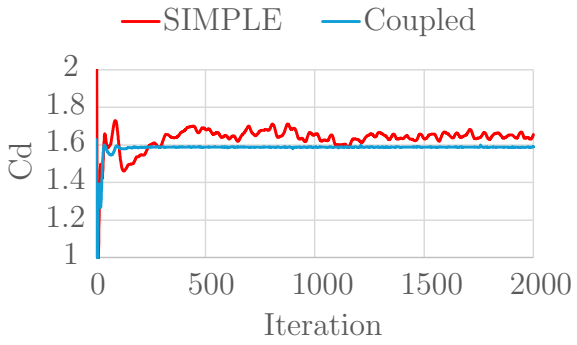

Figure 6.3: Stability of drag report comparison

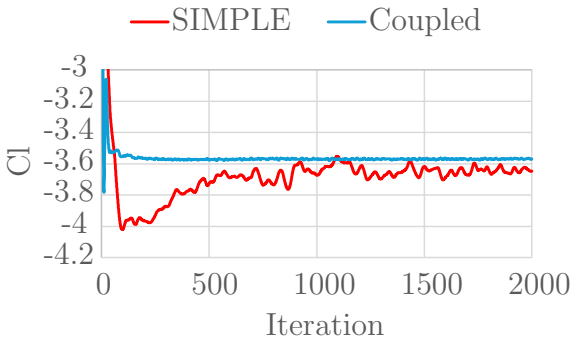

Figure 6.4: Stability of lift report comparison

From monitored results, we can assume that the coupled method is less "optimistic" in terms of downforce which is always the better assumption for FS race car CFD simulation since vehicle performance and lap time simulations are connected to the aerodynamical performance of the car. Too optimistic results might lead to false perception about the real vehicle performance. The computational speed difference has proved to be significant as well. Also, Figure 6.4 shows that the coupled method is more stable and converges faster, see the difference in the first 500 iterations. The result of the final Cl in the SIMPLE method could be surely approximated from a section of 1500 - 1600 iterations whereas the final Cl in the Coupled method could be surely approximated from a section of 300 - 400 iterations, resulting in a reduction of computational time of Coupled method to approximately 27 minutes which is even faster than the SIMPLE method.

This leads to the conclusion that the Coupled method with its current settings is more suitable for FS race car CFD simulation. However, further research within eForce Prague

Formula team might be done to find out whether the SIMPLE method with finer meshing might achieve faster computational time while maintaining the same accuracy.

6.3 CPU vs GPU Solver

CPU solver has been widely used and established for a long time however, in recent years, GPU acceleration became popular thanks to its speed incomparably faster than CPU solver. In the 2023/2024 FS season, the team eForce Prague Formula was for the first time in its 16-year history able to run CFD simulation calculated by GPU solver thanks to generous sponsorship from Lenovo, providing the team with powerful workstations capable of GPU acceleration and solution. This section compares CPU and GPU solvers in terms of their speed, stability, accuracy, and also available features within the solver.

The comparison was performed on the following hardware with model CTU.24_demo:

- CPU: AMD Ryzen Threadripper PRO 5995WX 64-Cores, Base speed: 2.7 GHz, Boost frequency: 4.5 GHz, Logical processors: 128[1], Threads: 7088
- GPU: 2x NVIDIA RTX A6000, Dedicated GPU memory: 47.5 GB, Shared GPU memory: 128 GB[2]
- RAM: 256 GB

The compared simulation setups are shown in Table 6.3. The reason for choosing single precision in GPU solver was the GPU's limited memory[3], however, in Section 6.1 we have compared the difference between single and double precision which showed to be irrelevant for our case. The computation length was set to 1500 iterations to observe possible difference in convergence and overall stability.

Table 6.3: Simulation setups

Solver	**CPU**	**GPU**
Solver method	Coupled	Coupled
Precison	Double	Single

[1]Only 30 processors were used during simulation. When selecting more than 30 processors, the PC tended to shut down, probably because of an insufficient power supply unit inside the PC

[2]Shared memory could not be used since the computer lacked NVLink

[3]Coupled method with double precision was performed as well but the simulation failed during initialization due to lack of GPU dedicated memory

Compared parameters were the following:

- Coefficient of drag - last 100 iterations average
- Coefficient of lift - last 100 iterations average
- Centre of pressure (CoP)
- Solver stability
- Time of numerical solution
- Convergence speed
- GPU memory usage
- RAM usage

The results are shown in Table 6.4 and Figures 6.5 and 6.6.

