UNIVERSITY OF ROME

TOR VERGATA

TOR VERGATA
UNIVERSITY OF ROME

FACULTY OF ENGINEERING

MASTER'S DEGREE THESIS IN ENERGY ENGINEERING

ADVANCED METHODS FOR AUTOMATIC SHAPE OPTIMIZATION OF ROAD VEHICLES DRIVEN BY RBF MESH MORPHING

Advisor:
Prof. Ing. Marco Evangelos Biancolini

Candidate:
Daniele Patrizi

Industrial advisor:
Ing. Torbjörn Virdung

Co-advisors:
Ing. Ubaldo Cella
Ing. Stefano Porziani

Academic year 2020/2021

Table of contents

ABSTRACT

In the analysis and design of vehicles, numerical simulation allows to investigate a wide range of solutions reducing the time and costs related to the prototyping of new models and their experimentation. In recent years, the interest of manufacturers and designers in an automated optimization workflow has grown. In industrial practices, however, it is not uncommon to manually carry out the optimization, making the whole process of design exploration slower and more laborious. This is because multiple programs and tools are usually required, and integrating them into a single, seamless workflow is not always easy; moreover, when adopting CAD-based geometric parametrization, geometry coherence must be preserved and the problem of re-meshing noise due to CAD reconstruction has to be faced. In recent years, advances in modern computer performance, as well as the availability of massively affordable processing capacity, have enabled ever-more precise analysis and forecasting of physical phenomena. In this work an automatic, file-based design exploration workflow built on the synergic use of a high-fidelity CFD solver and the meshless commercial morpher RBF Morph is presented. First, the studied geometry is parametrized employing the RBF Morph ACT Extension within the graphical user interface of ANSYS Mechanical; then, the parametric space is explored using an Optimal Space-Filling algorithm, and the morphing files (referring to each individual shape variant and each comprising the coordinates and relative displacements of appropriate source points used to map the geometry) are generated. These files containing the morphing instructions, together with the coordinates of the baseline mesh, are fed to a C++ code that processes the information contained therein, makes calls to the RBF libraries, and outputs the coordinates corresponding to the deformed geometry. Subsequently, through the use of UDF (*User-Defined Functions*) compiled inside a DLL (*Dynamic Linked Library*), the coordinates of the vertices of the mesh generated within the STAR-CCM+ commercial solver are updated and, following the set-up of the simulation file, the CFD analysis on the new shape is launched. At the end of the simulation, all the most relevant results of the analysis are automatically exported. The entire workflow described above is summarized within a MS-DOS script file, which, when launched from the command-line interface, initiates each task of the procedure. The proposed method effectiveness is proved with two examples: a drag reduction analysis on the ASMO (*Aerodynamics Studien Model*) idealized car body shape and a case study of a Volvo side-view mirror.

1. INTRODUCTION

Analyzing and solving problems involving fluid dynamics is an activity of extreme interest in various fields of engineering and other disciplines, such as the aerodynamics of airfoils, the hydrodynamics of naval vehicles, meteorology, the automotive sector, bioengineering, turbomachinery, and internal combustion engines. The equations that govern the physics of these phenomena are still the subject of study in many areas of research and are characterized by considerable mathematical complexity. Many problems remain in fact only partially analytically solvable and are often, therefore, treated with mainly numerical approaches, typically supported by IT tools with enormous demands for computing power. *Computational Fluid Dynamic* (CFD) is a branch of fluid mechanics that uses numerical techniques performed with the help of supercomputers for the solution of complex problems concerning the motion of fluids and constitutes a calculation technique that is progressively acquiring the role of vital component in the design process of industrial products. The main reason why the development of CFD has proceeded at a slower pace compared to other CAE (*Computer-Aided Engineering*) tools is mainly due to the enormous complexity of the phenomena involved, which precludes the possibility of obtaining a description that is at the same time inexpensive and sufficiently complete. The availability of reliable computer systems with higher computational capabilities and the introduction of more intuitive *graphical user interfaces* (GUI) have contributed to the recent amplification of interest in this discipline, and CFD has become part of the common design practice in the industrial sector. Clearly, the investment costs related to the technology needed to support the onerous CFD calculations are far from negligible, but the overall expenses to be sustained generally remain lower than those associated with an advanced experimental system. In fact, in a fluid systems design context, CFD offers numerous other advantages over an experimental campaign-based approach, including: a substantial reduction in the time and cost of producing new designs, the ability to study systems for which a controlled experiment environment is impossible or extremely difficult to obtain (as in the case of large and complex systems), the ability to study systems at the limits of their normal applications (such as in safety and accident situations), the possibility of analyzing a problem and obtaining a result with a theoretically unlimited level of detail. A CFD code can perform a simulation of an enormously articulated scenario, can produce results at virtually no additional costs, and allows to perform parametric studies in a relatively economical way making this approach always preferable in

the execution of optimization studies. The feasibility of a solution with a numerical approach to a fluid dynamic or aerodynamic problem is justified by recent advances in technology and information technology in aspects related to the computing capabilities of modern computers. *Moor's Law* is based on the observation that the number of transistors on an integrated circuit doubles approximately every two years; however, what impacts our lives is not the structure of these computers, but rather their capacity. *Fig. 1.1* shows how the computational capacity of computers has increased exponentially through the years, doubling approximately every 18 months.

