Università
degli Studi
di Padova

Master Thesis in Aerospace Engineering - Aeronautical Curriculum

Ground effect analysis of a Formula SAE car

Candidate

Antonio Minto

Student ID 2053135

Advisor

Prof. Federico Dalla Barba

University of Padua

Academic Year
2022/2023

Alla mia famiglia

I believe most things can be said in a few lines
- Enzo Ferrari

Abstract

This thesis aims to investigate the ground effect behavior of the Formula SAE car, SGe-06, a competitive vehicle of the University of Padua during the academic year 2022/2023.

Ground effect is a notable aerodynamic phenomenon resulting from the proximity of the car to the ground, playing a crucial role in generating downforce while minimizing drag.

The focus of this study is primarily on the vertical motion of the car induced by aerodynamic loads under steady-state conditions. This heaving motion will be analyzed using Computational Fluid Dynamics (CFD) simulations with the Star CCM+ software. The simulations will contribute to evaluating the operational height of the single seater at various velocities.

Furthermore, the dynamic effects of this vertical motion will be explored through simulations using OpenFOAM. These dynamic simulations will be compared with the static case and existing theoretical frameworks to gain a deeper understanding of the ground effect's influence on the car's aerodynamic behavior.

Sommario

Questa tesi si propone di analizzare il comportamento in effetto suolo della vettura Formula SAE, SGe-06, un veicolo competitivo dell'Università di Padova dell'anno accademico 2022/2023.
L'effetto suolo è un fenomeno aerodinamico derivante dall'interazione dell'auto con il suolo, che svolge un ruolo cruciale nella generazione di deportanza riducendo al minimo la resistenza aerodinamica.
Il focus di questo studio si incentra principalmente sul movimento verticale dell'auto indotto dai carichi aerodinamici in condizioni stazionarie. Questo movimento verticale sarà analizzato utilizzando simulazioni di Fluidodinamica Computazionale (CFD) con il software Star CCM+. Le simulazioni contribuiranno a valutare l'altezza operativa della monoposto a diverse velocità.
Inoltre, gli effetti dinamici di questo movimento verticale saranno esplorati attraverso simulazioni in OpenFoam. Queste simulazioni dinamiche saranno confrontate con il caso statico e con i modelli teorici esistenti per acquisire una comprensione più approfondita dell'influenza dell'effetto suolo sul comportamento aerodinamico dell'auto.

Contents

List of Figures

List of Tables

List of Codes

List of Acronyms

CFD Computational Fluid Dynamics

SAE Society of Automotive Engineers

CV Internal Combustion Engine Vehicle

EV Electric Vehicle

AWD All Wheel Driven

LE Leading Edge

TE Trailing Edge

COP Centre Of Pressure

GEV Ground - Effect Vehicle

WIG Wing - In - Ground

RANS Reynolds Averaged Navier Stokes

MRF Moving Reference Frame

FW Front Wing

RW Rear Wing

BS Base Size

TSS Target Surface Size

MSS Minimum Surface Size

SF Stretch Factor

NWT Near Wall Thickness

LPLT Last Prism Layer cell Thickness

NPL Number of Prism Layers

PLTT Prism Layer Total Thickness

TKE Turbulent Kinetic Energy

DNS Direct Navier - Stokes

LES Large Eddy Simulation

CFL Courant Friedrichs Lewy

AMG Algebraic Multi Grid

COM Center Of Mass

GAMG Generalized Geometric Algebraic Mltigrid

URANS Unsteady RANS

1

Introduction

1.1 Formula SAE

Formula SAE is an international student design competition firstly proposed by Society of Automotive Engineers (SAE).
It consists in the design and production of a single seater, that will be evaluated by judges in events that take place in the most important circuits of the World. Internal combustion or full electric cars that respect the regulations and pass all the technical tests planned, can be part of the competition. In particular, each contest consists of static and dynamic events.

Figure 1.1: The SGe-06 in action at the event of Formula SAE Italy

Statics are devided in three trials:

- Business Plan Presentation: each team has to create a business case for the vehicle built;
- Cost Analysis: teams need to evaluate the costs of every single component and process in building the car. This analysis needs to be very accurate and must include every single aspect of the life of the car;
- Engineering Design: expert judges assess the design of every single part of the car. Choices made need to be justified taking in consideration budget,

constraints, time and other important factors in the design and production phase.

Dynamic events consist in:

- Acceleration: a 75 m sprint from standing start;
- Skid-Pad: a 8 shaped track composed by two circles that tests the cornering speed of the car;
- Autocross: one lap of a circuit with short straights, turns and hairpins, to test speed and maneuverability of the car;
- Endurance and Efficiency: final test of the events that consists in a 22 km race. At the end of the race fuel or energy consumption is evaluated.

During every event, judges give points to the team. The sum of all the points received gives the overall classification.

1.1.1 Rulebook

With a lot of cars and Universities involved, competition is hard, but safety needs to comes first. Having a well defined rulebook is indeed essential in maintaining cars safe and at similar level of performance.
At the same time, it's really beneficial for teams trying to interpret the rules, as that could lead to significant gains in terms of speed of the car.
Regulations are slightly different in some competitions, but in general they cover all aspects of the car's construction and the progress of the race. Taking as reference the *Formula Student Rules 2023* [6], rulebook is devided in the following chapters:

- Administrative Regulations: this section includes general requirements for teams, giving rules of conduct, deadlines and documentation of the competition;
- General Technical Requirements: it gives technical requirements in various areas of the car, such as chassis design, aerodynamic devices and electrical components;
- Internal Combustion Engine Vehicles: requirements specific for Internal Combustion Engine Vehicle (CV);
- Electric Vehicles: requirements specific for Electric Vehicle (EV);
- Technical Inspections: this section gives all the information on how inspections are conducted, defining all the parts that must be presented and checked;
- Static Events: rules of the static events presented earlier in this chapter;
- Dynamic Events: rules for the dynamics, including tires regulations in rainy conditions and penalties.

It's important for this thesis to list some specific rules in order to understanding the constrains within which it operates.
In particular it's interesting the General Technical Requirements chapter (*T*).

> *T2.1.2 The vehicle must be open-wheeled, single seat and open cockpit (a formula style body) with four wheels that are not in a straight line.*
>
> *T2.1.3 Open wheel vehicles must satisfy the following:*
>
> - *The wheel/tire assembly must be unobstructed when viewed from the side.*

> - *No part of the vehicle may enter a keep-out-zone defined by two lines extending vertically from positions 75 mm in front of and 75 mm behind the outer diameter of the front and rear tires in the side view of the vehicle, with steering straight ahead.*
>
> *This keep-out zone extends laterally from the outside plane of the wheel/tire to the inboard plane of the wheel/tire assembly.*

It's important, during the design phase, to keep these zones unobstructed, although devices in this zone could lead to a smaller tire wake, giving an aerodynamic advantage.

> *T2.2.1 The minimum static ground clearance of any portion of the vehicle, other than the tires, including a driver, must be 30 mm.*
>
> *T2.2.2 Sliding skirts or other aerodynamic devices that by design, fabrication or as a consequence of moving, contact the track surface are prohibited.*

These two rules limit once again the aerodynamic performance that could be obtained. This aspect of ground clearance will be widely discussed in the following chapters.
Section *T8* describe precisely the constraints on aerodynamic devices. The most interesting for this thesis are reported:

> *T8.1.1 A specifically designed structure mounted on the vehicle to guide the airflow around the vehicle, increasing the downforce acting on the vehicle and/or lowering its drag. The mounting of this structure is not regarded as an aerodynamic device, unless it is intentionally designed to be one.*
>
> *T8.2.1 Height restrictions:*
>
> - *All aerodynamic devices forward of a vertical plane through the rearmost portion of the front face of the driver head restraint support, excluding any padding, set to its most rearward position, must be lower than 500 mm from the ground.*
> - *All aerodynamic devices in front of the front axle and extending further outboard than the most inboard point of the front tire/wheel must be lower than 250 mm from the ground.*
> - *All aerodynamic devices rearward of a vertical plane through the rearmost portion of the front face of the driver head restraint support, excluding any padding, set to its most rearward position must be lower than 1.2 m from the ground.*

T8.2.2 Width restrictions:

- *All aerodynamic devices lower than 500 mm from the ground and further rearward than the front axle, must not be wider than a vertical plane touching the most outboard point of the front and rear wheel/tire.*
- *All aerodynamic devices higher than 500 mm from the ground, must not extend outboard of the most inboard point of the rear wheel/tire.*

T8.2.3 Length restrictions:

- *All aerodynamic devices must not extend further rearward than 250 mm from the rearmost part of the rear tires.*
- *All aerodynamic devices must not extend further forward than 700 mm from the fronts of the front tires.*

T8.2.4 All restrictions must be fulfilled with the wheels pointing straight and with any suspension setup with or without a driver seated in the vehicle.

These rules create *boxes* in which aerodynamic devices can be placed.

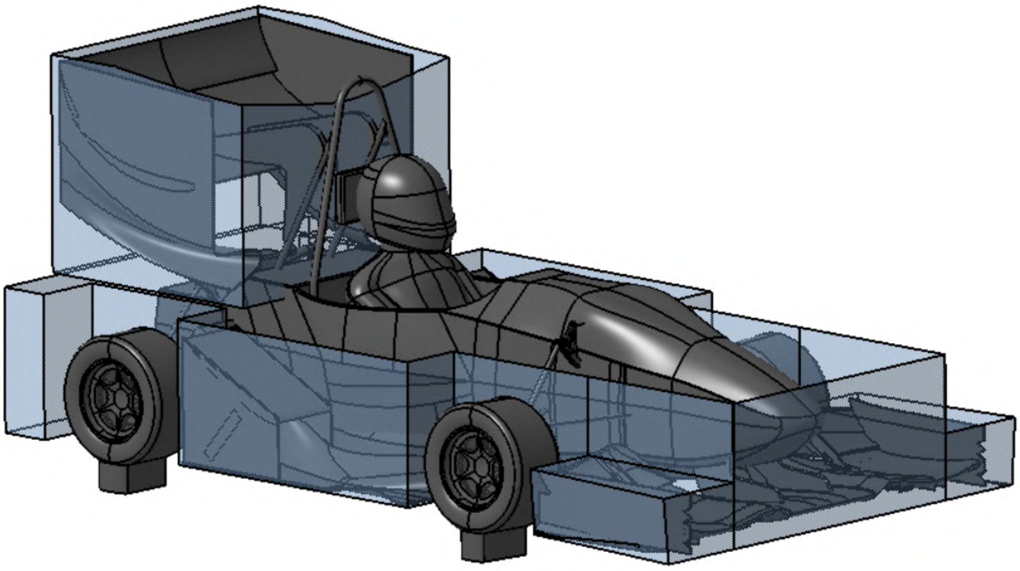


Figure 1.2: Boundary boxes for aerodynamic devices in the SGe-06 CAD

1.2 The SGe-06

The SGe-06 is the car that Race UP Team Electric of University of Padua designed and built in the academic year 2022/2023.
It competed in the events of Formula ATA, in Varano de' Melegari (Italy) and Formula Student East, at the Hungaroring (Hungary), obtaining respectively a 4th and an 8th place overall.
The vehicle is electric, with a LiPo battery with nominal capacity of 6,7 kWh. It's All Wheel Driven (AWD), with 4 synchronous motors of 35 kW with a maximum torque of 21 Nm each.
Car has a wheelbase of 1.535 m, with front and rear push rod suspension scheme, mounting 10 in *OZ Racing* rims and 16.2 in *Hoosier* tires.
It mounts a composite monocoque, composed by a sandwich of carbon fiber and aluminium honeycomb, and a composite aeropack, mainly composed of carbon fiber and rohacell.
Thanks to the materials used the car mass is around 200 kg.

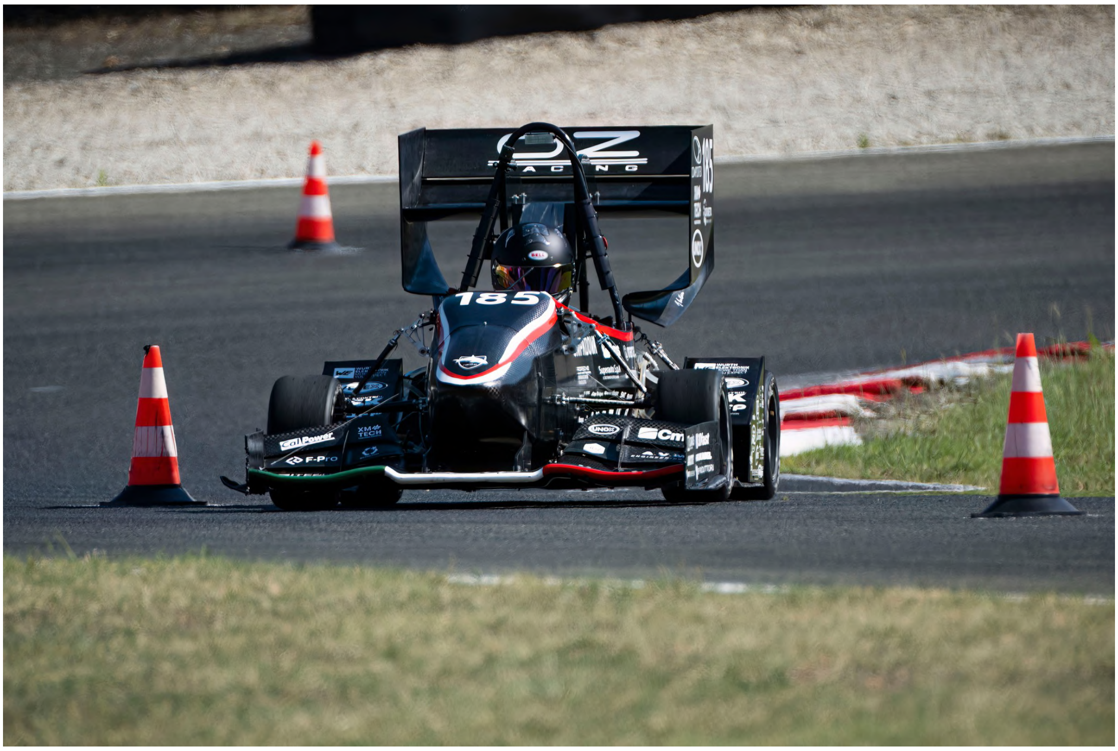


Figure 1.3: The SGe-06 in action

The aeropack includes:

- Front wing: contributes in creating downforce in the front part of the vehicle. It's a three elements wing, with the two flaps only connected to the endplates, in order to create a beneficial vortex in the zone next to the monocoque. Endplates are curved, creating an outwash effect, and mount footplates that create a vortex to the rear of the car. This wing aims to create a major outwash effect, while trying not to generate a massive upwash. This is seen to be optimal for the other aerodynamic components;
- Undertray: it's composed by two convergent - divergent channels mounted on the sides of the monocoque. These elements expands the air under the vehicle, working as Venturi's channels. It contributes for the major part of the total downforce, while minimizing drag thanks to the ground proximity;
- Diffuser: it creates a third channel using the underside of the chassis, expanding the remaining high - energy air. This component is effective in shifting the vehicle's center of pressure towards the rear;
- Rear wing: contributes in creating downforce at the rear. It's a two element wing, with curved endplates that helps creating a lower pressure at the exit of the undertray. This wing is designed to have an high efficiency and positioned to avoid the most of the wake generated by driver, mainhoop and other elements at the front.

These pieces work together to obtain the maximum possible level of downforce while keeping drag low, in order to reaching a good efficiency. All the aerodynamic effects cited will be better explained in the next chapters.

2

Case of Study

2.1 Ground effect

Ground effect, a significant aerodynamic and hydrodynamic phenomenon, manifests when an aircraft or a vehicle approaches close proximity to the ground or any solid surface. This phenomenon is of utmost importance in both aeronautics and automotive engineering, as it profoundly influences aerodynamic characteristics and overall performance.

In the realm of aeronautics [13], ground effect is experienced when an aircraft operates within a wingspan's distance from the ground. This proximity results in modifications to lift and induced drag, enhancing fuel efficiency and lift - to - drag ratio during takeoff, landing, and low - altitude flight.

Similarly, in automotive engineering, ground effect describes the aerodynamic interaction between a vehicle and the ground surface. Engineers and designers leverage this interaction to generate downforce, enhancing vehicle stability and maneuverability, particularly at higher speeds and during cornering.

In both environments, when an aerodynamic surface is close to the ground (which is creating either lift or downforce), wingtip vortices break, decreasing the biggest contribute in 3D wings drag, and thus increasing the generated force in the vertical direction, thanks to a smaller induced incidence angle, making the wing more efficient. A similar effect can be observed adding endplates or winglets at the tips of the wing, with the objective of reducing the influence of the wingtip vortex, that will start closer to the extremity of the wing, creating a wider low pressure surface and a more efficient wing.

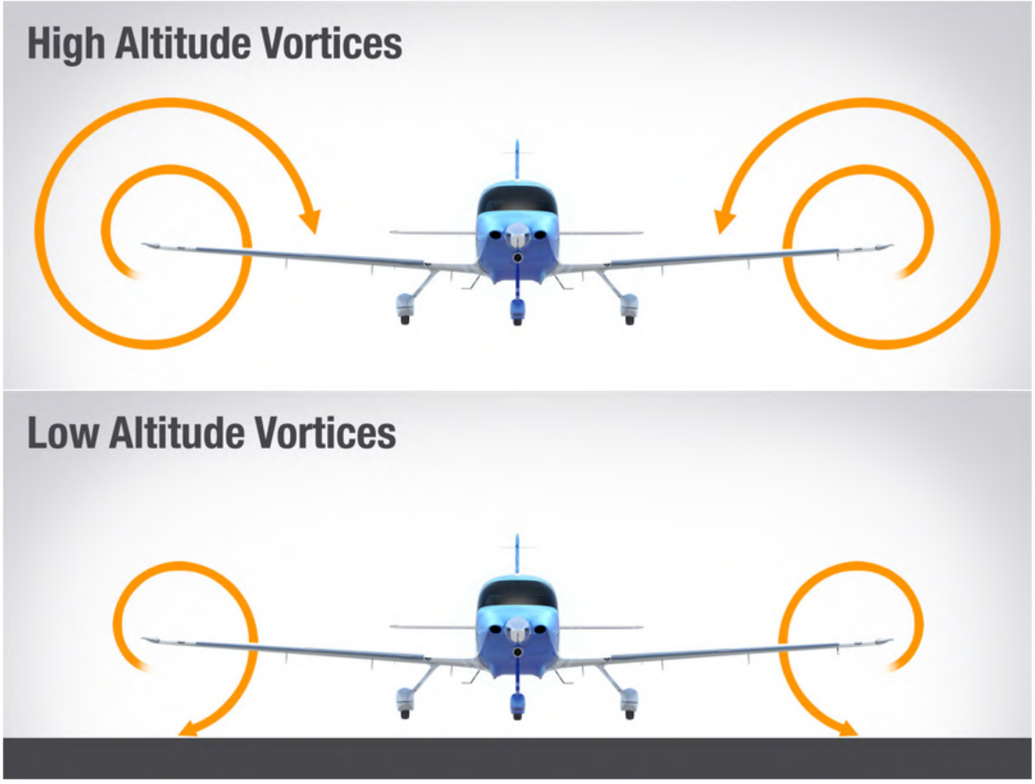


Figure 2.1: Tip vortices behaviour in ground effect

In racing cars ground effect is usually not only exploited with wings close to the ground, but also with the underside of the vehicle, working accordingly to the Bernoulli's Principle. This effect is indeed predominant in generating downforce close to the ground, as the car and the ground work together to create a Venturi channel.
In this thesis these listed effects will be analysed, evaluating the influence of the ground proximity at different heights.

2.1.1 Theoretical overview

It's necessary for this thesis to introduce some theory of what has been discussed earlier in this chapter, referring in particular to [8] and [1].

Firstly it's important to define the generation of lift around a 2D lifting airfoil profile with a rounded Leading Edge (LE) and a tapered Trailing Edge (TE). If there are no boundary layer separations present, then from Kutta - Joukowsky $L = \rho v \Gamma$, with Γ circulation, $\Gamma > 0$ if clockwise. What occurs in the initial moments when the profile is struck by a flow, is the generation of circulation Γ_1 around it, caused by the release of opposite circulation Γ_2 due to the counterclockwise vorticity born in the boundary layer. This vorticity causes the displacement of the stagnation point (the point where the pressure is maximum and the velocity is zero) posteriorly until it coincides with the trailing edge, as there are no expected boundary layer separation phenomena. The necessary Γ for all this to happen is determined by the Kutta hypothesis, equal to $\Gamma_{KUTTA} = \Gamma_1 = -\Gamma_2$.

It follows that the lift - generating process is related to viscous phenomena, which can be neglected only once established.

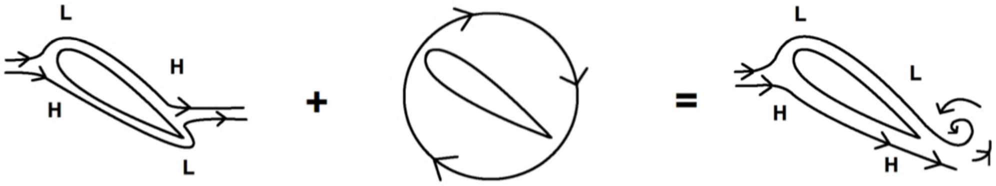


Figure 2.2: Kutta condition and lift generation

The Kutta condition states the necessity of having a finite angle or a cusp trailing edge to fulfill the Kutta hypothesis which asserts that at the posterior stagnation point, the jump in tangential velocity must be zero.

If we move to the three-dimensional case, additional effects that were absent in the previously discussed scenario need to be considered. In space, at the wingtips, vortices are observed, which influence the flow around the wings. High pressure air tends in fact to reach the lower pressure at the upper side of the lifting wing (the lower side in the downforce - production case). An induced angle of attack α_i is generated at the wing's leading edge due to these vortices.

This induced angle of attack can create a drag force, known as induced drag, given by: $D_i = L\alpha_i$.
In the three-dimensional case, even assuming the validity of potential theory, it is possible to resolve D'Alembert's paradox, which stated that $D = 0$.
With the lifting - line theory, which prescribes a continuous circulation release over the finite wing, gradually increasing from the center towards the wingtips, it is possible to derive the equation for a monoplane. This equation allows for the estimation of lift at each section of the finite wing, which can then be integrated over the entire wing to obtain the total lift $\hat{L}$:

$$\hat{L} = \int_{b/2}^{-b/2} L(y_0)\,dy_0 \tag{2.1}$$

From which the lift coefficient can be derived:

$$c_L = \frac{\pi \cdot AR \cdot A_1}{2} \tag{2.2}$$

where AR is the aspect ratio of the finite wing, defined as $AR = \frac{b}{c} = \frac{b^2}{S}$. Here, b represents the wingspan, c is the mean geometric chord, and S is the wing area. The parameter A_1 is a coefficient associated with the finite wing.
If we consider induced drag, it can be expressed as:

$$\hat{D} = \int_{b/2}^{-b/2} L(y_0) \cdot \alpha(y_0)\,dy_0 \tag{2.3}$$

This leads to the induced drag coefficient:

$$c_{D_i} = \frac{c_L^2}{\pi \cdot AR \cdot e} \tag{2.4}$$

where c_{D_i} is the induced drag coefficient and e is the Oswald efficiency factor related to the wing's efficiency.
This induced drag is really important in applications considered in this thesis. It's in fact, as just seen, related to the wingtip vortex that, in the case of an automotive profile that creates downforce, generates an upwash effect in the incident airflow, causing also the reduction of created downforce, other than the drag increase.

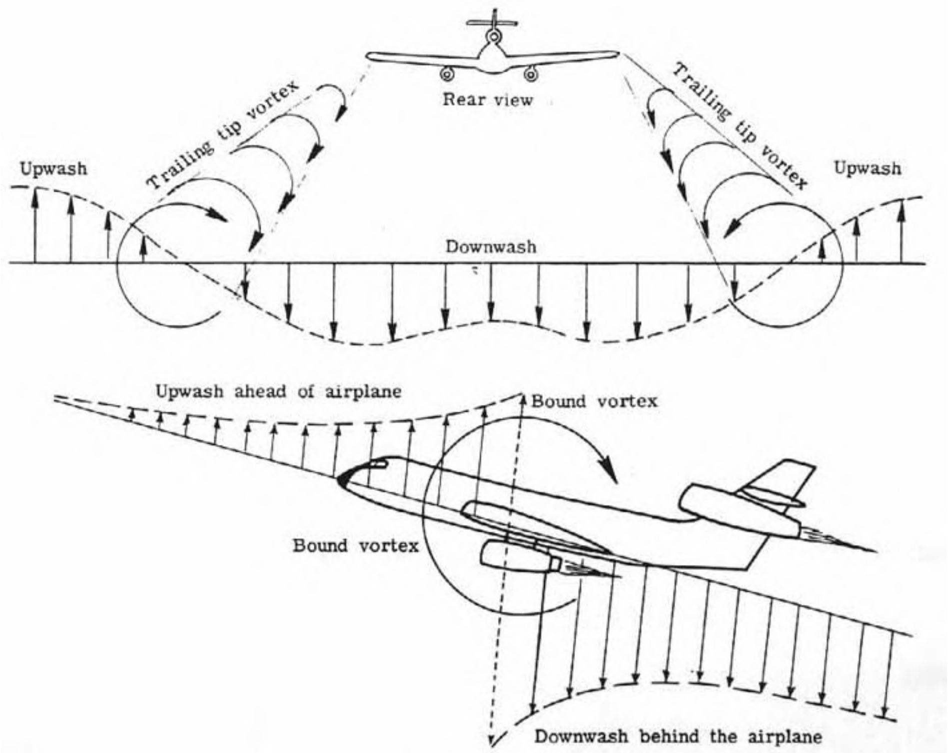


Figure 2.3: Vortices system around 3D wings

When this vortex is sufficiently close to the ground, its structure breaks, following a beneficial effect in both downforce and drag.
Together with this effect, a profile close to the ground creates a channel, that can be seen as the Venturi one.
Considering, in fact, a generic duct that undergoes a reduction in its cross-sectional area, with A_1 the larger area and A_2 the smaller area, according to the equation of continuity in fluid dynamics, in steady-state incompressible conditions, the mass flow rate into the first section must be equal to the mass flow rate into the second section ($A \cdot v = constant$).
Consequently, for a fluid with constant density (incompressible), as volumetric flow can be expressed as the product of fluid velocity and the cross-sectional area, it follows that the velocity in section A_2 is greater than that in A_1 ($v_1 < v_2$).
Given these considerations and assuming no difference in elevation between the two sections, we can adopt the axis of the duct as the height reference. This eliminates a term in the Bernoulli equation (energy conservation), leading to the following form:

$$p + \frac{1}{2}\rho v^2 = constant \tag{2.5}$$

where ρ represents density, p is pressure, and v is the fluid velocity.

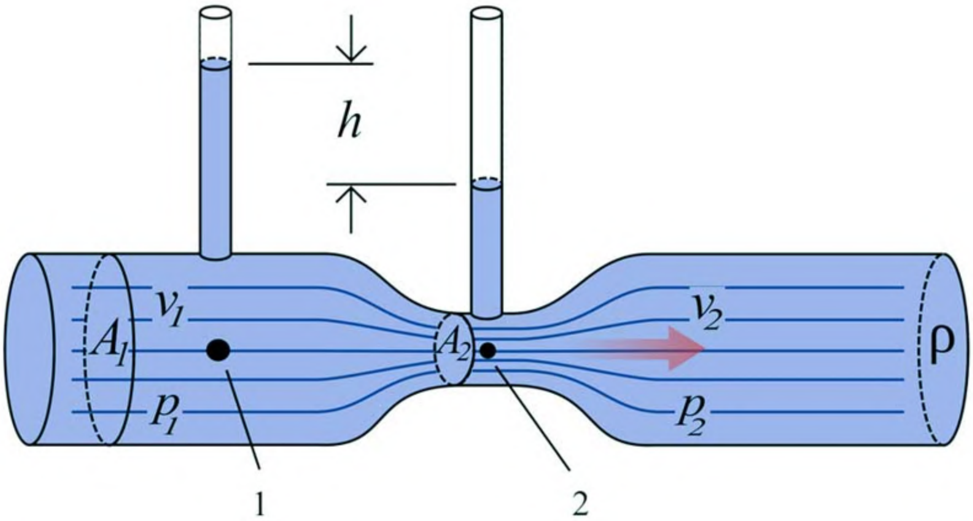


Figure 2.4: Venturi effect and Bernoulli's Principle

It is evident that an increase in fluid velocity results in a corresponding decrease in the internal pressure of the fluid. In this example, p_2 (pressure at A_2) is less than p_1 (pressure at A_1). Intuitively, one can comprehend the decrease in pressure within the fluid in the tube section characterized by a smaller area and higher velocity by considering that the pressure is utilized to accelerate the fluid. In simple terms, the pressure difference at the ends of the constriction or fitting zone is necessary to accelerate the fluid and ensure the constancy of the flow (continuity equation).

In the case of a 2D profile the velocity increase can be seen at the closest part to the ground. It's important to notice that this increase in velocity can be only present because after the narrowing, part or more of the initial area is recovered. The exterior pressure is, in fact, atmospheric (or close to this value) so, the ending area determines the mass flow in the considered channel and the lowest pressure obtained at the narrowest point, having the flux to restore the undisturbed conditions after the profile. This is similar to what happens also in freestream conditions around a wing, where part of the pressure is recovered at the TE.

Clearly this effect can be achieved using channels ad hoc, with a convergent (not necessary in the 2D case), a flat zone close to the ground where lowest pressure is achieved, and the essential divergent. Theoretically the bigger the ending area of the divergent and, at the same time, the smaller the minimum area, the lower will be the minimum of pressure underneath. This is generally true, but pressure gradients needs to be taken into account. Highly adverse pressure gradients can in fact create boundary layer separation, limiting the downforce creation.

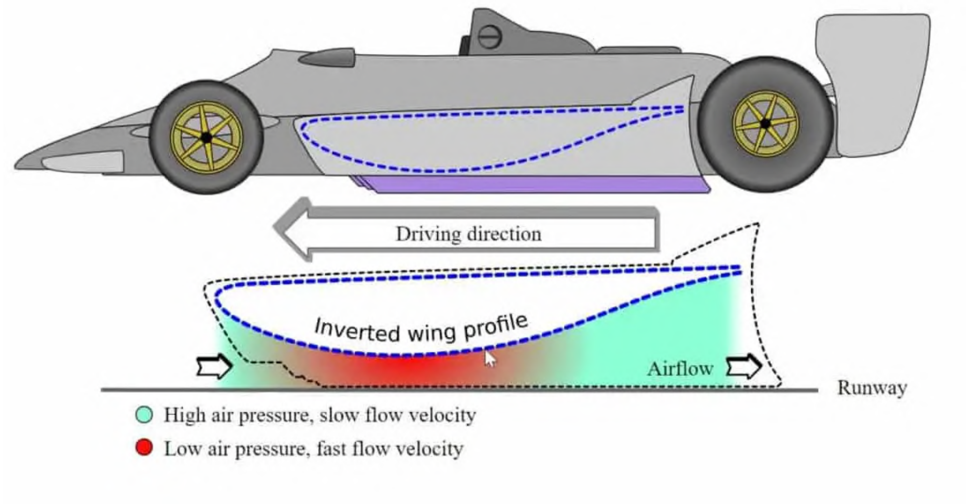

Driving direction
Inverted wing profile
Airflow
Runway
High air pressure, slow flow velocity
Low air pressure, fast flow velocity

Figure 2.5: Automotive application of the Venturi channel

Considering the 3D case, this effect is still valid, with some more complexities related to the non - infinite channel, with lateral higher pressure air that tends to reach the lower pressure. Originally in automotive *skirts* were used to increase the suction effect, then banned for safety reasons and substituted by other sealing solutions that include the use of beneficial vortices or a closer ride to the ground.

2.2 Applications

As mentioned, theories related to ground effect have been applied in both aeronautic and automotive contexts, aiming to achieve a high lift - to - drag ratio.

While this aerodynamic effect can manifest in any vehicle, some have been purposefully designed to exploit its advantages.

A Ground - Effect Vehicle (GEV), also referred to as a Wing - In - Ground (WIG) effect craft or ekranoplan (which translates to "screenglider" in Russian), is specifically engineered to glide over flat surfaces, often above water. Some models can operate over various level terrains such as frozen lakes or plains, similarly to an hovercraft. A GEV is sometimes described as a transitional form between a hovercraft and a seaplane. However, it lacks the hovercraft's ability to float at low speeds, given by onboard downward - directed fans that generate a cushion of pressurized air that provides the necessary support, and unlike a seaplane, a GEV is not typically designed to operate beyond ground effect [14].

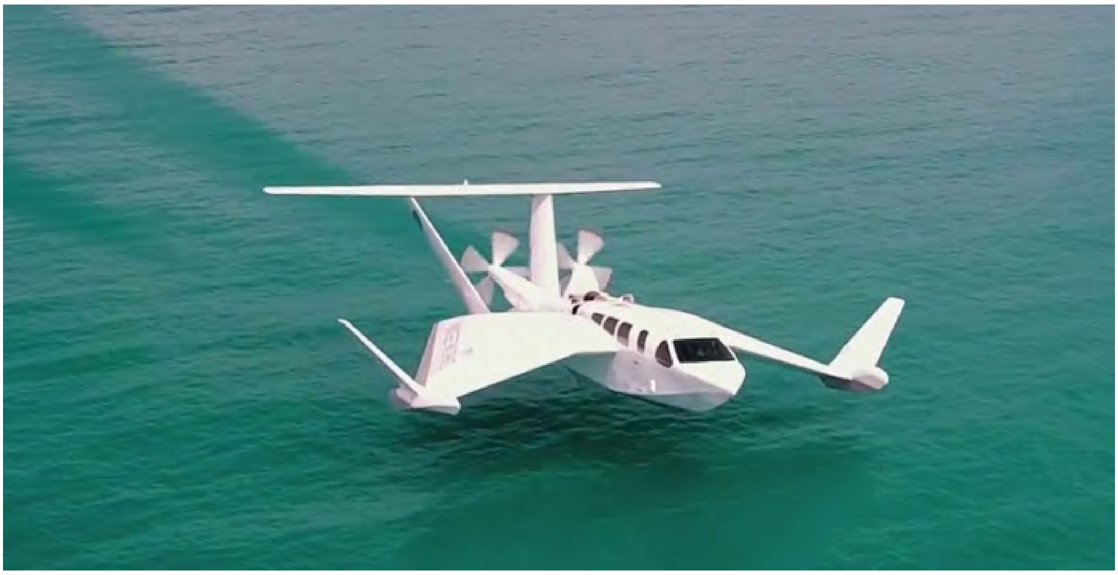


Figure 2.6: WIG effect aircraft

Staying in the maritime domain, modern America's Cup yachts (*AC-75*) also partially utilize this effect, aiding both lift generation and propulsion through sails.

On land, the ground-effect phenomenon is primarily explored in the realm of racing cars. Extreme applications have seen the integration of fans to draw air from beneath the car, accompanied by skirts that seal the car to the ground, as exemplified by the *Brabham - Alfa Romeo BT46B*. The *Lotus 78*, considered the pioneer of wing cars, introduced instead inverted aerofoil - shaped sidepods in Formula 1, a groundbreaking innovation that clearly paved the way for contemporary undertrays.

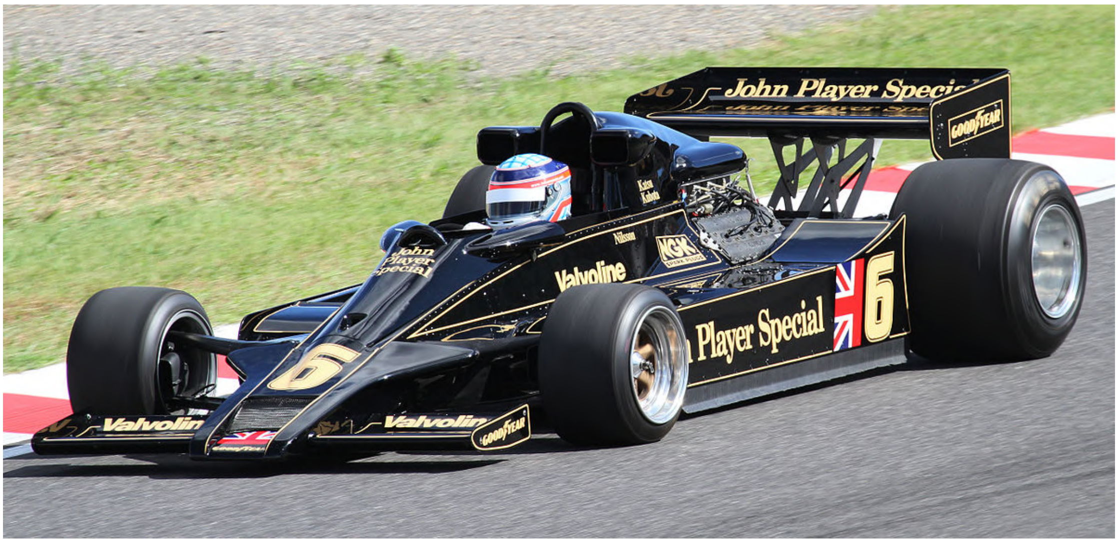


Figure 2.7: The Lotus 78 in action

3

Undertray design

3.1 The SGe-06 undertray

The purpose of this section is to introduce the SGe-06 undertray, explaining its features and the applied boundary conditions, while his design will be addressed in the next sections.
The objective at the basis was improving the previous year design for both aerodynamics and manufacturing, generating more downforce exploiting ground effect. A better integration with the new rear wing, monocoque and new tyres was also necessary to improve the overall performance of the vehicle.
Other than the general rule boxes described in the first chapter, boundary conditions included, as just said, the new monocoque and the smaller tires, which modified the rule boxes' volume, the necessity of keeping a flat zone for radiators positioning in front of the rear wheels and the imperative of maintaining a clean airflow for the battery ducts in the upper side of the undertray.

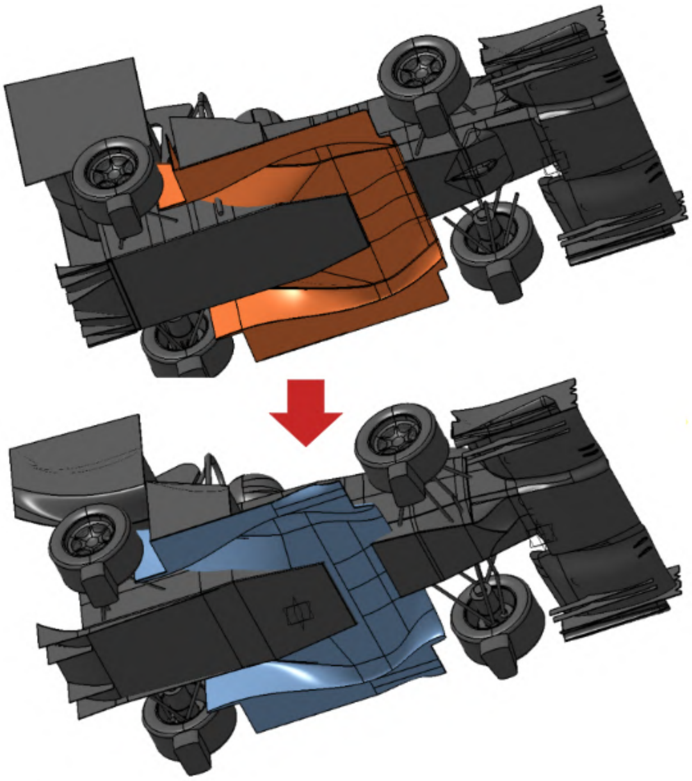


Figure 3.1: 2022 undertray (orange) and the 2023 evolution (light blue)

During the design phase different aerodynamics concepts were analysed with the most promising choice that turned out to be an evolution of last year's undertray. Prior to the optimization of the geometry, that will be discussed in

the next chapter, some choices were made in the aerodynamic concept, in order to fulfill boundary conditions and to better achieve theoretical advantages of the channel discussed earlier.
For the divergent:

– a large ending area inside the wheels to expand as much as possible without reaching stall conditions;

– the divergent starts early, closer to the front of the car, to prevent strong adverse pressure gradients, reducing stall risk;

– a low upwash effect is desirable, that would in fact reduce rear wing angle of attack, degrading the overall performance;

– a striction in the lower zone of the divergent aims to reduce losses in that area created by the air coming from the side of the car.