Table 6.4: CPU and GPU precision comparison

Solver	**CPU**	**GPU**
Coefficient of drag (last 100 it. avg.)	1.56	1.59
Coefficient of lift (last 100 it. avg.)	-3.58	-3.57
Centre of pressure (% rear axle)	54.3	54.3
Time of numerical solution	15 hrs	2.25 hrs
GPU memory usage	0 GB	69.3 GB
RAM usage	141 GB	99 GB

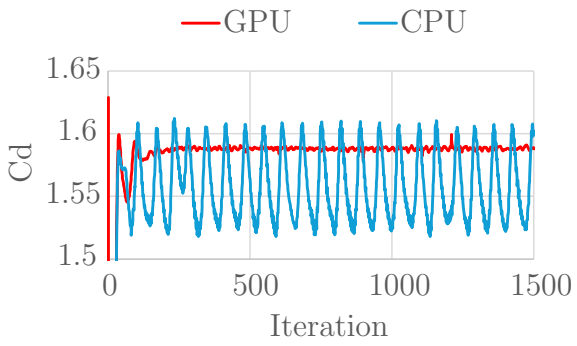

Figure 6.5: Stability of drag report comparison

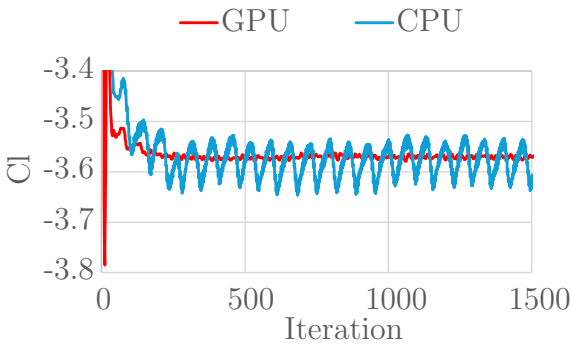

Figure 6.6: Stability of lift report comparison

It is clear that the GPU solver delivers a huge advantage in terms of computational time which was 85 % less than when the CPU solver was used while maintaining the same result

accuracy. The GPU solver also proved to be more stable even though single precision was used. This might be caused by the new update of Ansys Fluent (version 2024r1) where the GPU solver was introduced as non-Beta version for the first time. However, a GPU solver might not always be suitable for FS race car simulation because of the following reasons:

- Some features and boundary conditions like momentum sources (2D/3D fan zone) are not possible in GPU solver. MRF must be used instead, significantly increasing the number of cells in the mesh ($+\approx$ 7mil cells), resulting in possible dedicated GPU memory overload.
- This simulation with a simplified FS race car model as mentioned in Section 3.1 already reached 69.3 GB of dedicated GPU memory, out of the theoretical 95 GB in total.

To summarize this research, GPU solvers are the future of CFD simulations. They can provide very fast and accurate results which might be helpful especially when solving smaller meshes that are not very demanding for memory. For FS race car fully automated CFD simulation in eForce Prague Formula, the CPU solver is still a better option for the following reasons:

- Team eForce Prague Formula possesses with only one workstation capable of GPU simulations.
- The script for CPU simulation can be also used on university servers, where many simulations are still performed, even though the servers might be slower.
- The CPU solver can solve momentum sources such as 2D/3D fan zones which are more accurate approach when a precisely accurate CAD model of the fan is not available.

Chapter 7

Final Simualtion

When all important meshing and solver settings were compared, it is time to define a new CFD methodology for the development of FS race car external aerodynamics of eForce Prague Formula team for season 2025. The following changes were made compared to the CFD simulation used in season 2024:

- The tyre contact patch will be solved by precise surface wrap, not by degeneration of the tyre geometry.
- All trailing edges will have a minimum thickness of 1 mm. This approach is more accurate to the real-life geometry of the flaps used on FS race car. Moreover, this approach simplifies the mesh generation resulting in better quality cells, easier and more robust automatization, and a lower number of cells used in mesh.
- The bodies of influence were examined to be critical for computational stability however their size was reduced as well as the size of the tunnel domain.
- The setup of prisms for capturing the boundary layer was chosen according to Subsection 3.3.4 results. Each part of FS race car (flap, tyre, endplate, chassis etc.) has its unique values of growth ratio and total prism layer height according to the expected boundary layer thickness.
- SST k-ω turbulence model was chosen as the most accurate one within hardware computational power capabilities
- For the early stages of development where simpler geometry without source terms is used, a GPU solver might be used to maximize the number of simulations in a short period. As the geometry becomes more complex (more flaps, vortex generators, fans etc.) the simulation will be solved by the CPU.
- The simulation setup is supported by learning materials for new members of eForce Prague Formula team to ensure smooth knowledge transfer throughout many seasons.

7.1 Automatization

As stated in Section 1.3, CFD automatization is a vital feature for fast aerodynamic development within eForce Prague Formula team. There are two approaches to the automatization of Ansys Fluent simulation setup - Python script and Journal file.