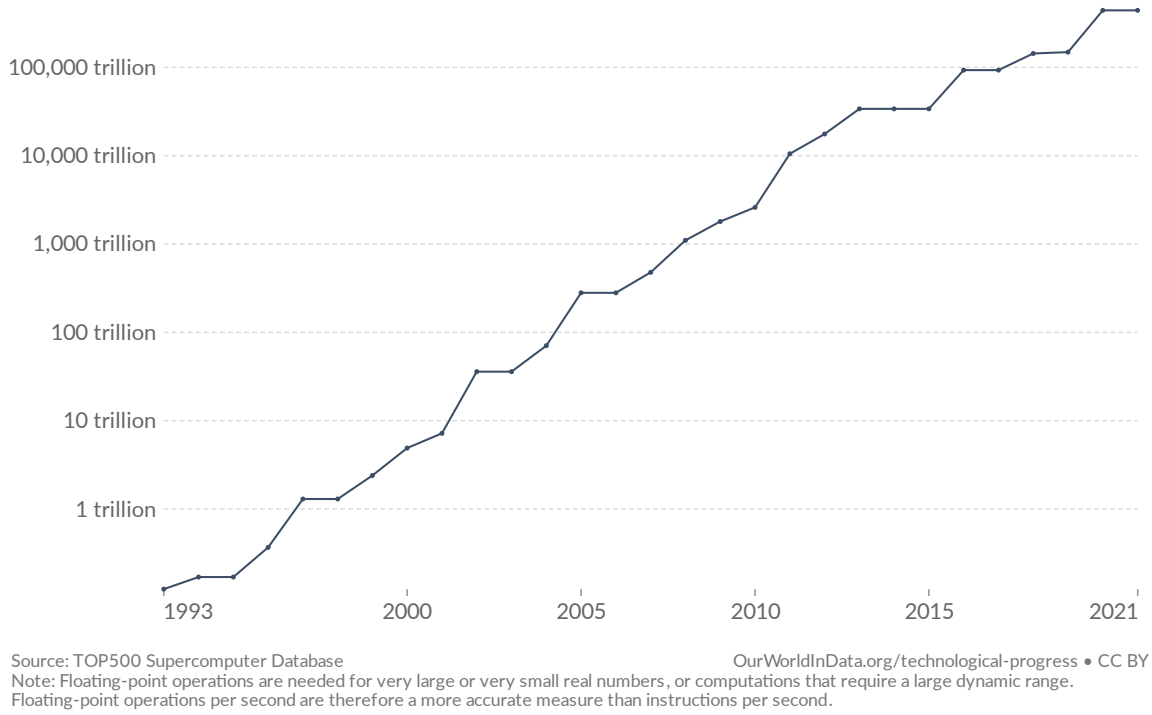

Figure 1.1: the computational capacity of the largest supercomputers in the world for any given year, based on the number of trillions (10^{12}) of 64-bit floating-point operations (FLOPS) carried out per second [1].

In this chart, the growth of supercomputer power is measured in terms of the number of floating-point operations carried out per second (FLOPS) by the largest supercomputer in any given year. FLOPS is a unit of measure for the numerical computing performance of a computer. FLOPS on an HPC-system (*High Performance Computing system*) can be calculated using this equation:

$$FLOPS = racks \cdot \frac{nodes}{rack} \cdot \frac{sockets}{node} \cdot \frac{cores}{socket} \cdot \frac{cycles}{second} \cdot \frac{FLOPs}{cycle} \qquad (1.1)$$

Floating-point operations are typically used in fields such as scientific computational research. A diverse range of technological measures such as processing speed, product price and memory capacity have also been progressing exponentially.