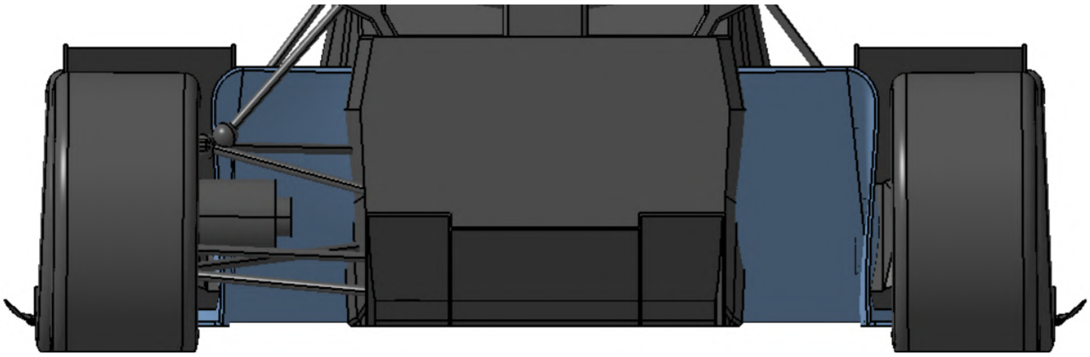


Figure 3.2: Divergent geometry

Flat - zone choices:

– this is the zone where low pressure occurs, so it's crucial to keep it as tightly sealed as possible by constructing a separate channel that is isolated from the lateral zone;

– flat zone dimensions and position determine convergent and divergent length in order to have a better optimized geometry.

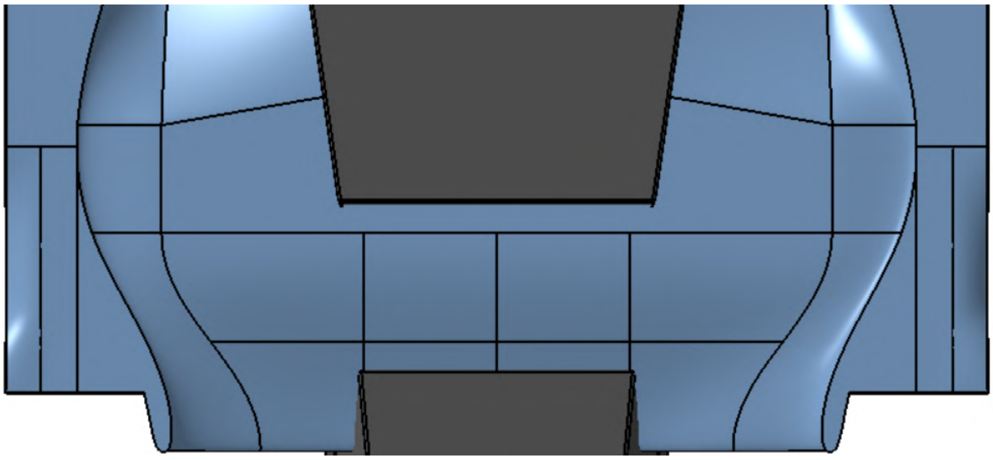


Figure 3.3: Flat - zone geometry

For the lateral - zone:

- smaller lateral flat plate compared to previous year, increasing channel dimensions;
- this zone is essential in keeping the front tire wake outside the channel and to create a first barrier for the air coming from the side;
- creation of a small lateral convergent - shaped geometry prevents losses from the entering air and helps generating a more stable vortex to seal off the undertray.

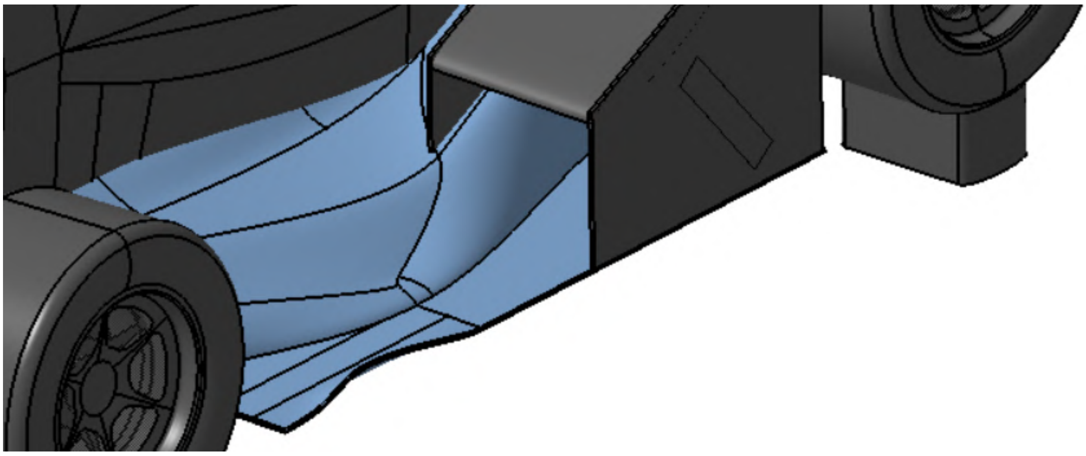


Figure 3.4: Lateral - zone geometry

Convergent choices:

- elimination of any mounting constraint under the monocoque, guaranteeing a more free geometry and thus a better airflow;

- a bigger leading edge is implemented to prevent losses;
- like the divergent it's inside the wheels, to avoid their wake.

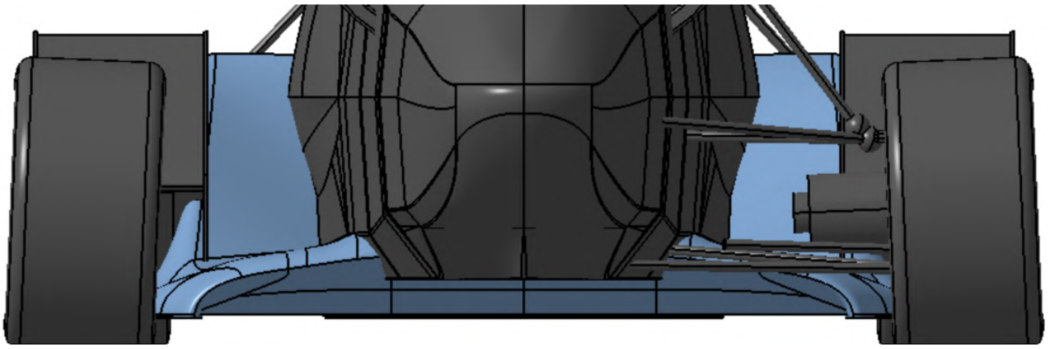


Figure 3.5: Convergent geometry

For the upper - zone:

- a different geometry between upper and lower zone is provided, shaping the undertray similar to a profile;
- this geometry can be optimized to achieve a smooth flow to the battery duct and to improve how the monocoque interacts with the undertray, reducing interference effects.

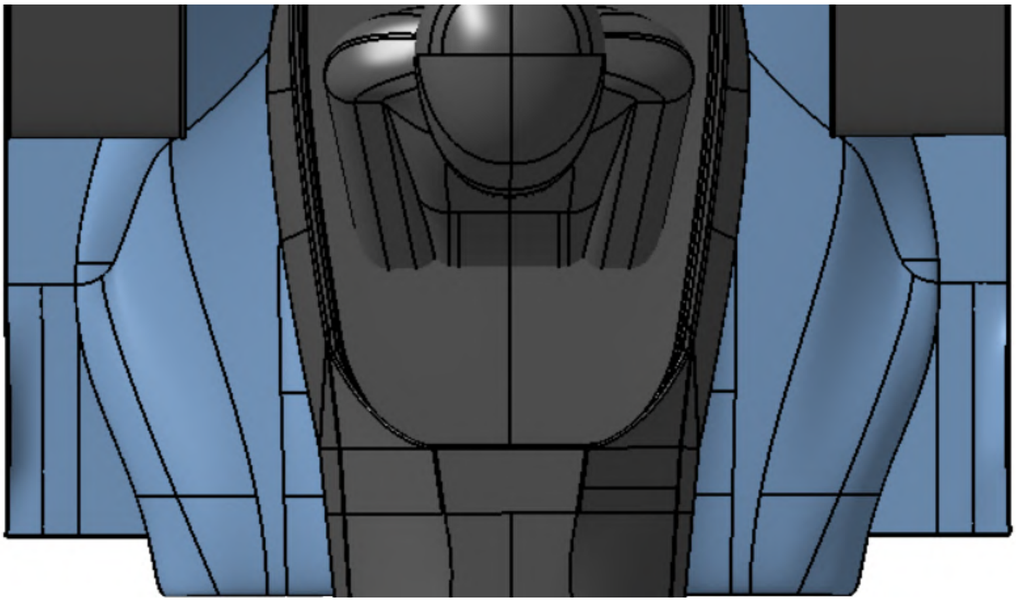


Figure 3.6: Upper - zone geometry

All these design choices, derived from a combination of theoretical considerations, the specific requirements of the car, and the evolution from previous concepts, have streamlined the subsequent optimization process. Additionally, the position of the Centre Of Pressure (COP) was carefully considered and monitored throughout the entire design process. This, in fact, coupled with downforce generation, significantly contributes to the car's performance and maneuverability.

3.2 Parametric CAD geometry

Undertray's design was developed in *Catia V5*.
The geometry is carefully constructed in order to be parameterized in all the interesting aspects, with parameters that define lengths, angles and curves tensions.
Firstly, basis lines and splines are determined, with the aim to be modifiable in their geometric features, without exceeding boundary boxes and following the choices explained in the previous chapter.
Surfaces are then constructed on these lines and closed together as a part body.

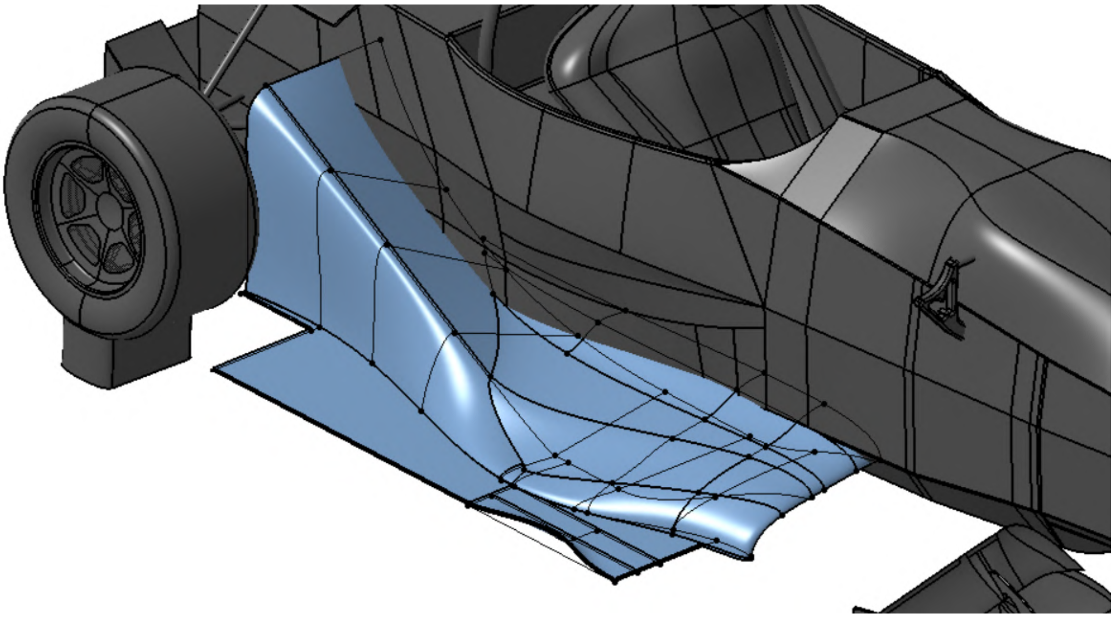


Figure 3.7: The *skeleton* of lines that describes and controls the undertray geometry

Some of the geometry features that can be controlled thanks to this CAD are:

- divergent height: one of the most important parameters to optimize;
- divergent tensions: two curves from the flat - zone describe the divergent shape internally and externally. Tension determines the grade of tangency with the horizontal flat - zone. Lower values led to sharper curves near the tangency constraint;
- flat - zone height: needs to be as lower as possible while higher than the 30mm from ground and the underside of the monocoque;

- flat - zone position and length: with undertray's start and end fixed, this value determines convergent and divergent lengths other than flat zone dimension. These values can assume independent values in the internal and external part;

- flat - zone wideness: determines the wideness of the widest part of the channel, determined from the most external part of the undertray;

- convergent height: important parameter to avoid losses earlier in the undertray;

- LE tension: the tensions of the splines that describe the LE determines its dimensions, inner sections are important for the undertray - monocoque interference;

- upper - zone height: determines the dimensions of the different - shaped upper surface of the undertray;

- lateral convergent wideness, height and position: to optimize the lateral convergent.

Parameters range and values are selected in order to have a good balance between optimization time and understanding of the parameter's influence.

3.3 CFD optimization

The optimization phase was done using the CFD software *Star CCM+*, changing geometric parameters between every simulation. In this section only the optimization procedure will be analysed, leaving the CFD implementation to the next chapters.

The procedure can be divided into two steps:

- coarse optimization: wide range for the chosen parameters with a low number of discrete values. This first step helps in understanding the influence of parameters and can identify the range within which the best results lie;
- fine optimization: with restricted ranges and a larger number of discrete values to obtain the best possible result.

Every individual step is also subdivided into groups of parameters that are optimized one after the other:

- flat - zone parameters;
- divergent parameters;
- convergent parameters;
- lateral - zone parameters;
- upper - zone parameters.

This choice derived from early studies that demonstrated which parameters mostly affected others, and from the zone's importance. It's in fact beneficial to initially determine the convergent and divergent lengths, modifying the flat - zone, and then starting the optimization from the most influencing parameters of the divergent to the least ones of the lateral - zone.

The objective of the optimization process for these parameters it's the maximum downforce generation, while drag and COP position are only checked at every iteration.

The upper - zone optimization it's the only step where downforce is not very influenced, and thus drag and wall shear stress scenes are evaluated.

The optimization detailed results analysis is not objective of this thesis, so only the results considered most influential and relevant to future studies will be presented.

Dividing as before the undertray in zones results showed for the flat - zone:

– best results are obtained for an external flat - zone length of 150 mm and an internal one of 200 mm. This difference is due to the proximity of the monocoque, with the internal part of the undertray that requires a longer flat - zone in order to avoid interference disturbs;

– the widest the flat - zone the best the results are. A value of 100 mm brings the higher value of downforce.

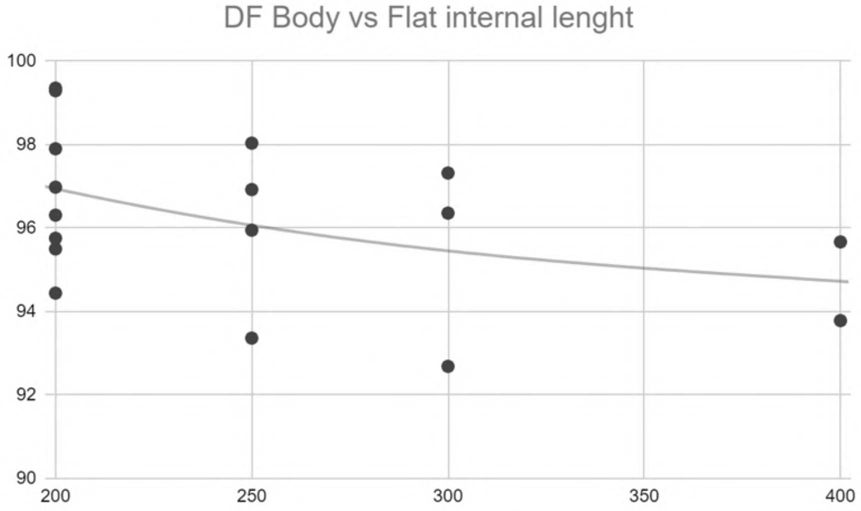


Figure 3.8: Body downforce [N] at 15 m/s vs flat internal lenght [mm]

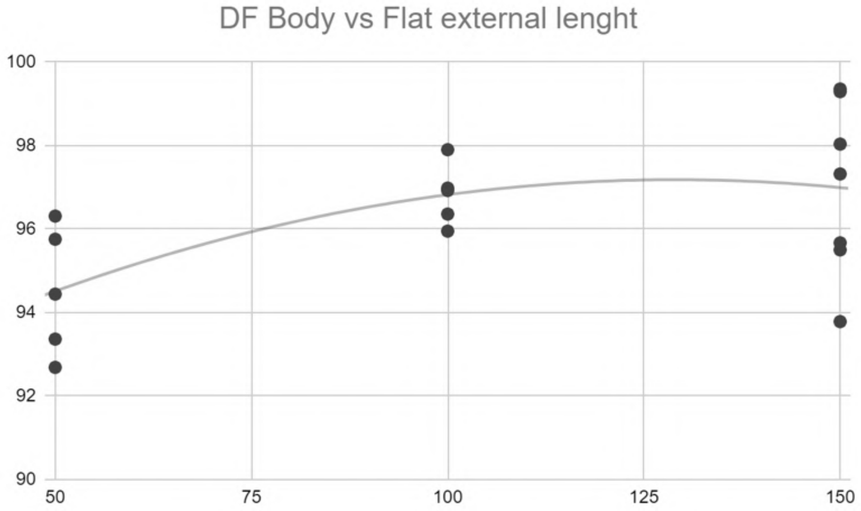


Figure 3.9: Body downforce [N] at 15 m/s vs flat external lenght [mm]

For the divergent:

- divergent height clearly shows how a bigger divergent final area led to higher values of downforce. A value of 370 mm from the monocoque lowest point is the safe maximum reached;
- divergent tensions are the most influencing parameter. A value of 0.1 leads to the best results for the external part, while the internal curvature tends to prefer higher values of tension, setting the optimal value to 1.3. This demonstrates how the internal portion of the undertray requires lower adverse pressure gradients due to the presence of the monocoque;
- the interaction of these divergent parameters and the flat - zone positioning is fundamental to obtain the best divergent geometry. During the optimization process higher heights and tensions and flat - zones that start rearward cause an higher upwash effect that makes the rear wing to lose performance.

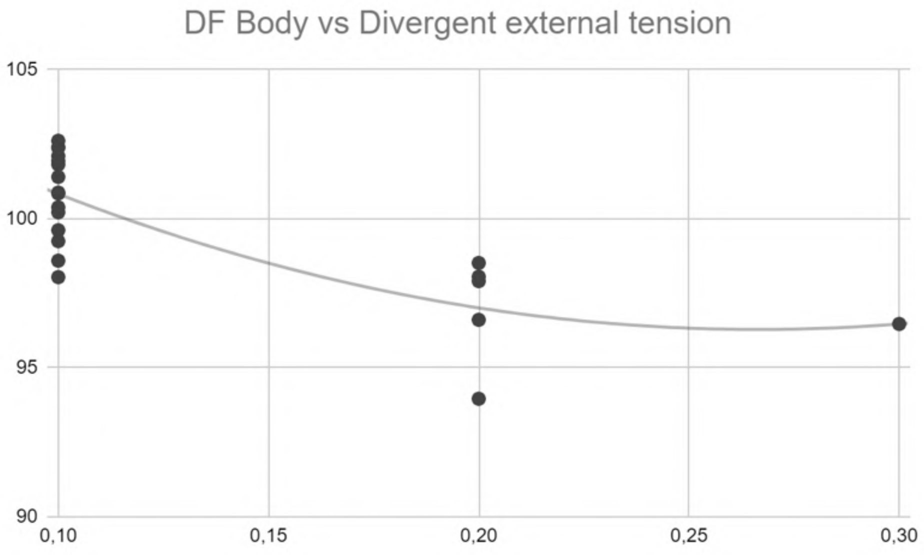


Figure 3.10: Body downforce [N] at 15 m/s vs divergent external tension

For the convergent:

- A maximum in downforce generation was found for a convergent height of 80 mm from the monocoque lowest point.

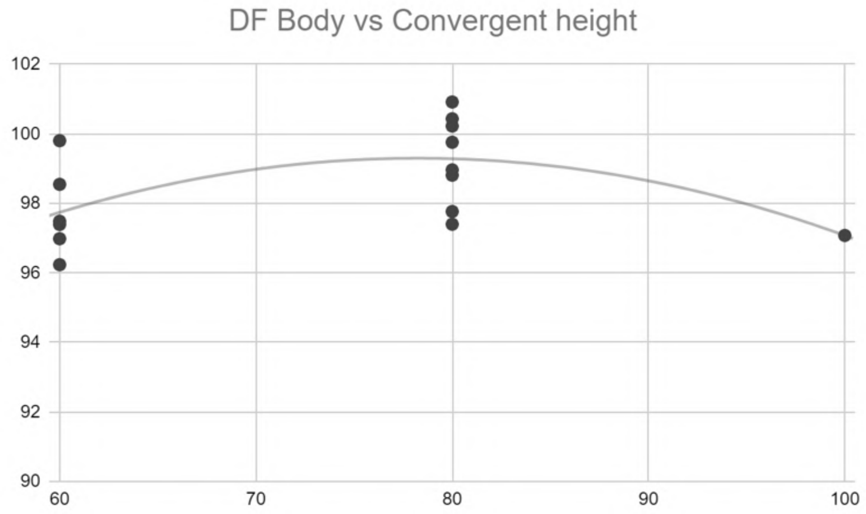


Figure 3.11: Body downforce [N] at 15 m/s vs convergent height [mm]

During the whole optimization phase the downforce of the car was set as objective and a SHERPA algorithm was used to determine how to vary parameters at each design iteration.

3.3.1 Theoretical overview

Optimization processes are commonly used in CFD.
One of the main methods used is the mathematical optimization, which involves formulating CFD problems into mathematical models that can be optimized. This often includes defining an objective function, which could be maximising downforce, optimizing flow rate, or minimizing drag, depending on the specific CFD problem. Constraints, such as geometric or physical limitations, are also considered in the optimization process.

- objective function: in CFD, this represents what needs to be optimized, in this case maximizing downforce;

- constraints: these can include limits on parameters, ensuring adherence to physical laws, or maintaining geometrical properties;

- variable parameter adjustment: varying parameters like mesh density, boundary conditions, or geometrical features to optimize the objective function while considering the specified constraints with specific algorithms.

The ultimate goal of mathematical optimization in CFD is to find the most efficient and effective solutions for fluid flow problems, providing engineers with insights to improve designs, reduce costs, enhance performance, and achieve design objectives in a multidimensional parameter space.
The weighted sum objective function, $F(x)$, is a commonly used algorithm that consists in the linear combination of individual objectives, each multiplied by a weight that reflects its relative importance or priority:

$$F(x) = \sum_{i=1}^{n} w_i \times f_i(x) \tag{3.1}$$

where:

- $F(x)$ is the aggregate (weighted sum) objective function to be maximized or minimized.
- x is the vector of decision variables.
- n is the number of individual objectives.
- $f_i(x)$ is the ith individual objective function.
- w_i is the weight associated with the ith objective.

The weights w_i play a crucial role in determining the trade-offs between different objectives. This approach assumes linearity in the combination of objectives, which may not always be the case. If non-linearity is a concern, other methods like Pareto optimization may be more appropriate.
In Pareto optimization, the objective is to find a set of solutions that represent the Pareto front, where no solution is superior to another concerning all objectives. An improvement in one objective comes at the expense of another.
The Pareto optimization problem can be formulated for a maximization problem as follows:

$$\text{Maximise } F(x), \text{ subject to: } F(x) = [f_1(x), f_2(x), \ldots, f_m(x)]$$

where:

- x is the vector of decision variables.
- m is the number of objectives to be maximized.
- $f_i(x)$ is the ith objective function to be maximized.

In practice, Pareto optimization is often addressed using multi-objective evolutionary algorithms as they are effective in finding and maintaining the Pareto front efficiently.
The Pareto front, once obtained, provides a set of trade-off solutions for decision-makers to choose from, depending on their preferences regarding the various objectives being optimized.
Algorithms in optimization process are instead used to find the best solution or configuration from a set of possible options while satisfying certain constraints. There are a lot of optimization algorithms that helps in finding the final solution sooner, learning from each iteration and avoiding some configurations based on previous solutions.

4

CFD simulation of the car

4.1 Simulation preparation

For the 3D CFD simulations the software *Star CCM+* was used. All these simulations consist in half car simulations in straight motion, using the symmetry property to lower the overall computational cost.
The modeled car is complete with all the main influencing details, including the presence of driver, rotating wheels and modeled radiators, in order to obtain results as close as possible to real conditions, while simplifying all the small details that will not influence the external flux, such as buttons, switches and small wires, other than all the internal part of the vehicle.

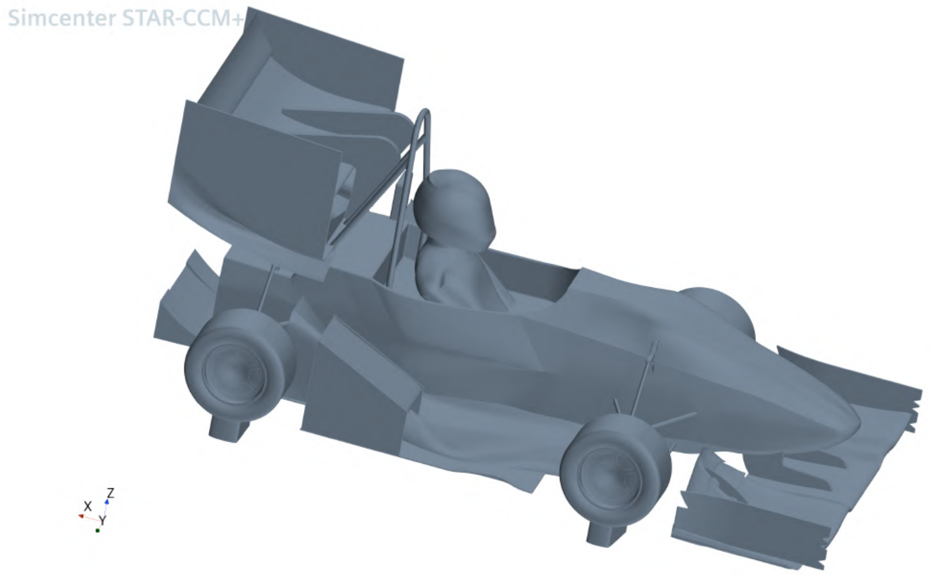


Figure 4.1: Geometry scene in *Star CCM+*

Before starting the CFD analysis, CAD geometry must be checked, ensuring the absence of low quality bodies or surfaces, bad defined tangencies and sharp edges (in this case all the TEs have a fillet of 0.1 mm radius), avoiding bad cells in the meshing phase.
All the simulations will be run at the velocity of 15 m/s, that is considered to be the mean velocity that the car maintain during a race. Considering standard air characteristics:

$$\rho = 1.225 \ kg/m^3$$

$$\mu = 1.812 \cdot 10^{-5} \ Pa \cdot s$$

and considering the car's approximated total length of 2.85 m as the reference length, Reynolds number can be computed:

$$Re = \frac{\rho \cdot u \cdot L}{\mu} \simeq 2.9 \cdot 10^6 \tag{4.1}$$

Even though it's not a very high Reynolds application, it's sufficient to expect a turbulent boundary layer, thus wall treatment will be crucial in setting the simulation.

4.1.1 Theoretical overview

Wall treatment is a really important aspect o deal with in external aerodynamics. Close to a body, as said earlier, viscous phenomena can't be neglected even at high Reynolds. This phenomenon takes place in a thin layer of fluid close to the wall, called boundary layer. On the wall boundary conditions need to be satisfied:

- no slip condition: $\vec{u}|_{\partial C} = 0$;
- impermeability condition: $\vec{v}|_{\partial C} = 0$.

Where $\vec{u}$ is the velocity parallel to the wall and $\vec{v}$ is the perpendicular one.
In the boundary layer the transition from boundary conditions to undisturbed flux happens. This can assume different profile velocities depending on Reynolds number:

- laminar boundary layer: the flow is smooth and streamlined. The velocity of the fluid increases gradually from zero at the solid surface to the free - stream velocity away from the surface. In particular, for $10^3 < Re < 10^6$ Blasius equations provide a solution;
- transition phase: for $Re \simeq 10^6$, shape and length depend on the boundary conditions;
- turbulent boundary layer: the flow is chaotic and characterized by irregular fluctuations in velocity and pressure. Fluid particles move in a less orderly fashion, causing mixing and eddies in the flow. Turbulent boundary layers have higher velocity gradients near the surface, causing major wall shear stresses;
- detached boundary layer: a geometry shape that creates strongly adverse pressure gradients can cause the boundary layer detachment. This is

delayed at higher Reynolds and with the presence of a turbulent boundary layer. In applications, tools such as vortex generators are used to energize the boundary layer, postponing its detachment.

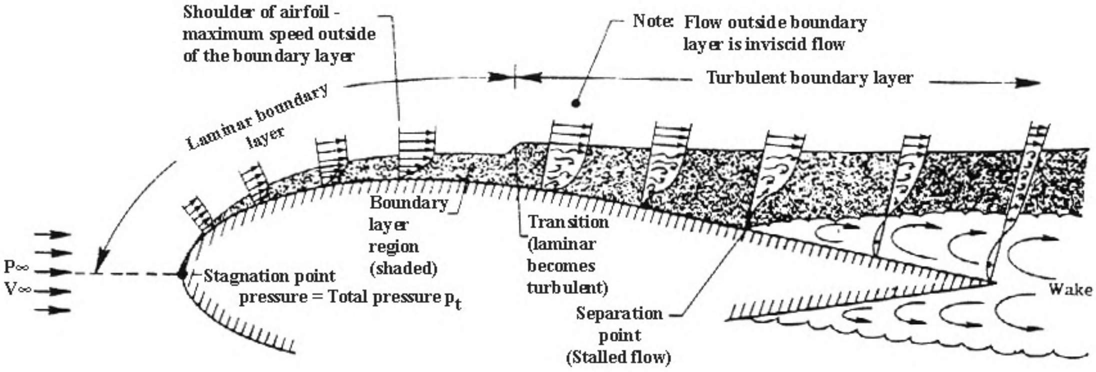

Shoulder of airfoil - maximum speed outside of the boundary layer
Note: Flow outside boundary layer is inviscid flow
Turbulent boundary layer
Laminar boundary layer
Boundary layer region (shaded)
Transition (laminar becomes turbulent)
P∞
V∞
Stagnation point pressure = Total pressure p_t
Separation point (Stalled flow)
Wake

Figure 4.2: Boundary layer definition

If the case of a turbulent channel is considered, some interesting notations on the turbulent boundary layer can be obtained and then extended and used also in external aerodynamics [12].
The mean velocity profile in the direction parallel to the wall, expressed in inner coordinates, is given by:

- $y^+ = \frac{y}{\delta_\nu} = \frac{U_\tau}{\nu} y$;
- $U^+ = \frac{<u>}{U_\tau}$.

Where δ_ν is viscous length, reference scale for the layer closest to the surface, the friction velocity U_τ is defined as:

$$U_\tau = \sqrt{\frac{\tau_w}{\rho}} \tag{4.2}$$

and the wall shear stress is defined as:

$$\tau_w = \nu\rho \left. \frac{d<u>}{dy} \right|_{y=0} \tag{4.3}$$

The three different types of dimensionless stresses with respect to τ_w as a function of the distance from the wall are as follows:

- viscous stresses (τ_ν): dominant in the region closest to the wall, defined as $\tau_\nu = \mu \frac{d<u>}{dy}$. They move closer to the wall with increasing Re;

- Reynolds stresses (τ_r): overcome by viscous stresses in the region closest to the wall, defined as $\tau_r = -\rho < u'v' >$, and important far from the wall. They dominate more and more channel with increasing Re;
- total stresses (τ_t): sum of viscous and Reynolds stresses, defined as $\tau_t = \tau_v + \tau_r$.

Two main regions can be defined (with δ half channel's height):

- inner region: close to the wall, dominated by viscous effects: $y/\delta < 0.1$;
- outer region: dominated by Reynolds stresses $y/\delta_v > 100$;
- overlap region: $y^+ > 50$, $y/\delta < 0.1$ in which there's a coexistence of the two regions.

In the inner region's other subdivisions can be defined:

- the viscous sublayer ($y^+ < 5$), $U^+ = y^+$, where the Reynolds stresses are negligible compared to viscous stresses;
- the log - law region ($y^+ > 30$, $y/\delta < 0.1$), $U^+ = \frac{1}{k}\ln(y^+) + B$, valid in the overlap region;
- the buffer layer between the viscous sublayer and the log - law region ($5 < y^+ < 30$).

Additionally the outer layer is observed for $y^+ > 50$, which connects with the log - law of the inner layers in the overlap region with the defect law: $B_1 - \frac{1}{k}\ln(\frac{y}{\delta})$. The two logarithmic laws must coincide.

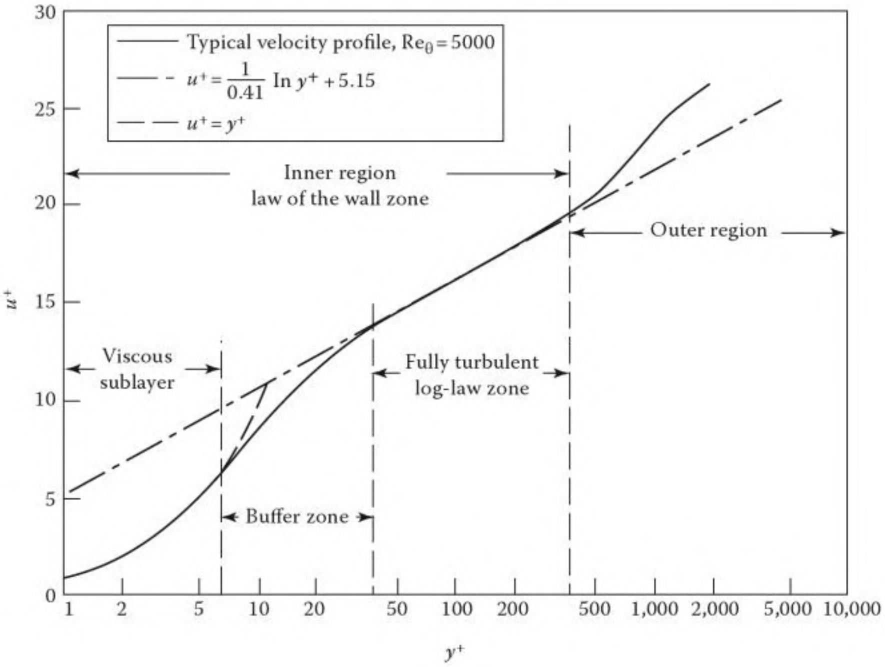


Figure 4.3: y^+ definition

During simulations, y^+ value of the first elements on the wall must be checked in order to clarify how well the boundary layer is described. If a value of $y^+ < 5$ is obtained, a precise description of the wall stresses is reached. For higher values stresses are underestimated, instead for $y^+ > 30$ the velocity profile is described by the log - law.

A precise estimation of the boundary layer profile is essential in external aerodynamics studies, with the aim of predict eventual detachments and stall conditions.

4.2 Physics

The physics and models selection is fundamental to reach a stable and accurate solution. In this case in *Star CCM+* the physics selections made are:

- space: tree dimensional, half of the car is simulated;
- time: steady, Reynolds Averaged Navier Stokes (RANS) are used, time step has no physical meaning;
- material: gas, air with $\rho = 1.225\ Kg/m^3$ and $\mu = 1.812 \cdot 10^{-5}\ Pa \cdot s$;
- flow: coupled flow;
- equation of state: constant density, $\rho = \rho_0$;
- viscous regime: turbulent, as said $Re \simeq 2.9 \cdot 10^6$;
- Reynolds - averaged turbulence: K - Omega Turbulence (SST (Menter) K - Omega);
- wall treatment: all y^+ treatment, trying to keep $y^+ < 5$.

The article *Best Practice Guidelines for Formula Student Cars* [15] is followed for the values set in the simulation.

4.2.1 Theoretical overview

It's essential to introduce Navier - Stokes equations and methods used for their solution in order to understand choices made [1], [2].
Navier - Stokes equations are a system of partial differential equations which describes the motion of viscous fluids, expressing the conservation of momentum and the conservation of mass for Newtonian fluids. For a compressible fluid, the Navier - Stokes equations (mass and momentum conservation) can be expressed together with the equation for the conservation of energy in the so called conservation form as:

$$\begin{cases} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \\ \frac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) = -\nabla p + \nabla \cdot \boldsymbol{\sigma} + \rho \mathbf{f} \\ \frac{\partial (\rho E)}{\partial t} + \nabla \cdot (\rho \mathbf{u} E) = -\nabla \cdot (p \mathbf{u}) + \nabla \cdot (\boldsymbol{\sigma} \cdot \mathbf{u}) + \nabla \cdot (k \nabla T) + \rho \mathbf{f} \cdot \mathbf{u} + Q \end{cases} \tag{4.4}$$

with the viscous stress tensor $\boldsymbol{\sigma}$:

$$\boldsymbol{\sigma} = \mu(\nabla\mathbf{u} + (\nabla\mathbf{u})^T) + \lambda\nabla\cdot\mathbf{u}\mathbf{I} = 2\mu\mathbf{e} - \frac{2}{3}\mu(\nabla\cdot\mathbf{u})\mathbf{I} \tag{4.5}$$

Let us consider an incompressible flow such that $\rho = \rho(x,t)$, $k = k(x,t)$, and $\mu = \mu(x,t)$ are constants. The equations in conservative form become:

$$\begin{cases} \nabla\cdot\mathbf{u} = 0 \\ \frac{\partial\mathbf{u}}{\partial t} + \nabla\cdot(\mathbf{u}\otimes\mathbf{u}) = -\frac{\nabla p}{\rho} + \nu\nabla^2\mathbf{u} + \mathbf{f} \end{cases} \tag{4.6}$$

Here, $\nu = \frac{\mu}{\rho}$ represents the kinematic viscosity, where μ is the dynamic viscosity, ρ is the density, $\mathbf{u}$ is the velocity vector, p is the pressure, and $\mathbf{f}$ represents any external forces or body forces acting on the fluid.

Most fluid flows of engineering interest are characterized by irregularly fluctuating flow quantities. Often these fluctuations are at such small scales and high frequencies that resolving them in time and space comes at excessive computational costs. Rather than employing Direct Navier - Stokes (DNS), which involves solving the complete governing equations for turbulent flows, a more cost-effective approach is to solve for averaged or filtered variables while approximating the influence of the smaller turbulent structures. Turbulence models offer various methodologies to represent and model these turbulent structures. RANS turbulence models provide closure relations for the Reynolds - Averaged Navier - Stokes equations, that govern the transport of the mean flow quantities. To obtain the Reynolds-Averaged Navier-Stokes equations, each solution variable ϕ in the instantaneous Navier-Stokes equations is decomposed into its mean, or averaged, value $\overline{\phi}$ and its fluctuating component ϕ':

$$\phi = \overline{\phi} + \phi' \tag{4.7}$$

where ϕ represents velocity components, pressure, energy, or species concentration. Incompressible Reynolds - Averaged Navier - Stokes equations are obtained:

$$\begin{cases} \nabla\cdot\overline{\mathbf{u}} = 0 \\ \frac{\partial}{\partial t}\overline{\mathbf{u}} + \nabla\cdot(\overline{\mathbf{u}}\otimes\overline{\mathbf{u}}) = -\frac{\nabla p}{\rho} + \nabla\cdot(2\nu\overline{\mathbf{e}}) - \nabla\cdot < \mathbf{u}'\mathbf{u}' > \end{cases} \tag{4.8}$$

with $< \mathbf{u}'\mathbf{u}' >$ Reynolds stresses that:

$$tr(< \mathbf{u}'\mathbf{u}' >) = 2k \tag{4.9}$$

k is the Turbulent Kinetic Energy (TKE) and its equation can be written as:

$$\frac{\partial k}{\partial t} = C + F - D + P \tag{4.10}$$

and its mean:

$$\frac{\partial \overline{k}}{\partial t} = \mathbb{C} + \mathbb{F} - \mathbb{D} + \mathbb{P} \tag{4.11}$$

C is convective transport, F is spatial flux of k, D dissipation and P production. In the mean field, for $Re >> 1$ and in steady state conditions, energy accumulated in the term $\mathbb{C}$ it's not dissipated by the term $\mathbb{D}$ that tends to 0, it follows that all the energy is used to produce fluctuations ($\mathbb{C} \simeq \mathbb{P}$), that will be entirely dissipated at the smaller scales in the fluctuation field, where $\mathbb{P} \simeq D$. Smaller scales can be considered universal.

It's necessary to model Reynolds stresses in order to solve the equations. Eddy viscosity models are based on the analogy between the molecular gradient-diffusion process and turbulent motion and the most common model is known as the Boussinesq approximation:

$$\begin{cases} \mathbf{T} = -p\mathbf{I} + 2\nu\mathbf{e} \\ - < \mathbf{u}'\mathbf{u}' >= -\frac{2}{3}k\mathbf{I} + 2\nu_T\overline{\mathbf{e}} \end{cases} \tag{4.12}$$

with ν_T turbulent viscosity, that's the variable that needs to be modeled in order to obtain a N equation system in N unknowns:

$$\begin{cases} \nabla \cdot \overline{\mathbf{u}} = 0 \\ \frac{\partial}{\partial t}\overline{\mathbf{u}} + \nabla \cdot (\overline{\mathbf{u}} \otimes \overline{\mathbf{u}}) = -\nabla(\frac{p}{\rho} + \frac{2}{3}k) + \nabla \cdot [2(\nu + \nu_T)\overline{\mathbf{e}}] \end{cases} \tag{4.13}$$

Most frequently used turbulence models are the k - Epsilon, the K - Omega and the Spalart - Almaras models. The first two are two equations models, while the third one it's a one equation model, valid for simple fluxes in external aerodynamics.