7.1.1 PyFluent

Ansys software offers an additional Python package called PyAnsys, respectively PyFluent in the case of Fluent software. This additional tool offers large control over Fluent automatization with a wide range of adjustments relevant to standard code functions such as "if" conditionals. However, this package is a relatively new function with minimal information online and also requires good knowledge of Python environment. Moreover, some of the standard commands in Fluent GUI or console are not established in PyFluent yet. These factors create a very complex problem for Fluent automatization of cases such as FS race car external aerodynamic simulation. However, further research should be focused on this automatization process.

7.1.2 Fluent Journal File

Another, much simpler, automatization approach called Journal exists. Fluent can remember the simulation workflow and choice of parameters in GUI which is then transferred into a text file. This text file can then be read to the new mesh and all GUI functions will be automatically selected by the Journal file. The biggest issue with this approach is that all variables and parameters are fixed by the Journal file. When repairing the mesh, the number of iterations depends on the quality and number of bad cells. This might result in an unnecessary number of repair iterations for good meshes and vice versa. Moreover, each Journal file defines all parts of the car and their name within its code so when adding a new part or removing some old one, all code lines with appropriate parts must be removed, adjusted, or a new Journal file must be created. Another problem with the Journal script is its inability to incorporate 3D fan zone boundary conditions. Every 3D fan zone requires upload of a fan curve text file which contains ID number of the 3D fan zone. Unfortunately, the ID of every zone changes randomly in every simulation so it can not be predicted in the text file, resulting in different zone ID that creates error while importing the fan curve. Only MRF or 2D fan zone can be used with the Journal file. Nevertheless, this issue can be bypassed by optimization of 2D fan zone according to the 3D fan zone or MRF results. On the other hand, the biggest advantage over the Pyfluent script is the simplicity and speed of creating a Journal file. One can simply manually set up a random simulation within a few hours and the Journal file can then be used for other simulations that follow the Journal script rules. This means that every Fluent user can create his own automatization script regardless of his abilities with other development software. Therefore, the Journal file was chosen as the better option for CFD automatization of our specific case of FS race car simulation (chosen for season 2025 and until PyFluent automatization is established).

The Journal created as a part of this thesis is not only responsible for meshing and solver but also for postprocessing when simulation results, pictures, and report are created and saved.

7.1.2.1 Knowledge Transfer

Last, but not least I can not help but highlight the importance of a good knowledge database and knowledge transfer throughout the FS team. The best FS teams are known to slowly but steadily improve their multiple-season lasting concept and vision. Every design decision is supported by thorough research before changes are implemented and time, personal, and financial aspects are always the main factor of these design decisions. These teams and their members can grow because, from every research, development, success or even failure, there is a lesson learned which is passed every season to the new generation. On the other hand, most of FS teams try to design and manufacture the best possible race car within a single season without fully understanding the whole process of race car development, manufacturing, and data gathering. These teams can win only once in a few years if they happen to be lucky that season. Without a proper understanding of past development, there is no move forward in race car design. A similar situation, with CFD, occurred at eForce aerodynamics group during the 2024 season. The group used conventional CFD software with full automatization, similar to Ansys Fluent. This CFD automatization and its CFD software setup was created by a former CTU Cartech member in season 2021, as already mentioned. Unfortunately, only a few years after the release of the CFD automatization, there was only one person in the team who was able to somehow adjust the automatization script. The rest of the aerodynamic group became disinterested in CFD and its methodology. This led to the overall team's misconception about CFD when many aerodynamicists within the team did not understand the vast majority of the results generated from CFD simulations because they were never explained the theory and basic principles behind the automatization. Also, it is worth mentioning that this method has not significantly improved since its release in 2021 and if it was not for this thesis, the method would not have changed until the 2026 season at least.

A similar scenario should never happen again so more effort should be put into the theory of aerodynamics and CFD, instead of grinding every possible gain of downforce by spending all time designing more and more aerodynamic parts on the FS race car. Furthermore, CFD tutorials for novices were made during the 2024 season to provide better knowledge transfer and to enlighten all members about why specific setups, methods, and variables were chosen.

7.2 Postprocessing

This section aims to demonstrate how simulation results can be visualised during post-processing and how this visualization can help during results understanding, following the right design decisions. It is not necessarily the topic of this thesis however not many

publications show or explain how to approach this problem. Because of that, the most important visualisations for FS race car simulation will be explained in two separate subsections regarding their automatization. Car axis and cutting planes will be mentioned so just to clarify the axis system of FS race car, pitch is around the Y-axis, roll is around the X-axis, and rotation is around the Z-axis.