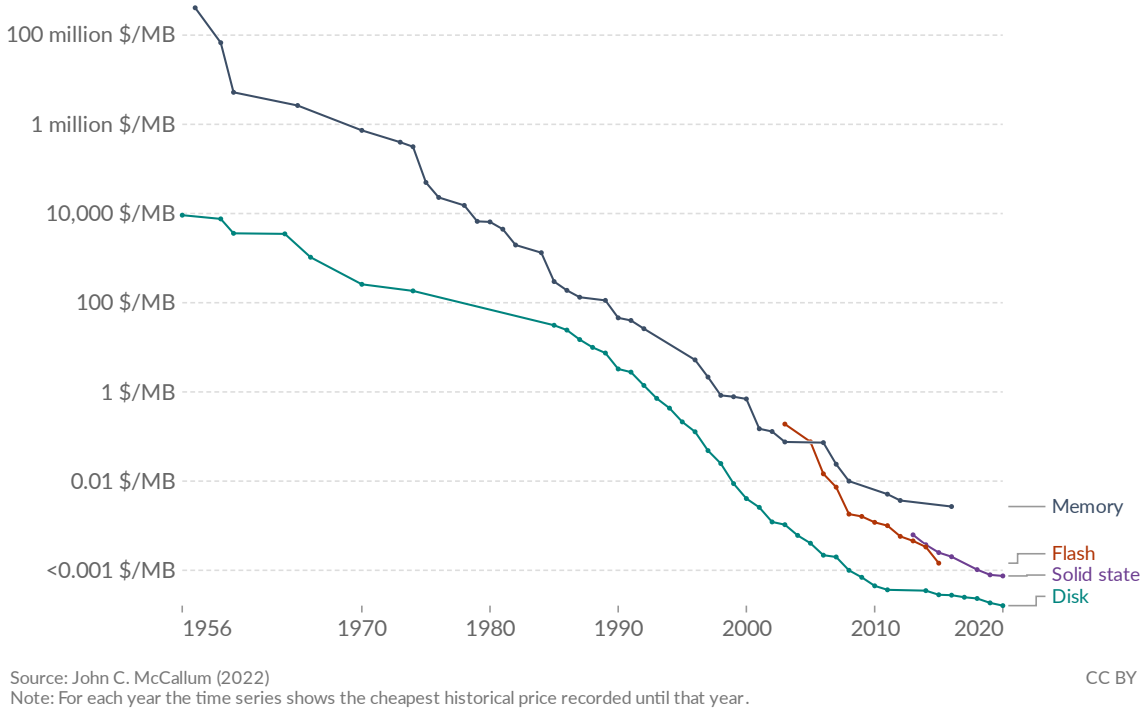

Figure 1.2: Historical cost of computer memory and storage, measured in US$ per megabyte *[1]*.

A huge sector of the market in which CFD analysis techniques are used is the automotive industry. Vehicles are still mostly powered by fossil fuels, although their availability is increasingly limited. The use of fossil fuels also causes significant environmental damage such as pollution, ozone depletion and global warming. As part of the current renewable energy transition, which aims to the gradual reduction of the use and production of fossil fuels and to the consequent replacement with new renewable energy sources, the phase-out of fossil fuels has begun. Current efforts in fossil fuel phase-out involve replacing it with sustainable energy sources in sectors such as transport and heating [2]. The automotive industry is working to introduce electric vehicles to adapt to current and upcoming restrictions [3] and electric car sales reached a record 3 million in 2020, up 40% from 2019 [4]. Considering the thrust produced by international policies to reduce global pollutant emissions, with the aim of reaching net zero by 2050, it is expected that by the next ten years about 300 million electric vehicles will circulate on the roads, accounting for more than 60% of new car sales.

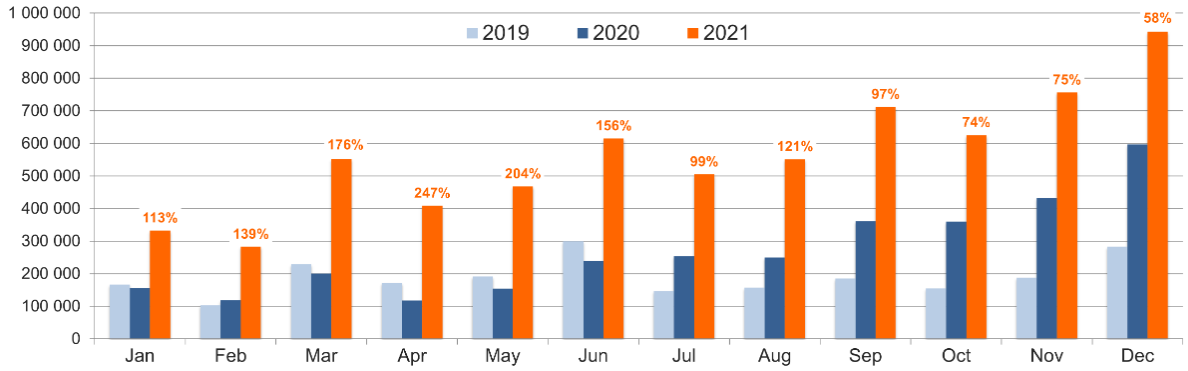

Figure 1.3: global monthly plug-in vehicles sales & year-on-year growth *[5]*.