The K -Epsilon model:

$$\begin{cases} \nu_T = C_\mu \frac{k^2}{\epsilon} \\ \frac{\partial \epsilon}{\partial t} + \nabla \cdot (\mathbf{u}\epsilon) = \nabla \cdot [(\frac{\nu_t}{\sigma_\epsilon})\nabla \epsilon] + C_{\epsilon 1} \frac{p\epsilon}{k} - C_{\epsilon_2} \frac{\epsilon^2}{k} \end{cases} \tag{4.14}$$

This model shows good results outside the boundary layer, while the K - Omega model works better in representing the adverse pressure gradient in turbulent boundary layers:

$$\begin{cases} \nu_T = C_\mu \frac{k}{\omega} \\ \frac{\partial \omega}{\partial t} + \nabla \cdot (\mathbf{u}\omega) = \nabla \cdot [(\frac{\nu_t}{\sigma_\omega})\nabla \omega] + C_{\omega_1} \frac{p\omega}{k} - C_{\omega_2} \omega^2 \end{cases} \tag{4.15}$$

This model is sensitive to the input variables, leading to different results with slightly different ω_{in}. For this reason combined models have been formulated:

- k - ω - BSL: k - ϵ far from the walls and k - ω close to them;
- k - ω - SST: BLS that transfer shear stresses at the walls.

A more expensive strategy of simulation could be the Large Eddy Simulation (LES). These kind of simulations create models only for the smaller universal scales, dissipating all the energy at the scale Δ chosen, while bigger eddies are simulated. This allows to obtain a more universal model with only a constant in the Smagorinski model and zero in the dynamic Smagorinski model.

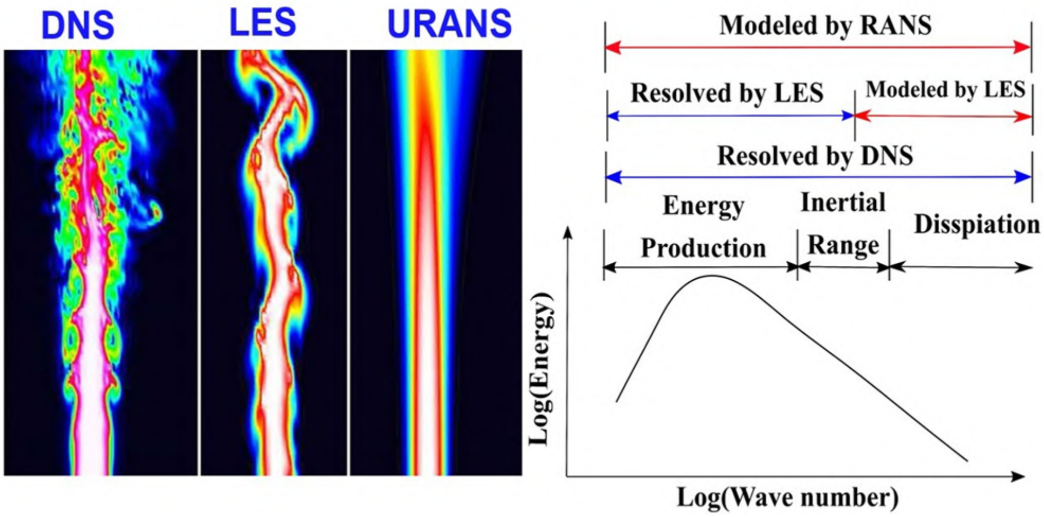


Figure 4.4: Solution approaches comparison

4.3 Regions

The half car is placed in a domain big enough to represent a real free stream condition, with the presence of the moving ground. With respect to the reference coordinate system origin, placed at half wheelbase of the car on the ground, the domain extends for 24 m in the negative x and 42 in the positive x, it's 11 m wide (positive y direction) and 11 m high (z direction).

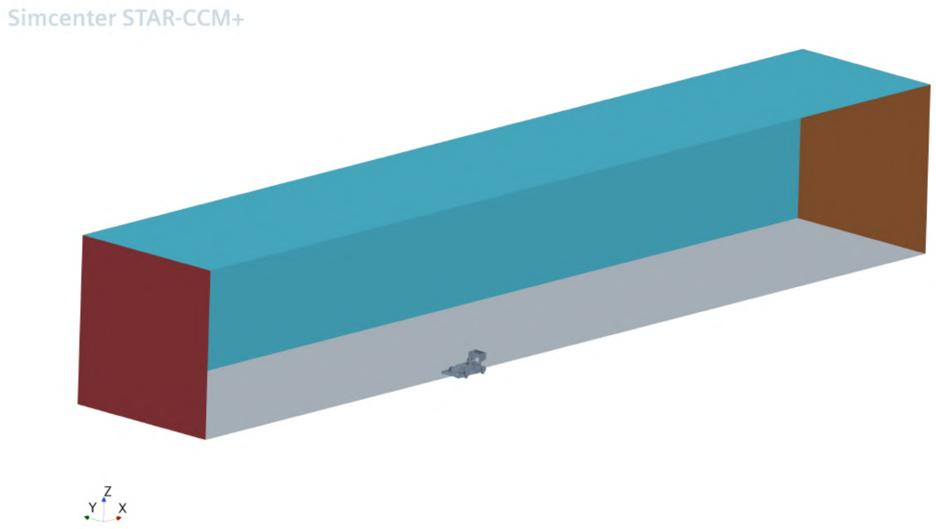


Figure 4.5: Domain of simulation

Surfaces representing the car and the domain are divided in regions' boundaries, in order to get assigned the right boundary conditions and to be better defined in post processing.
Four regions define the whole simulation:

- One car and domain surfaces fluid region: the main surfaces are divided in groups and assigned to specific boundary conditions;
- Two rotating Moving Reference Frame (MRF) fluid regions (front and rear): essentials in defining the interfaces of the rims, to simulate the rotating wheels;
- One porous region to simulate a simplified radiator geometry.

Each region is composed by boundaries. For the main region:

- Front Wing (FW): defined as a wall with no slip, contains all the front wing elements;

- Rear Wing (RW): wall with no slip, contains all the rear wing elements;
- Body: wall with no slip, contains all the remaining parts of the car that don't rotate. This boundary is the one used to evaluate the undertray performance;
- Front axle: wall with no slip with an imposed rotation of 73.819 rad/s around the wheel axis. It includes front tire, rim and brake disk;
- Rear axle: wall with no slip with an imposed rotation of 73.819 rad/s around the wheel axis. It includes rear tire, rim and brake disk;
- Inlet: velocity inlet given in components, with an x velocity of 15 m/s (that corresponds to the wheels angular velocity);
- Outlet: pressure outlet that impose atmospheric pressure at the end of the domain;
- Ground: wall with no slip with a velocity x component of 15 m/s, in order to simulate the moving car;
- Symmetry: symmetry plane of the half car;
- Top and Side: patches far from the car, *Siemens* suggests to put the symmetry condition, that in this case will not mirror anything;
- Shroud + Fan: group of boundaries that defines the interfaces of the channel at the rear of the radiator and at end of the duct, where the fan is placed. Fan interface is set with a specified volumetric flow rate.
- Other boundaries to create interfaces between different regions: these are all wall regions with no slip condition that needs to be set in order to include MRFs and radiator in the simulation.

The MRFs regions rotate at the same rate of the wheels and contains only two boundaries. One determines the interfaces with the main region described earlier, the other specifies the internal solid part of the rims (wall with no slip). The porous region contains the radiator geometry, that is simplified as a block with a porous inertial resistance of 192.529 kg/m^4 in the direction orthogonal to radiator's front surface (0 kg/m^4 in the other two directions) and a porous viscous resistance of 773.353 $kg/(m^3 \cdot s)$ in the same direction, while 10000 $kg/(m^3 \cdot s)$ in the others. Interfaces are set as for the other regions.

4.4 Meshing

Meshing phase is critical to achieve accurate and stable results. Computational cost must be considered, avoiding over defined meshes saving a lot of time.
Mesh is built in the software *Star CCM+*. Three different mesh groups define the meshes for car and domain, MRFs and the porous region.
As said earlier the wall treatment aims to reach a $y^+ < 5$. To do so the first cell thickness is set to an optimal value of 0.03 mm on wall boundaries, an optimized value that derives from a first estimation done as follows:

- Prandtl friction coefficient for a turbulent boundary layer:

$$C_f = 0.0576 \cdot Re^{-\frac{1}{5}} \simeq 2.94 \cdot 10^{-3} \tag{4.16}$$

- wall shear stress estimation:

$$\tau_w = \frac{1}{2} C_f \rho U_{inf}^2 \simeq 0.40 \ Pa \tag{4.17}$$

- friction velocity:

$$U_\tau = \sqrt{\frac{\tau_w}{\rho}} \simeq 0.57 \ m/s \tag{4.18}$$

- and thus the first cell thickness, with $y^+ = 1$:

$$y = \frac{\nu \cdot y^+}{U_\tau} \simeq 2.57 \cdot 10^{-5} \ m \tag{4.19}$$

The main domain is meshed with trimmed cells that provides a robust and efficient method of producing a high-quality grid for both simple and complex mesh generation problems. It combines a number of highly desirable meshing attributes in a single meshing scheme:

- predominantly hexahedral mesh with minimal cell skewness;
- refinement that is based upon surface mesh size and other user-defined refinement controls;
- surface quality independence;
- alignment with a user-specified coordinate system.

For these reasons this is the mesh chosen also for the porous region of the inclined radiators.

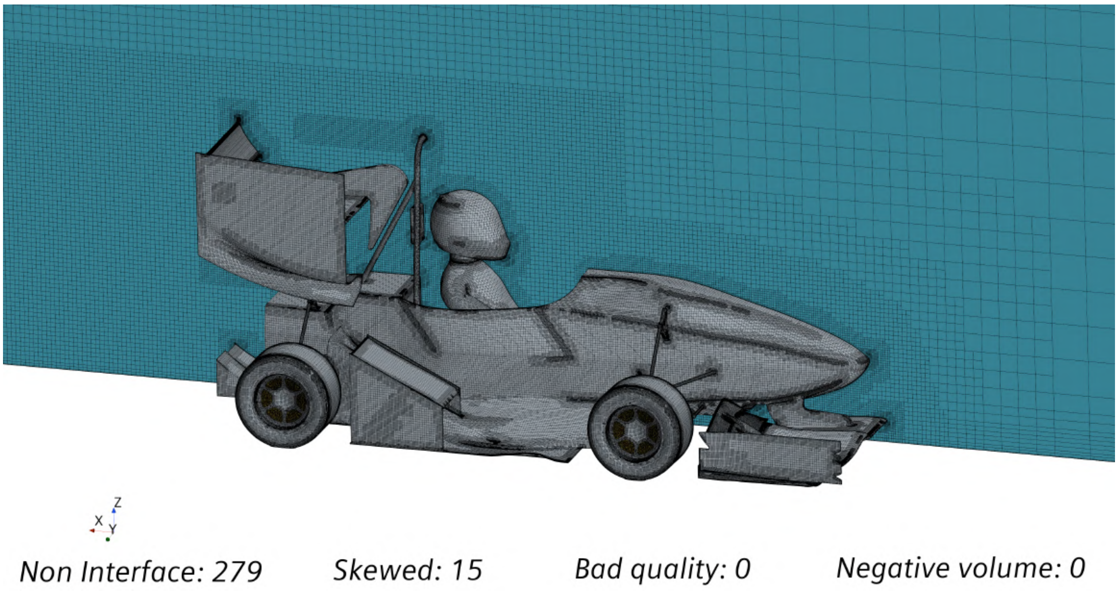


Figure 4.6: Trimmed mesh for the car

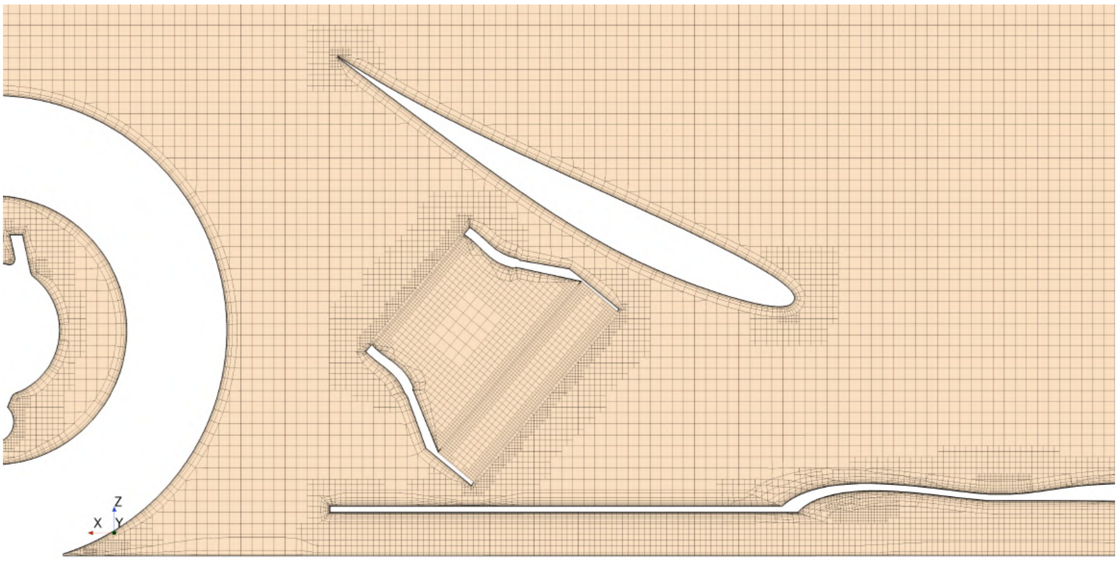


Figure 4.7: Inclined radiators trimmed mesh section

For the MRFs polyhedral mesh is preferred. It provides a balanced solution for complex mesh generation problems and it's recommended for MRFs interfaces.

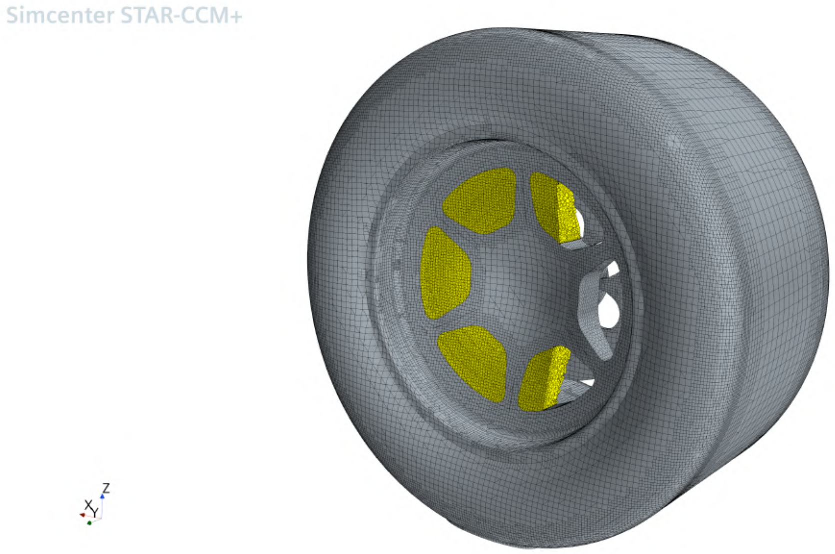


Figure 4.8: MRFs polyhedral mesh particular

The most important commands in the mesh generation are the ones that control the surface cells size and the prism layer, that represent the connection between the first cell that determines the y+ and the first surface cell. The first ones are:

- Base Size (BS): set to 0.06 m
- Minimum Surface Size (MSS);
- Target Surface Size (TSS).

The base size determines the dimension of the reference cell. All the other cells commands are set as relative to this value. Target and minimum surface size are different for every car's component, with values that were previously studied in a mesh sensitivity analysis. With the trimmed cell mesh, cells can only double in dimension at every step, so the values set in these commands will only be of the type: $2^n, n \in \mathbb{Z}$.
For the prism layer definition:

- prism layer Near Wall Thickness (NWT): 0.03 mm;
- Prism Layer Total Thickness (PLTT);
- Number of Prism Layers (NPL).

The values of the last two commands are set differently for every component. The sensitivity analysis, in fact, led to a correlation between these values and the target surface size for each component with the method that follows:

– quality parameters of the prism layer optimised in the sensitivity study:

  - Stretch Factor (SF): defines the gradient of the geometric progression in the prism layer, values range $[1.2, 1.5]$;
  - TSS and Last Prism Layer cell Thickness (LPLT) ratio: determines the continuity between prism layer and core mesh. Safe maximum value found is 3;

– NPL calculation from:

$$LPLT = NWT \cdot SF^{NPL-1} \tag{4.20}$$

$$\implies NPL = \log_{SF} \frac{LPLT}{NWT} + 1 \tag{4.21}$$

– PLTT calculation from the geometric progression:

$$PLTT = NWT \frac{1 - SF^{NPL}}{1 - SF} \tag{4.22}$$

This led to a continuous mesh close to the walls, fundamental for the simulation quality.

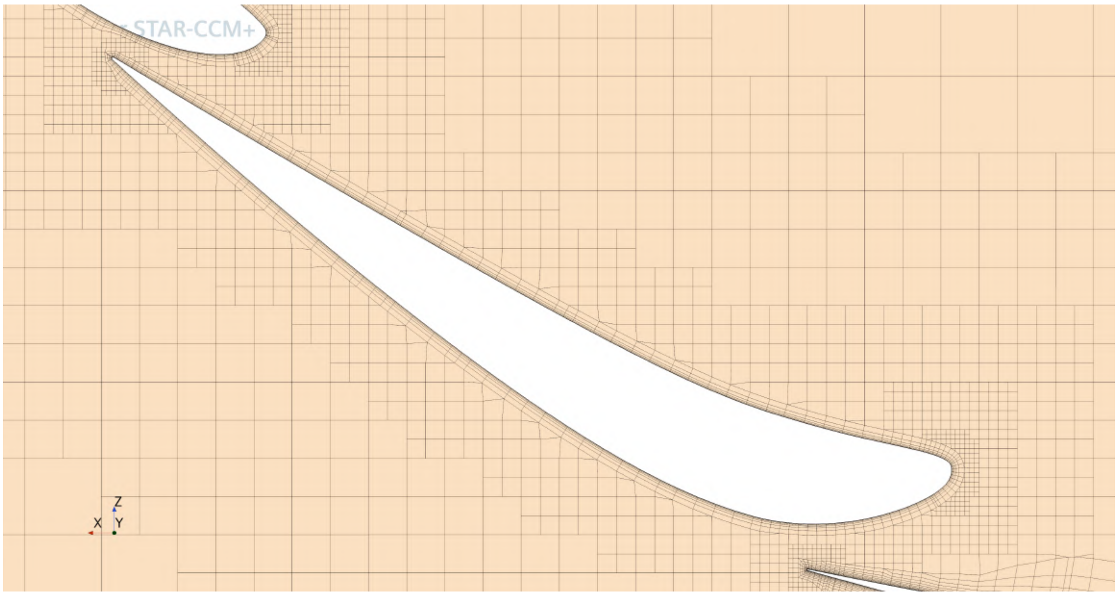


Figure 4.9: Particular of the mesh for a front wing flap

For interfaces (non solid parts of MRFs and radiators and fan) the prism layers number is set to 1 as recommended.
Volumetric refinements are also present to better capture wheels wake and the rear wake of the car, with different steps in refinement.

4.4.1 Sensitivity analysis

A sensitivity analysis for the mesh was done in order to calibrate the values presented earlier. It's in the interest of this thesis bringing the example on how ground effect parts of the car have been studied and not entering in the detail of every aspect studied during the analysis.

In general every critical part of the simulation was studied separately, starting from a over refined mesh and slowly decreasing the number of elements, verifying where the best trade off between computational cost and simulation accuracy was. During this phase not only the presented commands were modified, but also wake and edge refinements were evaluated, adding controls or volume refinements if necessary.

To study the meshing in ground effect, a generic double element wing was chosen, with a 500 mm chord main and a 200 mm chord flap. Other than forces, also TKE volume (above 6 J/kg) and vortices using Q - Criterion (1000 s^{-2}) are monitored.

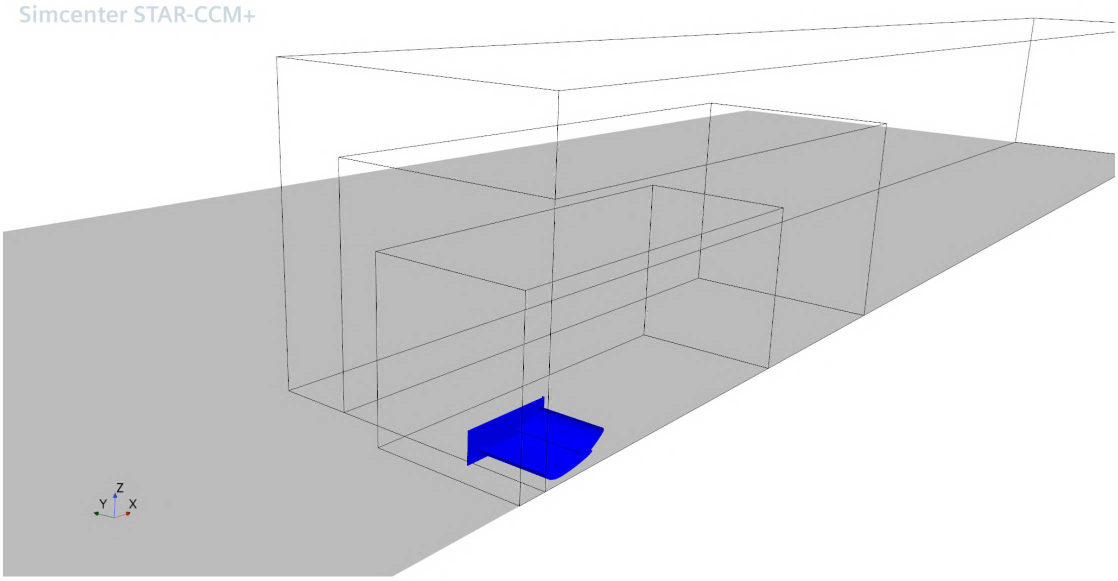


Figure 4.10: Sensitivity study geometry and volume refinements for ground effect treatment

After determining the volume refinements, the base size is changed, keeping these values fixed:

- SF = 1.4;
- LPLT = 2;

- NWT = $3.0 \cdot 10^{-5}$ m;
- main element:
  - TSS = 16 % (relative to base size);
  - MSS = 4 %;
- flap element:
  - TSS = 8 %;
  - MSS = 1 %;
- TEs absolute size = $1 \cdot 10^{-4}$ m.

Results showed:

BS [m]	ClA [m^2]	CdA [m^2]	Vortices volume [m^2]	TKE volume [m^2]
0.04	0.464	0.088	3.34	1.79
0.045	0.465	0.088	3.34	1.80
0.05	0.465	0.088	3.34	1.85
0.055	0.461	0.088	3.34	1.90
0.06	0.464	0.089	3.32	1.92
0.065	0.468	0.091	3.15	1.58
0.07	0.469	0.091	3.24	1.52

Table 4.1: Base size variation for mesh sensitivity analysis

BS [m]	ClA error [%]	CdA error [%]	Vortices error [%]	TKE error [%]
0.04	0	0	0	0
0.045	0.2	0	0	0.6
0.05	0.2	0	0	3.4
0.055	0.6	0	0	6.1
0.06	0	1.1	0.6	7.3
0.065	0.9	3.4	5.7	11.7
0.07	1.1	3.4	3.0	15.1

Table 4.2: Errors in the simulation for different base sizes

It can be seen how the volume of TKE is a critical value, clearly influenced by cells dimensions. A lower accuracy in the wake could lead to worse results in components at the back, directly influenced by its volume. Forces are good for values of base size lower than 0.6 m, that is the value used in the complete mesh with final refinements that keep the TKE error lower and a surface proximity refinement that fills spaces lower than 60 mm with a minimum number of 8 cells (excluding prism layer cells).

For each base size, prism layer values varied automatically as described before. With a fixed base size of 0.06 m, the stretch factor is then varied to determine its influence:

SF	ClA error [%]	CdA error [%]	Vortices error [%]	TKE error [%]
1.3	0	0	0	0
1.4	0.2	0.2	4.2	1.4
1.5	1.1	0.5	4.3	4.6
1.6	1.6	0.8	4.4	6.3
1.7	1.8	0.9	4.8	6.8
1.8	2.6	1.5	4.8	9.2

Table 4.3: Errors in the simulation for different stretch factors

The safe maximum stretch factor turned out to be 1.5. A value of 1.4 is chosen for the final mesh.

A final check of the previously calculated NWT was done to see its influence in the solution. New values of maximum y^+ and surface average y^+ are taken into account:

NWT [m]	ClA [%]	CdA [%]	Vor. [%]	TKE [%]	Max y^+	Avg y^+
$7.5 \cdot 10^{-6}$	0	0	0	0	3.4	0.18
$1.5 \cdot 10^{-5}$	0.2	0.6	2.5	1.2	3.6	0.36
$3 \cdot 10^{-5}$	0.4	1.3	6.1	3.6	4.2	0.72

Table 4.4: Errors in the simulation for different near wall thickness

The value of 0.03 mm gives a good estimation of forces and wakes, while keeping the average $y^+ < 1$ and the maximum $y^+ < 5$. Higher values will cause a lower quality in wall treatment, bringing errors in predicting boundary layer detachments and drag forces.

This example of analysis was done for every other component of the car, and then the final mesh was assembled.

4.5 Solvers

As mentioned previously, the preference here is for a coupled solver over a segregated one.

In the segregated flow solver, the integral conservation equations for mass and momentum are tackled sequentially. The nonlinear governing equations are iteratively solved one after another for the solution variables. The segregated solver employs a pressure - velocity coupling algorithm to satisfy the mass conservation constraint on the velocity field. This is achieved by solving a pressure - correction equation derived from the continuity and momentum equations. Essentially, it seeks a predicted velocity field that fulfills the continuity equation, which is achieved through pressure - correction, a method often called predictor - corrector approach. The pressure as a variable is obtained from this pressure - correction equation.

In contrast, with the coupled solver, the conservation equations for continuity, momentum, energy, and species are simultaneously solved as a vector of equations. The velocity field is derived directly from the momentum equations. The pressure is calculated from the continuity equation, and the density is determined from the equation of state when necessary. This method is more computationally demanding but tends to achieve better convergence with fewer iterations.

The discretization schemes for the convective and diffusive fluxes chosen is the MUSCL - 3rd order/CD scheme that is a blend between a MUSCL 3rd - order upwind scheme and the 3rd - order central - differencing reconstruction scheme. It provides improved (reduced) dissipation when compared with the second-order scheme.

The coupled system of equations can be solved using either an implicit or an explicit time - integration scheme. In transient simulations this is a key aspect for achieving time consistent solutions, with the Courant Friedrichs Lewy (CFL) number that is crucial for the solution accuracy [2]:

$$CFL = \frac{u\Delta t}{\Delta h} \tag{4.23}$$

with Δt timestep and Δh grid dimension. This number must be < 1 or less with explicit time integration schemes in transient simulation. It in fact represents the rate at which information, denoted by the velocity u, traverses a computational grid cell of size Δh during a unit of time Δt. If the Courant number exceeds the unity, it implies that information propagates through more than one grid cell during each time step. This can result in inaccuracies in the solution and may potentially lead to non - physical results or divergence of the solution.

Implicit schemes alleviate such strict CFL condition allowing the use of time steps where the Courant number is higher than 1, however, this limit should not be exceeded in large areas of the computational domain, as this can lead to incorrect results.

CFL can be controlled varying the mesh dimension and the timestep. Adaptive timestep can be good in keeping the CFL number under a specified value, but if mesh definition is inadequate, it could lead to very small timestep and thus too long simulation time.

In the case of steady state simulations, timestep lose its physical meaning, allowing higher CFL number, that can make the simulation faster.

In *Star CCM+* an automatic CFL feature is used for implicit schemes. It adjusts the CFL number in response to Algebraic Multi Grid (AMG) solver convergence behavior to maintain the specified target number of cycles. It targets a balance between the cost of forming the linear system and the cost of solving it. Traditional iterative solvers like Jacobi, Gauss - Seidel, or ILU slow down as mesh sizes increase, causing a quadratic rise in computational time. Multigrid methods enhance solver speed by iteratively addressing error reduction in a hierarchy of coarsened linear systems. This hierarchy allows efficient reduction of low - frequency errors, which standard iterative methods struggle with. Key steps in multigrid methods include cell agglomeration to form coarser grid levels, transferring residuals from fine to coarser levels (restriction), and moving corrections from coarser to finer levels (prolongation). *Star CCM+* offers two methods for accelerating linear solvers: preconditioned conjugate - gradient for incompressible flows and bi - conjugate gradient stabilized for other cases. The second one will be used in the simulations presented together with Gauss - Seidel as relaxation scheme.

The higher the target cycles, the higher the CFL number that can be obtained, with a faster convergence in terms of iteration, but an higher iteration time. 6 AMG target cycles are chosen as trade off, that, together with the defined mesh, keeps the CFL number varying between 50 and 200 during the simulation. The AMG maximum cycles are instead 30, with a convergence tolerance of 0.05 that exits the cycle when reached.

Explicit relaxation is a scaling factor that is used to relax all coupled flow corrections explicitly before they are applied to the flow solution. This generally improves numerical stability and convergence, particularly when running at a high CFL number. For the unsteady solver, the default is 1 in order to be time accurate, in this case of steady state simulation the value of 0.2 showed better results.

Field are initialized with a grid sequencing, that performs the normal initialization followed by the computation of an approximate inviscid solution to the flow problem, reaching higher values of CFL, that is beneficial for the fisrt iterations, acting as a relaxation.

For the k - omega turbulence an under - relaxation factor of 0.8 is used, together with an AMG cycle with an higher convergence tolerance of 0.1.

4.6 Post processing

Post processing simulation results it's really important in order to understand the airflow around the car and to verify theoretical hypothesis or predicted behaviours.

Scenes in *Star CCM+* are used for this purpose, with vector and scalar field that are investigated. Interesting scenes of the simulation with the car at 40 mm from the ground (standard height) are presented.

Pressure coefficient scene together with TKE above 7 J/kg shows well how undertray works. Lowest pressure is located at the end of the flat - zone and then recovered towards the end of the divergent. It can be seen how front wing works in a similar way, and how its outwash effect helps in keeping front tyres wakes outside the undertray. The major part of turbulences are created by the driver and components around him, which implies that the rear wing had to be carefully designed and positioned to avoid the majority of these disturbances, that would lead to great losses in downforce creation.

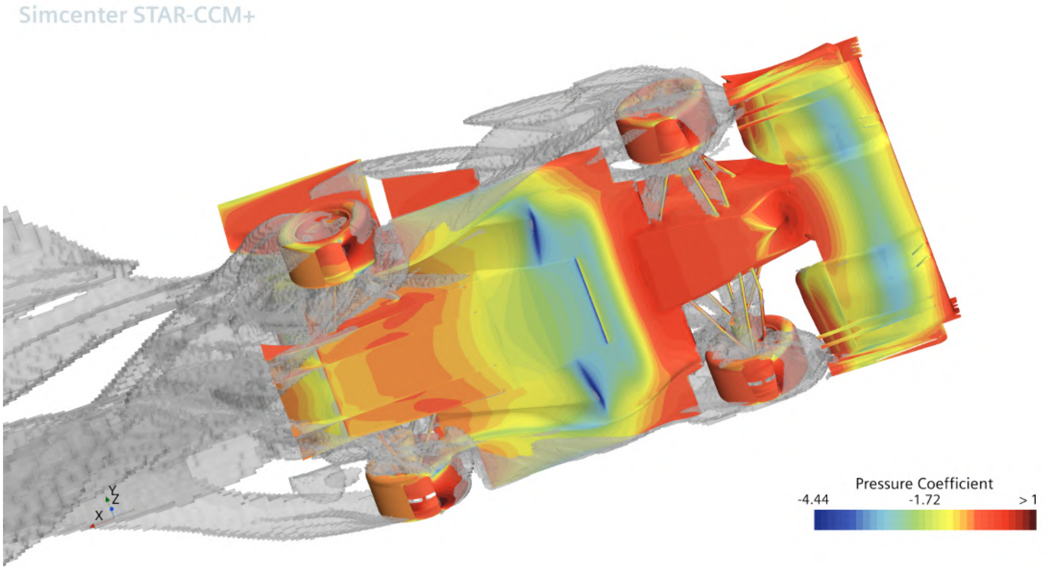


Figure 4.11: Pressure coefficient scene + TKE - underside view

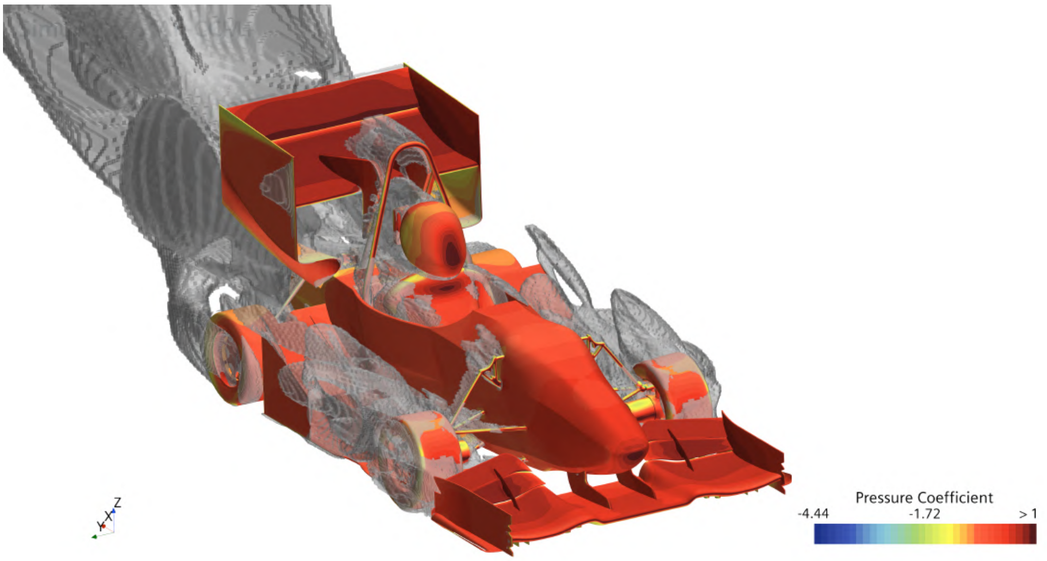


Figure 4.12: Pressure coefficient scene + TKE - upper side view

Also the wall shear stresses scene is significantly important to check. Here boundary layer detachments can be seen easily, and it gives a good understanding on how airflow behaves close to the walls. It can be seen how the paths under the car are quite smooth, with the exception of the monocoque - undertray interference disturbs that create some losses in the inner part. The upper - zone of the undertray keeps clean the area close to the monocoque, where battery cooling duct inlet lays.

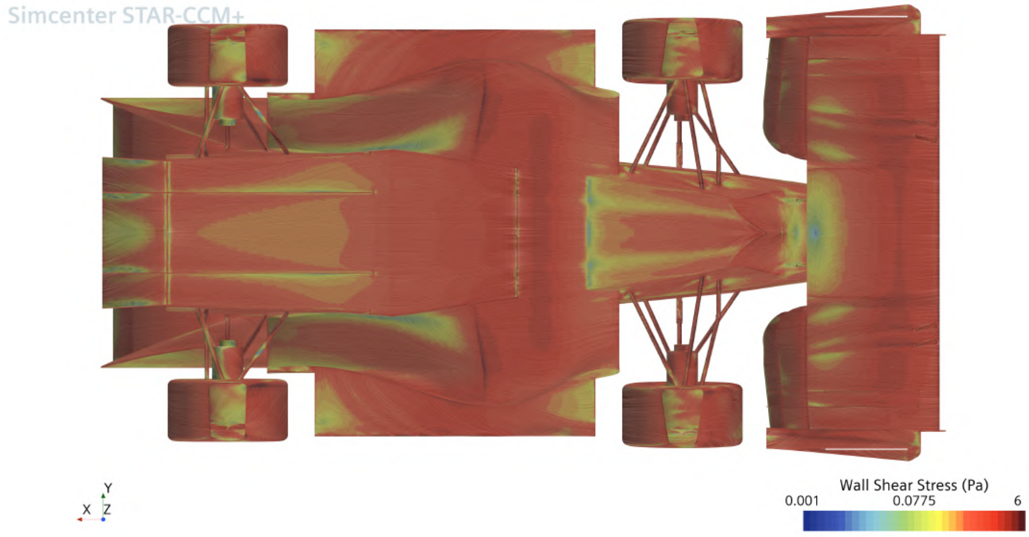


Figure 4.13: Wall shear stresses - underside view

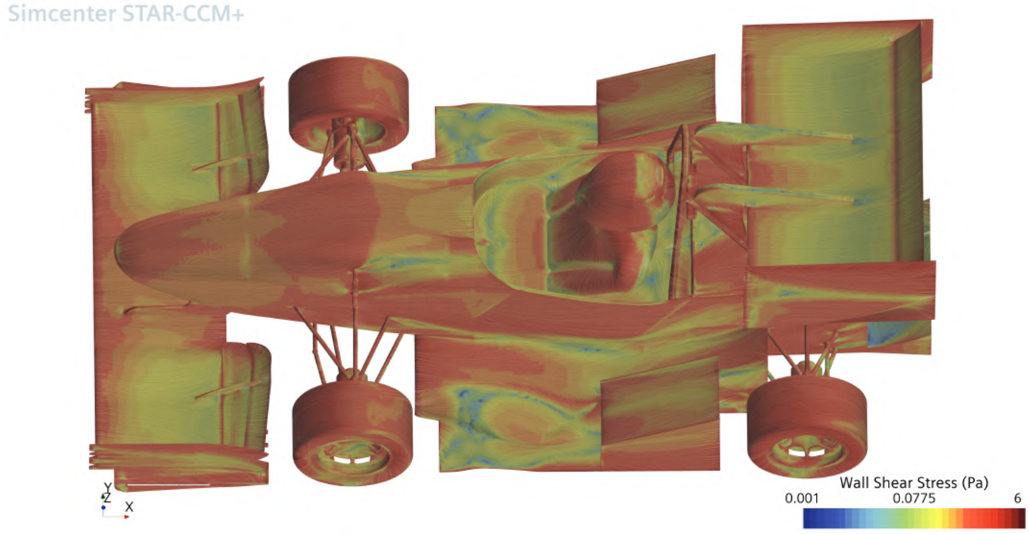


Figure 4.14: Wall shear stresses - upper side view

Velocity scene is instead used only at some interesting sections of the car, to check the presence of detachments or the airflow in the radiators. From the presented figures, it can be seen how undertray and rear wing works together, with a low upwash effect from the undertray that doesn't influence rear wing angle of attack. Radiator flow and tyres wakes can be checked, with the rear tyre wake significantly smaller thanks to the presence of the radiator duct in front of it.