7.2.1 Automatized Visualizations

These visualizations are an essential part of every FS race car simulation results so they can be automatized to save postprocessing time and also ensure the same properties of the picture so that all simulations can be compared with the same postprocessing setup. Some of these visualizations are also saved as a scene image which allows the user to read the scene in Ansys Viewer where the user can rotate, move, and zoom in/out as required. Other visualisations are saved only as a PNG image from a specific point of view or cutting plane.

7.2.1.1 Pressure Coefficient

This is the absolute standard of every FS race car CFD simulation. The pressure coefficient is a non-dimensional value so that various design iterations can be compared equally. This contour can show areas with lower or higher pressure and also outline possible flow separation to the experienced user.

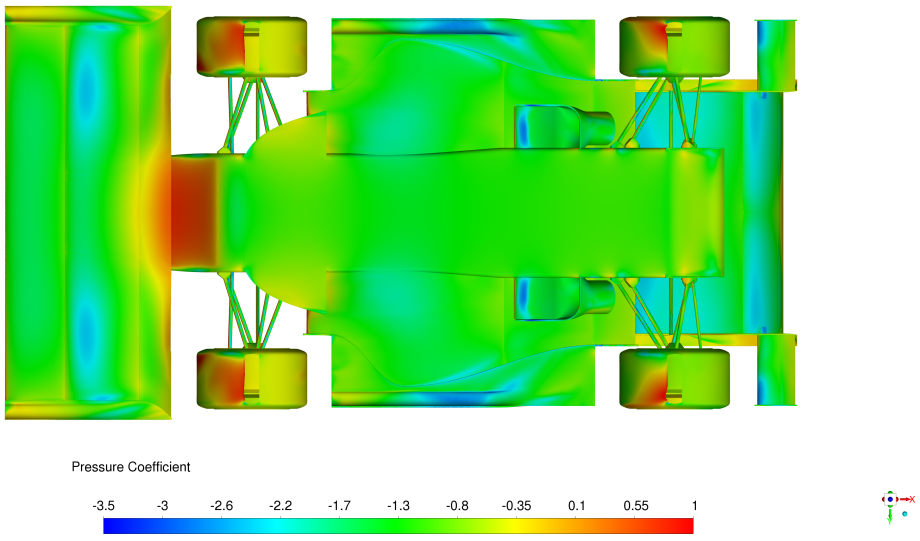

Figure 7.1: Contour of pressure coefficient

7.2.1.2 Velocity and Iso-surface

The velocity magnitude contour as shown in Figure 7.2 shows velocity magnitude across the whole domain of the cutting plane. The velocity magnitude contour, when combined with other visualizations such as the pressure coefficient, can help understand the airflow behaviour better. The pictures are created in an XZ-plane with a Y-axis spacing of 0.05 m. The second velocity visualisation, as shown in Figure 7.3, shows something different. The iso-surface shows an area of a constant value which is set to 0 m/s. By doing so, the user receives an image of the airflow separation where the green area divides the flow velocity of positive and negative values.