As a result of the enormous growth of the electric vehicle market around the world, a disruptive use of increasingly advanced technologies on board is also developing. Autonomous vehicles (AVs) are expected to be designed as ACES (*Autonomous, Connected, Electrified, and Shared*). For AVs to infiltrate the streets by the next few years, consumers note the need for faster-charging infrastructure, battery durability, longer range, safe and regulated technology, and protection of personal and vehicle data. Therefore, because autonomous driving technology integrates well with electric motors, systems and equipment, electric vehicles are contributing very effectively to the widespread advancement and adoption of autonomous vehicles; the main reasons that justify this trend are attributable to the fact that, generally, buyers of this technology seek both characteristics in the same product, but also to the various advantages that the presence of an all-electric system can entail, including: the relative ease of integration and implementation of autonomous driving features (thanks to the reduced presence of moving parts of the vehicle), the extension of ranges (notoriously a critical aspect of electric vehicles) is facilitated by the presence of an efficient autonomous driving system, and finally the greater compatibility and flexibility that electronic control systems offer. Nevertheless, a big breakthrough in battery technology to offer plenty of range and autonomous functions is needed to achieve pure electric AVs. Additionally, other factors that would allow for and support AVs are still in the gray area: the regulatory environment remains murky, liability guidelines are undetermined, social acceptance is still questionable, road and highway infrastructures for many countries are not ready, to name a few. Most consumers will prefer to buy vehicles that employ technology to assist drivers rather than completely autonomous cars for at least the next decade. Examples include automated emergency braking and assistive parking. The willingness of consumers to pay for features based on assistive technologies like radar and computer vision will determine their deployment, and most are not willing to pay much more than they currently do. However, as volume increases, costs will be reduced. If automakers want to preserve their

high safety ratings, they'll have to install more advanced safety measures. Other functions, such as advanced cruise control, lane-departure warnings and cross traffic alerts, will rely on the same fundamental hardware and may be enabled through software upgrades.

Figure 1.4: Volvo side-view mirror warning light indicating the presence of another car on the changing lane.

Since the 2010s, multiple automatic technology features have been developed to assist the driver. Blind spots while utilizing side-view mirrors became more problematic on passenger vehicles as wider pillars became more common due to safety legislation relevant to rollover risks. Already in 2007 Volvo developed its *Blind Spot Information System* and included it on its S80 sedan, which created a visible alert for the driver when changing lanes occupied by other cars. Ford was Volvo's parent company at the time and started using the same system for all of its brands. Mazda used a similar system in 2008 for its Mazda CX-9 model and in 2013 started making it more available through its range, and other companies followed suit. Blind spot monitoring systems are now also used for rear-cross-traffic safety systems.

It is evident that all driver assistance technologies mounted on board a vehicle maintain an effective and reliable operation over time only if their integrity is preserved and if the environmental conditions in which they operate are controlled. This is especially true when considering the efficiency of the visibility sensors mounted outside the vehicle casing, as these are openly exposed to the surrounding weather conditions [6].

In autonomous driving, cameras are an important part of the sensor package. Surround-view cameras are directly exposed to the outside environment and are vulnerable to get soiled. When compared to other sensors, cameras have a substantially higher rate of performance

deterioration due to soiling. As a result, accurately detecting soiling on cameras is critical, especially if taken into consideration the increasing number of circulating AVs.

This work provides an advanced numerical method applied to the automotive sector for finding optimal design solutions. The procedure that will be illustrated in the following chapters aims to automate the geometry shape optimization process of road vehicles, evaluating the aerodynamic behavior of a large number of variants of car body shape (or details of it) with the use of high-fidelity CFD models. The shape variants studied were generated using the functionality of the commercial morpher RBF Morph implemented within the modular architecture of ANSYS Workbench through an add-on logic with ACT (*Application Customization Tool*) technology. The RBF Morph ACT extension manages the task of updating the calculation meshes built on the studied geometries using an ANSYS Mechanical integrated interface. Each shape variant is then processed by C++ code, which contains calls to the RBF Morph function library and whose output is the new coordinates of the morphed geometry. These coordinates are then transferred to the STAR-CCM+ commercial solver through UDF (*User-Defined Functions*) contained in a dynamic linked library. After updating the coordinates of the calculation mesh, the solver is ready to launch the CFD simulation on the morphed geometry. All these steps are contained inside a routine described by a single batch file in MS-DOS, able to automate the entire parametric analysis through a series of commands to be executed by the command-line interpreter in a Microsoft Windows environment. The implemented workflow was tested on two, mostly demonstrative, technical applications. The first application consists in the aerodynamic optimization of the publicly available ASMO (*Aerodynamics Studien Model*) idealized car body shape, aimed at the reduction of the drag coefficient (C_D). The second application, in collaboration with Volvo Cars and RBF Morph, consists of a design exploration of a detail located around the lens of a camera placed below the side-view mirror of a Volvo vehicle. The analysis is meant to minimizing the fluid film layer thickness deposited on the camera in adverse weather and soiling conditions.