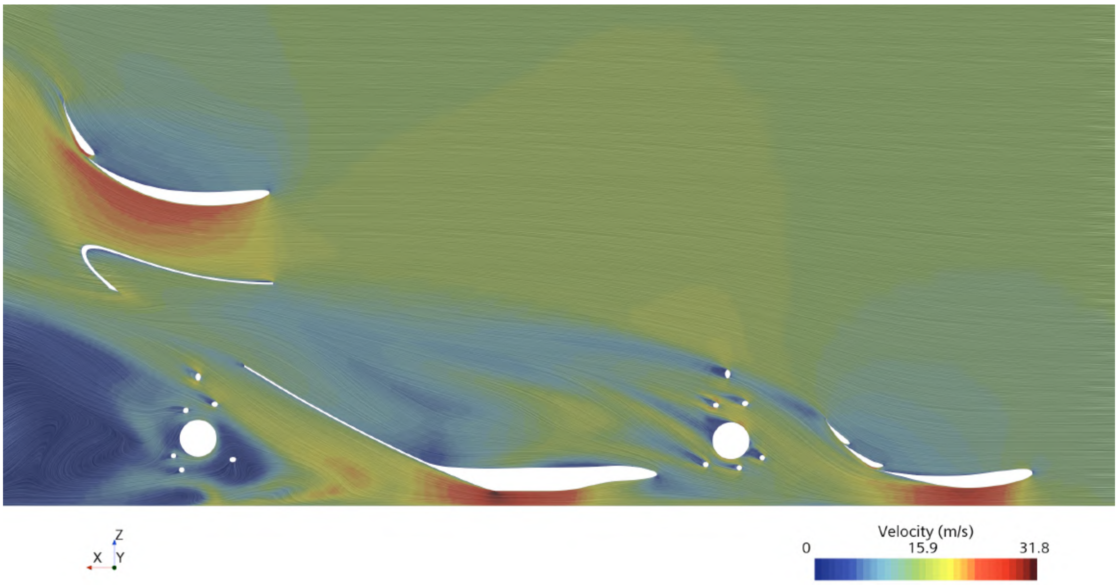


Figure 4.15: Velocity scene - $y = 0.3$ m section

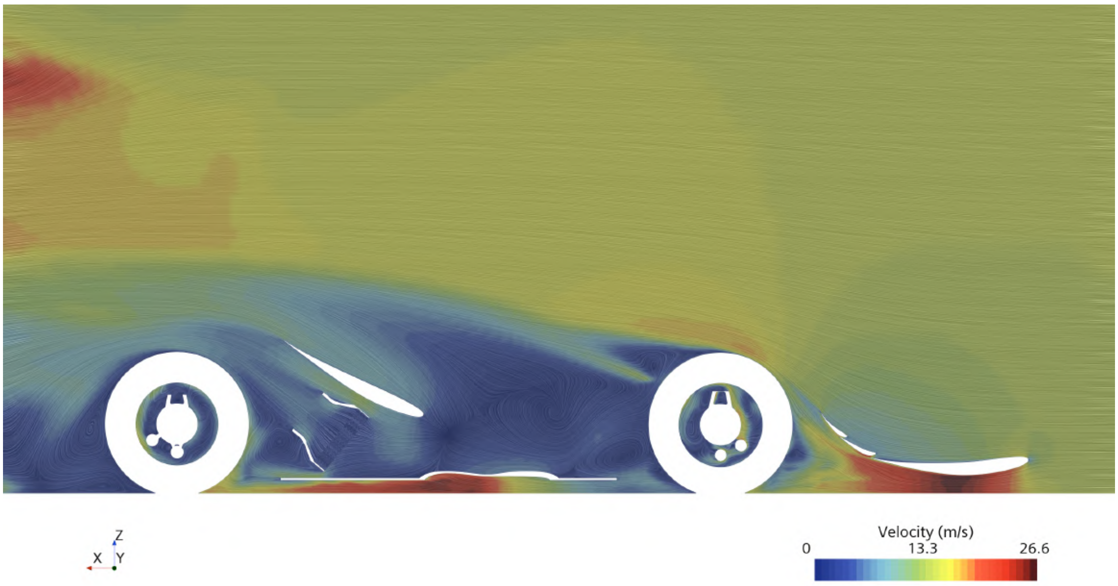


Figure 4.16: Velocity scene - $y = 0.4$ m section

Also vortex scene is commonly used to check vortices presence and their intensities. It can be immediately seen the big vortex created by the rear wing, that can be compared with the one at the outboard points of the front wing, that's significantly smaller. This is in accordance to the ground effect theories that have been discussed in the previous chapters. The undertray is sealed by two vortices: one that follows the external edge, born after the lateral convergent; the other one that follows the inner edge of the channel.

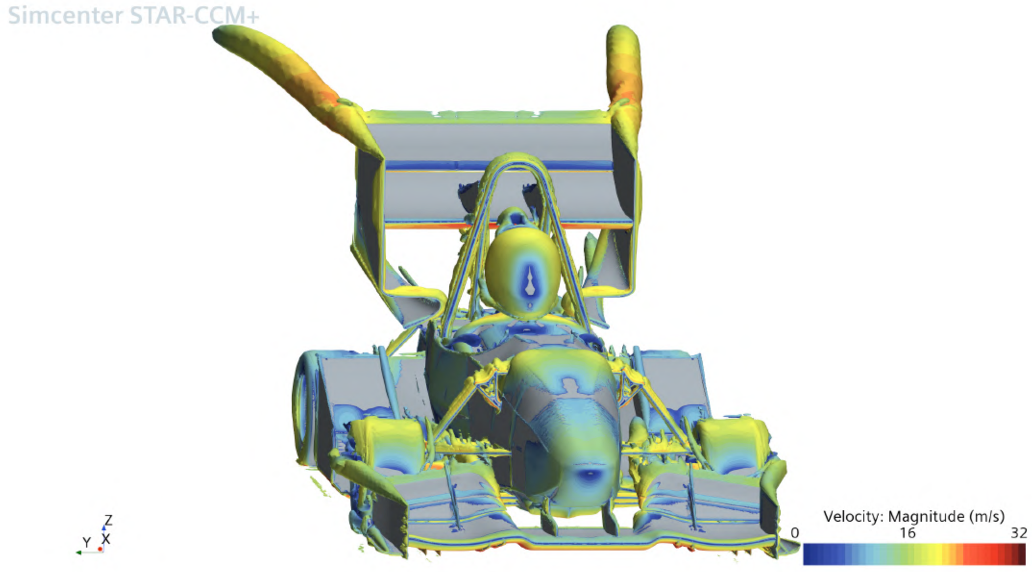


Figure 4.17: Vortex scene - upper side view

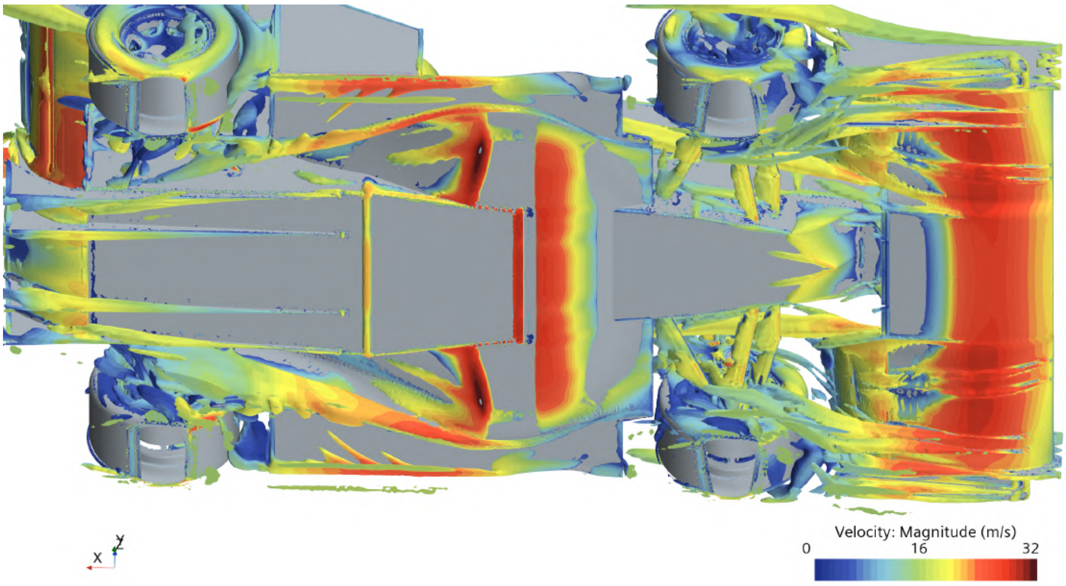


Figure 4.18: Vortex scene - underside view

Streamlines can be also used, mostly for checking the general behaviour of the 3D flow.

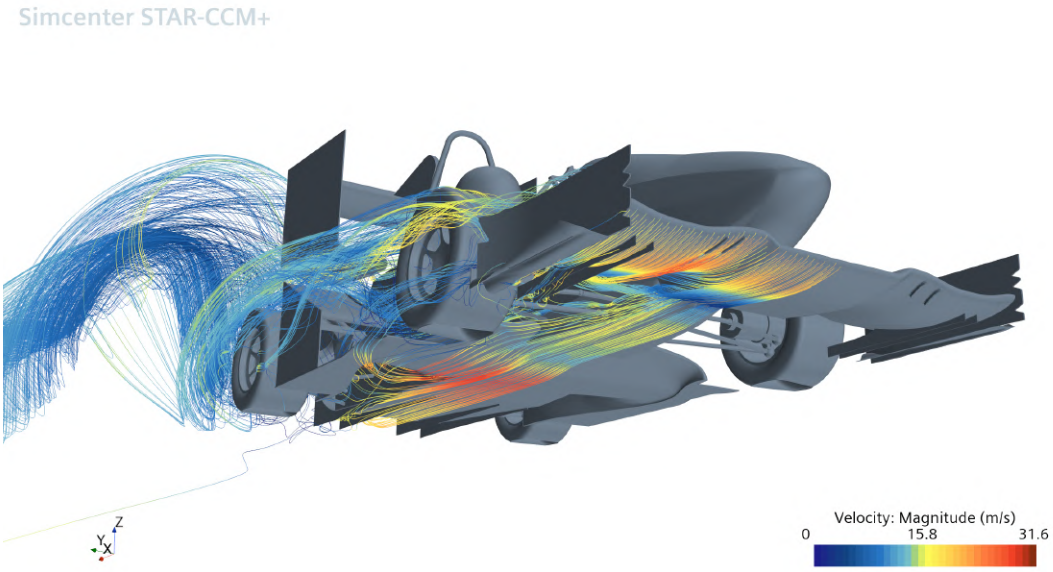


Figure 4.19: Streamlines scene

4.7 Results of the analysis and validation

The simulation converges and gives as an output the values of downforce, drag and y momentum of the entire car and its main components singularly.

Convergence is checked during the whole simulation through the residuals monitor and the forces plotting. The stopping criteria of the simulation is the downforce plot, when it becomes asymptotic (0.2 N of maximum variation in 100 iterations) the simulation stops. Residuals all reach a value of 10^{-4}, that could be even lower, but this early stop in the simulation saves a lot of calculation time. Also the y^+ value is monitored, ensuring that the expected maximum value is confirmed during the whole simulation. This can be done through the wall y^+ plot and the scene, that shows better the behaviour of y^+ around the car (higher shear stresses brings higher values of y^+).

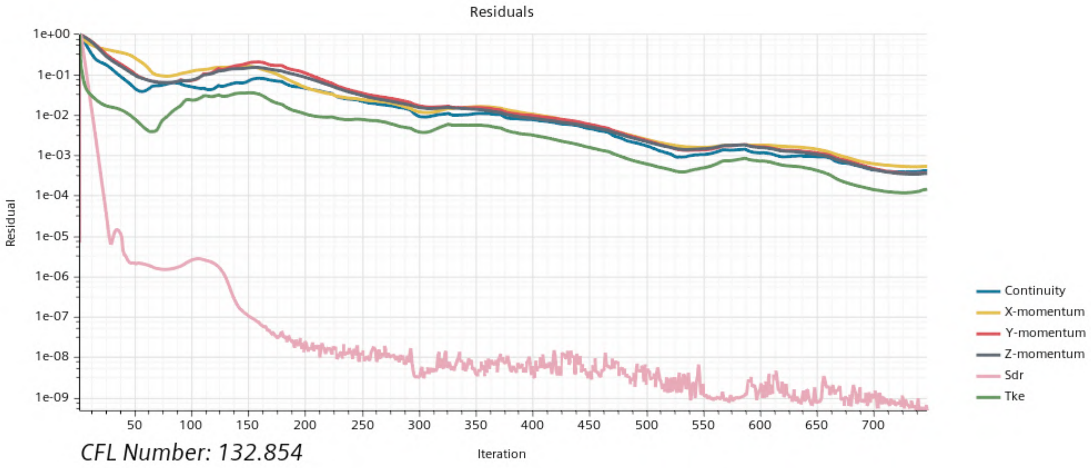


Figure 4.20: Residuals of the simulation

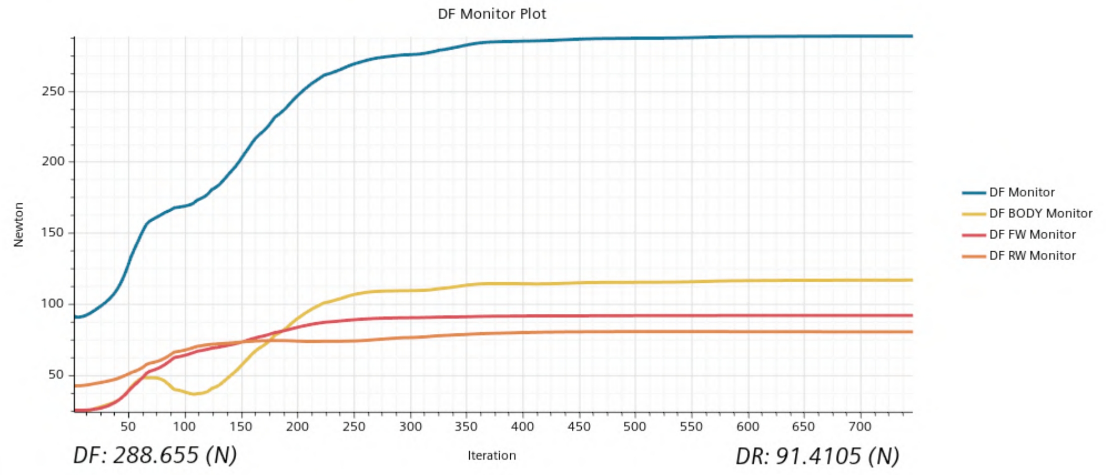


Figure 4.21: Downforce monitor plot

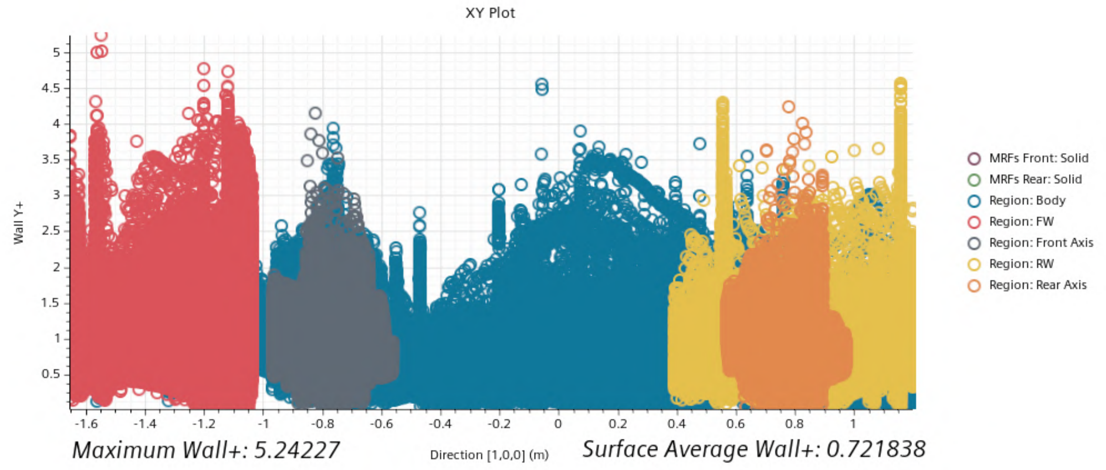


Figure 4.22: y^+ monitor plot

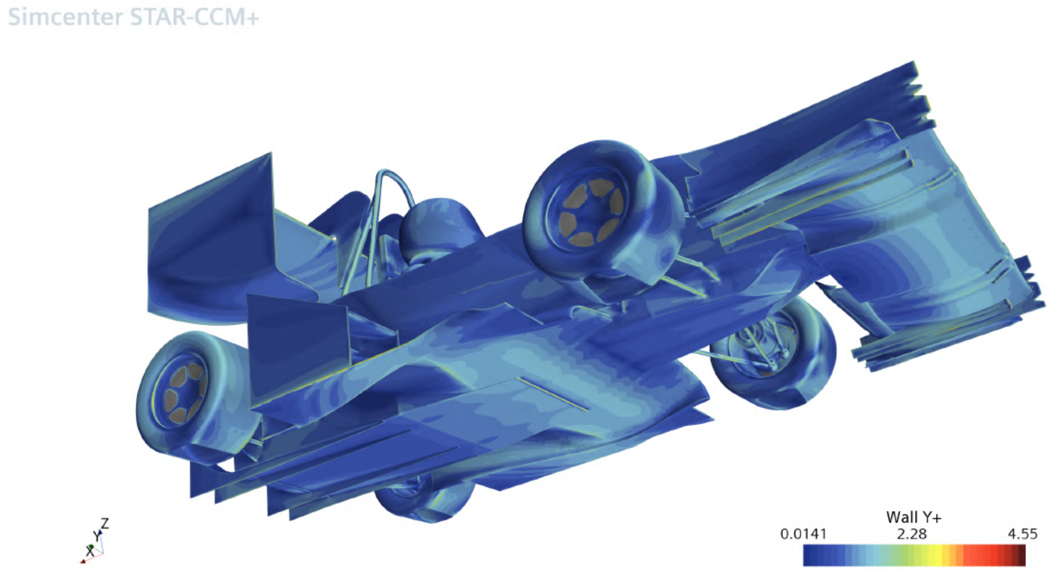


Figure 4.23: y^+ scene

As done before with the scenes, results for the simulation at standard height of 40 mm are presented and analysed:

Region	Downforce [N]	ClA [m^2]	Drag [N]	CdA [m^2]
Front wing	183.4	1.331	23.4	0.170
Body	233.4	1.694	97.4	0.707
Rear Wing	160.4	1.164	69.4	0.504
Full car	577.4	4.190	182.8	1.326

Table 4.5: Forces and moments results at 40 mm height

Undertray creates the major part of the downforce and, together with the front wing, they exploit the ground proximity. Rear wing isn't affected by ground effect, this can be seen from the high drag creation with respect to the front wing.

All the components works together generating 577.4 N of downforce at 15 m/s ($\rho = 1.225\ kg/m^3$), and a corresponding ClA (Cl per unit area) of 4.19 m^2. The moment generated by the forces of drag and lift with respect to the mean point of the wheelbase around the y axis is of $-0.26\ N \cdot m$, that correspond to a center of loads 40 mm before the origin, in the direction of the negative x. This means that the car has almost a 50 - 50 distribution of loads, slightly more loaded at the front axis. This distribution, together with the forces, is expected to change with a reduced height from the ground, with ground effect components that will be

highly affected by the height change.
This simulation setting will be the base for the next chapter results. The models, mesher and solvers have all been validated inside the Formula SAE team of the University of Padua, and on track loads acquisition has been argument of recent thesis. The most recent comparison between acquired data and simulations showed a mean error of 10% in forces estimation, that is an expected value considering also the possibility of measuring errors due to the presence of wind gusts or variations in the air density, together with car's manufacturing imperfections, that after some reverse engineering, thanks to the 3D scan of some components and the entire car, showed errors of similar nature and magnitude.

5

Heaving motion - Static case

5.1 Parametric CAD geometry

In this chapter the main objective is to repeat the results presented earlier for a set of different car's height from the ground. To do so a parametric CAD of the car in *Catia V5* is created, capable of changing car's suspensions setup by modifying some parameters. This is achieved by assembly constraints, that correlates the desired position of the car to the suspensions geometry. In the case of a change in heave (ride height), car translates along the z direction and suspension elements follow this movement keeping the same original contact points with both chassis and the hubs. With this CAD geometry it's easy to change the configuration of the car from straight line to cornering, including pitch and roll angles.
As already discussed, the standard minimum height of the standing still car in ready to race conditions is 40 mm. Other heights that will be studied are in the range of $[10, 60]$ mm, to have a good understanding of the phenomena.
Geometries are exported in STP extension and then imported in $StarCCM+$ to run the simulations as described.

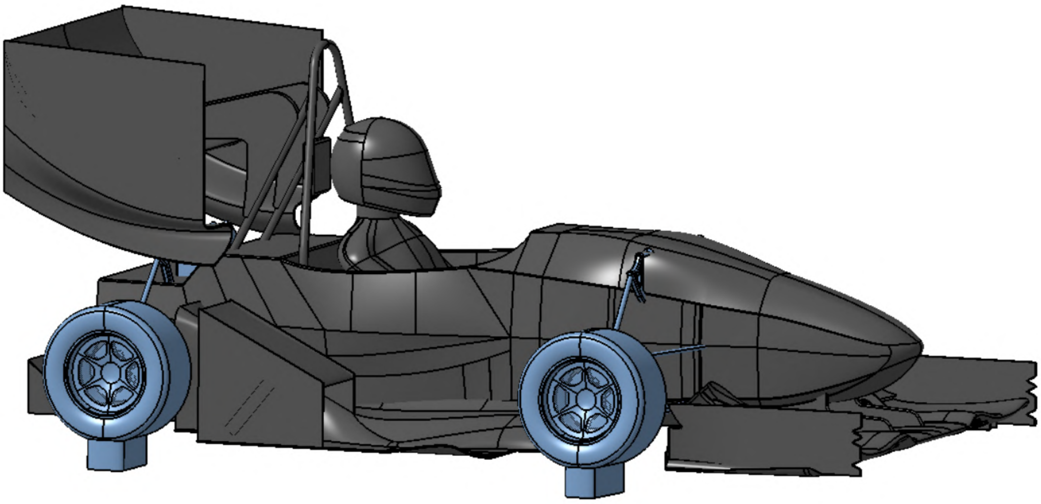


Figure 5.1: Parametric CAD at 10 mm height from the ground

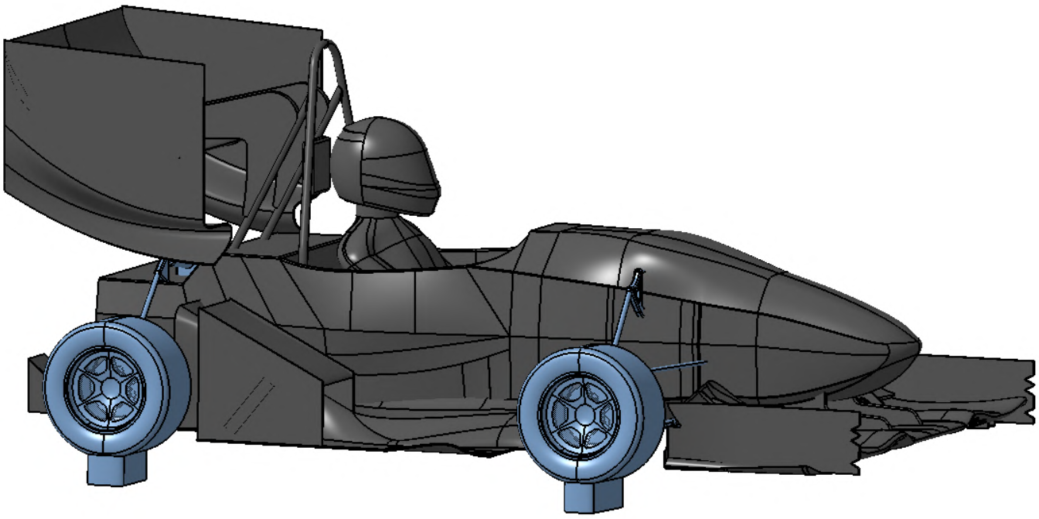


Figure 5.2: Parametric CAD at 60 mm height from the ground

5.2 Simulations results

Results obtained from the simulations are reported in the following tables. For the lift coefficient per unit area:

h [mm]	ClA [m^2]	ClA Body [m^2]	ClA FW [m^2]	ClA RW [m^2]
10	2.635	0.566	1.003	1.067
15	2.918	0.791	1.061	1.068
20	3.251	1.039	1.117	1.094
22.5	3.913	1.382	1.482	1.049
25	4.063	1.540	1.467	1.054
27.5	4.255	1.724	1.441	1.090
30	4.331	1.784	1.416	1.131
35	4.286	1.765	1.374	1.145
40	4.190	1.694	1.331	1.164
45	4.010	1.548	1.281	1.180
50	3.852	1.415	1.252	1.184
55	3.709	1.312	1.215	1.183
60	3.606	1.228	1.186	1.193

Table 5.1: ClA at different ride heights

For the drag coefficient per unit area:

h [mm]	CdA [m^2]	CdA Body [m^2]	CdA FW [m^2]	CdA RW [m^2]
10	1.132	0.553	0.163	0.464
15	1.190	0.596	0.164	0.479
20	1.210	0.610	0.167	0.486
22.5	1.218	0.624	0.168	0.477
25	1.228	0.630	0.170	0.480
27.5	1.257	0.649	0.170	0.491
30	1.286	0.668	0.171	0.499
35	1.309	0.694	0.171	0.498
40	1.326	0.707	0.170	0.504
45	1.337	0.717	0.167	0.506
50	1.337	0.715	0.165	0.506
55	1.329	0.707	0.164	0.508
60	1.342	0.721	0.164	0.511

Table 5.2: CdA at different ride heights

It's immediately visible how variations in force coefficients are important. This means that an estimation of the real ride height at different velocities is essential to comprehend the real loads acting on the car.
Data obtained are plotted into graphs to better understand the behaviour at different heights.

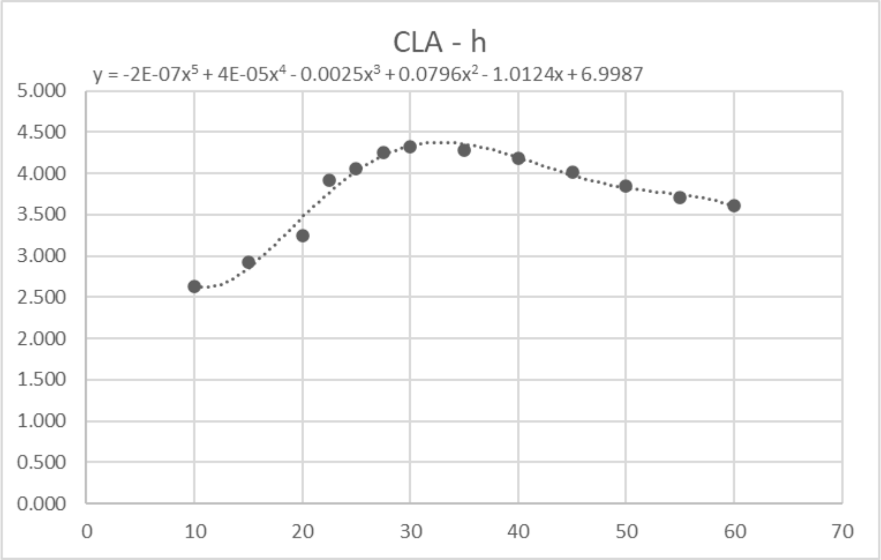


Figure 5.3: ClA - height

The expected behaviour is obtained. There's a peak in the total downforce around 30 mm height, with higher rides that show a quasi - linear decrement in downforce. For too low ride heights there's a fast drop in the ClA, created by the stall of the ground effect elements. Even if this is just a static analysis, it's important to verify that in operational heights car is far from stall conditions.

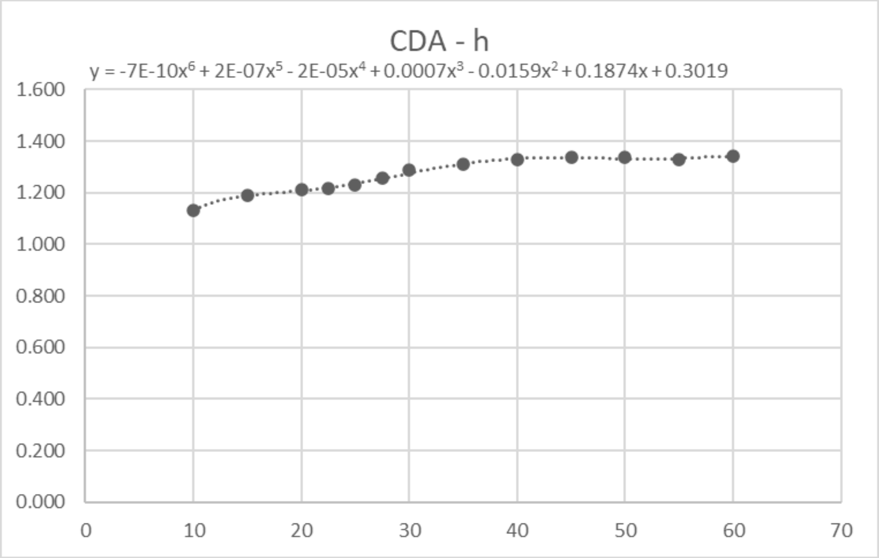


Figure 5.4: CdA - height

The Cda - h graph is really good in showing the benefits of ground effect. The gains in downforce are in fact followed by a continuous drag reduction, caused by the reduced intensity of tip vortices. The downforce loss at low rides is instead the reason of the final drop in the drag monitor.

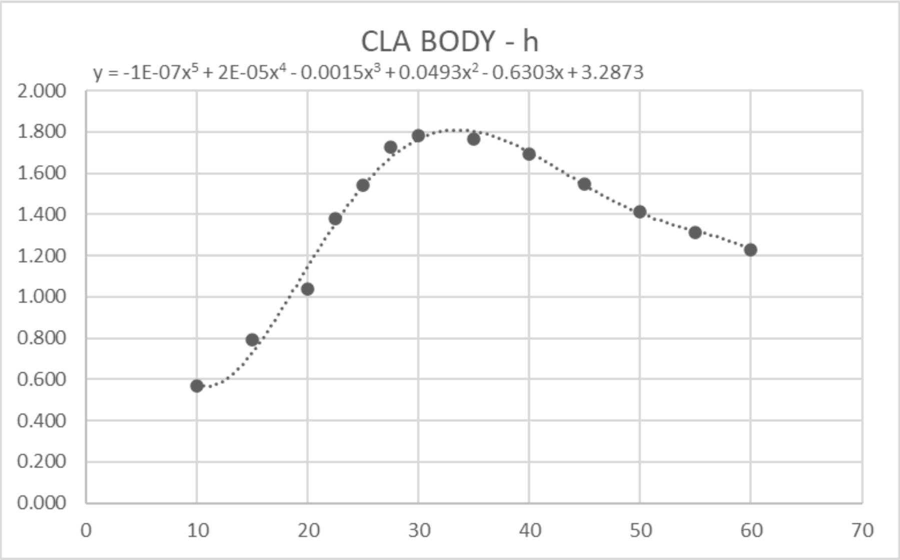


Figure 5.5: ClA Body - height

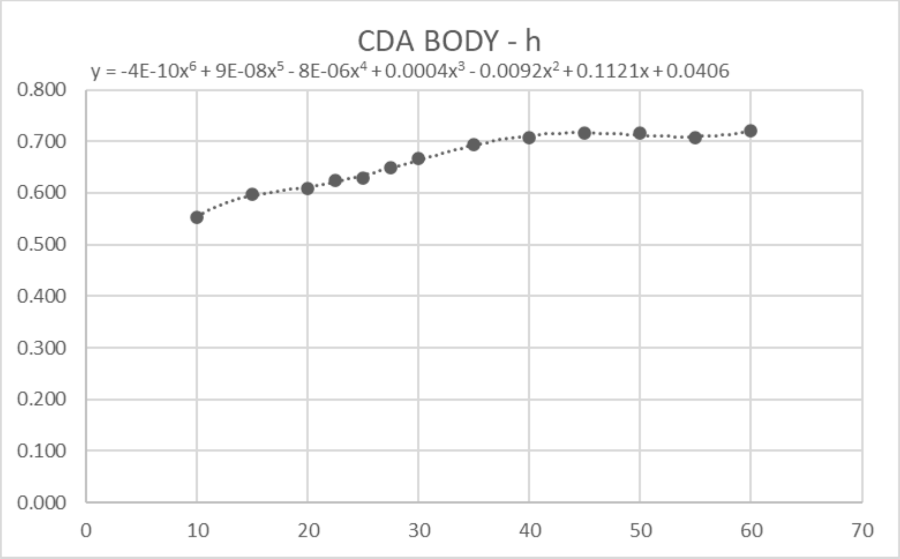


Figure 5.6: CdA Body - height

The body behaves similarly to the whole car. Its performance decreases rapidly with heights lower than 25 mm, with losses that arise from the inner part of the undertray and can be clearly seen from scenes.

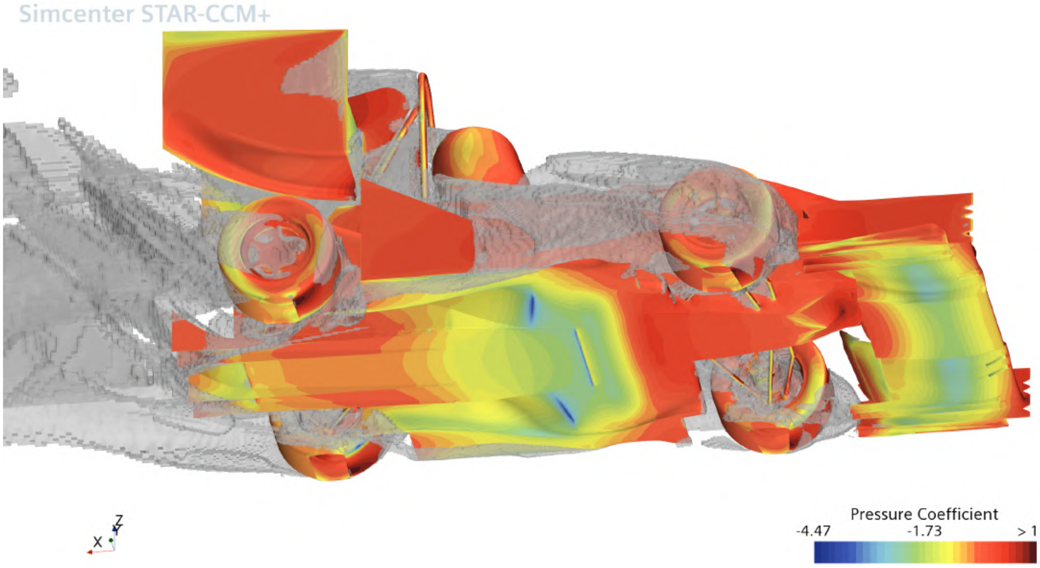


Figure 5.7: Pressure coefficient scene + TKE at 50 mm height from the ground

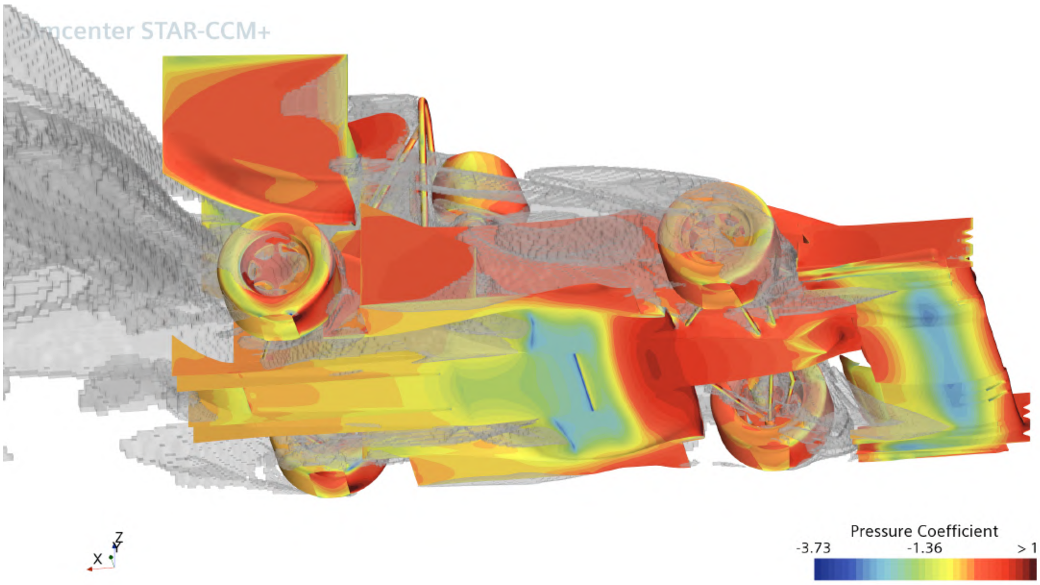


Figure 5.8: Pressure coefficient scene + TKE at 20 mm height from the ground

Differences between an high setup and a too low one can be clearly seen. Stall of the undertray is the main cause of the downforce loss.

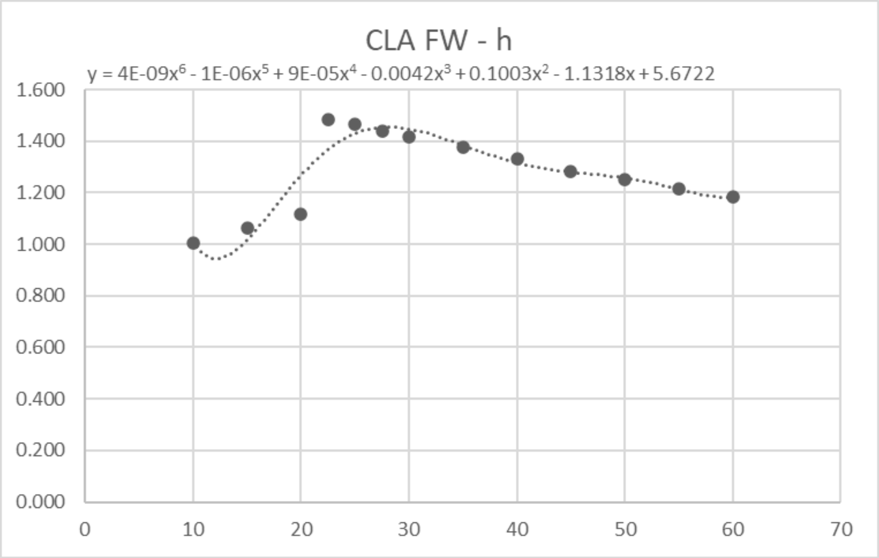


Figure 5.9: ClA Front Wing - height

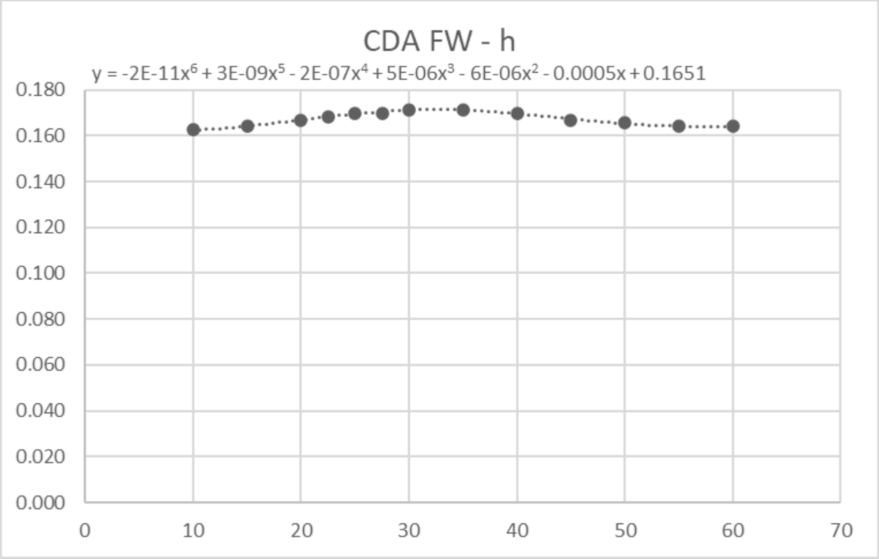


Figure 5.10: CdA Front Wing - height

Front wing increases its ClA linearly reducing height, with a maximum at 22.5 mm. A sudden stall creates a big loss in downforce. The three elements wing appears to be really sensitive at height changes, with a degradation in performance that's not gradual and predictable as the one experienced by the body.

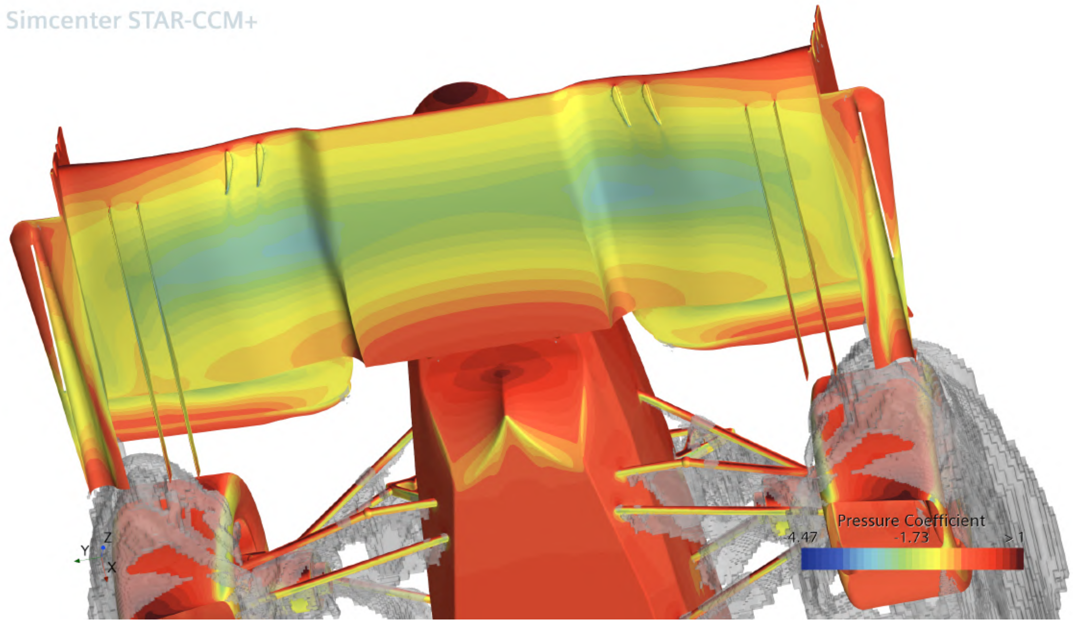


Figure 5.11: Pressure coefficient scene + TKE at 50 mm height from the ground

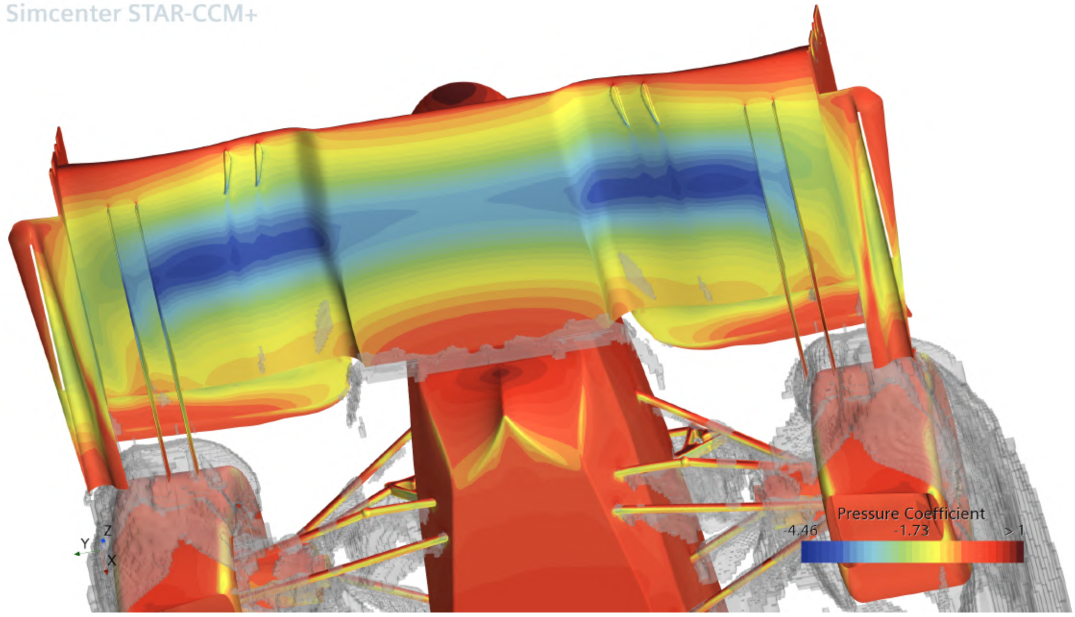


Figure 5.12: Pressure coefficient scene + TKE at 22.5 mm height from the ground

High differences of pressure distribution under the front wing are visible between an high case and the maximum ClA case.