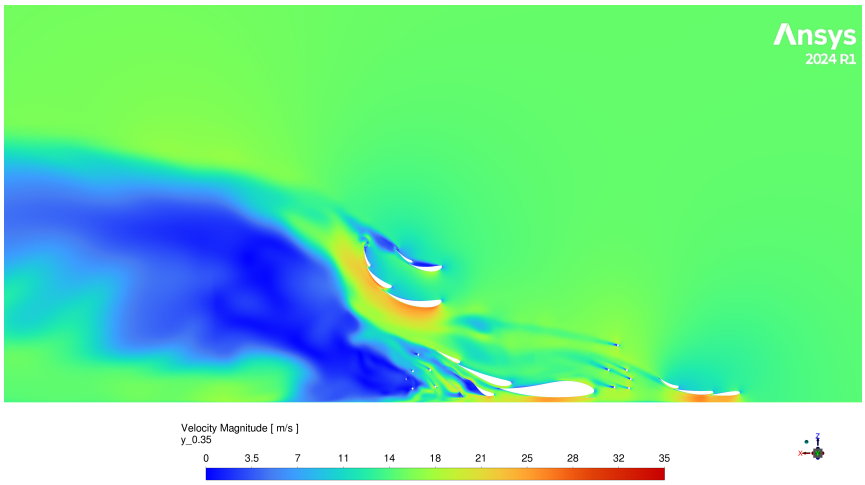

Figure 7.2: Contour velocity magnitude, XZ-plane

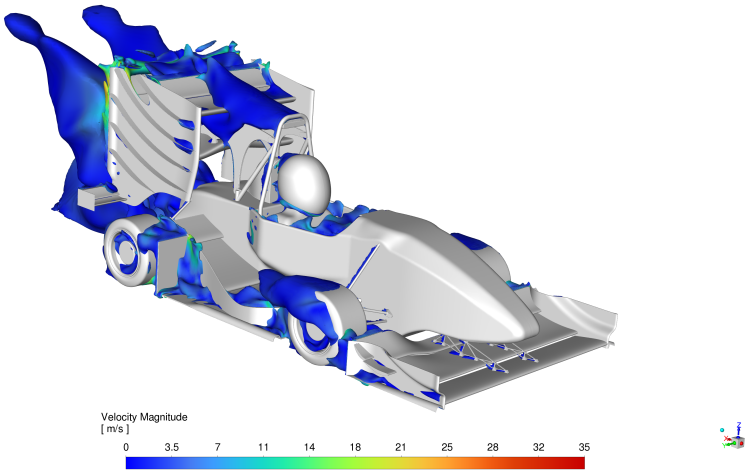

Figure 7.3: Contour velocity magnitude, iso-surface

7.2.1.3 Total Pressure and Iso-surface

Figure 7.4. shows total pressure across the domain in a YZ-plane with an X-axis spacing of 0.05 m. This contour helps to understand the vortices and their origin. Areas with lower values of total pressure are less favourable for aerodynamical devices and their performance so engineers should try to avoid such areas by correct channelling of these vortices. Figure 7.5 shows the iso-surface of total pressure 0 Pa (note that the 0 Pa is a relative value to the reference pressure which is 101 352 Pa). This can reveal parts with high drag and also show how parts especially in front of the car influence the airflow towards the rear.

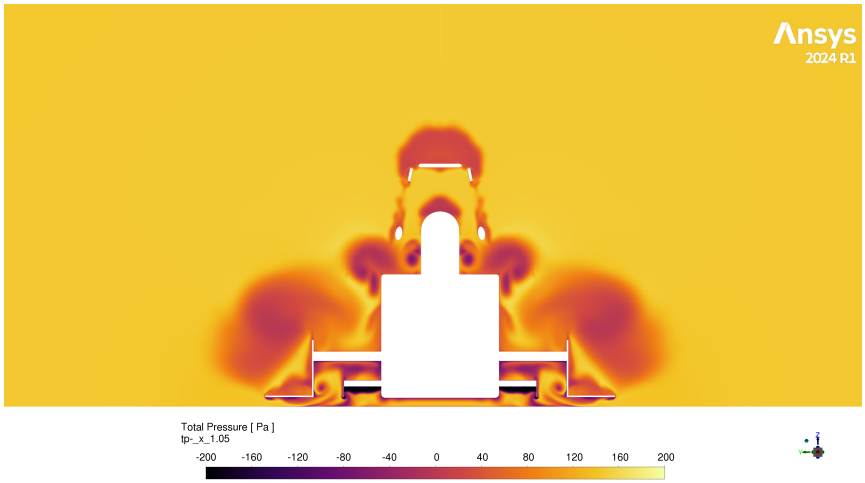

Figure 7.4: Contour of total pressure, YZ-plane

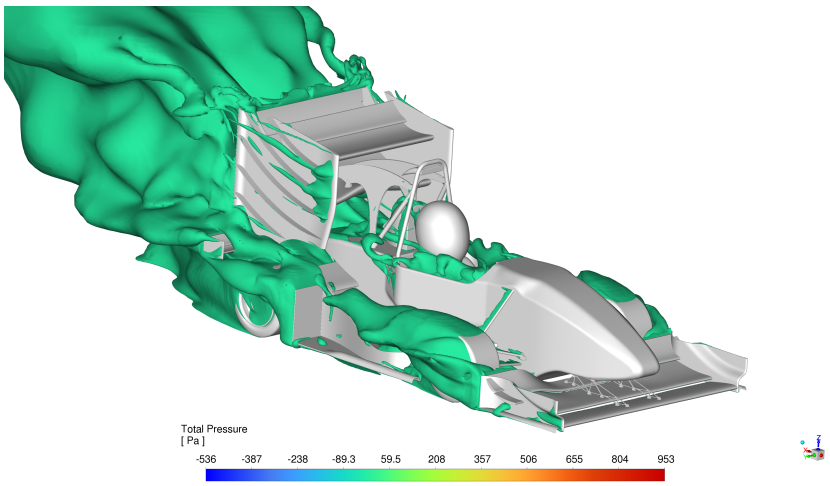

Figure 7.5: Contour of total pressure, iso-surface

7.2.1.4 X-wall Shear Stress

The name of this subsection already tells what this visualization shows. The contour range is specific for this case as only the border between the positive and negative values is important. Airflow separation from the surface happens when the colour changes from pink to blue (X-axis positive-direction shear stress becomes negative-direction). This can highlight the areas where design change is needed, assuming the flow separation is not by design of course.