2. FUNDAMENTALS OF FLUID MECHANICS

Fluid mechanics is the branch of physics that deals with studying the properties of fluids. One can distinguish between fluid statics, which is the study of fluids at rest, and fluid dynamics, which is the study of fluids in the state of motion. The solution to a fluid dynamic problem generally involves the calculation of various properties of the fluid such as velocity, pressure, density, and temperature expressed as functions of space and time. Fluid mechanics, especially fluid dynamics, is a particularly active and mathematically complex field of research. Many problems are only partially solved and, more frequently, are addressed with numerical methods, typically with the use of computer resources. Computational fluid dynamics (CFD) is a modern discipline aimed precisely at these aspects, using a numerical approach for the solution of complex fluid dynamic problems.

When analyzing phenomena involving fluid dynamics, the focus is mainly on what happens at macroscopic scales rather than microscopic ones. In addition, the continuity hypothesis is assumed for the fluid, establishing that the properties of the flow are defined at each point of space. With this assumption the behavior of the flow can be categorized as *Newtonian* and *non-Newtonian*: a *Newtonian* fluid is characterized by exhibiting, by means of the dynamic viscosity coefficient of the fluid, a direct proportionality between the strain stress exerted by the fluid and the velocity gradient perpendicular to the direction of the deformation.

$$\vec{\tau} = \mu \frac{d\vec{u}}{dy} \tag{2.1}$$

In contrast, for a *non-Newtonian* fluid the previous relation is no longer valid since the proportionality between the two terms is not linear. Other fluid classification criteria distinguish between single-phase and multiphase flow, stationary and transient, ideal (inviscid) or real (viscous), compressible or incompressible, laminar or turbulent.

Mathematically, flows can be classified according to the nature of the differential equations that describe them. Fluids, as will be explored later, are governed by the *Navier-Stokes equations*, which are highly nonlinear second-order partial differential equations. Nonlinearity, and the coupling between variables, is what makes these equations difficult or impossible to solve and is the main factor contributing to the turbulent phenomena that these equations describe.

1.1. GOVERNING EQUATIONS

The fundamental equations governing computational fluid dynamics are based on the law of conservation of mass, through the continuity equation, on the law of conservation of momentum, historically represented by the Navier-Stokes equations, and on the law of conservation of energy. More recently, in the scientific community the name of Navier-Stokes equations is used to refer to the set of conservation laws of mass, momentum and energy. Before proceeding with the derivation of these conservation laws, it is necessary to present and derive the Reynolds Transport Theorem.

2.1.1 REYNOLDS TRANSPORT THEOREM

The Reynolds Transport Theorem allows to study the variations over time of a physical quantity associated with a domain, and therefore to evaluate quantities related to a material volume *V(t)* known those relating to an initial control volume $V_0 = V_{(t=0)}$. Consider an extensive magnitude *B*, such that:

$$B = \int_V \rho b dV \tag{2.2}$$

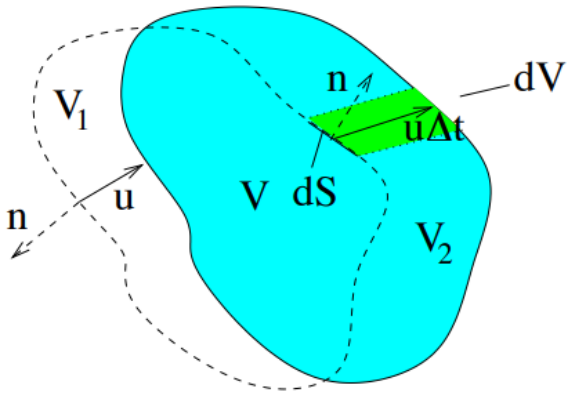

Figure 2.1: representation of control volume V_0 and material volume *V(t)*.