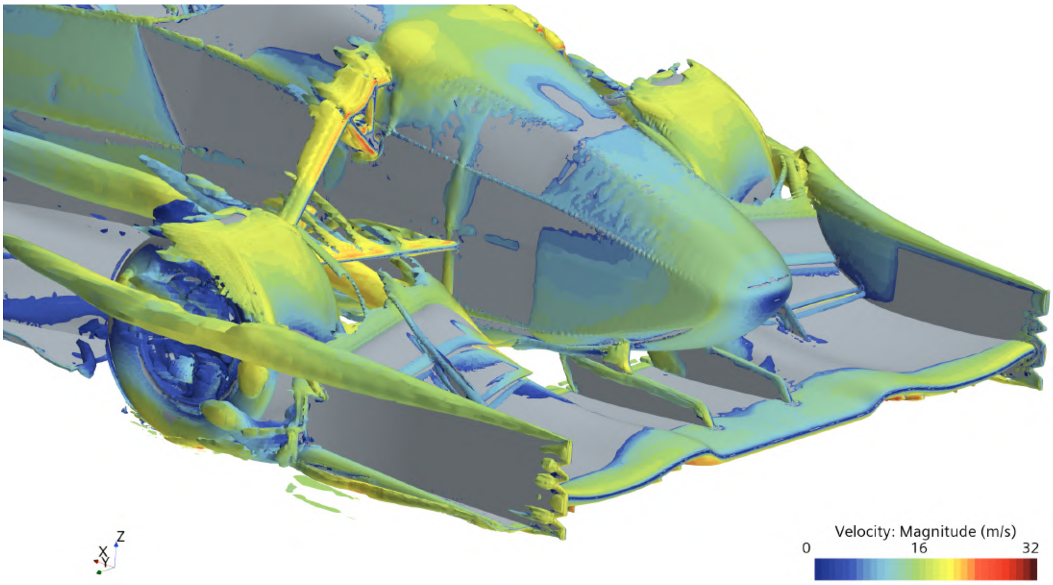


Figure 5.13: Vortex scene at 50 mm height from the ground

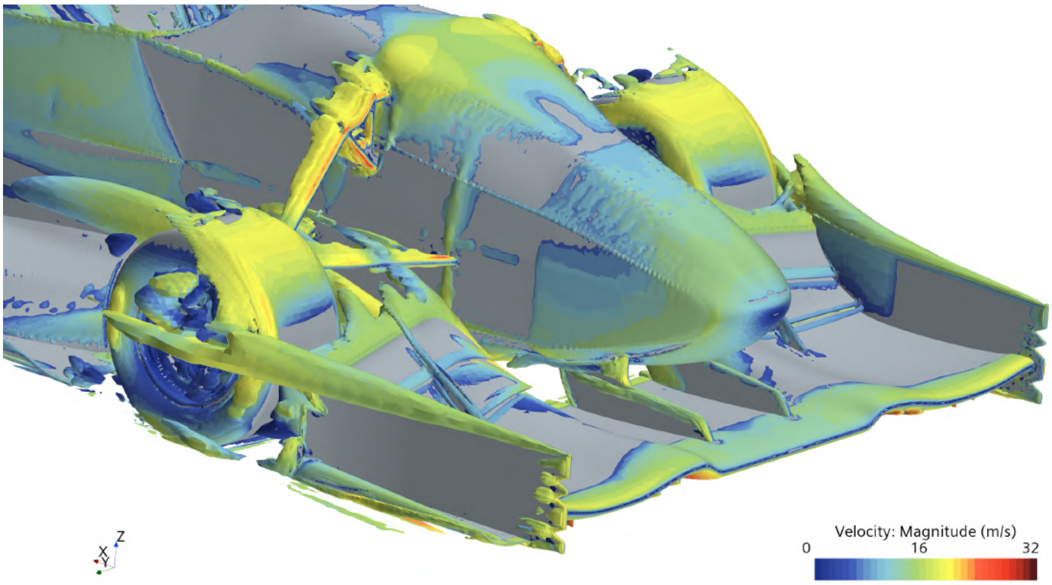


Figure 5.14: Vortex scene at 22.5 mm height from the ground

It's clearly visible how the tip vortex is reduced in a lower case, causing a drag reduction.

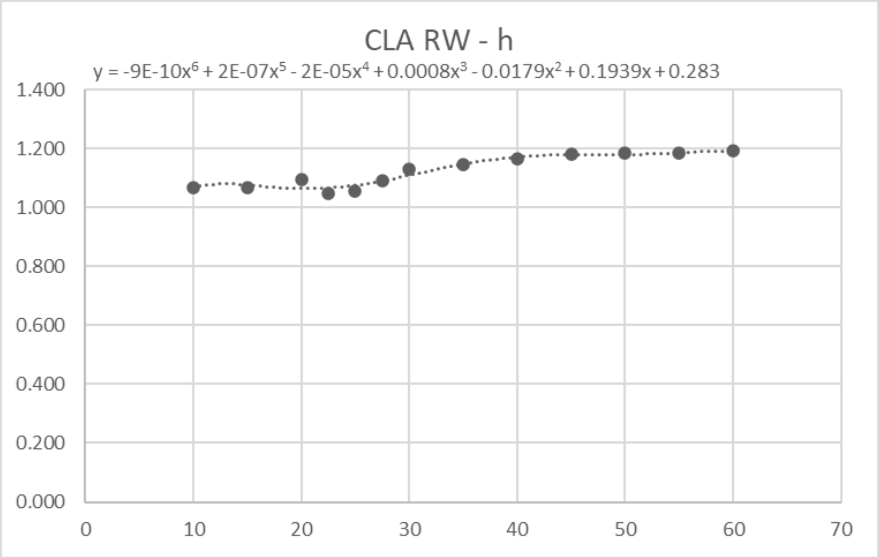


Figure 5.15: ClA Rear Wing - height

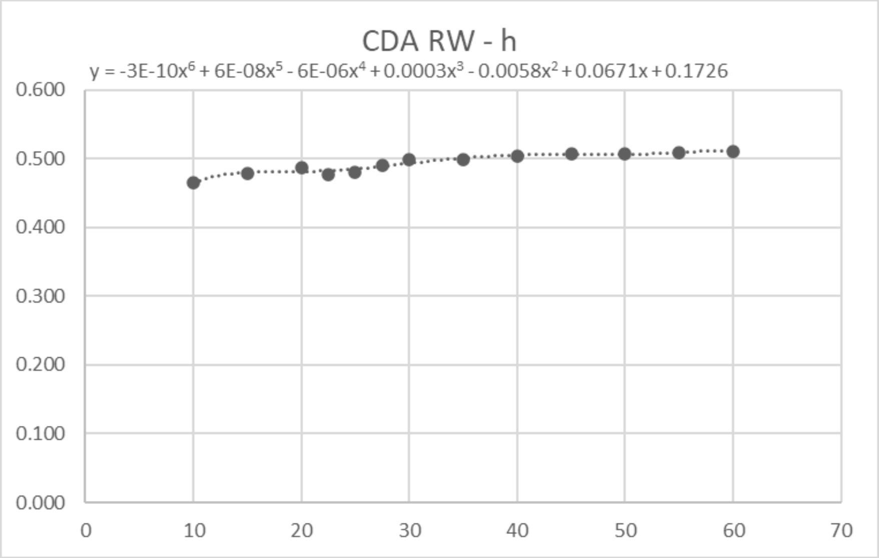


Figure 5.16: CdA Rear Wing - height

Rear wing is not strongly affected by ground effect, but a correlation to the rest of the car behaviour can be noted. When the front wing creates more downforce, in fact, rear wing loses a bit of performance due to the major upwash effect generated at the front, reducing rear wing's angle of attack. A similar pattern to the front wing ClA can be verified. For what concerns CdA, once again it's possible to see how much more drag this wing creates compared to the front one.

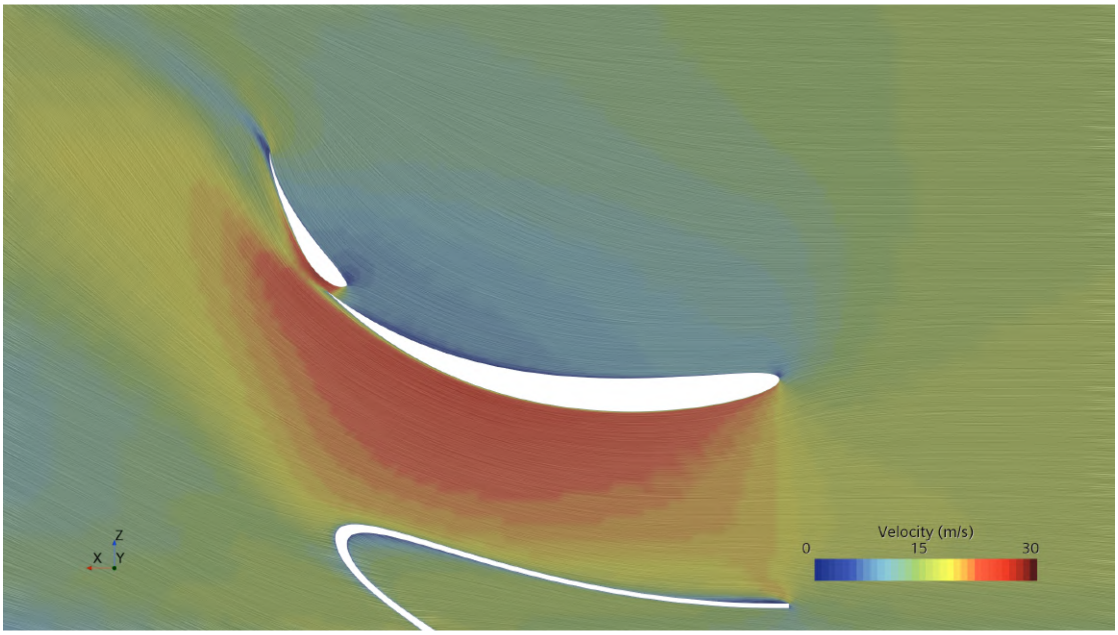


Figure 5.17: Velocity scene at 50 mm height from the ground

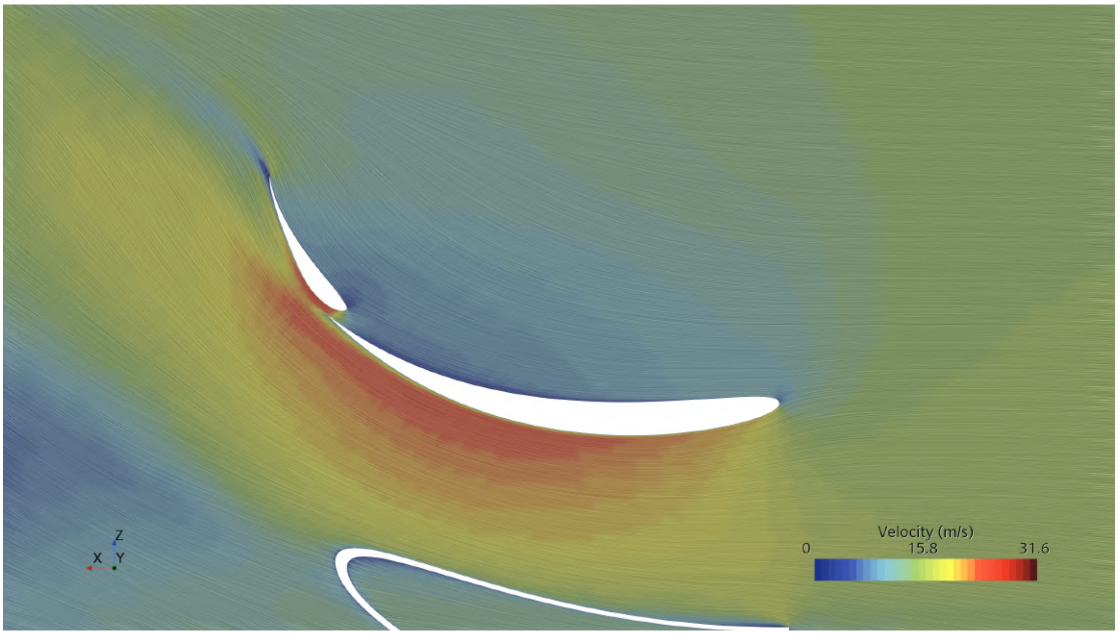


Figure 5.18: Velocity scene at 22.5 mm height from the ground

A slightly major upwash effect reducing rear wing angle of attack is visible when front wing generates more downforce.

5.3 Ride height prevision

Assuming that these coefficients are not function of the velocity, assumption perfectly fine with the range of velocities the car can reach, with a maximum velocity of 110 km/h, at different velocities, ride height will change due to the load on the car. As seen, this load is also function of the ride height, so an iterative approach is necessary to find the real final height.

Total ClA and CdA curves will be used for this cause, with the assumption that the phenomenon of change in ride height is sufficiently small and slow that the variation of loads acting on the car can be considered quasi - steady, so curves can be considered valid. The transient phase is neglected, with interest in only the final static equilibrium.

A *Matlab* script is done for this cause. It includes the suspension geometry, thanks to which a load applied to the car can be converted in a heaving motion using the iterative approach in order to find the equilibrium point.

The CFD analysis is done in straight line, thus in the code only the straight line speed will be put as input, without considering any acceleration, that would need a further study for force determination.

At the mean velocity of 15 m/s:

- heave = 2.46 mm;
- ClA = 4.29 m^2;
- CdA = 1.32 m^2.

The heave is minimum, but the increased ClA with respect to the standard height (ClA = 4.19), makes this heaving motion preciser.

At the higher speed reachable by the car, 30 m/s:

- heave = 9.69 mm;
- ClA = 4.30 m^2;
- CdA = 1.28 m^2.

Heave and ClA increase, CdA decreases as height from the ground reduces.

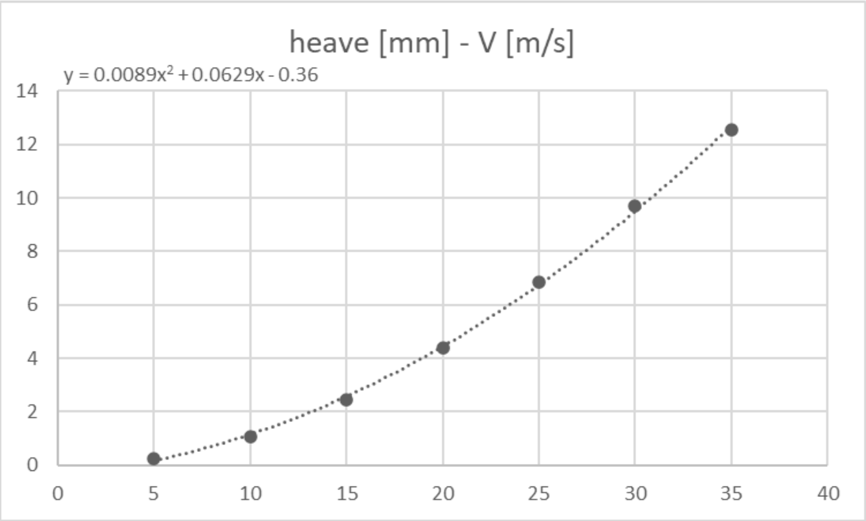


Figure 5.19: Heave predicted in function of car's velocity

Similarly to forces, heave increase is almost quadratic with velocity, with variations due to the non constant ClA considered.
It can be seen how for a velocity up to 30 m/s the z displacement of the car is maximum at around 10 mm. For this range of heigh, [30, 40] mm, all the components of the car can exploit their best performance, with stalls that happens at lower heights.

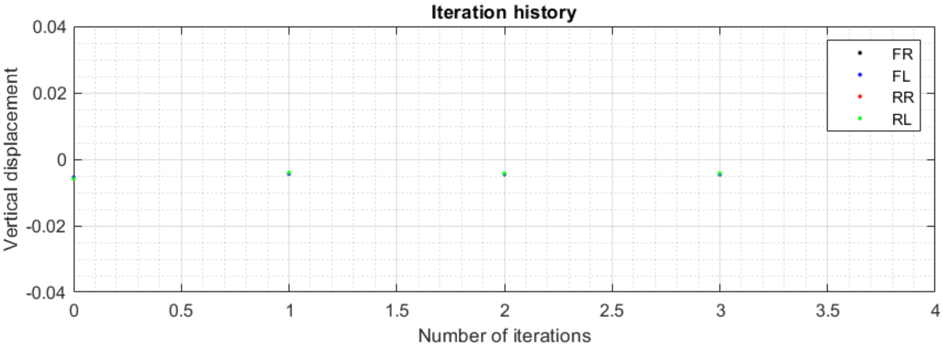


Figure 5.20: Rear and front heaves iterations plot for a velocity of 20 m/s

From graphs it's possible to see how front suspensions gets a slightly major heave compared to the rear ones. This is due to the different springs resistance coefficients (k) caused by different motion ratios between front and rear suspensions. Other factors such as Center Of Mass (COM) and COP can modify front and rear suspensions heave, but don't really affect the mean car's heave. For example, a center of pressure slightly forward the wheelbase center (55% front - 45% rear), that could be the case at higher speeds, with the car closer to the ground and the front wing moving the COP towards the front thanks to its increased ClA, values obtained at 30 m/s are:

- heave = 9.73 mm;
- ClA = 4.30 m^2;
- CdA = 1.28 m^2;
- pitch = 0.1 deg.

Even with an increased displacement at the front axle, the total pitch angle of the car can be neglected, and the total heave can be considered unchanged.

5.3.1 The code

The code from which results are taken is shown in this section, where some features are explained.

At first essential data are presented, ClA and CdA included, that are vectors fitted by a polynomial 6^{th} degrees function. Motion ratios interpolation functions are also defined to change the motion ratio values of front and rear suspensions when they compress [7]. Initial conditions are set:

```
%% Data

w = 1.535; % wheelbase
tf = 1.23; % track front
tr = 1.2; % track rear
m_s = 270-4*9; % sprung mass
z_g = 0.30; % CoM height
a = 0.55*w; % CoM front axle
b = 0.45*w; % CoM rear axle
z_a = 0; % CoP height
a_a = 0.45*w; % Cop front axle
b_a = 0.55*w; % Cop rear axle
m = 270; % total mass
rho = 1.225; % air density
H = [-0.030 -0.025 -0.020 -0.015 -0.010 -0.005 0 0.005 0.010 0.015
    0.020];
CLA = -4.190; % ClA at standard height
CLAH=-[2.635 2.918 3.251 4.063 4.331 4.286 4.190 4.010 3.852 3.709
    3.606];
pCLA=polyfit(H,CLAH,6);
```

```
CDA = 1.326; % CdA at standard height
CDAH = [1.132 1.190 1.210 1.228 1.286 1.309 1.326 1.337 1.337 1.329
    1.342];
pCDA = polyfit(H,CDAH,6);
k_f0 = 61294; % front spring k
k_r0 = 61294; % rear spring k
k_t = 88000; % Hoosier tires k
k_arb_f = 372255; % k anti roll bar front
k_arb_r = 425914; % k arb rear
g = 9.81;
pr =    1.0e+04 * [-1.552620638539592   0.170806966768406
    -0.016940792238554   0.001396349979041   0.000120742987204
    0.000114063370171];
% 5th degree interpolation motion ratio rear
pf = 1.0e+05*[-8.079809067062426  -0.256226574060929
    -0.000446723507707  -0.000040120084565   0.000028045499374
    0.000011860530909];
% 5th degree interpolation motion ratio front
paf = [-14021.7414291225  -1032.33348807123 -140.699590315231
    2.60685298092043  1.70680464991742];
% 4th degree interpolation mr arb front
par = [-6009.69325415841  -605.939783087960 -79.8948555541866
    11.4729158462236  1.63381160238139];
% 4th degree interpolation mr arb rear

%% Initial values

% X
xfr0 = 0;
xfl0 = 0;
xrr0 = 0;
xrl0 = 0;
% Motion Ratio
mrfr = 1.14;
mrfl = 1.14;
mrrr = 1.187;
mrrl = 1.187;
% Motion Ratio Arb
mr_arb_fr = 1.708;
mr_arb_fl = 1.708;
mr_arb_rr = 1.633;
mr_arb_rl = 1.633;
% k springs
```

```
k_fr = k_f0/(mrfr^2);
k_fl = k_f0/(mrfl^2);
k_rr = k_r0/(mrrr^2);
k_rl = k_r0/(mrrl^2);
% dz
dz_g = 0;
dz_a = 0;
% dF
dF_f = 0;
dF_r = 0;
% F arb
F_arb_f = 0;
F_arb_r = 0;
% k ride
k_rfr = k_fr;
k_rfl = k_fl;
k_rrr = k_rr;
k_rrl = k_rl;
% heave and roll
hf = 0;
hr = 0;
r_f = 0;
r_r = 0;
```

Code 5.1: Data and initial conditions for heave roll and pitch calculation

Calculation can then begin. At first it execute an evaluation of the different heights from the ground for every tire (the two fronts and the two rears obtain identical results for straight line analysis), it renew the values of motion ratios, the various stiffness and accelerations and then execute the first real iteration. The values of ClA and CdA are renewed for the new height reached and another iteration starts. Every iteration's displacement results are compared to the previous ones. Convergence is reached when this error is lower than a value close to 0.0001 (scaled every iteration to help convergence).

```
%% Calculation

ef0 = 0.0001; % max error
ef = ef0; % max error first iteration
e = 1; % initial error
i = 0;
xm = 0.035; % max displacement
s = 0; % damping
```

```
n = 1000; % max iterations

while e>ef && i<n
    if i<2
        xfr = -(0.5*(- 0.5*rho*v^2*CLA*b_a/w - m_s*a_x/w*(z_g + dz_g) - 0.5*rho*v^2*CDA*(z_a + dz_a)) + dF_f)/k_rfr;
        xfl = -(0.5*(- 0.5*rho*v^2*CLA*b_a/w - m_s*a_x/w*(z_g + dz_g) - 0.5*rho*v^2*CDA*(z_a + dz_a)) - dF_f)/k_rfl;
        xrr = -(0.5*(- 0.5*rho*v^2*CLA*a_a/w + m_s*a_x/w*(z_g + dz_g) + 0.5*rho*v^2*CDA*(z_a + dz_a)) + dF_r)/k_rrr;
        xrl = -(0.5*(- 0.5*rho*v^2*CLA*a_a/w + m_s*a_x/w*(z_g + dz_g) + 0.5*rho*v^2*CDA*(z_a + dz_a)) - dF_r)/k_rrl;

        xfr=xfr-s*(xfr-xfr0); % similar to a damping
        xfl=xfl-s*(xfl-xfl0);
        xrr=xrr-s*(xrr-xrr0);
        xrl=xrl-s*(xrl-xrl0);
    end

    % roll front e rear
    r_f = 180/pi*atan((xfl-xfr)/tf); % roll front in degrees
    r_r = 180/pi*atan((xrl-xrr)/tr); % roll rear in degrees

    hf = (xfr+xfl)/2;
    hr = (xrr+xrl)/2;

    % limits
    if abs(xfr)>xm
        xfr=xm*sign(xfr);
    end
    if abs(xfl)>xm
        xfl=xm*sign(xfl);
    end
    if abs(xrr)>xm
        xrr=xm*sign(xrr);
    end
    if abs(xrl)>xm
        xrl=xm*sign(xrl);
    end
    % dz
    dz_g = ((xfr+xfl)*b/w+(xrr+xrl)*a/w)/2;
    dz_a = ((xfr+xfl)*b_a/w+(xrr+xrl)*a_a/w)/2;
    % Motion Ratio Front
```

```
 mrfr = pf(1)*xfr.^5 + pf(2)*xfr.^4 + pf(3)*xfr.^3 + pf(4)*xfr.^2
+ pf(5)*xfr + pf(6);
 mrfl = pf(1)*xfl.^5 + pf(2)*xfl.^4 + pf(3)*xfl.^3 + pf(4)*xfl.^2
+ pf(5)*xfl + pf(6);
 % Motion Ratio Rear
 mrrr = pr(1)*xrr.^5 + pr(2)*xrr.^4 + pr(3)*xrr.^3 + pr(4)*xrr.^2
+ pr(5)*xrr + pr(6);
 mrrl = pr(1)*xrl.^5 + pr(2)*xrl.^4 + pr(3)*xrl.^3 + pr(4)*xrl.^2
+ pr(5)*xrl + pr(6);
 % Motion Ratio Arb
 mr_arb_fr = paf(1)*hf.^4 + paf(2)*hf.^3 + paf(3)*hf.^2 + paf(4)*
hf.^1 + paf(5) + 0.025*r_f;
 mr_arb_fl = paf(1)*hf.^4 + paf(2)*hf.^3 + paf(3)*hf.^2 + paf(4)*
hf.^1 + paf(5) - 0.025*r_f;
 mr_arb_rr = par(1)*hr.^4 + par(2)*hr.^3 + par(3)*hr.^2 + par(4)*
hr.^1 + par(5) + 0.1*r_r;
 mr_arb_rl = par(1)*hr.^4 + par(2)*hr.^3 + par(3)*hr.^2 + par(4)*
hr.^1 + par(5) - 0.1*r_r;
 % k springs
 k_fr = k_f0/(mrfr.^2);
 k_fl = k_f0/(mrfl.^2);
 k_rr = k_r0/(mrrr.^2);
 k_rl = k_r0/(mrrl.^2);
 % k arb
 ka_fr = k_arb_f/(mr_arb_fr^2);
 ka_fl = k_arb_f/(mr_arb_fl^2);
 ka_rr = k_arb_r/(mr_arb_rr^2);
 ka_rl = k_arb_r/(mr_arb_rl^2);
 % k tot (equivalent spring)
 Kfr = k_fr+ka_fr;
 Kfl = k_fl+ka_fl;
 Krr = k_rr+ka_rr;
 Krl = k_rl+ka_rl;
 % k ride
 k_rfr = Kfr*k_t/(Kfr + k_t);
 k_rfl = Kfl*k_t/(Kfl + k_t);
 k_rrr = Krr*k_t/(Krr + k_t);
 k_rrl = Krl*k_t/(Krl + k_t);
 % k ride axles
 k_r_f = k_rfr+k_rfl;
 k_r_r = k_rrr+k_rrl;
 k_torx_f = k_r_f*tf^2/2; % torsional stiffness front axle
 k_torx_r = k_r_r*tr^2/2; % torsional stiffness rear axle
```

```
    k_torx = k_torx_f+k_torx_r;
    % roll centers height
    df_x = 1.4167*(xfr+xfl)/2+0.032;
    dr_x = 1.8*(xrr+xrl)/2+0.048;
    df_r = -0.0003172*r_f^2;
    dr_r = -0.0008894*r_r^2;
    df = df_x+df_r;
    dr = dr_x+dr_r;
    d = df*b/w+dr*a/w; % roll center
    dF_f = k_torx_f/k_torx*m_s*a_y*(z_g-d)/tf+m*a_y*b/w*df/tf; % Fz front
    dF_r = k_torx_r/k_torx*m_s*a_y*(z_g-d)/tr+m*a_y*a/w*dr/tr; % Fz rear

    if i>1
        xfr = -(0.5*(- 0.5*rho*v^2*CLA*b_a/w - m_s*a_x/w*(z_g + dz_g) - 0.5*rho*v^2*CDA*(z_a + dz_a)) + dF_f)/k_rfr;
        xfl = -(0.5*(- 0.5*rho*v^2*CLA*b_a/w - m_s*a_x/w*(z_g + dz_g) - 0.5*rho*v^2*CDA*(z_a + dz_a)) - dF_f)/k_rfl;
        xrr = -(0.5*(- 0.5*rho*v^2*CLA*a_a/w + m_s*a_x/w*(z_g + dz_g) + 0.5*rho*v^2*CDA*(z_a + dz_a)) + dF_r)/k_rrr;
        xrl = -(0.5*(- 0.5*rho*v^2*CLA*a_a/w + m_s*a_x/w*(z_g + dz_g) + 0.5*rho*v^2*CDA*(z_a + dz_a)) - dF_r)/k_rrl;

        xfr=xfr-s*(xfr-xfr0);
        xfl=xfl-s*(xfl-xfl0);
        xrr=xrr-s*(xrr-xrr0);
        xrl=xrl-s*(xrl-xrl0);
    end

    if i==0

        plot (i,xfr,'.k')
        hold on
        plot (i,xfl,'.b')
        plot (i,xrr,'.r')
        plot (i,xrl,'.g')
        title ('Iteration history')
        xlabel ('Number of iterations')
        ylabel ('Vertical displacement')

        xfr = -(0.5*(- 0.5*rho*v^2*CLA*b_a/w - m_s*a_x/w*(z_g + dz_g) - 0.5*rho*v^2*CDA*(z_a + dz_a)) + dF_f - F_arb_f)/k_rfr;
        xfl = -(0.5*(- 0.5*rho*v^2*CLA*b_a/w - m_s*a_x/w*(z_g + dz_g)
    - 0.5*rho*v^2*CDA*(z_a + dz_a)) - dF_f + F_arb_f)/k_rfl;
        xrr = -(0.5*(- 0.5*rho*v^2*CLA*a_a/w + m_s*a_x/w*(z_g + dz_g)
    + 0.5*rho*v^2*CDA*(z_a + dz_a)) + dF_r - F_arb_r)/k_rrr;
        xrl = -(0.5*(- 0.5*rho*v^2*CLA*a_a/w + m_s*a_x/w*(z_g + dz_g)
    + 0.5*rho*v^2*CDA*(z_a + dz_a)) - dF_r + F_arb_r)/k_rrl;
        i = 1;
    end

    h = (hf+hr)/2
    CLA = polyval(pCLA,h)
    CDA = polyval(pCDA,h)

    F_arb_f = -(xfr/mr_arb_fr^2-xfl/mr_arb_fl^2)*k_arb_f/2;
    F_arb_r = -(xrr/mr_arb_rr^2-xrl/mr_arb_rl^2)*k_arb_r/2;

    % error
    e = max([abs(xfr-xfr0) abs(xfl-xfl0) abs(xrr-xrr0) abs(xrl-xrl0)
   ]);
    % convergence help
    s = (1-exp(-i/n)); % change s to retry iteration
    ef = ef0*(1-s)^(1/s); % scaled error at i

    i = i+1;
    xfr0 = xfr;
    xfl0 = xfl;
    xrr0 = xrr;
    xrl0 = xrl;
end
```

Code 5.2: Heave roll and pitch calculation

Displacements results are plotted every iteration.

5.4 Results analysis and further studies

Ground effect has seen to be relevant in force coefficients variations with height and with the consideration of the real forces acting at different heights, it was possible to determine a more precise height of the car from the ground.
In the range of heights in which car operates, all the aerodynamic components works without reaching stall conditions, so the design of all the components won't need any modification to adapt at lower heights from the ground.
If smaller ground clearance were reached, the loss in performance could lead to aerodynamic instabilities, changing the car's behaviour, that's not desirable. A solution for the undertray performance loss could be the use of vortex generators similar to the ones already present at the LE of the front wing. These elements could help energizing the boundary layer in the inner part of the undertray, where detachment starts, creating a small increase in drag, but bringing a strong benefit in the overall performance at lower heights, were undertray starts its ClA decay.
All the considerations done in the present chapter are for a steady or quasi - steady case, in which aerodynamic components operate at every instant as they operate in steady conditions and transient phases are neglected. This assumption is really strong, but justified by the slightly changes in ClA and CdA in the small range of displacement of the car.
A dynamic analysis of the entire car heaving would be interesting to see how really coefficient varies with time, but extremely expensive in computational terms.
If we consider the case of a wing creating downforce in freestream, only its dynamic movement would create changes in forces, with clearly no effect in static case [3],[11]. The downward motion reduces the wing's angle of attack, with an upwash effect that sums its velocity component with the one of the incoming flow. The upward motion would cause instead an increased angle of attack. This behaviour is classical and for an oscillating airfoil (in heave and/or pitch) can be described with the Theodorsen function $C(k) = F(k) + iG(k)$, that model the changes in amplitude and phase of the sinusoidal unsteady aerodynamic forces relative to the quasi - steady forces for different reduced frequencies k:

$$k = \frac{\omega c}{2V} = \pi \frac{T_{aer}}{T_m} \tag{5.1}$$

with $c/2 = b$, $T_{aer} = c/V$ characteristic time of interaction between a fluid particle and the airfoil, T_m oscillation time. The quasi - steady approach provides reasonably accurate results for $T_{aer} << T_m$, alternatively it can be found that for a oscillating motion of heave $z = z_0 e^{i\omega t}$ and pitch $\theta = \theta_0 e^{i\omega t}$, the lift force generated by the profile is:

$$\begin{aligned} L = \{\pi\rho b^2[-\omega^2 z_0 + i\omega V\theta_0 + \omega^2 ba\theta_0]+ \\ +2\pi\rho Vb(F+iG)[i\omega z_0 + V\theta_0 + i\omega b(0.5-a)\theta_0]\}e^{i\omega t} \end{aligned} \tag{5.2}$$

In the case of an heaving airfoil in ground effect, forces response could not be harmonic due to the presence of the moving ground, making this relation not valid. A simulation of the case is needed to try to predict a possible behaviour. This will be the main objective of the next chapter.

6

Heaving motion - Dynamic Case

6.1 OpenFOAM

A simulation of a 2D airfoil in oscillating heaving motion is chosen to verify the dynamic behaviour of an aerodynamic component in ground effect [5]. To do so, the open source software *OpenFOAM* is chosen.
OpenFOAM allows highly controllable simulations, with models, discretization schemes and solvers that can be easily changed in order to achieve a better convergence. An *OpenFOAM* case is divided into 3 main directories:

– system: contains the information on parameters associated with the solution procedure itself. It contains at least the following 3 files:

  - controlDict, which contains information on run control parameters including start time, end time, time step and data output;
  - fvSchemes, which contains information on the discretization schemes used in the solution;
  - fvSolution, which contains information on the equation solvers, tolerances and other algorithm controls.

  Additional files can be included, such as the blockMeshDict which can be used to automatically generate a mesh;

– constant: contains all the information that are kept constant during a simulation. It can contain the following dictionaries:

  - physicalProperties, a file containing informations on quantities such as viscosity;
  - polyMesh, a full description of the case mesh;

– 0: contains information on initial conditions and boundary conditions.

Firstly a steady state simulation at different instant heights is brought on to replicate the results obtained for the car in the previous chapter and used to validate the analysis with respect to two PhD thesis [4], [9].
Then an oscillating motion is imposed to the airfoil in ground effect thanks to a morphing mesh and results are analysed and compared with the static case.
The airfoil, whose geometry is taken from [4], has chord $c = 1$ m, it's immersed in air $\rho = 1\ kg/m^3$ $\nu = 3.26 \cdot 10^{-5}\ m^2/s$ at the usual velocity of 15 m/s, that produce a Reynolds number of $Re = 4.6 \cdot 10^5$, as the one used in the references.

6.2 Static analysis

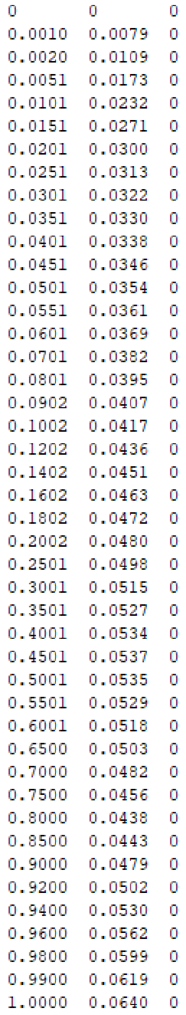
As said before, a static analysis of the phenomenon is done to initialize the case and validate the models.
Heights, measured from the closest point of the profile to the ground, chosen for the study are:

- 100 mm
- 120 mm
- 130 mm
- 140 mm
- 160 mm
- 180 mm
- 200 mm
- 300 mm

Every height correspond to a RANS simulation with k - omega turbulence model.

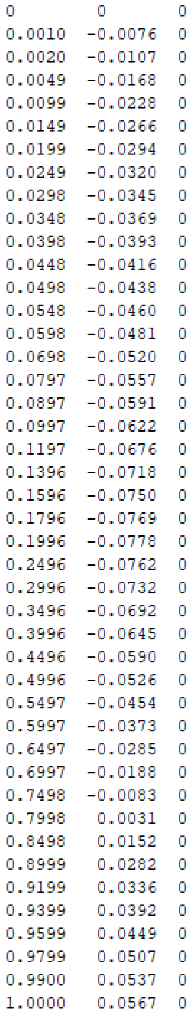
6.2.1 Geometry and meshing

The aerofoil geometry investigated is a derivative of the LS(I)-0413 MOD profile of NASA, whose coordinates are taken from the reference case, has a chord of 1 m and an angle of attack of 3.6 deg.



0	0	0
0.0010	0.0079	0
0.0020	0.0109	0
0.0051	0.0173	0
0.0101	0.0232	0
0.0151	0.0271	0
0.0201	0.0300	0
0.0251	0.0313	0
0.0301	0.0322	0
0.0351	0.0330	0
0.0401	0.0338	0
0.0451	0.0346	0
0.0501	0.0354	0
0.0551	0.0361	0
0.0601	0.0369	0
0.0701	0.0382	0
0.0801	0.0395	0
0.0902	0.0407	0
0.1002	0.0417	0
0.1202	0.0436	0
0.1402	0.0451	0
0.1602	0.0463	0
0.1802	0.0472	0
0.2002	0.0480	0
0.2501	0.0498	0
0.3001	0.0515	0
0.3501	0.0527	0
0.4001	0.0534	0
0.4501	0.0537	0
0.5001	0.0535	0
0.5501	0.0529	0
0.6001	0.0518	0
0.6500	0.0503	0
0.7000	0.0482	0
0.7500	0.0456	0
0.8000	0.0438	0
0.8500	0.0443	0
0.9000	0.0479	0
0.9200	0.0502	0
0.9400	0.0530	0
0.9600	0.0562	0
0.9800	0.0599	0
0.9900	0.0619	0
1.0000	0.0640	0

Figure 6.1: High pressure airfoil side



0	0	0
0.0010	-0.0076	0
0.0020	-0.0107	0
0.0049	-0.0168	0
0.0099	-0.0228	0
0.0149	-0.0266	0
0.0199	-0.0294	0
0.0249	-0.0320	0
0.0298	-0.0345	0
0.0348	-0.0369	0
0.0398	-0.0393	0
0.0448	-0.0416	0
0.0498	-0.0438	0
0.0548	-0.0460	0
0.0598	-0.0481	0
0.0698	-0.0520	0
0.0797	-0.0557	0
0.0897	-0.0591	0
0.0997	-0.0622	0
0.1197	-0.0676	0
0.1396	-0.0718	0
0.1596	-0.0750	0
0.1796	-0.0769	0
0.1996	-0.0778	0
0.2496	-0.0762	0
0.2996	-0.0732	0
0.3496	-0.0692	0
0.3996	-0.0645	0
0.4496	-0.0590	0
0.4996	-0.0526	0
0.5497	-0.0454	0
0.5997	-0.0373	0
0.6497	-0.0285	0
0.6997	-0.0188	0
0.7498	-0.0083	0
0.7998	0.0031	0
0.8498	0.0152	0
0.8999	0.0282	0
0.9199	0.0336	0
0.9399	0.0392	0
0.9599	0.0449	0
0.9799	0.0507	0
0.9900	0.0537	0
1.0000	0.0567	0

Figure 6.2: Low pressure airfoil side

Domain extends for 4 m in front of the airfoil and 9 m at the back and it's 3.24 m high. The defined mesh is structured with 89,428 quad elements with non orthogonality value less than 40 (that is important for schemes and solvers definition).

y^+ value is keep as low as possible both for the ground and the airfoil, trying to find a good compromise between results accordance and computational cost, knowing that this mesh will be the same for the dynamic case, which would require too much computational time and resources for over defined meshes. The mesh is 3D, with only one element in the y direction (normal to the 2D case) of unit length, in order to obtain a reference unit area of the airfoil.