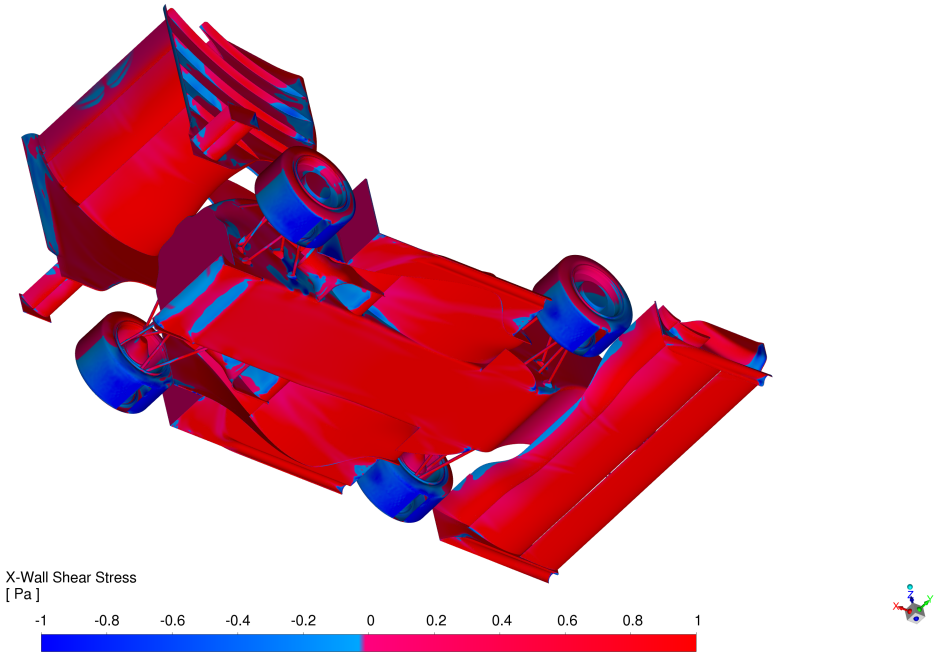

Figure 7.6: Contour of wall shear stress in X-direction

7.2.1.5 Wall y+

As already discussed in Sections 3.3.3 and 5.3, the $y+$ value visualisation shows the refinement quality of the prism layer. Where $y+ \in \langle 5, 30 \rangle$, the mesh must be adjusted to meet the desired $y+$ value. This contour is not especially useful for engineers developing aerodynamic devices but is extremely important for CFD engineers. When $y+ \in \langle 5, 30 \rangle$ at some aerodynamical part of the car, the mesh must be refined immediately otherwise further development of that part might be useless as the airflow is not predicted correctly there. An example of the $y+$ contour is shown in Figure 5.3. at page 38.

7.2.1.6 RMSE Static Pressure

The RMSE (Roor Mean Square Error) shows the fluctuation between the expected and observed values of Static Pressure. In CFD application this visualization shows the areas with the highest result oscillation which might lead to solution instability. The solution to this might be local mesh refinement. Another meaning of this visualisation might be an indication of unsteady behaviour in the flow. The RMSE contour, as the $y+$, is mainly an informative tool for CFD engineers.