After a time $t+\Delta t$, the material volume *V(t)* will have moved by a certain amount, and the variation over time of the extensive magnitude *B* can be expressed as:

$$\frac{dB}{dt} = \frac{d}{dt}\int_{V(t)} \rho b \, dV = \lim_{\Delta t \to 0} \frac{\int_{V(t+\Delta t)} \rho b \, dV - \int_{V(t)} \rho b \, dV}{\Delta t} \tag{2.3}$$

According with *fig. 2.1*, eq. (2.3) can also be rewritten as:

$$\frac{dB}{dt} = \lim_{\Delta t \to 0} \frac{\int_V (\rho b)_{t+\Delta t} \, dV + \int_{V_2} (\rho b)_{t+\Delta t} \, dV - \int_V (\rho b)_t \, dV - \int_{V_1} (\rho b)_t \, dV}{\Delta t} \tag{2.4}$$

and since for $t \to 0$, $V(t) \to V_0$:

$$\lim_{\Delta t \to 0} \frac{\int_V (\rho b)_{t+\Delta t} \, dV - \int_V (\rho b)_t \, dV}{\Delta t} = \int_{V_0} \frac{d(\rho b)}{dt} dV \tag{2.5}$$

Considering that the two infinitesimal volumes dV_1 and dV_2 can be defined as shown in eq. (2.6), then the previous equation becomes (eq. (2.7)):

$$\begin{cases} dV_2 = +\vec{u} \cdot \vec{n} \, \Delta t \, dS \\ dV_1 = -\vec{u} \cdot \vec{n} \, \Delta t \, dS \end{cases} \tag{2.6}$$

$$\lim_{\Delta t \to 0} \frac{\int_{V_2} (\rho b)_{t+\Delta t} \, dV - \int_{V_1} (\rho b)_t \, dV}{\Delta t} = \lim_{\Delta t \to 0} \left(\int_{S_2} (\rho b)_{t+\Delta t} \, \vec{u} \cdot \vec{n} \, dS + \int_{S_1} (\rho b)_t \, \vec{u} \cdot \vec{n} \, dS \right) = \int_{S_0} \rho b \, \vec{u} \cdot \vec{n} \, dS \tag{2.7}$$

It is then obtained by replacing the eq. (2.5) and (2.7) in eq. (2.4):

$$\frac{dB}{dt} = \int_{V_0} \frac{d(\rho b)}{dt} dV + \int_{S_0} \rho b \vec{u} \cdot \vec{n} \, dS \tag{2.8}$$

Applying Gauss's Theorem to the second right term of the previous equation ultimately leads to the integral form of Reynolds Transport Theorem, represented by eq. (2.9).

$$\frac{dB}{dt} = \int_{V_0} \frac{d(\rho b)}{dt} dV + \int_{V_0} \vec{\nabla} \cdot (\rho b \vec{u})\, dV \tag{2.9}$$

It is worth noting that the velocity *u* that appears in the eq. (2.9) represents the absolute velocity of the fluid in the case of a control volume V_0 that holds a fixed position in space. In the case of a moving control volume with a certain speed *v*, the correct speed to use would be the relative speed $u_r = u - v$.

2.1.2 CONSERVATION OF MASS

Consider the following three-dimensional fluid element of dimensions δx, δy, δz:

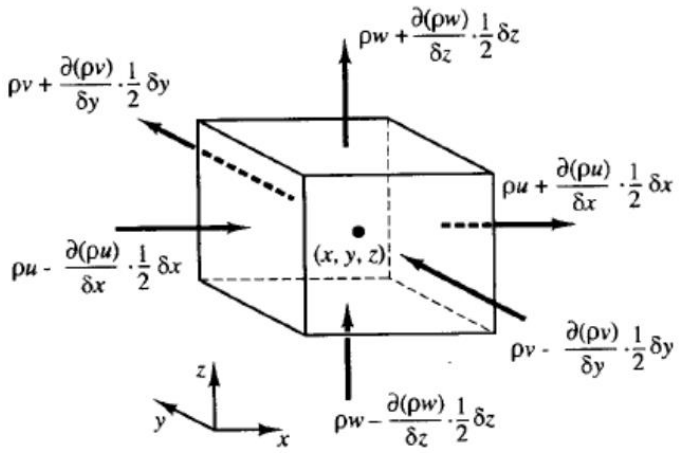

Figure 2.2: fluid reference element for mass balance.