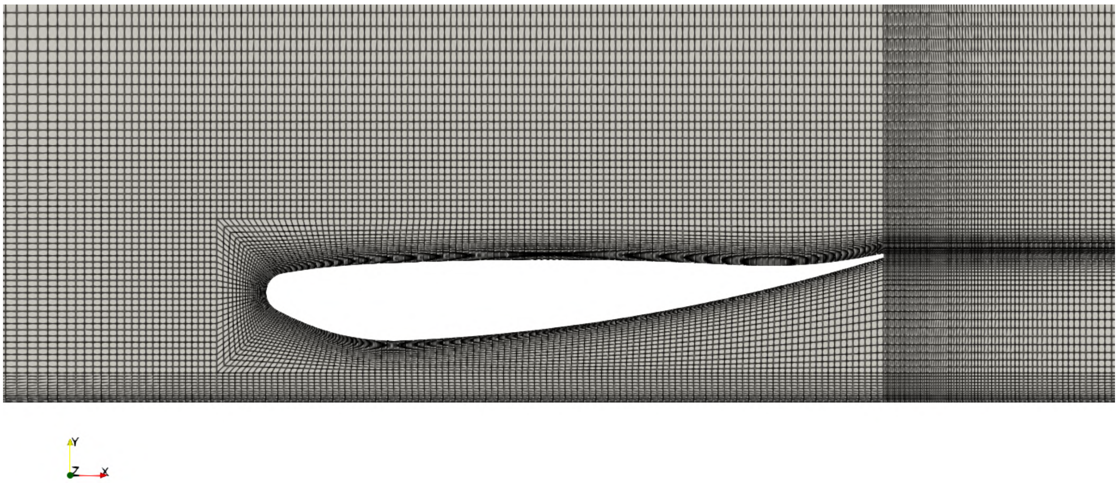


Figure 6.3: Generated *OpenFOAM* mesh for the airfoil at minimum height

Mesh boundary conditions are set all to patches with exception to the ground and the airfoil, set as walls.
In the 0 folder all the initial conditions are set. U file is reported due to its importance in determining the boundary conditions:

```
dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (15 0 0);

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           $internalField;
    }

    outlet
    {
        type            inletOutlet;
        inletValue      uniform (0 0 0);
        value           $internalField;
    }

    ground
    {
      type    fixedValue;
      value   $internalField;
    }

    wing
    {
        type            movingWallVelocity;
        value           uniform (0 0 0);
    }

    top
    {
        type            slip;
    }

    front
    {
        type            empty;
    }

    back
    {
        type            empty;
    }
}
```

Code 6.1: U initial conditions

6.2.2 Models, schemes and solvers

As said, for the static case simulations RANS with k - omega SST turbulence are used.
In the constant folder, momentumTransport, in fact, shows:

```
simulationType RAS;

RAS
{
    model           kOmegaSST;

    turbulence      on;

    printCoeffs     on;
}
```

Code 6.2: momentumTransport

which identifies the model used.
In system, controlDict indicates the application to run the simulation, simpleFoam is the best one for steady state incompressible flows. Simulation stops after 4000 iterations and print out forces and forces coefficient, residuals and y^+ value on walls thru the *OpenFOAM* libraries:

```
application     simpleFoam;

startFrom       latestTime;

startTime       0;

stopAt          endTime;

endTime         4000;

deltaT          1;

writeControl    runTime;

writeInterval   100;

purgeWrite      0;

writeFormat     binary;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

functions
```

```
{
    forces
    {
        type            forces;
        libs            ("libforces.so");
        writeControl    timeStep;
        writeInterval   10;
        patches         (wing);
        rho             rhoInf;
        log             true;
        rhoInf          1;
        CofR            (0 0 0);
    }
    forceCoeffs1
    {
        type            forceCoeffs;
        libs            ("libforces.so");
        writeControl    timeStep;
        writeInterval   10;
        log             yes;
        patches         (wing);
        liftDir         (0 -1 0);
        dragDir         (1 0 0);
        pitchAxis       (0 0 1);
        rho             rhoInf;
        rhoInf          1;
        magUInf         15;
        lRef            1;
        Aref            1;
        CofR            (0 0 0);
    }
    residuals
    {
        type            residuals;
        libs            ("libutilityFunctionObjects.so");
        writeInterval   100;
        fields (p U e k omega nuTilda);
    }
    yPlus
    {
        libs            ("libfieldFunctionObjects.so");
        type            yPlus;
        writeControl    adjustableRunTime;
        writeInterval   100;
    }
}
```

Code 6.3: controlDict - static case

fvSolution set the solvers for p, U, k and omega, defines simple algorithm and relaxation factors:

```
solvers
{
    p
    {
        solver          GAMG;
        tolerance       1e-7;
        relTol          0.1;
        smoother        GaussSeidel;
    }

    U
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        tolerance       1e-8;
        relTol          0.1;
        nSweeps         1;
    }

    k
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        tolerance       1e-8;
        relTol          0.1;
        nSweeps         1;
    }

    omega
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        tolerance       1e-8;
        relTol          0.1;
        nSweeps         1;
    }
}

SIMPLE
{
    nNonOrthogonalCorrectors 0;
}

relaxationFactors
{
    fields
    {
        p               0.3;
    }
    equations
    {
        "(U|k|omega)"   0.7;
    }
}

cache
{
    grad(U);
}
```

Code 6.4: fvSolution - static case

linear solver used are Generalized Geometric Algebraic Mltigrid (GAMG) for pressure (recommended) and the smoothSolver for the others quantities, which consists in a solver smoothed by another linear solver, in this case the Gauss Seidel method. Non orthogonal correctors for the simple algorithm are set to 0 since there's no necessity to recalculate the value of p for meshes with non orthogonality less than 40. Relaxation factors are set to a default value for this external aerodynamics incompressible case.

fvSchemes defines discretization schemes for convective, diffusive and gradient terms:

```
ddtSchemes
{
    default steadyState;
}

gradSchemes
{
    default         Gauss linear;
    grad(p)         Gauss linear;
    grad(U)         Gauss linear;
}

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

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

wallDist
{
    method meshWave;
}
```

Code 6.5: fvSchemes - static case

time discretization isn't necessary in steady state. Convective terms (divSchemes) use a second order accurate scheme bounded (Gauss linear upwind), more stable than the unbounded one (Gauss linear). laplacianSchemes and snGradSchemes (diffusive terms) use a fully orthogonal scheme, that's the best choice for good quality meshes. Gradient terms avoid limiters increasing accuracy.

6.2.3 Plots and post processing

Residuals are saved during the iterations and its plots monitored through the application foamMonitor. These plots define how well the simulation converges and represent only the initial residual, before the linear solver. This is a default in *OpenFOAM*, with initial residuals that give a better understanding of the simulation convergence, while final residuals are easily lowered earlier.
Forces are collected in files from which a *Python* script extract their values in a format more compatible with *Fortran90* and *Gnuplot*. Thanks to *Gnuplot* it's possible to print out the force coefficient plot with respect to iterations.
For example at the height 160 mm:

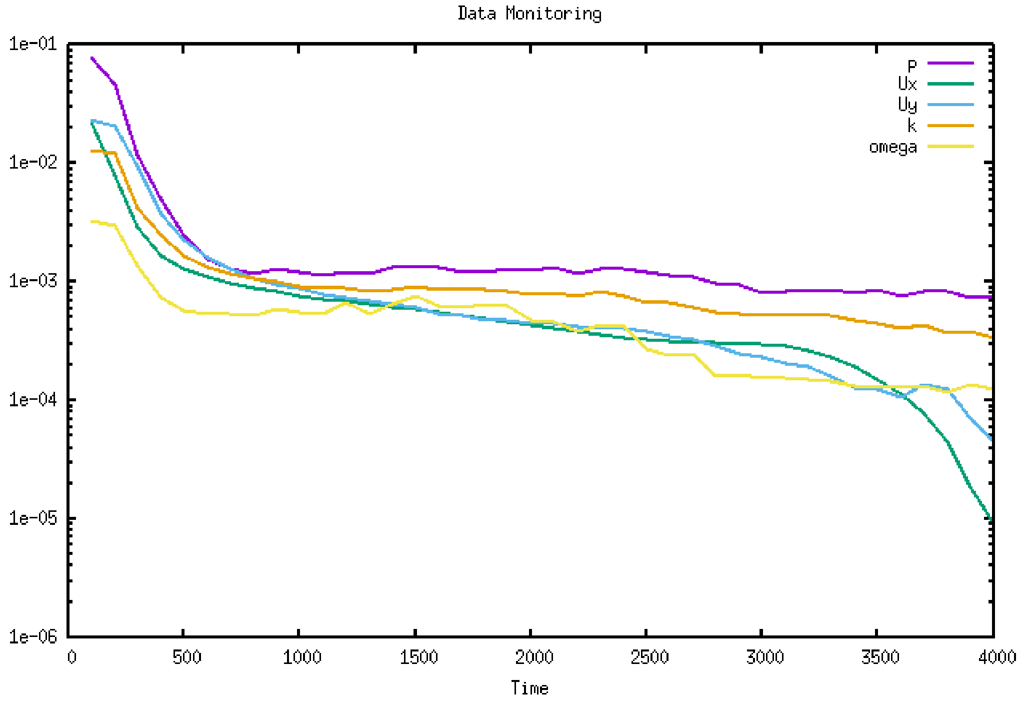


Figure 6.4: Residuals plot - 160 mm static case

Initial residuals reach sufficiently low values and force coefficients show a good convergence of the values.

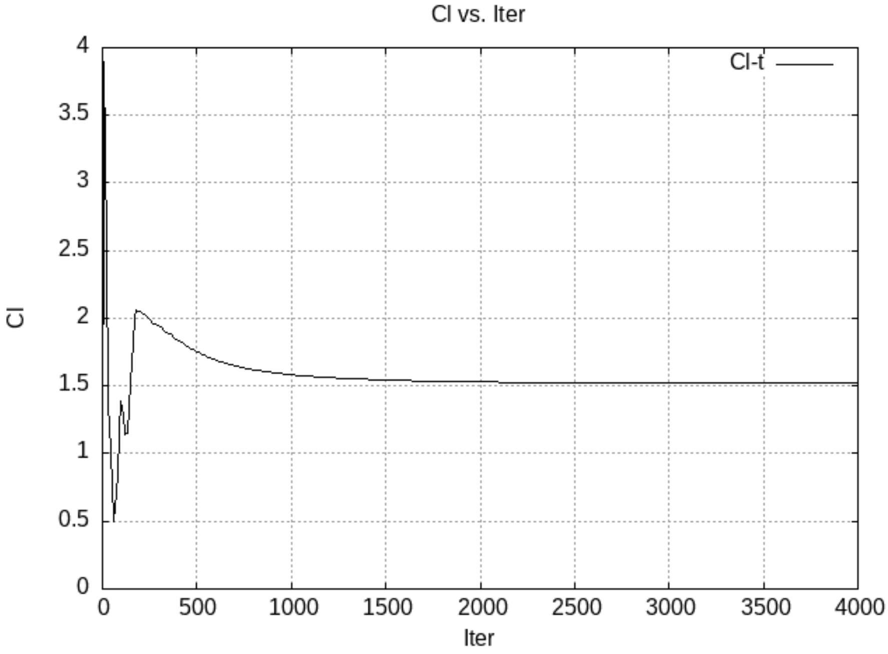


Figure 6.5: Cl plot - 160 mm static case

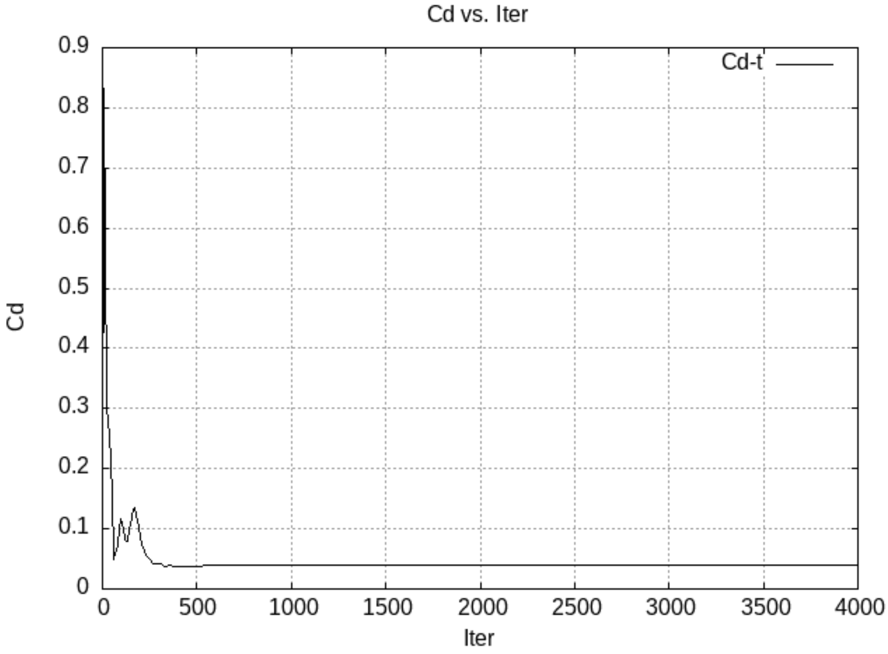


Figure 6.6: Cd plot - 160 mm static case

The post processing is executed in *Paraview*. Contours of velocity and pressure are created and the plot of the pressure coefficient around the airfoil is extracted. Scenes and plots extracted in this phase have mainly the objective to validate the simulations, in order to have a good starting point for the dynamic analysis. Scenes for the case at 160 mm height:

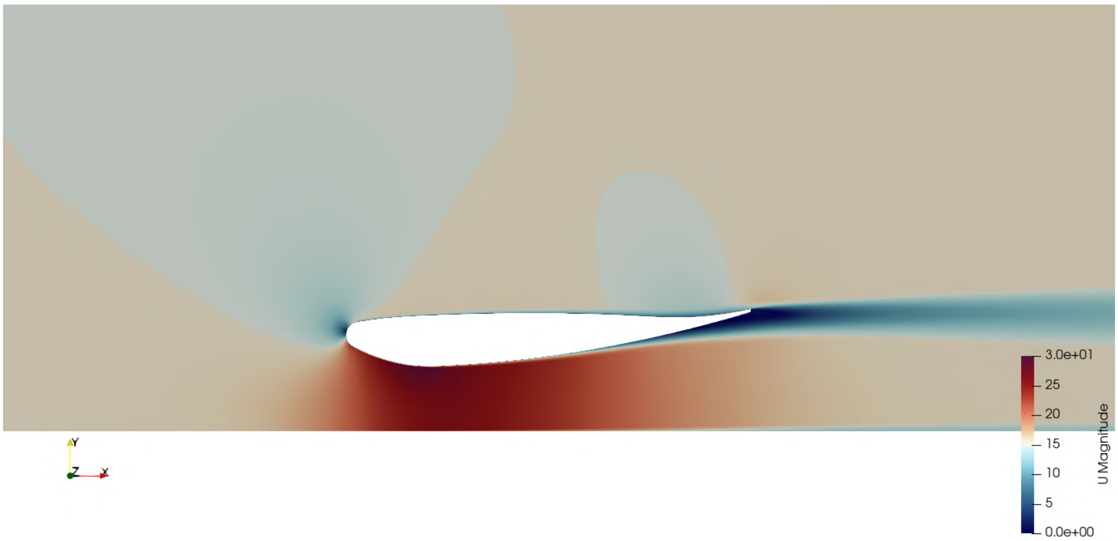


Figure 6.7: Velocity scene - 160 mm static case

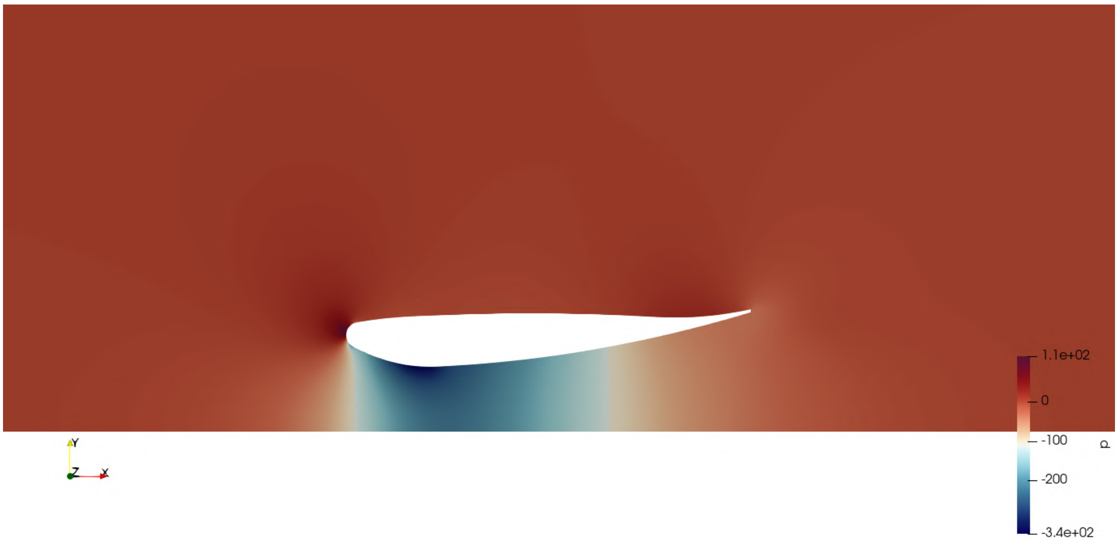


Figure 6.8: Pressure scene - 160 mm static case

6.2.4 Results of the analysis and validation

Studying a ground effect case in 2D, the Cd won't be an important value to take in consideration for the analysis. For this reason the Cl values will be the ones reported more frequently.
Values of Cl and Cd for different height from the ground are:

h [mm]	Cl	Cd
100	1.396	0.056
120	1.403	0.049
130	1.406	0.046
140	1.412	0.043
160	1.413	0.039
180	1.400	0.035
200	1.379	0.033
300	1.242	0.025

Table 6.1: Force coefficients - *OpenFOAM* static case

It can be immediately clear that the behaviour seen in the static analysis of the whole car is confirmed also for a 2D airfoil, with an increase in Cl with smaller heights from ground, with the profile that behaves as a Venturi channel, and a decrease after the profile gets too close, due to the stall. In 2D Cd grows constantly moving towards the ground as expected, since there aren't tip vortices and induced drag.
The Cl - h curve can be compared with the one obtained by David and Zerihan [4] through a wind tunnel simulation of the same case:

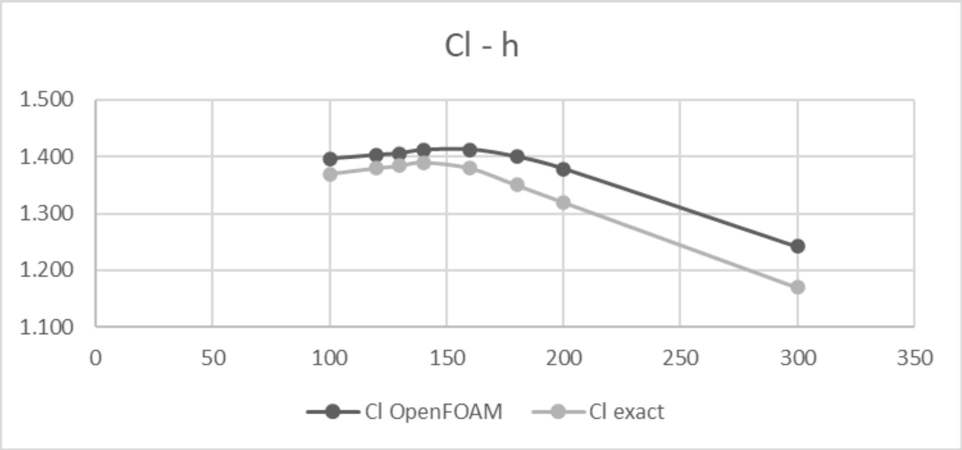


Figure 6.9: Cl plot with respect to h validation

Curve obtained has the same trend of the exact one, with a slightly overestimation of Cl. It's possible to notice how both maximum in downforce are around the height of 150 mm and the stall is well predicted by the model.
Pressure coefficient around the airfoil are also compared at some height from the ground:

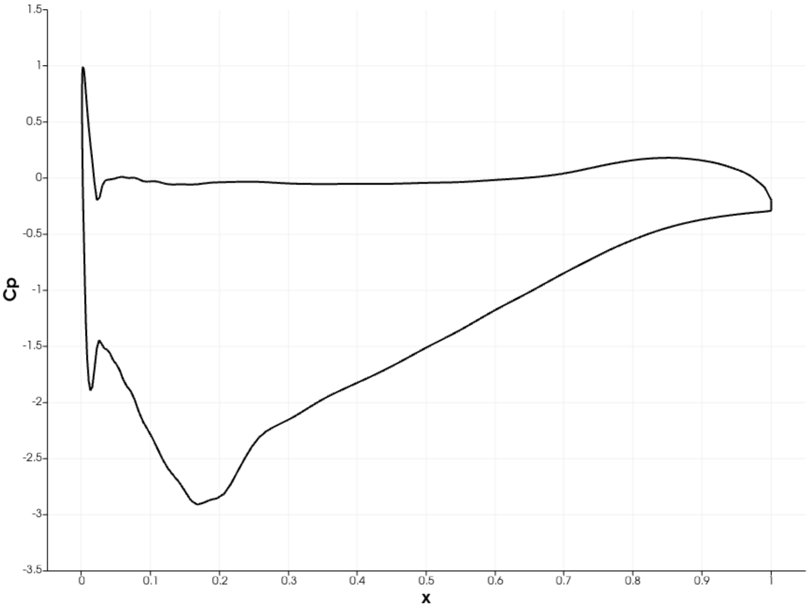


Figure 6.10: Cp plot - 180 mm height

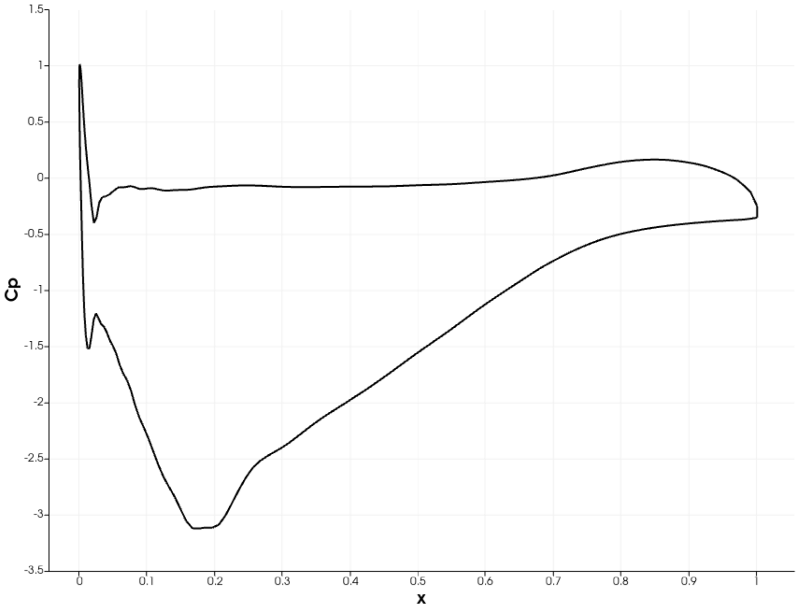


Figure 6.11: Cp plot - 130 mm height

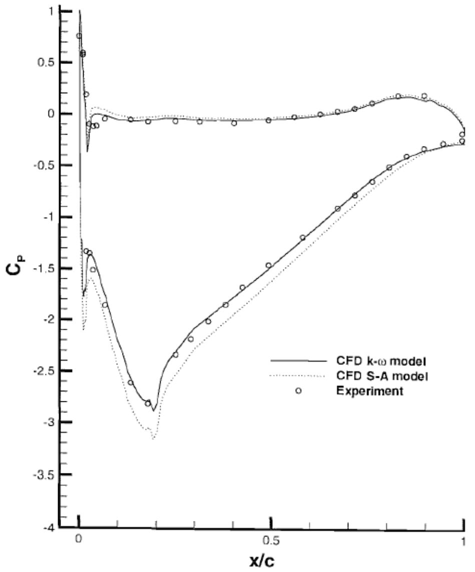


Figure 6.12: Cp plot - 180 mm height real case

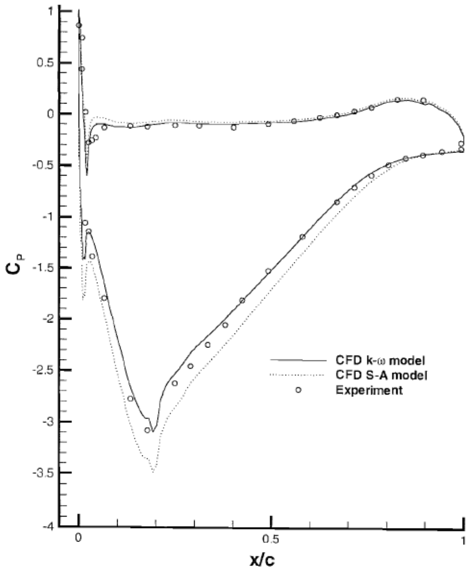


Figure 6.13: Cp plot - 130 mm height real case

For both heights considered, Cp is sufficiently well predicted, confirming what has been seen for the Cl. A slightly minor Cp in the low pressure side justifies the small overestimation seen for the downforce.
Comparing the results obtained between different heights it's noticeable how lower pressure coefficients are obtained at lower heights, with a stronger adverse pressure gradient that at a too low ground proximity creates a stall, visible both from the velocity scene and the Cp plot, with a characteristic shape in the lower part of the profile near the trailing edge, with higher values of Cp reached earlier.

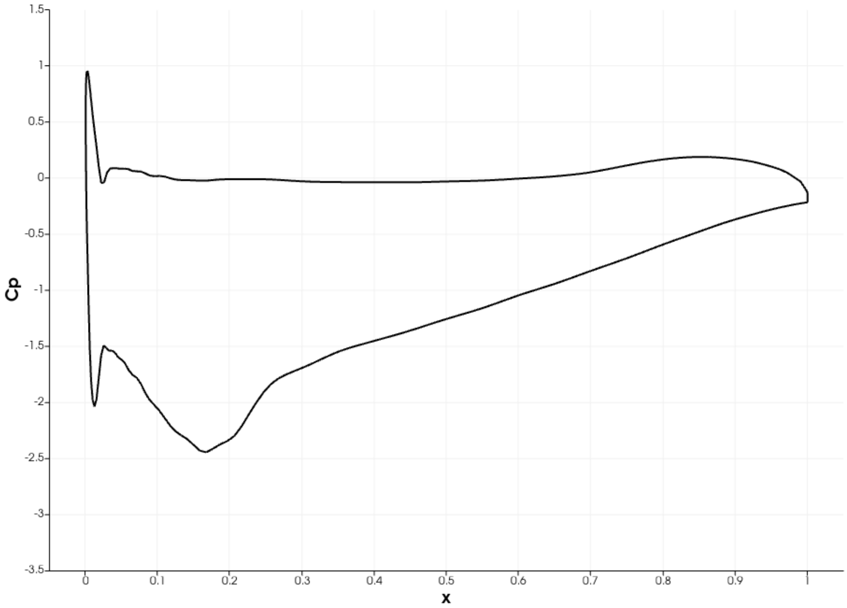


Figure 6.14: Cp plot - 300 mm height

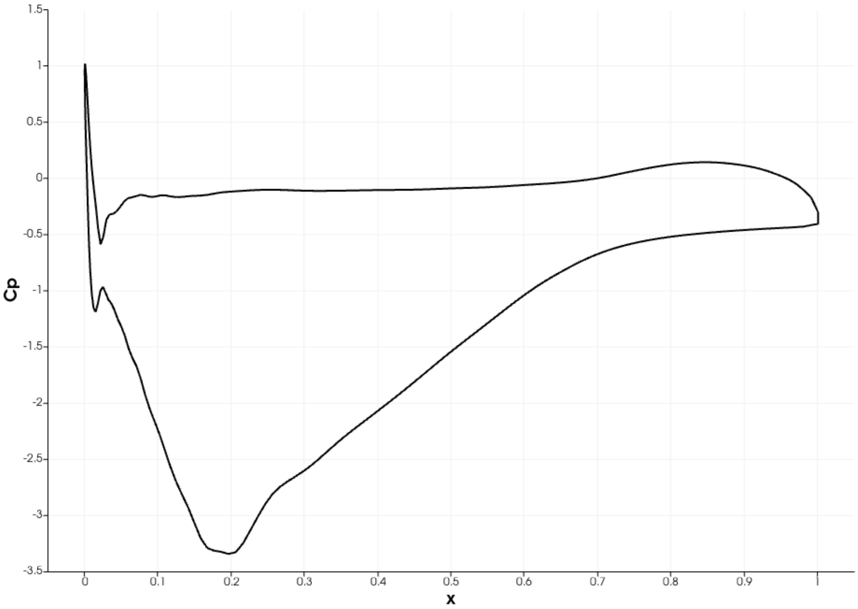


Figure 6.15: Cp plot - 100 mm height

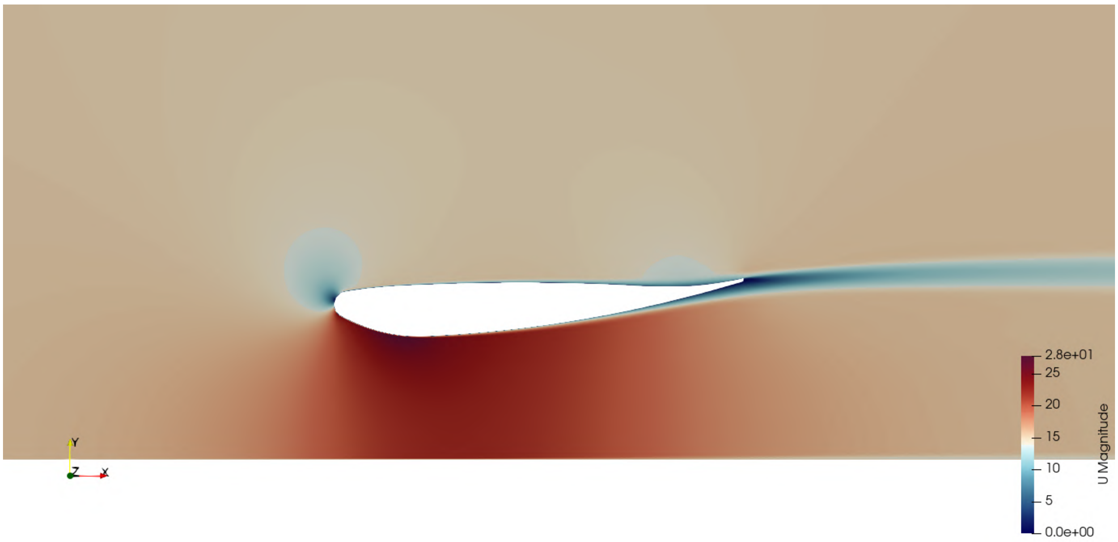


Figure 6.16: Velocity scene - 300 mm height

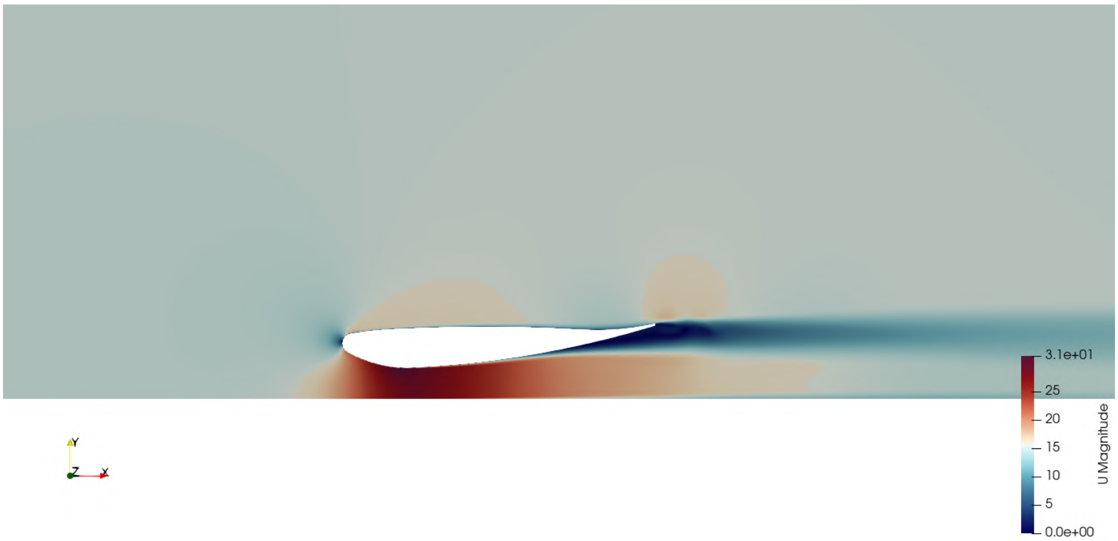


Figure 6.17: Velocity scene - 100 mm height

6.3 Dynamic analysis

The same airfoil in the domain considered earlier is then put in an oscillatory motion, to verify its dynamic behaviour in ground effect. Results will then be analysed and some hypothesis of the car's behaviour in a similar situation, extending the results obtained, will be made.
Qualitative results can also be compared to the similar work bring out by Molina and Zhang [10].
Three different analysis will be made, varying the amplitude and the period of the cosinusoidal motion:

- $A = 50$ mm, $\omega = 6.28$ rad/s, $T_m = 1$ s;
- $A = 15$ mm, $\omega = 6.28$ rad/s, $T_m = 1$ s;
- $A = 15$ mm, $\omega = 3.14$ rad/s, $T_m = 2$ s.

The first simulation has a wide oscillation in a short period of time, exaggerating the dynamic effects, with a fast moving airfoil between the minimums and maximums of the function. The second one has a reduced amplitude, closer to the one experienced by the car, that will imply slower transitions between maximum and minimum positions. The last simulation doubles the period of oscillation, halving ω and so the reduced frequency k, theoretically bringing the solution closer to the steady state one.

6.3.1 Dynamic mesh

The method chosen to simulate the oscillating airfoil is the one using a morphing mesh. The starting mesh from the steady state analysis is deformed following the airfoil movement, changing cell sizes and positions. This can lead to bad quality cells and is in fact used only because the oscillation amplitudes are small, making the remeshing unnecessary.
To do so, in constant, the file dynamicMeshDict is added:

```
mover
{
    type            motionSolver;

    libs            ("libfvMeshMovers.so" "libfvMotionSolvers.so");

    motionSolver    displacementSBRStress;

    diffusivity     inverseDistance (wing ground);
}
```

Code 6.6: dynamicMeshDict

This file defines the solver for the motion, that in this case is a displacement solver called SBR Stress, that's an evolution of the Laplacian displacement solver, based on solving the cell centre solid body rotation stress equations for the motion displacement:

$$\nabla \cdot (\gamma \nabla \mathbf{x}) + \nabla \cdot \gamma [(\nabla \mathbf{x}^T - \nabla \mathbf{x}) - tr(\nabla \mathbf{x})\mathbf{I}] = 0 \tag{6.1}$$

with γ motion diffusivity, **x** point displacement and the first term of the equation $\nabla \cdot (\gamma \nabla \mathbf{x})$ Laplacian displacement.

Diffusivity is important to maintain cells around the walls fixed, deforming only the ones that are in zones of less interest. In this case inverseDistance is used on the walls (the airfoil and the ground), a more restrictive diffusivity method such as quadratic inverseDistance would have fixed the position of too many cells between airfoil and ground, causing a wide deformation of few cells that will create a bad quality mesh.

In the 0 folder is necessary to add the desired displacements of mesh boundaries in the file pointDisplacement:

```
dimensions      [0 1 0 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    wing
    {
        type            myoscillatingDisplacement;
        amplitude       (0 -0.05 0);
        omega           6.28;
        value           uniform (0 0 0);
    }

    front
    {
        type            empty;
    }

    back
    {
        type            empty;
    }

    ".*"
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
}
```

Code 6.7: pointDisplacement

inlet, outlet, top patch and ground are fixed. Wing is instead described by an oscillatory movement, with an amplitude and a omega. It can be noticed how the type of motion is myoscillatingDisplacement instead of the *OpenFOAM* standard oscillatingDisplacement. This is due to a change in the standard definition of the oscillation motion, that is a sine, into a cosine function, to better adapt to the problem as will be explained later, by creating a new *OpenFOAM* library. To do so in $WM_PROJECT_USER_DIR a path equal to the one that identifies the standard oscillatingDisplacement is created, with the last folder called myoscillatingDisplacements containing the original files .C and .H modified with a *my* at the beginning. In .C, the line 111 that defines the sine motion function is changed to:

```
        Field<vector>::operator=(amplitude_*cos(omega_*t.value())-
    amplitude_)
```

Code 6.8: myoscillatingDisplacement function changes

To make this change effective, the Make folder relative to the oscillatingDisplacement must be created in the new path in the same position of the original one, with inside the files called *files* and *options*. In particular files must be modified with the new libraries:

```
derivedPoint = pointPatchFields/derived

$(derivedPoint)/myoscillatingDisplacement/
    myoscillatingDisplacementPointPatchVectorField.C

```

```
LIB = $(FOAM_USER_LIBBIN)/libmyfvMotionSolvers
```

Code 6.9: new library definition in files

Running the wmake, *OpenFOAM* compiles the new libraries and creates the new motion myoscillatingDisplacement.

This choice, as anticipated, was necessary to obtain the desired oscillatory motion. In fact, airfoil lowest point from the ground (100 mm) is chosen as starting position in order to have only bigger cells under the profile during the whole morphing process, avoiding too compressed cells or the creation of invalid cells at the lower heights from the ground, bringing bad quality cells when they are more important. So the new motion is a cosine function, that needs to be translated in the origin:

$$A \cdot \cos(\omega \cdot t) - A \tag{6.2}$$

to start from a minimum, A should be negative.

6.3.2 Models, schemes and solvers

The simulation starts from the fields of the last iteration of the static case at 100 mm height from the ground. To do so the application mapFields is used. Also this set of simulations will be RANS solved with kOmegaSST model, with the difference of being transient simulations, often referred to as Unsteady RANS (URANS), in which time discretization is important and Courant number must be checked constantly as already anticipated in the previous chapters.

In this case controlDict is set as follows:

```
libs ("libmyfvMotionSolvers.so");

application     pimpleFoam;

startFrom       latestTime;

startTime       0;

stopAt          endTime;

endTime         2;

deltaT          1e-5;

```

```
writeControl    adjustableRunTime;

writeInterval   1e-2;

purgeWrite      0;

writeFormat     binary;

writePrecision  10;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

adjustTimeStep  yes;

maxCo           0.8;
```

Code 6.10: controlDict - dynamic case

It's immediately clear how time treatment changes from the static case. The starting delta time imposed in the simulation is $1 \cdot 10^{-5}$ s and the adjustableRunTime feature is chosen. This control allows the automatic change of the timestep in order to respect the maximum Courant number indicated, that is set to be 0.8 to stay in safe conditions. *OpenFOAM* automatically increase the timestep when it's possible, making the simulation faster, and lowers it when CFL reaches critical values, slowing down the simulation in favour of accuracy.

Functions requested are the same from the steady state case, but the additional library libmyfvMotionSolvers needs to be mentioned to run the simulation with the new command myoscillatingDisplacement.

For every case the end time is set after two periods are completed, 2 s in the first two cases, 4 s in the third one. Notably, the behaviour observed in the second period mirrors that of the third period in early studies. Consequently, for analytical purposes, it is deemed reasonable to treat the second period as a regime condition.