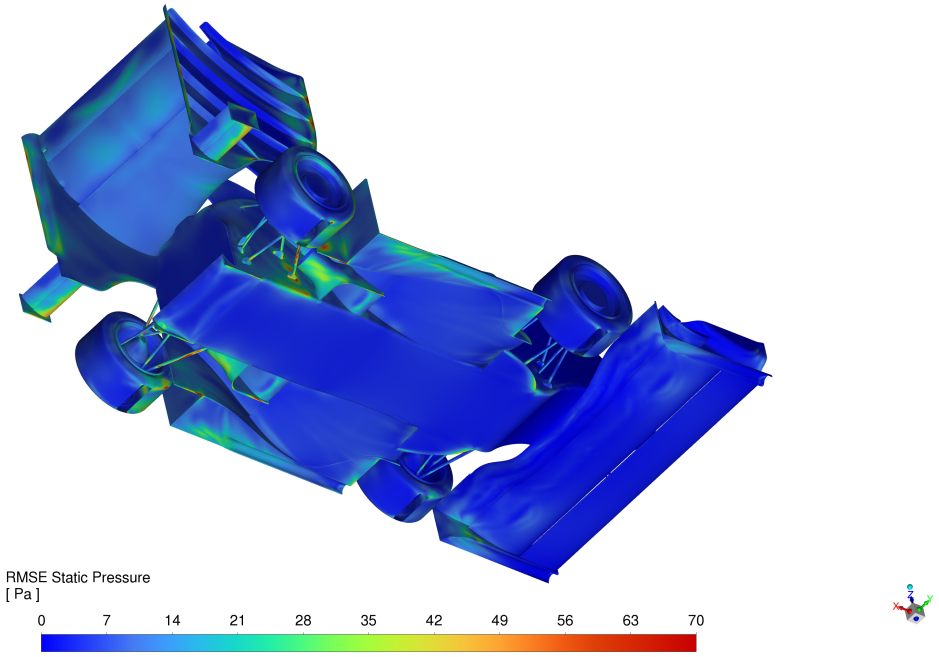

Figure 7.7: Contour RMSE static pressure

7.2.2 User-defined Visualizations

The user-defined visualizations have no limit in terms of customization. Moreover, the standard automatized visualizations are great for the display of whole-car properties but they might not always provide the required picture or angle of view in some specific cases. Because of that, all engineers participating in the development of aerodynamic devices should be able to display their own customized visualizations.

7.2.2.1 Pathlines

One example for all might be the development of the front wing endplate which is a device to enhance the aerodynamic performance of the front wing by separating low and high

pressure areas between the front wing flaps and freestream next to the wing. Moreover, the endplate vortex generators help to guide the airflow around the car, reducing the overall drag of the car and redirecting the dirty air from leading to other aerodynamic parts of the car such as the underbody. Correctly defined pathlines might be extremely useful for a better understanding of the airflow exiting the front wing. Pathlines as shown in Figure 7.8 could never be automatized because, for every specific case, the origin and amount of pathlines must be customized.