The conservation of mass will be given by the equality between the rate of accumulation of the same within the element and the net balance of the incoming and outgoing flows through the contour surfaces. Using the notation shown in the figure and assuming negative the outgoing flows and positive the incoming ones, it is possible, with simple steps, to get to the equation:

$$\frac{\partial \rho}{\partial t} = -\left[\frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} + \frac{\partial(\rho w)}{\partial z}\right] \tag{2.10}$$

In vectorial form, the continuity equation for a non-stationary flow can be written as:

$$\frac{\partial \rho}{\partial t} + \vec{\nabla} \cdot (\rho \vec{u}) = 0 \tag{2.11}$$

where ρ is the density of the fluid and $\vec{u}$ the velocity vector. In the case of a sufficiently low Mach number, generally less than the value of $M = 0.3$, one can reasonably consider the flow as incompressible and thus simplify the previous expression as follows:

$$\vec{\nabla} \cdot \vec{u} = 0 \tag{2.12}$$

An alternative procedure for deriving the continuity equation uses Reynolds Transport Theorem (eq. (2.9)), establishing that:

$$B = M = \int_{V(t)} b \, dV \tag{2.13}$$

where it is used that $b = 1$. Then, substituting the previous equation in eq. (2.8), the following is obtained:

$$\frac{dM}{dt} = \int_{V_0} \frac{d\rho}{dt} dV + \int_{S_0} \vec{\nabla} \cdot \rho \vec{u} \cdot \vec{n} \, dS = 0 \tag{2.14}$$

Which represents the integral form of the continuity equation.
Applying the Divergence Theorem to the previous expression:

$$\frac{dM}{dt} = \int_{V_0} \left[\frac{d\rho}{dt} + \vec{\nabla} \cdot (\rho \vec{u})\right] dV = 0 \tag{2.15}$$

Finally, since the integral is valid for an arbitrary volume V_0, the following expression is obtained, which instead constitutes the differential form of the continuity equation.

$$\frac{\partial \rho}{\partial t} + \vec{\nabla} \cdot (\rho \vec{u}) = 0 \tag{2.16}$$

The equation just obtained, as is evident, coincides with the eq. (2.11).

2.1.3 CONSERVATION OF MOMENTUM

Newton's second law states that the change in momentum of a fluid particle must equal the summation of the forces acting on the particle, namely:

$$\frac{d\vec{Q}}{dt} = \vec{F} = \vec{F}_S + \vec{F}_V \tag{2.17}$$

where with $\vec{F}_S$ and $\vec{F}_V$ are indicated respectively the surface forces and the volume forces, while with $\vec{Q}$ is indicated the momentum of a fluid particle, defined as follows:

$$\vec{Q} = \int_{V_0} \rho \vec{u}\, dV \tag{2.18}$$

Surface forces can be written by distinguishing between forces due to pressure *P* alone, and all other forces (including friction forces).

$$\vec{F}_s = -\int_{S_0} P\vec{n}\, dS + \vec{F}_S' \tag{2.19}$$

Using Reynolds Transport Theorem (eq. (2.9)), the change in momentum can be rewritten in integral form.

$$\frac{d\vec{Q}}{dt} = \int_{V_0} \frac{d(\rho\vec{u})}{dt} dV + \int_{S_0} \rho\vec{u}\vec{u} \cdot \vec{n}\, dS \tag{2.20}$$

By entering the eq. (2.17) within the one just written:

$$\int_{V_0} \frac{d(\rho\vec{u})}{dt} dV + \int_{S_0} \rho\vec{u}\vec{u} \cdot \vec{n}\, dS + \int_{S_0} P\vec{n}\, dS = \vec{F}_S' + \vec{F}_V \tag{2.21}$$

Let:

$$\begin{cases} \vec{F}_S = \int_{S_0} \tilde{T} \cdot \vec{n}\, dS \\ \vec{F}_V = \int_{V_0} \rho\vec{f}\, dV \end{cases} \tag{2.22}$$

where is indicated by $\tilde{T}$ the stress tensor and $\vec{f}$ the density of volume forces, such as the force of gravity and electromagnetic forces. In the case of a moving fluid, the stress tensor, consisting of an isotropic part and a deviatory part, is shown in eq. (2.23).

$$\tilde{T} = -P\tilde{I} + \tilde{\tau} \tag{2.23}$$

Substituting these expressions for the surface and volume forces in the equation describing Reynolds Transport Theorem leads to the integral formulation of the momentum balance.

$$\int_{V_0} \frac{d(\rho\vec{u})}{dt} dV + \int_{S_0} \rho\vec{u}\vec{u} \cdot \vec{n}\, dS = -\int_{S_0} P\tilde{I} \cdot \vec{n}\, dS + \int_{S_0} \tilde{\tau} \cdot \vec{n}\, dS + \int_{V_0} \rho\vec{f}\, dV \tag{2.24}$$

Then, by applying the Divergence Theorem:

$$\int_{V_0} \left[\frac{d(\rho\vec{u})}{dt} + \vec{\nabla} \cdot (\rho\vec{u}\vec{u}) \right] dV = \int_{V_0} \left(-\vec{\nabla}P + \vec{\nabla} \cdot \tilde{\tau} + \rho\vec{f} \right) dV \tag{2.25}$$