The fvSchemes are kept similar to the steady state ones, that showed good results, with small changes in the divSchemes for k and omega, in search of more accuracy:

```
ddtSchemes
{
    default CrankNicolson 0.7;
}

gradSchemes
{
    default         Gauss linear;
    grad(p)         Gauss linear;
    grad(U)         Gauss linear;
}

divSchemes
{
    default         none;
    div(phi,U)      Gauss linearUpwind grad(U);
    div(phi,k)      Gauss limitedLinear 1;
    div(phi,omega)  Gauss limitedLinear 1;
    div((nuEff*dev2(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

wallDist
{
    method meshWave;
}
```

Code 6.11: fvSchemes - dynamic case

Time discretization scheme chosen is a Crank - Nicholson blended with Euler at the value of 0.7. This means that the used scheme is almost a pure Crank - Nicholson second order accurate implicit scheme, with a small value of relaxation to obtain a better stability.
Fundamental for the simulation convergence is also the solver used for the calculations, already indicated in the controlDict: pimpleFOAM, a transient solver for incompressible, turbulent flow of Newtonian fluids on a moving mesh. Its settings are indicated in fvSolution, together with the definition of the other linear solvers and the relaxation factors, that in this case are:

```
solvers
{
    "pcorr.*"
    {
        solver           GAMG;
        tolerance        1e-7;
        relTol           0;
        smoother         GaussSeidel;
    }

    p
    {
        $pcorr;
        tolerance        1e-6;
        relTol           0.01;
    }

    pFinal
    {
        $p;
        tolerance        1e-7;
        relTol           0;
    }

    "(U|k|omega)"
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-06;
        relTol          0.1;
    }

    "(U|k|omega)Final"
    {
        $U;
        tolerance       1e-07;
        relTol          0;
    }

    cellDisplacement
    {
        solver          GAMG;
        tolerance       1e-7;
        relTol          0;
        smoother        GaussSeidel;
    }
}

PIMPLE
{
    momentumPredictor        yes;
    correctPhi               yes;
    nOuterCorrectors         50;
    nCorrectors              3;
    nNonOrthogonalCorrectors 1;

    outerCorrectorResidualControl
    {
      p
      {
        tolerance    1e-4;
        relTol       0;
      }
    }
}

relaxationFactors
{
    fields
    {
        p               0.3;
        pFinal          1;
    }
    equations
    {
        "(U|k|omega)"   0.7;
        "(U|k|omega)Final" 1.0;
    }
}

cache
{
    grad(U);
}
```

Code 6.12: fvSolution - dynamic case

Changes with respect to the static case are visible. Final quantities identify the last value of the field for each timestep. They are requested with stricter tolerances in order to obtain a more precise final iteration, while keeping the first iterations of the same timestep faster thanks to the smaller tolerances used.
The presence of the pimple algorithm permits to still have relaxation factors for the value of p, U, k and omega, but their final values at each timestep needs to be calculated without any relaxation in order to be time consistent.
The pcorr and cellDisplacement quantities are strictly related to the morphing mesh and are solved at the start of every timestep to correct the solution to the mesh change.
What coordinates the whole solution is the pimple solver, defined in its section, whose algorithm will be treated in the next theoretical overview.

6.3.3 Theoretical overview

The pimple algorithm, which stands for piso (Pressure Implicit with Splitting of Operators) and simple (Semi-Implicit Method for Pressure-Linked Equations), it's an iterative method that combines the two algorithms, the first one typical of transient simulations, while the first one, as already said, is classical for the steady state ones.
In general, within in time the algorithms solve a pressure equation, to enforce mass conservation, with an explicit correction to velocity to satisfy momentum conservation. They optionally begin each step by solving the momentum equation, the so-called momentum predictor [2].
While all the algorithms solve the same governing equations, the algorithms principally differ in how they loop over the equations. In the case of the pimple algorithm, loop is controlled in the fvSolution file by these input parameters:

- momentumPredictor: switch that controls solving of the momentum predictor. Typically set to *no* for some flows, including low Reynolds number and multiphase, set to *yes* in the case treated;

- nOuterCorrectors: it enables looping over the entire system of equations within one time step, representing the total number of times the system is solved. It must be ≥ 1, if it's set to 1 the algorithm chosen is the piso one. Larger numbers correspond to the pimple algorithm, that loops like a simple algorithm inside the same timestep. In the simulations a value of 50 will be used, but a residual control is added to stop the loops when the tolerance on pressure is reached;

- nCorrectors: sets the number of times the algorithm solves the pressure equation and momentum corrector in each step. It's typically set to 2 or 3. It's a great parameter to help convergence, but computational demanding. In the simulations the value of 3 will be used;

- nNonOrthogonalCorrectors: specifies repeated solutions of the pressure equation, used to update the explicit non - orthogonal correction of the Laplacian term $\nabla^2 p$; typically set to 0 for steady-state and 1 for transient cases. Major numbers could help in the case of meshes with a high non orthogonality. In the case treated a value of 1 is chosen due to the good quality mesh used.

In particular, the pimple loop is essentially a predictor - corrector algorithm (as discussed in [16]), that in an incompressible problem can be described as follows:

- predictor: predicts the pressure field p, discretizes the momentum equation based on p, and solves the linear system of **u**;

- corrector: constructs and discrete the pressure equation based on the updated velocity field;

- solves the produced linear system of p;

- updates the velocity field based on the new pressure field.

In terms of equations, for 2 loops (nOuterCorrectors = 2):

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{\nabla p}{\rho} + \nu \nabla^2 \mathbf{u} \tag{6.3}$$

$$\nabla^2 p = f(\mathbf{u}, \nabla p) \tag{6.4}$$

$$\rightarrow p_{\text{new}}$$

$$\rightarrow \text{corrects } \mathbf{u} \text{ with } p_{\text{new}} \rightarrow \mathbf{u}_{\text{corrected}}$$

$$\rightarrow \mathbf{u}_{\text{corrected}} \text{ and } p_{\text{new}} \text{ for the second loop}$$

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{\nabla p}{\rho} + \nu \nabla^2 \mathbf{u} \tag{6.5}$$

$$\nabla^2 p = f(\mathbf{u}, \nabla p) \tag{6.6}$$

$$\rightarrow p_{\text{new}}$$

$$\rightarrow \text{corrects } \mathbf{u} \text{ with } p_{\text{new}} \rightarrow \mathbf{u}_{\text{corrected}}$$

A number > 1 for the nCorrectors forces the solution of the equation $\nabla^2 p = f(\mathbf{u}, \nabla p)$ more times in the same loop.
A number > 0 for nNonOrthogonalCorrectors means that the pressure field is corrected more often: $\nabla^2 p = f(\mathbf{u}, \nabla p) \rightarrow p_{\text{new}} \rightarrow \nabla^2 p_{\text{new}} = f(\mathbf{u}, \nabla p_{\text{new}})$.
In the case of study the additional parameter correctPhi is set to *yes*, this is required for moving meshes, when non - conservative fluxes may adversely affect the prediction - part of the solution algorithm.

6.3.4 Plots and post processing

Similarly to the static case, initial residuals can be plotted, ensuring the convergence of the analysis, the example of the first dynamic case is reported:

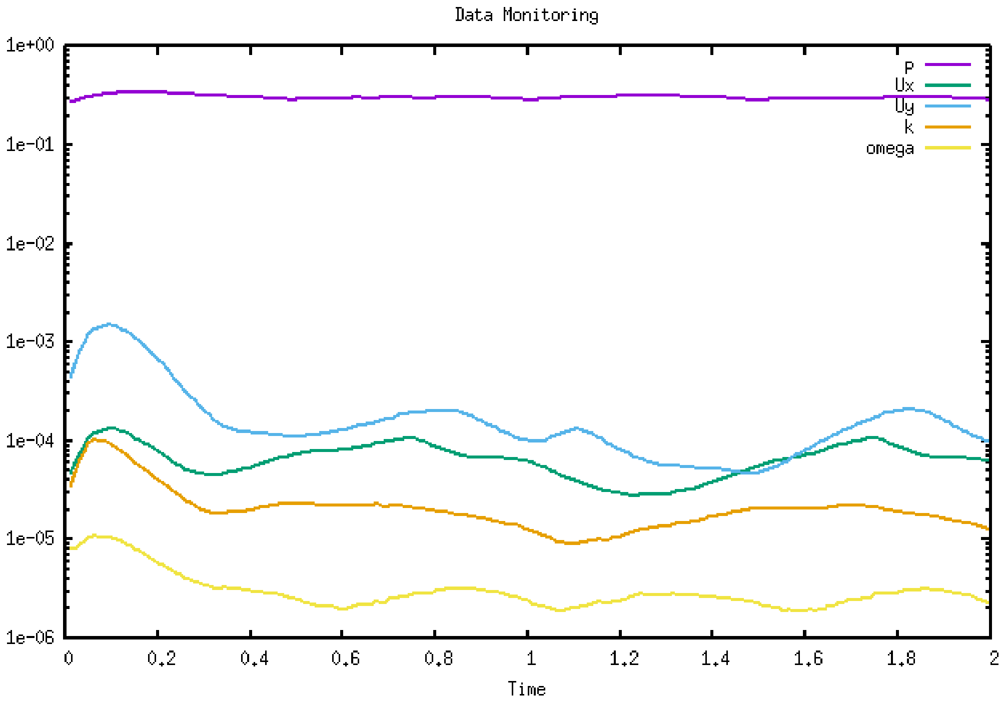


Figure 6.18: Initial residuals plot - first dynamic case

Final residuals are difficult to extract due to the high number of loops and correction per timestep, but a classical timestep solution from the log file of the first dynamic simulation is reported:

```
Courant Number mean: 0.03830375014 max: 0.7961923356
deltaT = 9.615384615e-05
Time = 0.998173s

PIMPLE: Iteration 1
GAMG:  Solving for cellDisplacementx, Initial residual = 3.98915685e
    -06, Final residual = 6.849449974e-08, No Iterations 8
GAMG:  Solving for cellDisplacementy, Initial residual =
    0.01379196604, Final residual = 5.542660852e-08, No Iterations 4
GAMG:  Solving for pcorr, Initial residual = 1, Final residual =
    5.864329887e-07, No Iterations 1000
GAMG:  Solving for pcorr, Initial residual = 0.2199107821, Final
    residual = 9.957794281e-08, No Iterations 512
```

```
time step continuity errors : sum local = 3.984107427e-16, global = 2.162928274e-17, cumulative = -8.457096724e-10
smoothSolver:  Solving for Ux, Initial residual = 6.254373643e-05, Final residual = 1.099463018e-07, No Iterations 1
smoothSolver:  Solving for Uy, Initial residual = 0.0001016654866, Final residual = 1.358196202e-07, No Iterations 1
GAMG:  Solving for p, Initial residual = 0.2901361726, Final residual = 0.002067847672, No Iterations 6
GAMG:  Solving for p, Initial residual = 0.008191287276, Final residual = 7.959008533e-05, No Iterations 111
time step continuity errors : sum local = 8.468891828e-11, global = 3.939484692e-12, cumulative = -8.417701877e-10
GAMG:  Solving for p, Initial residual = 0.188488328, Final residual = 0.001569122041, No Iterations 6
GAMG:  Solving for p, Initial residual = 0.005904987227, Final residual = 5.734323036e-05, No Iterations 111
time step continuity errors : sum local = 6.095309446e-11, global = 2.803457387e-12, cumulative = -8.389667303e-10
GAMG:  Solving for p, Initial residual = 0.1886594434, Final residual = 0.001653022362, No Iterations 6
GAMG:  Solving for p, Initial residual = 0.005957906408, Final residual = 9.86365954e-08, No Iterations 513
time step continuity errors : sum local = 1.048508515e-13, global = 3.306971376e-15, cumulative = -8.389634233e-10
PIMPLE: Iteration 2
smoothSolver:  Solving for Ux, Initial residual = 2.26879405e-05, Final residual = 4.813264249e-08, No Iterations 1
smoothSolver:  Solving for Uy, Initial residual = 5.739080022e-05, Final residual = 1.143557692e-07, No Iterations 1
GAMG:  Solving for p, Initial residual = 0.02610869272, Final residual = 0.0002588895453, No Iterations 14
GAMG:  Solving for p, Initial residual = 0.00183913786, Final residual = 1.805928988e-05, No Iterations 88
time step continuity errors : sum local = 1.871998739e-11, global = -1.110581282e-12, cumulative = -8.400740046e-10
GAMG:  Solving for p, Initial residual = 0.01801475291, Final residual = 0.0001651179362, No Iterations 15
GAMG:  Solving for p, Initial residual = 0.001264752893, Final residual = 1.235106228e-05, No Iterations 93
time step continuity errors : sum local = 1.279651069e-11, global = -7.26412467e-13, cumulative = -8.40800417e-10
GAMG:  Solving for p, Initial residual = 0.01808153755, Final residual = 0.0001803164008, No Iterations 14
```

```
GAMG:  Solving for p, Initial residual = 0.001279932576, Final residual = 9.97009243e-08, No Iterations 438
time step continuity errors : sum local = 1.033105753e-13, global = -3.296566796e-15, cumulative = -8.408037136e-10
...
PIMPLE: Iteration 6
smoothSolver:  Solving for Ux, Initial residual = 2.12412882e-07, Final residual = 2.12412882e-07, No Iterations 0
smoothSolver:  Solving for Uy, Initial residual = 3.746545564e-07, Final residual = 3.746545564e-07, No Iterations 0
GAMG:  Solving for p, Initial residual = 6.660833102e-05, Final residual = 7.224679115e-07, No Iterations 5
GAMG:  Solving for p, Initial residual = 2.678958171e-06, Final residual = 7.55360342e-07, No Iterations 1
time step continuity errors : sum local = 7.759388169e-13, global = 3.549794054e-15, cumulative = -8.409622078e-10
GAMG:  Solving for p, Initial residual = 5.370588674e-05, Final residual = 9.962541915e-07, No Iterations 6
GAMG:  Solving for p, Initial residual = 2.537929804e-06, Final residual = 7.86334491e-07, No Iterations 3
time step continuity errors : sum local = 8.077543894e-13, global = -2.323130225e-14, cumulative = -8.409854391e-10
GAMG:  Solving for p, Initial residual = 5.276359697e-05, Final residual = 9.911565032e-07, No Iterations 7
GAMG:  Solving for p, Initial residual = 2.489207039e-06, Final residual = 9.82512751e-08, No Iterations 94
time step continuity errors : sum local = 1.009277038e-13, global = -4.438484572e-15, cumulative = -8.409898776e-10
PIMPLE: Converged
        Doing final iteration
PIMPLE: Iteration 7
smoothSolver:  Solving for Ux, Initial residual = 1.048341791e-07, Final residual = 2.204477177e-09, No Iterations 1
smoothSolver:  Solving for Uy, Initial residual = 1.882534584e-07, Final residual = 1.301356568e-09, No Iterations 1
GAMG:  Solving for p, Initial residual = 0.2743782153, Final residual = 0.002514568522, No Iterations 5
GAMG:  Solving for p, Initial residual = 0.01202640473, Final residual = 0.0001187479976, No Iterations 109
time step continuity errors : sum local = 1.064038545e-10, global = -4.580082365e-12, cumulative = -8.455699599e-10
GAMG:  Solving for p, Initial residual = 0.2069682659, Final residual = 0.001932275222, No Iterations 5
```

```
GAMG:  Solving for p, Initial residual = 0.008458428204, Final residual = 8.342702986e-05, No Iterations 111
time step continuity errors : sum local = 7.488150863e-11, global = -3.207179161e-12, cumulative = -8.487771391e-10
GAMG:  Solving for p, Initial residual = 0.2070622312, Final residual = 0.001930625997, No Iterations 5
GAMG:  Solving for p, Initial residual = 0.008461074543, Final residual = 9.924075659e-08, No Iterations 519
time step continuity errors : sum local = 8.909250192e-14, global = -2.752655844e-15, cumulative = -8.487798918e-10
smoothSolver:  Solving for omega, Initial residual = 2.395115419e-06, Final residual = 1.16337789e-08, No Iterations 1
smoothSolver:  Solving for k, Initial residual = 1.295011867e-05, Final residual = 6.660959204e-08, No Iterations 1
PIMPLE: Converged in 7 iterations
ExecutionTime = 77357.87624 s  ClockTime = 77984 s

forces write:
    sum of forces:
        pressure : (4.995938838 183.7683503 1.496218718e-11)
        viscous  : (1.414762971 0.05697095052 6.336857517e-15)
        porous   : (0 0 0)
    sum of moments:
        pressure : (91.88417516 2.497969419 -67.44080192)
        viscous  : (0.02848547526 0.7073814855 0.05101090163)
        porous   : (0 0 0)
forceCoeffs write:
    Cm    = -0.5990203646
    Cd    = 0.05698401608
    Cl    = 1.634002856
    Cl(f) = 0.2179810633
    Cl(r) = 1.416021793
```

Code 6.13: pimpleFoam timestep log

The loop refers to the time 0.998173s, it can be noticed how Courant number is kept under the limit value of 0.8 while the timestep is increased to $9.6 \cdot 10^{-5}$ s, almost ten times the starting value.

There are 6 pimple iterations, then the convergence of the nOuterCorrectors is reached and the final iteration with no relaxation to regain the time consistency is started. Each iteration the pressure field is solved 3 times (nCorrectors) and corrected twice (nNonOrthogonalCorrectors).

Residuals are lowered easily in the first 6 pimple loops (that are actually like

simple ones), then in the last pimple iteration initial residual for p grows again due to the change in relaxation, but it's brought down again inside the loop in a faster way that it would have been in the case of a piso algorithm.
After some predefined timesteps, forces and force coefficients are printed out. The already mentioned *Python* script, together with a *Fortran* script, launch *Gnuplot*, plotting and saving plots of Cl - h, Cl - t, Cd - h and Cd - t. The *Fortran* script reads and treats forces data as follows:

```fortran
    program

    use,intrinsic :: iso_fortran_env,only: rp  => real64
    implicit none

    character*256 :: ctmp
    integer :: fid,err
    integer :: i=0,num_lines=0
    real(rp), allocatable :: time(:),lift(:),drag(:),momentz(:),height(:),cl(:),cd(:)

    open(newunit=fid,file="forces.txt",status="unknown",iostat=err)

    do while(err==0)
      num_lines=num_lines+1
      read(fid,*,iostat=err) ctmp
    end do

    num_lines=num_lines-1

allocate(time(num_lines),lift(num_lines),drag(num_lines),momentz(num_lines),height(num_lines),cl(num_lines),cd(num_lines))

    rewind(fid)
    do i=1,num_lines
      read(fid,*) time(i),lift(i),drag(i),momentz(i)
    end do

    do i=1,num_lines
      height(i)=-0.05_rp*cos(6.28_rp*time(i))+0.05_rp
      cl(i)=lift(i)/(0.5_rp*15_rp**2_rp)
      cd(i)=drag(i)/(0.5_rp*15_rp**2_rp)
    end do

    close(fid)

    open(newunit=fid,file="forces_plot.txt",status="unknown",iostat=err)

    if(err==0) then
      do i=100,num_lines
        write(fid,*) height(i),time(i),cl(i),cd(i)
      end do
    end if

    close(fid)

    deallocate(time,lift,drag,momentz,height,cl,cd)

    call execute_command_line("gnuplot -p // graph.dat")

end program
```

Code 6.14: Fortran90 script for post processing

The initial timesteps are skipped to improve result visualization, removing the initial oscillations during the less informative transition phase between the steady state (which initializes the solution) and the oscillating problem.
Also the satisfaction of regime conditions indicates that the simulation has converged.
Field of p, U and TKE will be analysed in *Paraview*.

6.3.5 Results of the analysis

The results for the three simulations are reported and analysed in this chapter.
The first simulation features an oscillation amplitude of 50 mm, leading to a maximum displacement of 100 mm from the starting point, at 100 mm from the ground. $\omega = 6.28$ rad/s, which implies a period of 1 s. Speaking in terms of reduced frequencies: $k = 0.21$. The high number shows the predominance of the dynamic effects with respect to the steady state ones, so a really different behaviour from static case is expected.

Cl and Cd are reported in function of time:

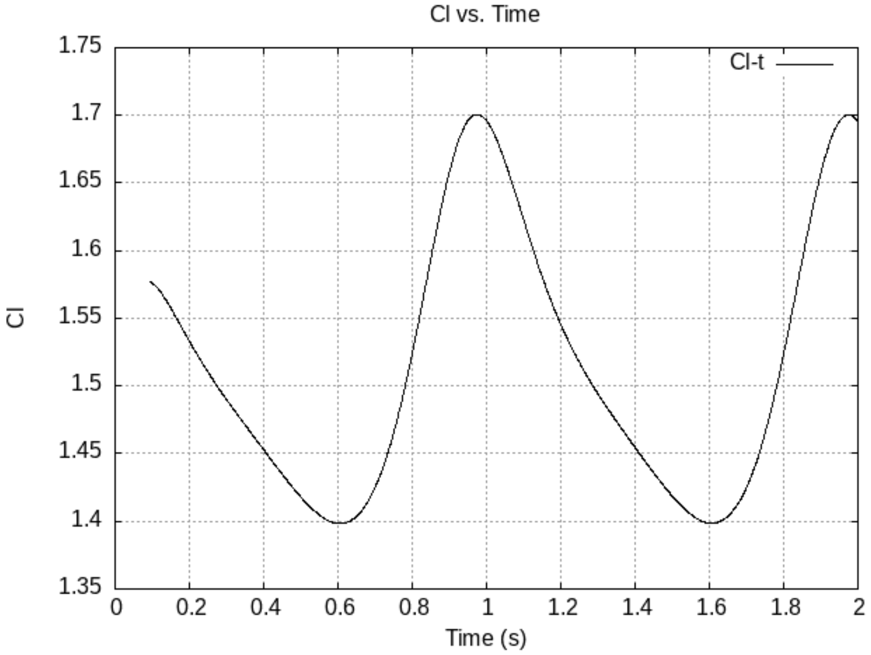


Figure 6.19: Cl vs t - first dynamic case

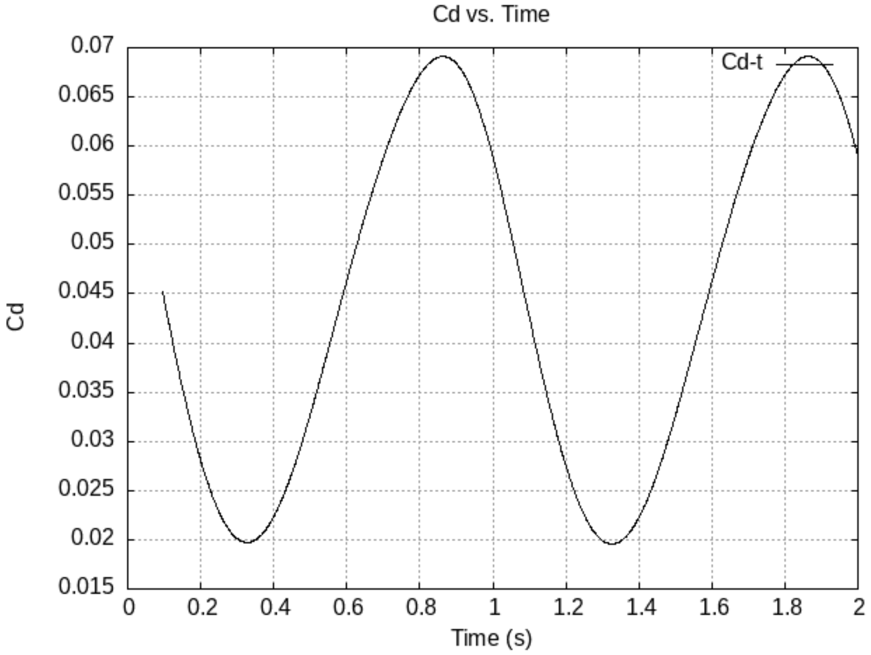


Figure 6.20: Cd vs t - first dynamic case

The force coefficients oscillate with the airfoil in time, with the Cd that assumes a quasi - cosinusoidal shape with a phase shift, similar to the theory of Theodorsen.

The Cl curve is instead different, with smaller peaks of Cl. It's immediately clear that the simulation reaches regime conditions early, with a behaviour that is different between the descending and ascending phase, noticeable from the non symmetric oscillation.
To better understand the forces variations, Cl and Cd are referred to the height from the ground:

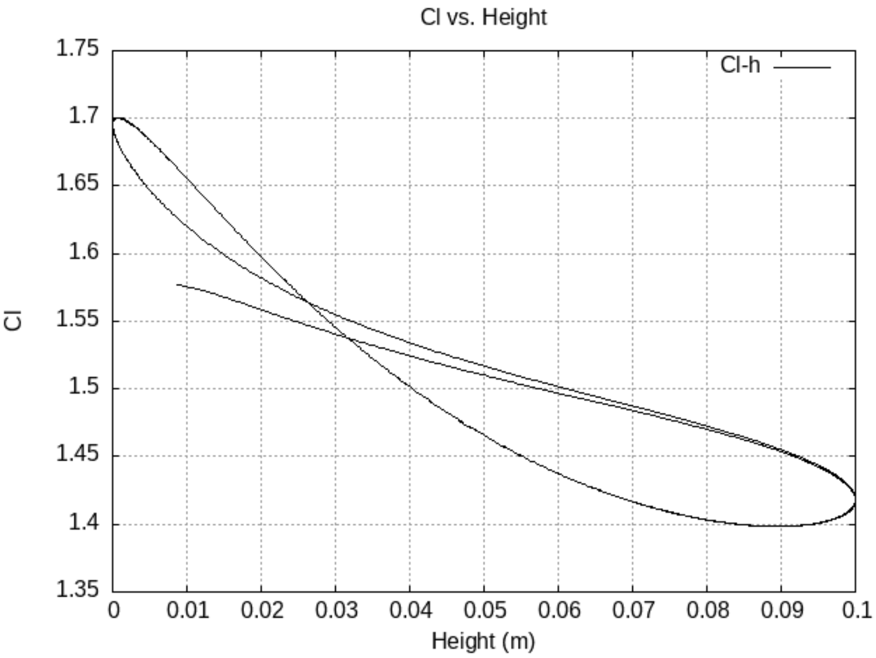


Figure 6.21: Cl vs h - first dynamic case

Cl is considerably higher that the one measured in steady state conditions. The peak is also shifted closer to the ground, with the airfoil that reaches a Cl value around 1.7 at an height close to 100 mm (that correspond to the zero in the graphs) instead of the 1.4 at 150 mm of the static case. An hysteresis behavior is also present:

- in the upward motion a loss in Cl is reported, strong for low height where the motion velocity is slow, reduced for heights higher than 130 mm, with vertical speeds of the profile that contrast the downforce loss generating an augmented relative angle of attack;
- in the downward motion the opposite behaviour happens, with the minimum in Cl reached at 190 mm during the upward motion, where both effects of relative angle of attack and ground effect are in contrast for downforce creation. At low heights the reduced angle of attack is instead beneficial, helping the profile to delay the boundary layer separation and

the partial stall of the profile, causing early downforce losses. The peak in downforce is in fact reached in this phase, with boundary layer that detaches really close to 100 mm.

- Boundary layer reattaches during the upward motion phase at higher heights than the downward motion.

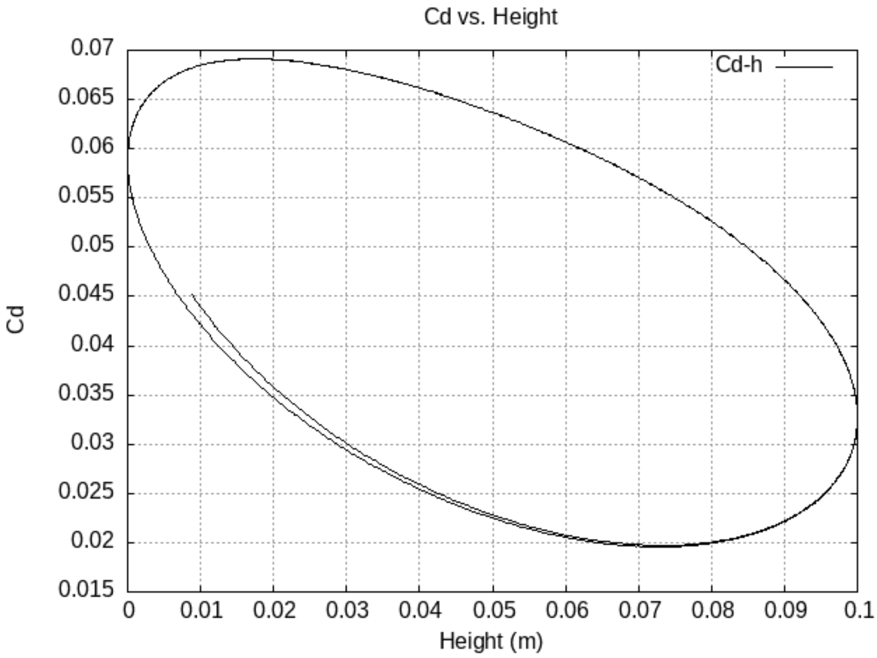


Figure 6.22: Cl vs h - first dynamic case

It's clear how the motion of the profile increases the effect of downforce gain with ground proximity, with the profile that can go really low without stalling. This is confirmed by the scenes extracted in paraview:

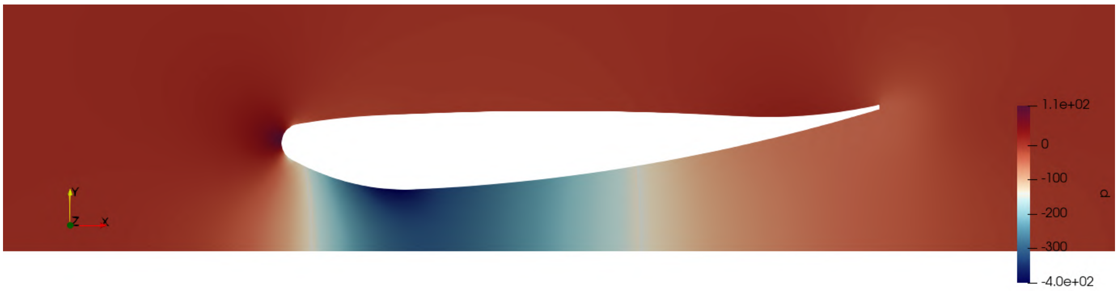


Figure 6.23: Pressure scene at peak Cl - first dynamic case

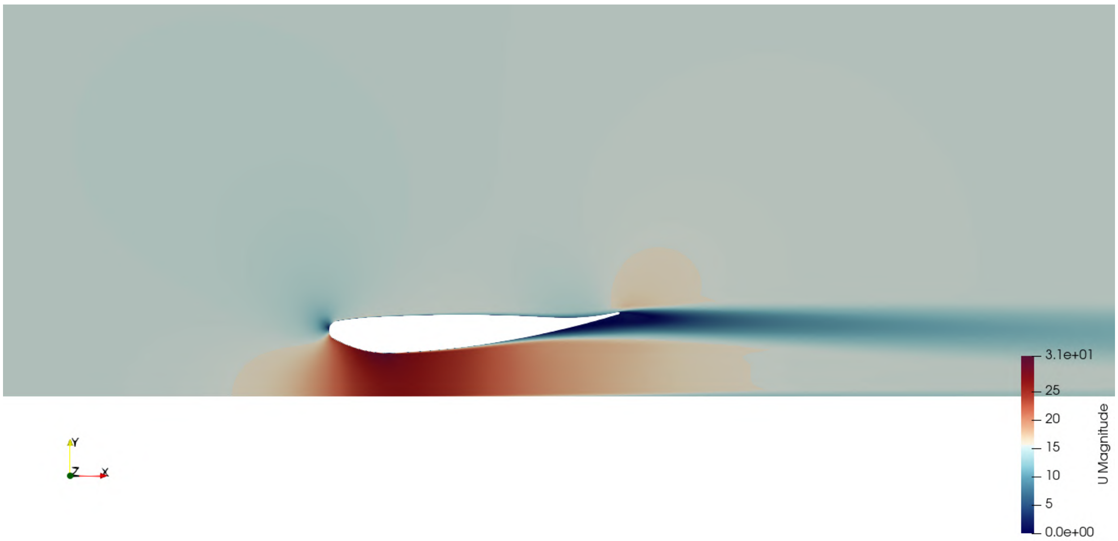


Figure 6.24: Velocity scene at 1.25 s - first dynamic case

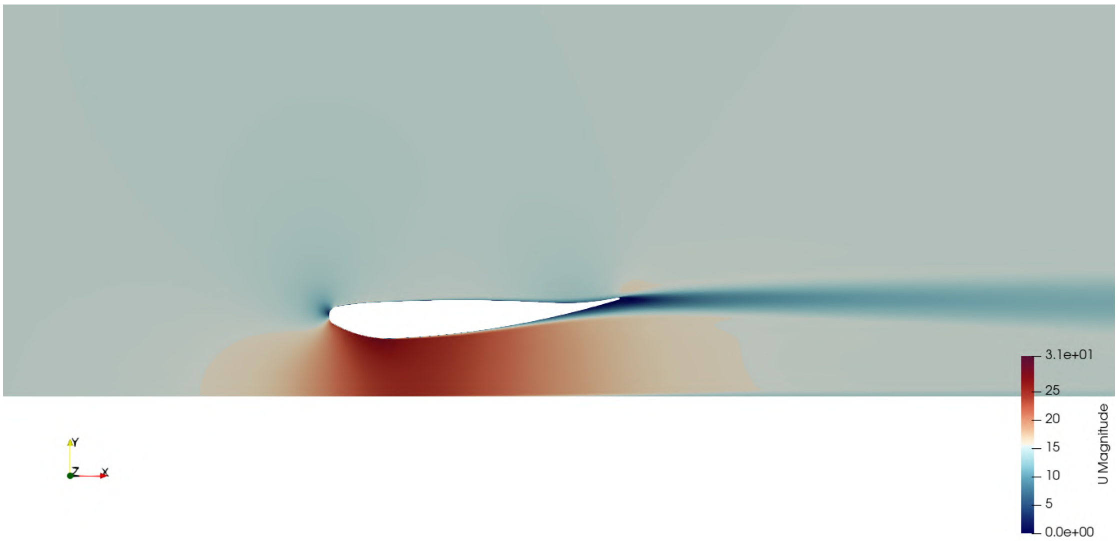


Figure 6.25: Velocity scene at 1.5 s - first dynamic case

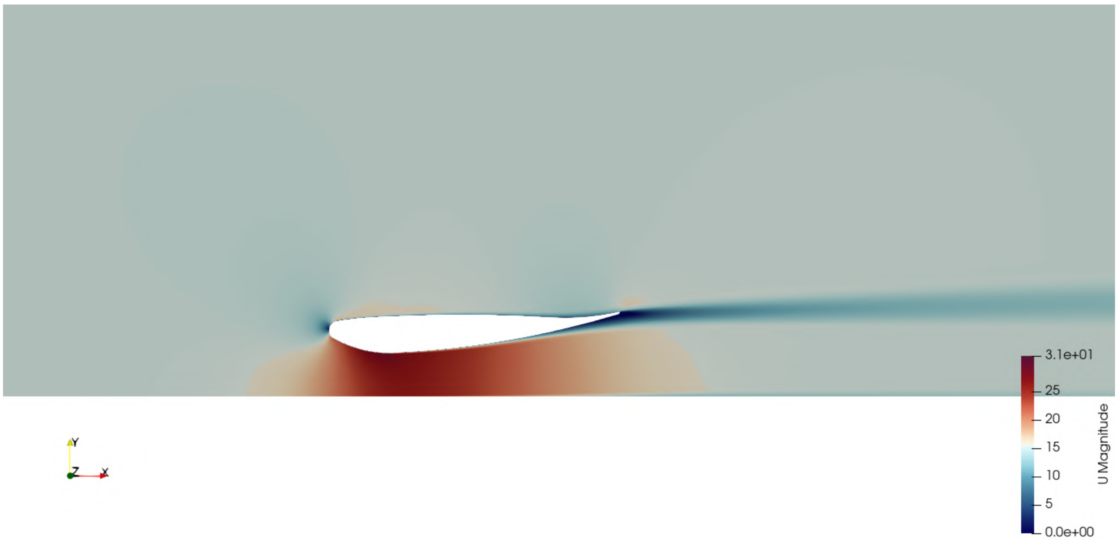


Figure 6.26: Velocity scene at 1.75 s - first dynamic case

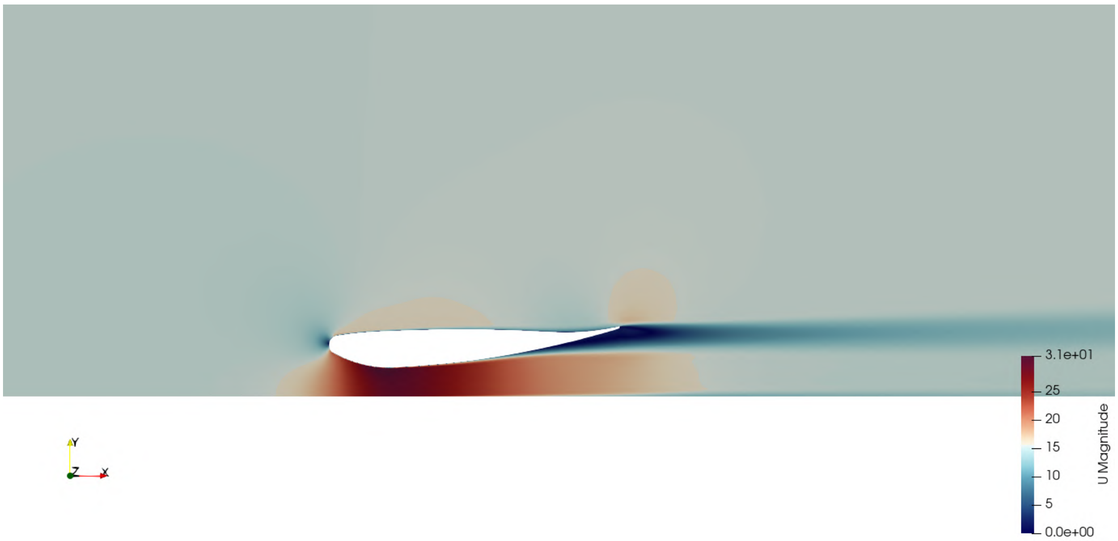


Figure 6.27: Velocity scene at 2 s - first dynamic case

This first case uses a wide range of displacement, that involves a fast motion of the airfoil. To analyse a case closer to the one of the car, the second one sets the amplitude value to 15 mm, that implies a maximum vertical displacement of 30 mm (from 100 mm to 130 mm), with the same period of 1 s.
This choice will introduce slower movements, that should bring the solution a little bit closer to the steady state case, even if k is unchanged.