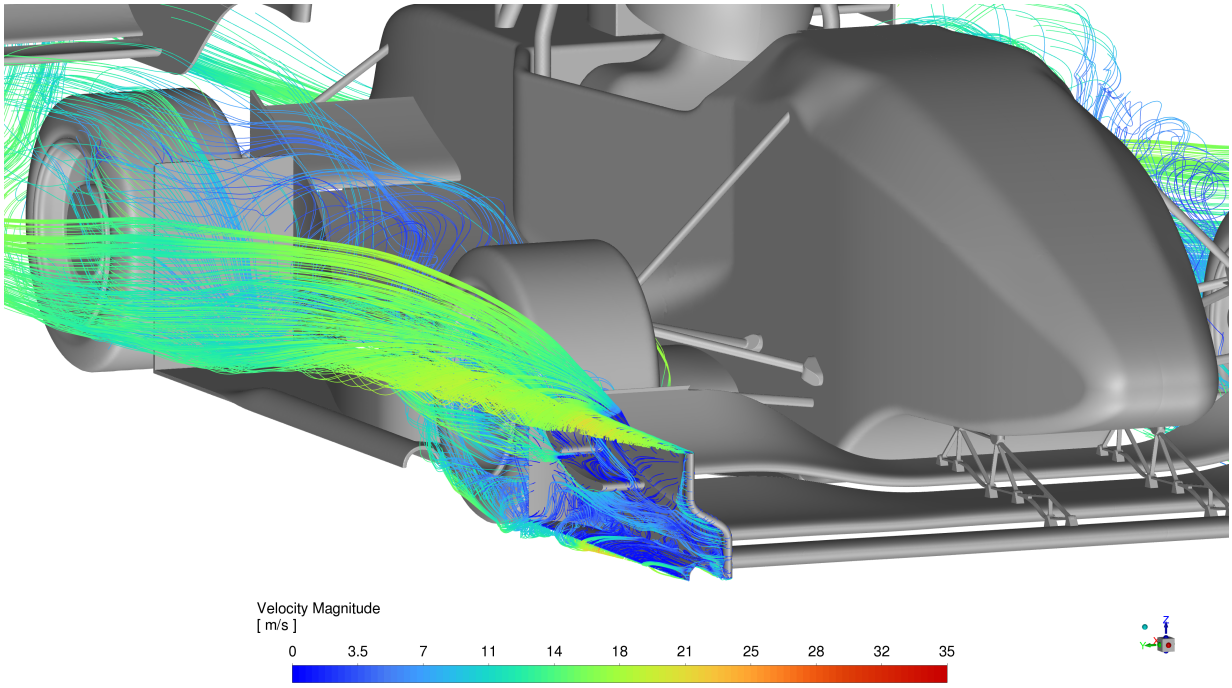

Figure 7.8: Velocity pathlines of front wing endplate

Chapter 8

Conclusions

As a result of this thesis, a new CFD methodology was developed according to the requirements of FS team eForce Prague Formula. The requirements are specified in the first chapter, which also explains the motivation behind this thesis and the principle of Formula Student. In the second chapter, the ideal CAD geometry of car tyre and flaps trailing edge for this new methodology was established. The additional geometry between the tyre and the road has proved to be unnecessary if the tyre is wrapped correctly. The trailing edge was set to be modelled with 1mm thickness in CAD in order to preserve the reality of the FS race car aerodynamic devices as well as to simplify the meshing process. Chapter 3 showed the importance of local mesh refinement (BOI) and also showed the importance of boundary layer capture which proved to be crucial for the accuracy. After this research, all parts of the car have their unique setup for the prism layer. The different approaches to axial fan and radiator setup were compared in Chapter 4. Because of the CFD automatization, the 2D fan zone was chosen as the best solution. The inaccurate turbulence model used by eForce Prague Formula team during previous seasons was compared and replaced by a more accurate turbulence model SST k-ω in Chapter 5. Chapter 6 compared various approaches to simulation solver. The single precision was established as a sufficient method for this CFD simulation thanks to its lower demand on computational power while maintaining the same accuracy. The Coupled method was selected as a better option for season 2025 however, future research might be done to find the limits of the SIMPLE method. To summarize this chapter, CPU and GPU solvers were compared. The GPU solver should be used during the early stages of development thanks to its magnificent speed advantage over the CPU solver. Unfortunately, the GPU solver proved to be suboptimal for simulations of axial fans and the PC hardware capabilities were too low for the whole FS race car with complex geometries. Because of that, the CPU solver should be used during the later stages of development or when an axial fan is present in a FS race car CFD simulation. The last chapter demonstrates the final simulation which might be used for season 2025. This simulation was created and set up based on results from all previous chapters.

8.1 Future Work

The following research might be done as a success to this thesis:

- The change of tyre radius when a normal load is applied, as mentioned in Section 2.2.3, might be examined to find a possible influence of aerodynamic devices behind the wheels.
- To find the best possible ratio between the mesh size and solution accuracy, the size of BOI and tunnel might be thoroughly examined as well.
- Section 3.3 established a new methodology for the generation of prism layers where every part of the car has its unique setup. In pursuit of the smallest mesh possible, research about different prism layer setups on a single part would be very interesting. For example chassis of the car requires a fine prism layer on the underbody whereas the sides and top of the monocoque can be computed by wall functions. This might lead to the use of, for example, 15 prism layers on the underbody and 2 prism layers on the other faces. The problem is however far more complex as connecting such different prisms on a single part leads to cell quality issues.
- The SIMPLE method, as discussed in Section 6.2, would deserve its own deep research in order to reach the limits of this method compared to the Coupled method.
- And finally, a complete CFD automatization via PyFluent would be a breakthrough not only for eForce Prague Formula, but for the whole FS community.

Bibliography

[1] Joseph Katz. *Race Car Aerodynamics*. EAN 9780837601427, Robert Bentley Inc., 1996, 280 pp.

[2] Martin Ševčík. *Metodika návrhu aerodynamiky vozu Formula Student*. Master's thesis, Czech Technical University, Faculty of Mechanical Engineering, 2021, 102 pp.

[3] ANSYS. Introduction to Turbulence Modeling in ANSYS Fluent - Lesson 1. [online], April 2024. Available from: https://courses.ansys.com/index.php/courses/turbulence-modeling-in-ansys-fluent/lessons/introduction-to-turbulence-modeling-in-ansys-fluent-lesson-1/

[4] Wikimedia Commons. *Law of the wall*. April 2024. Available from: https://commons.wikimedia.org/w/index.php?curid=15672321

[5] *Ansys Fluent User's Guide 2024R1*. Ansys, Inc., 2024, 5666 pp.

[6] AirfoilTools. Cl v Alpha. [online], March 2024. Available from: http://airfoiltools.com/airfoil/details?airfoil=goe226-il

[7] Schubeler. DS-86-AXI HDS® (120 mm). [online], February 2024. Available from: https://www.schubeler.com/product/ds-86-axi-hds/?v=928568b84963

[8] *Ansys Fluent Theory Guide 2024R1*. Ansys, Inc., 2024, 1144 pp.

[9] Lukáš Pacoň. *Ověření CFD simulací v aerodynamickém tunelu*. Bachelors's thesis, Czech Technical University, Faculty of Mechanical Engineering, 2016, 49 pp.

[10] Xilinx. Single Precision vs Double Precision: Main Differences. [online], April 2024. Available from: https://www.xilinx.com/applications/ai-inference/single-precision-vs-double-precision-main-differences.html

[11] *Lecture 5 - Solver Settings*. Customer Training Material, Ansys, Inc. Proprietary, 2010, 32 pp.