Cl and Cd with respect to time:

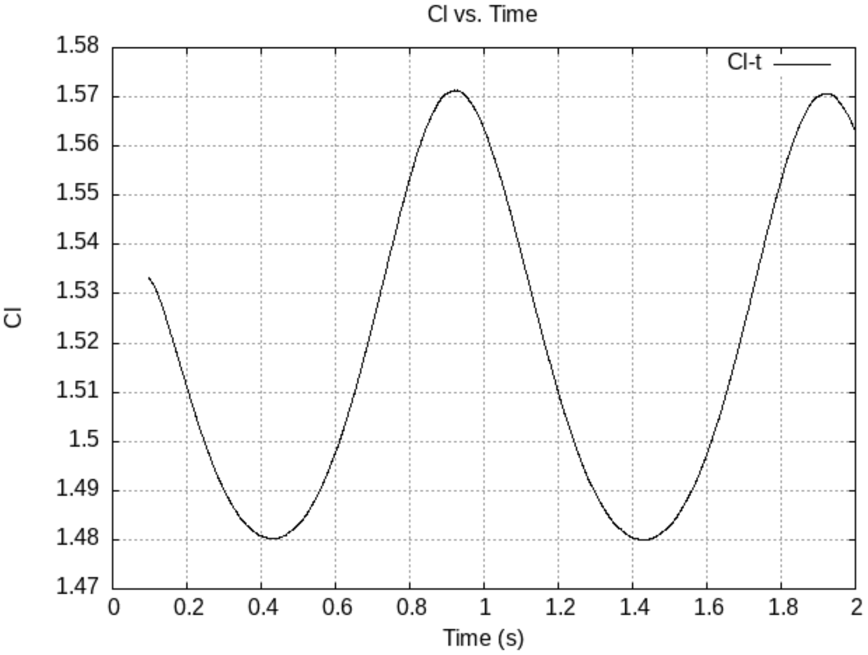


Figure 6.28: Cl vs t - second dynamic case

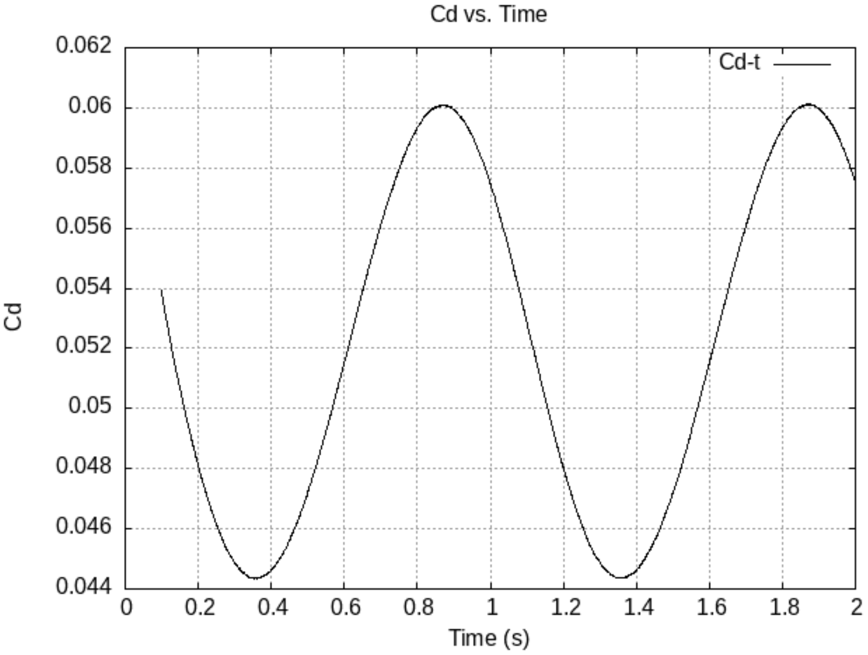


Figure 6.29: Cd vs t - second dynamic case

Curves are similar to a shifted cosinusoid. Peak in Cl is close to the minimum height (that happens at 1 s and 2 s) also in this case. Lower maximum values of Cl and Cd can be noted, while lower values are clearly higher thanks to the smaller maximum height from the ground. With the height in the x axis:

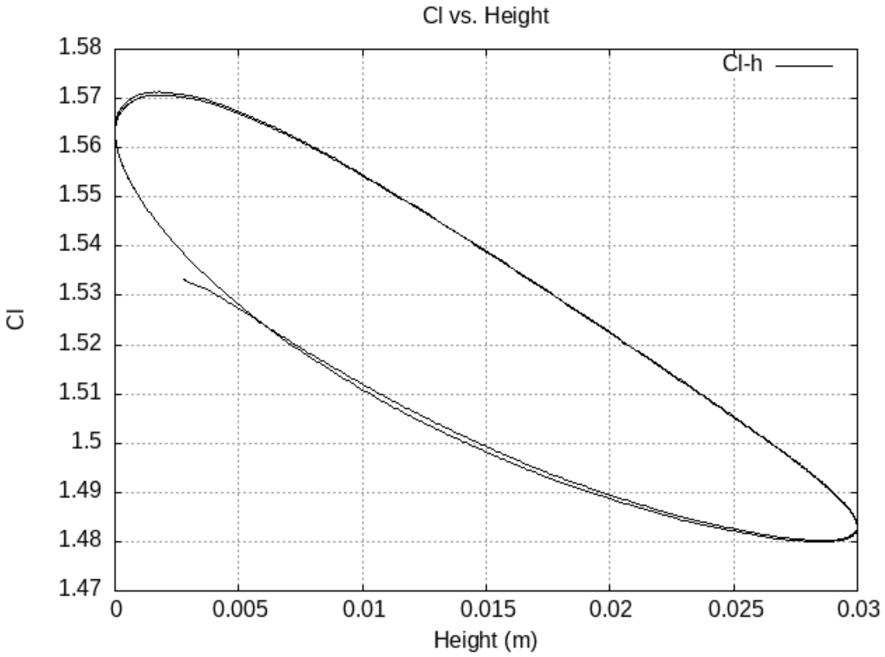


Figure 6.30: Cl vs h - second dynamic case

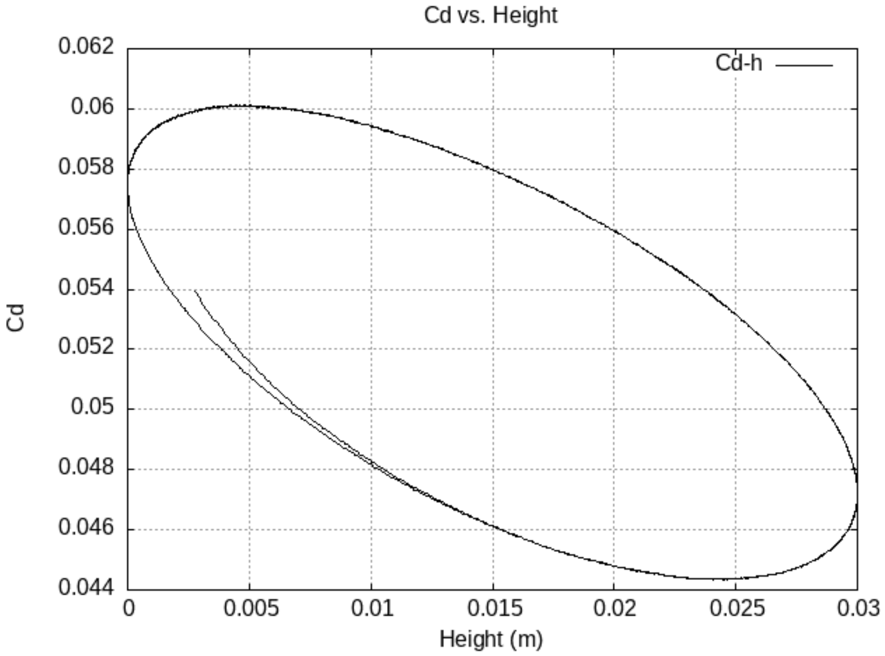


Figure 6.31: Cd vs h - second dynamic case

Hysteresis is still present and the shape is clearly different from the first case,

but the main effect is confirmed: the boundary layer reattaches at higher heights in the upward motion that it detaches at during the downward motion. The smaller range of motion doesn't allow the profile to exit the ground proximity and benefit from the relative angle of attack created by the air downwashing the profile, so the Cl - h curve isn't eight - shaped like the previous one.
Cd curve is substantially unchanged in shape, but lower values of drag coefficient are reached.
Again, scenes confirm what graphs told. A smaller pressure gradient can be noticed with the respect of the previous case. Scenes of TKE are reported for this case to better show the trend followed by the airfoil wake. Higher values of k imply bigger detachments.

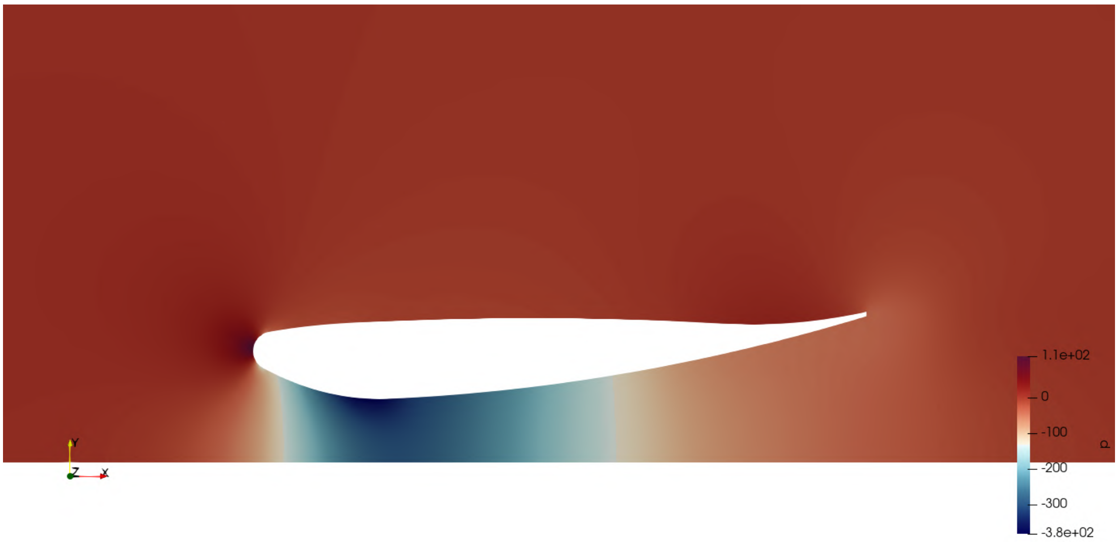


Figure 6.32: Pressure scene at peak Cl - second dynamic case

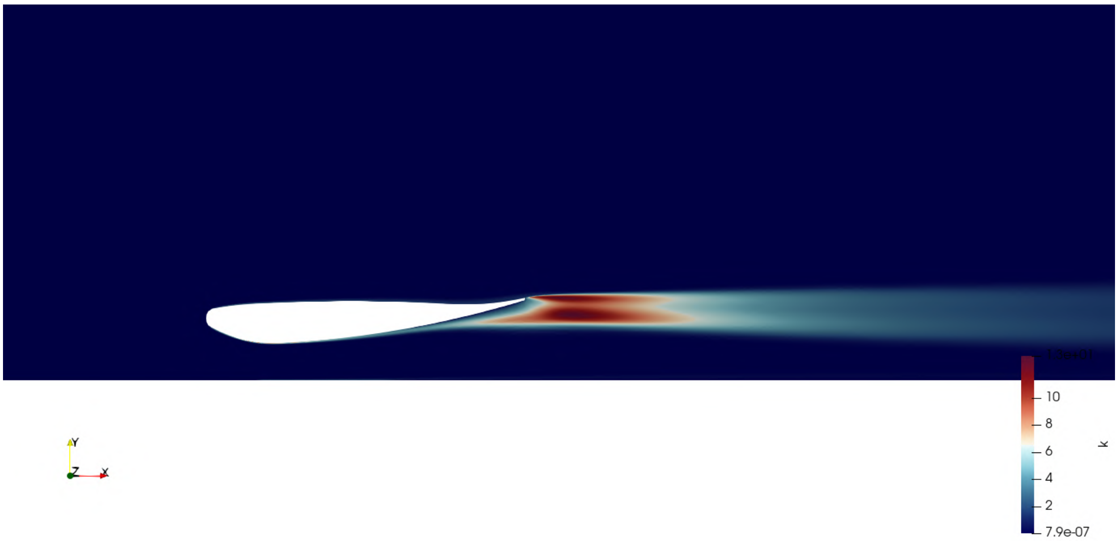


Figure 6.33: TKE scene at 1.25 s - second dynamic case

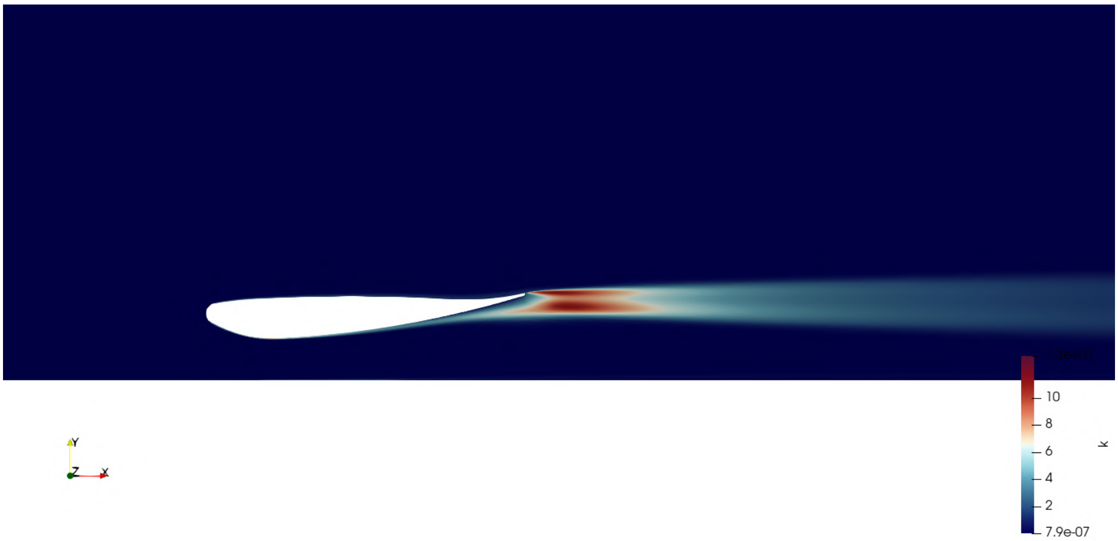


Figure 6.34: TKE scene at 1.5 s - second dynamic case

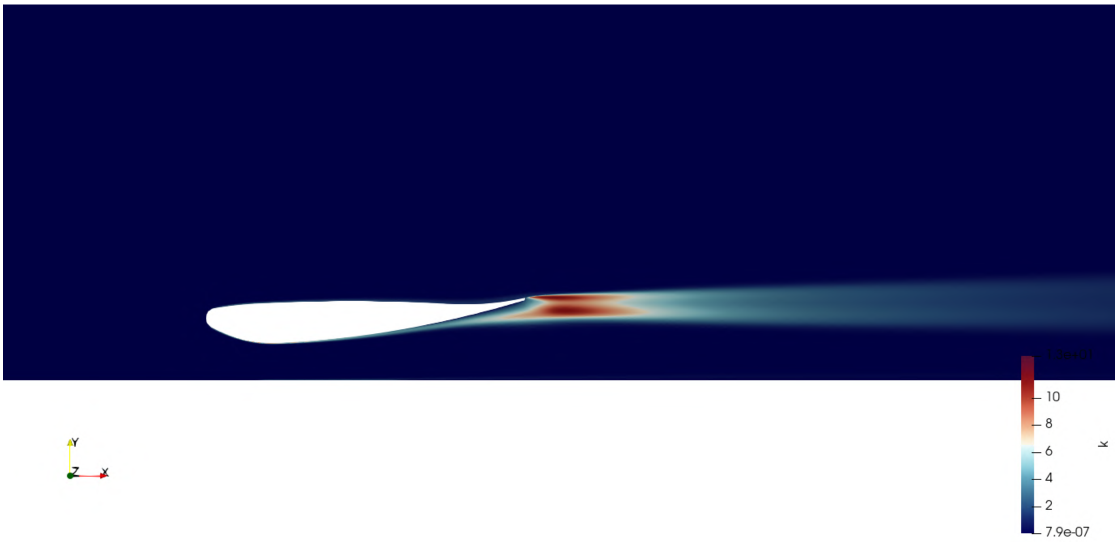


Figure 6.35: TKE scene at 1.75 s - second dynamic case

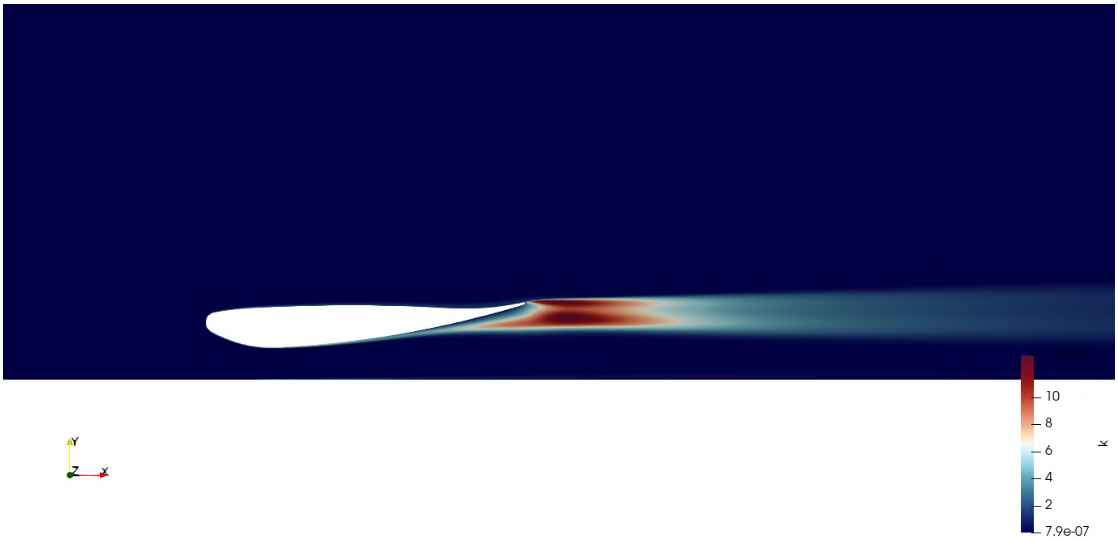


Figure 6.36: TKE scene at 2 s - second dynamic case

Force coefficients obtained from these first two simulations have higher values than the static case, confirming that a downward movement close to the ground can enhance the ground effect. The third analysis aims to obtain a behaviour closer to the steady state one making the reduced frequency k smaller, following Theodorsen theory. This is done by halving ω, doubling the period of motion. The chosen range of motion is the one of the second case, that featured a more interesting behaviour, remaining highly influenced by the ground for the entire motion.

Curves in time for this case:

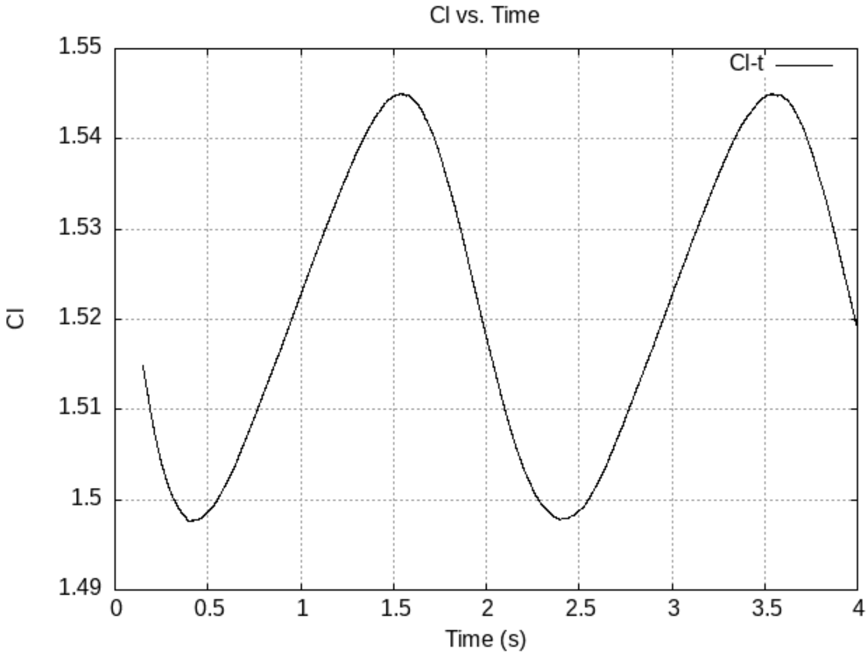


Figure 6.37: Cl vs t - third dynamic case

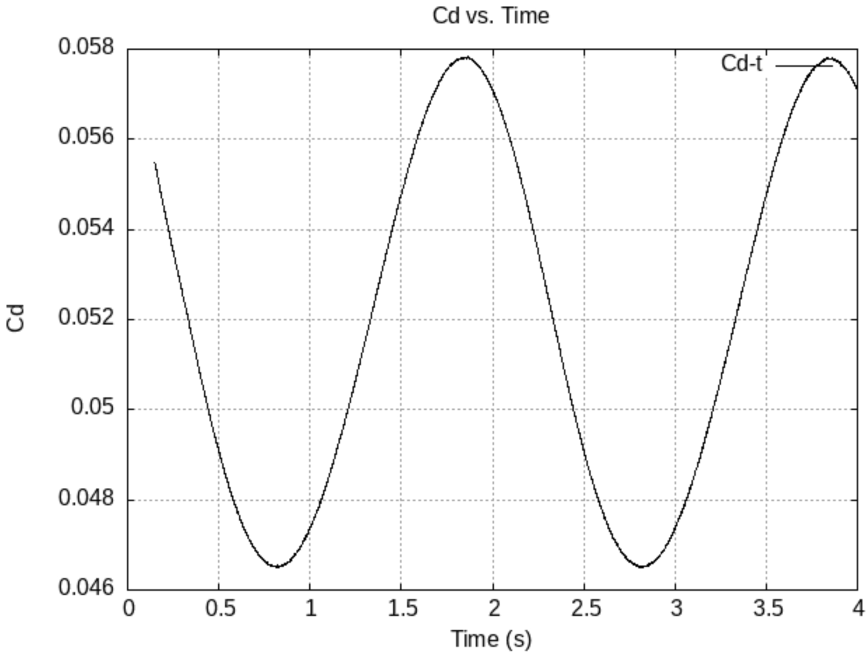


Figure 6.38: Cd vs t - third dynamic case

Curves are similar in shape to the previous case considered, but phase shift from a cosinusoidal function is here important, with maximum that happens around 0.5 s earlier than the time of 2 s that corresponds to the minimum height. Minimum value of Cl are again higher than the previous case, while the maximum Cl reached is significantly lower. With the height in the x axis:

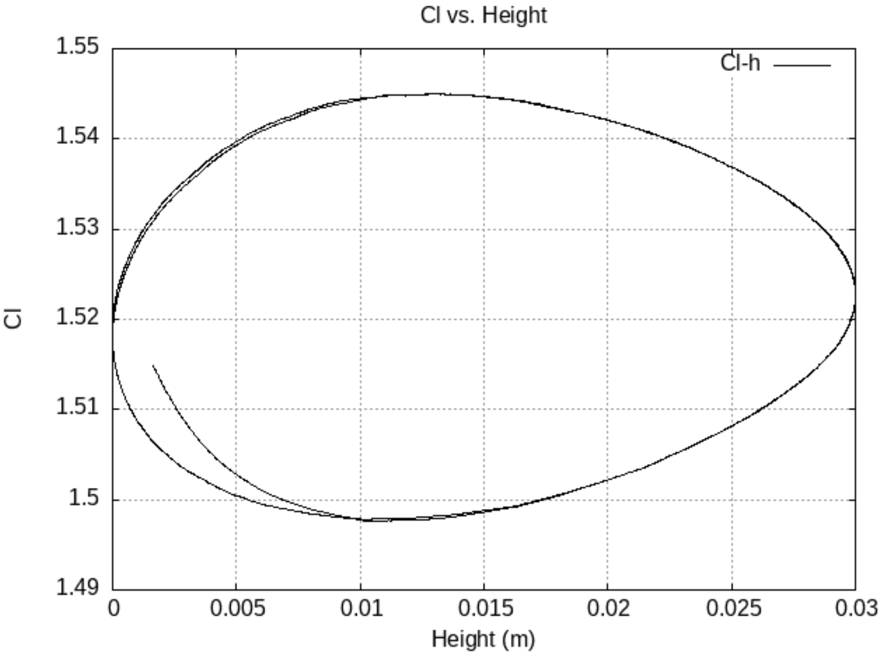


Figure 6.39: Cl vs h - third dynamic case

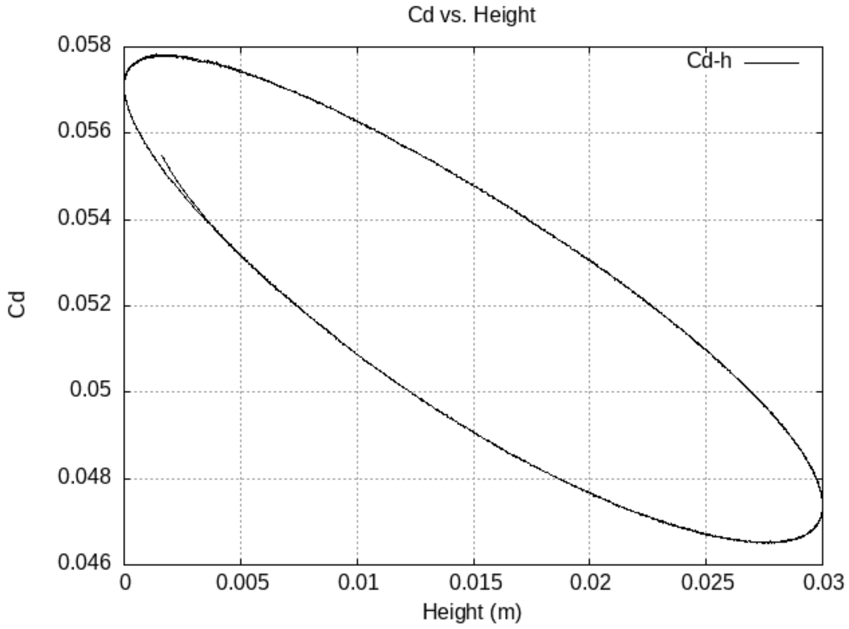


Figure 6.40: Cd vs h - third dynamic case

While the Cd behaviour was quite similar in the past two cases, here smaller variations between upward and downward motion can be found, identifying a behaviour that is getting closer to the one of the steady state case.
Cl - h curve is once again different. The maximum in downforce generation is reached at bigger heights, with the stall that is anticipated compared to the previous case, clearly caused by the slower movement, reducing the effect discussed before. In the upward motion, for the same reason, the reattachment happens earlier and downforce can increase again. At these heights the airfoil behaviour is similar to the one of the static case, this is also clear from the scenes, with pressure gradients that keeps getting closer to the ones experienced in the static case and velocity scenes that show a bigger detachment of the boundary layer for small heights from the ground.

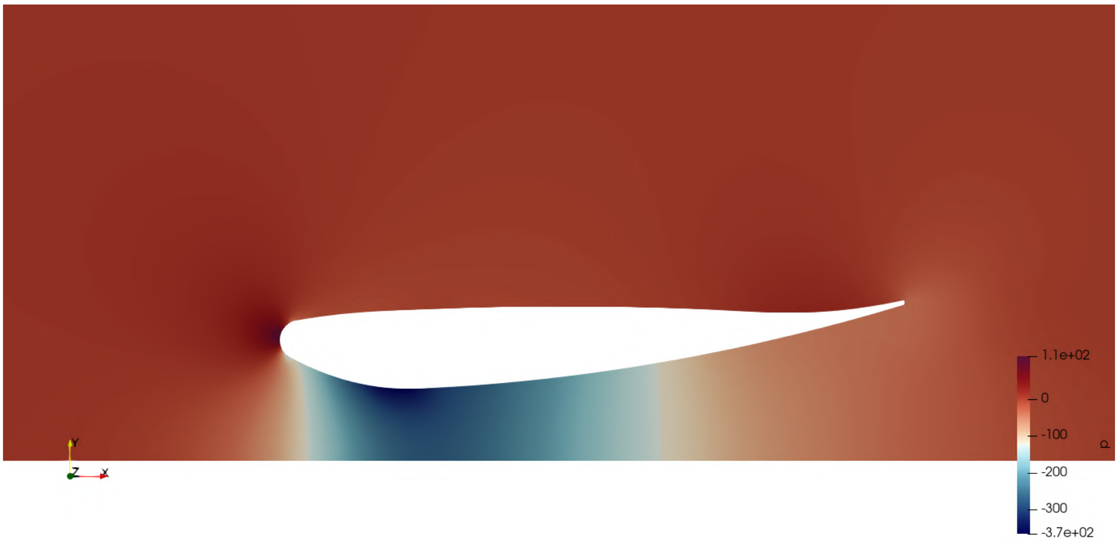


Figure 6.41: Pressure scene at peak Cl - third dynamic case

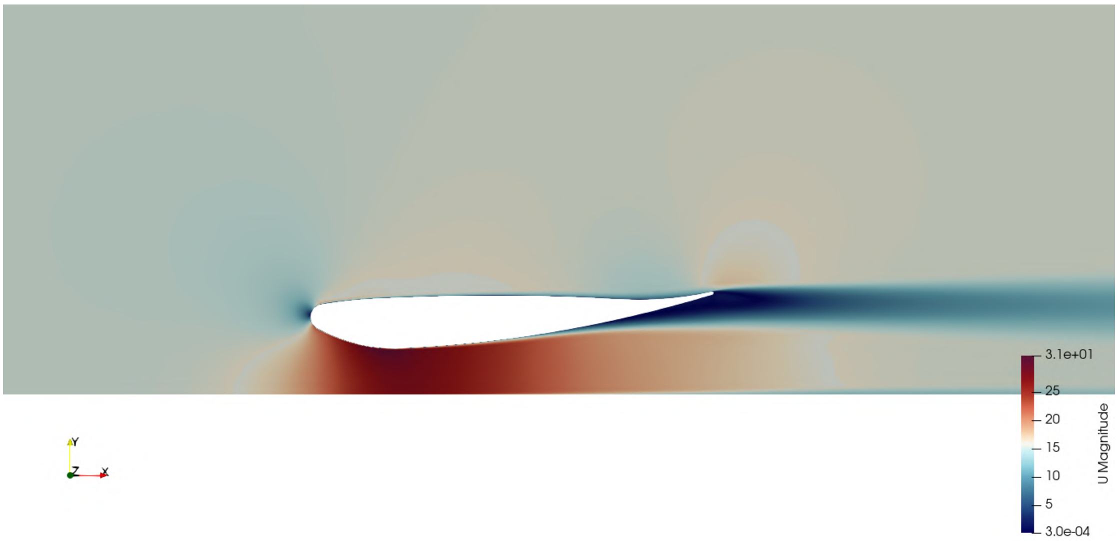


Figure 6.42: Velocity scene at 2.5 s - third dynamic case

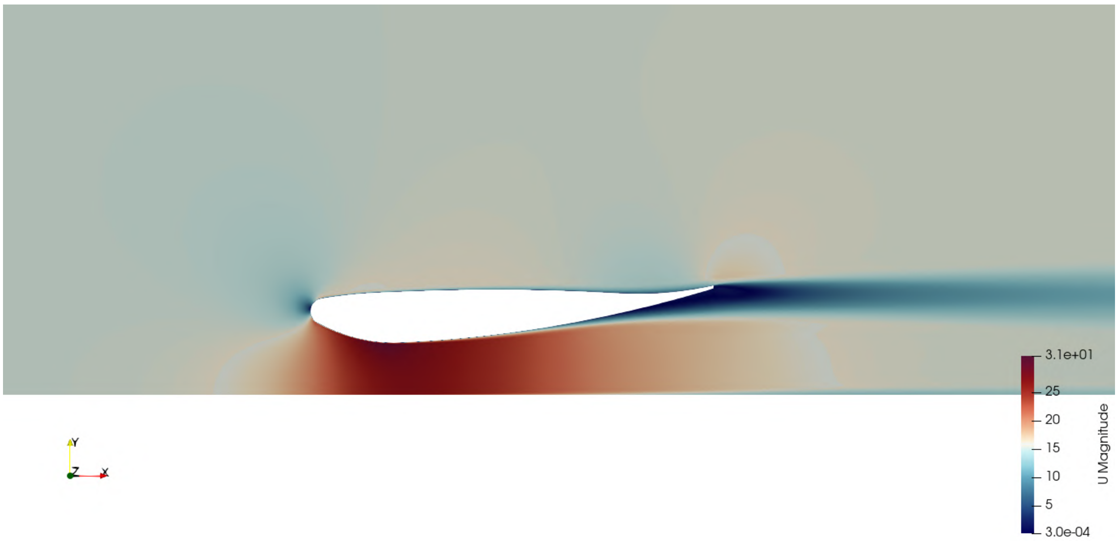


Figure 6.43: Velocity scene at 3 s - third dynamic case

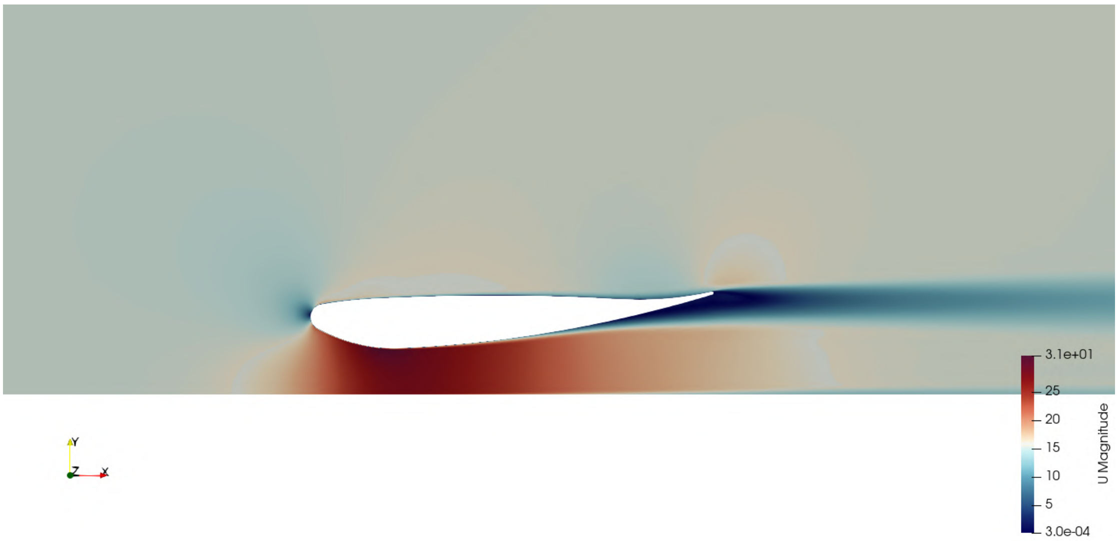


Figure 6.44: Velocity scene at 3.5 s - third dynamic case

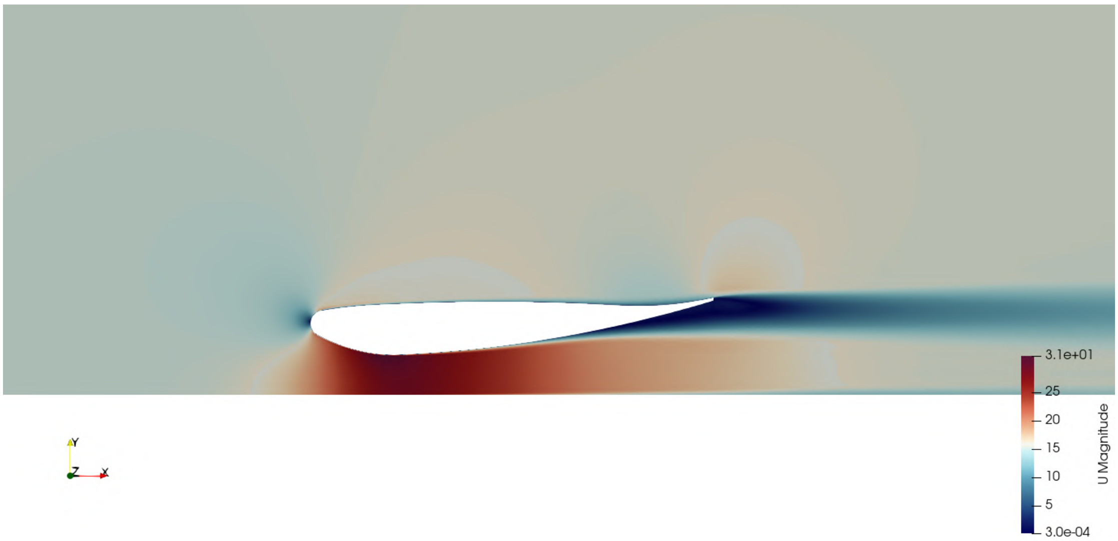


Figure 6.45: Velocity scene at 4 s - third dynamic case

7

Conclusions

7.1 Results comparative analysis

An overview of the ground effect phenomenon has been covered, studying at first the behaviour of a Formula SAE car, analysing how its aerodynamic components react to the ground proximity, then a an oscillating airfoil in ground effect has been studied, trying to discover the dynamic effect of an aerodynamic surface close to the ground.

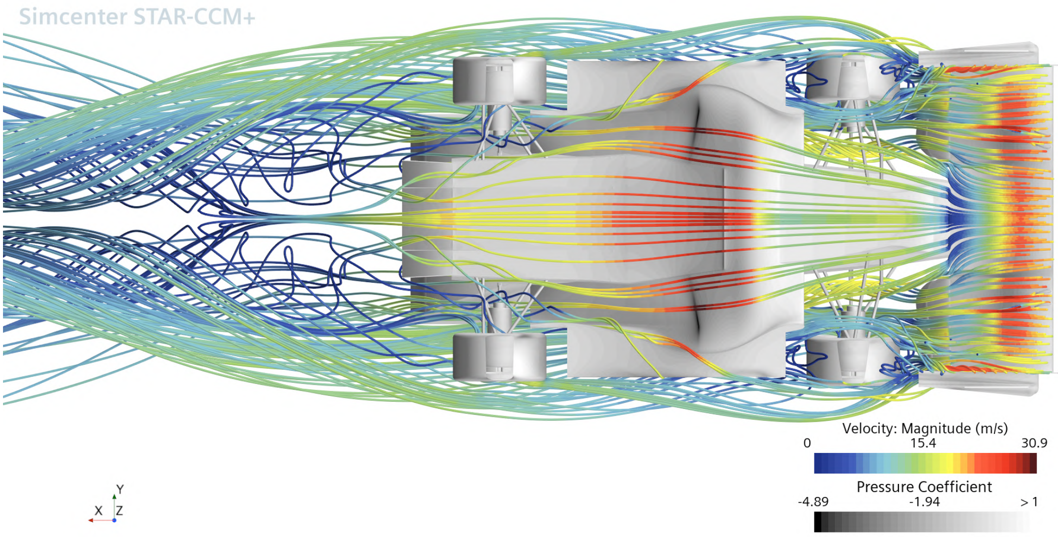


Figure 7.1: Streamlines in the undertray at 30 mm height from the ground

The comparative analysis between the static behaviors of the car and the 2D airfoil demonstrates remarkable consistency. Both exhibit analogous interactions with the ground, displaying Cl curves in alignment with established theories. Notably, parallels exist in pressure reduction at decreased heights and the consequent loss of downforce due to boundary layer detachments is for both cases present.

Acknowledged this, some evaluation of the possible behaviour of the car in dynamic conditions can be made from what has been discovered from the transient simulations in *OpenFOAM*.

It's possible to verify that in dynamic conditions ground effect is augmented, with the 2D airfoil that could reach smaller heights without losing performance, increasing the value of Cl created, depending on the frequency and amplitude of the oscillation. For a car that reaches 100 km/h in 2.7 s, characteristic oscillation periods can be reasonably low, similar to the ones considered in the cases simulated, telling that important dynamic effects can be experienced, even if the amplitude of motion is very restricted and in very few race conditions is possible to experience these fast height variations.

It's imperative to acknowledge the inherent complexity of three-dimensional car study, where not all effects can be observed by extrapolating results from simpler two-dimensional case. However, the general behaviour obtained, considering the good match between the other studies, could be considered a possible qualitative prediction of the phenomenon and good starting point for eventual further studies.

In summary, the ground effect plays a pivotal role in generating downforce for road vehicles. In specific instances, this effect can be amplified, potentially leading to higher Cl peaks due to downward vertical motions. In extreme cases this effect could cause dynamic instabilities, with car subject to high variations of the value of downforce and sudden changes in car height, creating a dangerous vicious cycle. In Formula 1 this theory could be associated as one of the responsible of the *porpoising* phenomenon, where extreme velocities and small ride heights increase the effect observed in this thesis.

7.2 Further studies

To achieve better results ensuring the actual influence in downforce variations of dynamic height oscillations, more computationally demanding analysis should be done.

The inherently unsteady nature of the problem emphasizes the need for heightened accuracy in predicting boundary layer detachment and wake behavior. These aspects have been identified as pivotal factors in comprehending and effectively addressing this phenomenon, so higher fidelity LES simulations should be ran, exiting from the 2D domain.

As said this thesis had the purpose of giving an overview of the phenomenon, so oscillations analysed derived directly from the 2D steady state validated case. To better comprehend if the car effectively experience similar oscillation frequencies, discovering how strong is the effect applied, further analysis of the car behaviour in motion should be done.

References

[1] John D. Anderson. “Fundamentals of Aerodynamics”. In:

[2] H. Aref S. Balachandar. “A First Course in Computational Fluid Dynamics”. In: 2017.

[3] Jan R. Wright Jonathan E. Cooper. “Introduction to Aircraft Aeroelasticity and Loads”. In: 2015.

[4] Jonathan David and Chaim Zerihan. “An Investigation into the Aerodynamics of Wings in Ground Effect”. In: 2001.

[5] Ira H. Abbott Albert E. Von Doenhoff. “Theory of Wing Sections”. In: 1959.

[6] Formula Student Germany. “Formula Student Rules 2023”. In: 2023.

[7] Thomas D. Gillespie. “Fundamentals of Vehicle Dynamics”. In: 1992.

[8] Giorgio Graziani. “Aerodinamica”. In: 2010.

[9] Stephen Alexander Mahon. “The Aerodynamics of Multi - Element Wings in Ground Effect”. In: 2005.

[10] Juan Molina and Xin Zhang. “Aerodynamics of a Heaving Airfoil in Ground Effect”. In: 2011.

[11] Dewey H. Hodges G. Alvin Pierce. “Introduction to Structural Dynamics and Aeroelasticity”. In: 2002.

[12] Stephen B. Pope. “Turbulent Flows”. In: 2000.

[13] Kirill V. Rozhdestvensky. “Aerodynamics of a Lifting System in Extreme Ground Effect”. In: 2000.

[14] Kirill V. Rozhdestvensky. “Wing - in - ground effect vehicles”. In: 2006.

[15] Claudio Santarelli and Léon Reketat. “Best Practice Guidelines for Formula Student Cars”. In: 2019.

[16] Shuai Ye Yufei Lin Liyang Xu and Jiaming Wu. “Improving Initial Guess for the Iterative Solution of Linear Equation Systems in Incompressible Flow”. In: 2020.

Ringraziamenti

Desidero ringraziare tutti coloro che mi hanno sostenuto nel mio percorso accademico. In primo luogo, la mia preziosa famiglia per il costante supporto e l'incoraggiamento che mi hanno donato lungo questa strada. Un sentito ringraziamento va anche agli amici e a tutte le persone che, in modi diversi, hanno contribuito al mio apprendimento e alla mia crescita, consentendomi di raggiungere questo importante traguardo.
Ringraziamenti speciali vanno al Prof. Federico Dalla Barba e al Prof. Francesco Picano, che mi hanno dato l'opportunità di trattare questo argomento come tesi magistrale.