Under the hypothesis of isotropy and remembering that, without losing generality, the choice of the control volume V_0 is arbitrary, the equation in differential form of Navier-Stokes describing the conservation of the momentum of the fluid is as follows:

$$\frac{\partial \rho\vec{u}}{\partial t} + \vec{\nabla} \cdot (\rho\vec{u}\vec{u}) = -\vec{\nabla}P + \vec{\nabla} \cdot \tilde{\tau} + \rho\vec{f} \tag{2.26}$$

while the expressions for the conservation of momentum along *x* and along *y*, derived from eq. (2.26), are reported respectively in the eq. (2.27) and (2.28).

$$\frac{\partial \rho u}{\partial t} + \vec{\nabla} \cdot (\rho u\vec{u}) = -\frac{\partial P}{\partial x} + \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} + \rho f_x \tag{2.27}$$

$$\frac{\partial \rho v}{\partial t} + \vec{\nabla} \cdot (\rho v\vec{u}) = -\frac{\partial P}{\partial y} + \frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \tau_{yy}}{\partial y} + \frac{\partial \tau_{zy}}{\partial z} + \rho f_y \tag{2.28}$$

where have been denoted with *u* and *v* the velocity components respectively along *x* and along *y* directions, with *P* the pressure, with τ the components of the tensor of the shear stresses and with *f* the volume forces. The Navier-Stokes equations allow to describe and model a very wide range of phenomena concerning fluid dynamics.
To solve the eq. (2.26), (2.27) and (2.28), two fundamental difficulties must be overcome:

- the convective terms present in the eq. (2.27) and (2.28) contain non-linear quantities that must be treated iteratively with a guessed initial velocity
- the pressure field is unknown (the variables are coupled) and needs to be solved iteratively as well.

A solution to the problem consists in the use of a *staggered* mesh: some variables are evaluated at the center of the cell (pressure) and other variables are evaluated on the faces of the control volume (speed components). The staggering of the velocity components is obtained by translating the coordinates of the computational nodes of the components *u* and *v* respectively

along the *x* axis and along the *y* axis, with reference to the scalar pressure nodes *P*, as shown in *fig. 2.3*.

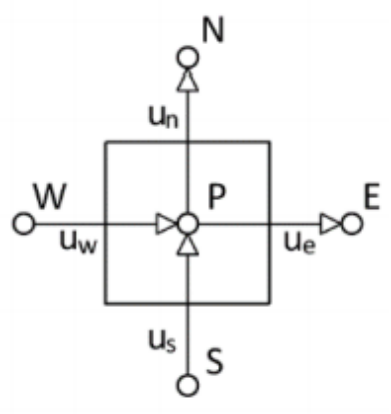

Figure 2.3: displaying the control volume of a staggered grid.

Through the staggering of the calculation grid described in *fig. 2.3* the problem-solving strategy through iterative approaches becomes viable. To undertake the calculation it is first necessary to proceed with the discretization of the equations of conservation of momentum using the *Finite Volume Method* (FVM) that will be presented in the following chapter.

2.1.4 CONSERVATION OF ENERGY

The principle of conservation of energy derives directly from the application of the first law of thermodynamics and establishes that the temporal variation of the internal energy of a fluid particle is equivalent to the sum of the thermal power and mechanical power exchanged from the outside with the particle.

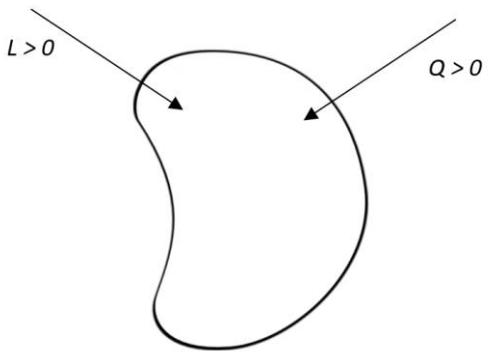

Figure 2.4: representation of the sign conventions on heat and work exchanged by the system.

$$\frac{dE}{dt} = \dot{L} + \dot{Q} \tag{2.29}$$

Through Reynolds Transport Theorem, introducing the intensive quantity ε, one can rewrite the eq. (2.29) in integral form as follows:

$$\frac{dE}{dt} = \frac{d}{dt}\int_V \rho\varepsilon \, dV = \int_{V_0} \frac{d(\rho\varepsilon)}{dt} dV + \int_{S_0} \rho\varepsilon\vec{u} \cdot \vec{n} \, dS \tag{2.30}$$

Distinguishing between surface contributions and volumetric contributions, it can be written that:

$$\dot{L} = \dot{L}_S + \dot{L}_V \tag{2.31}$$

$$\dot{Q} = \dot{Q}_S + \dot{Q}_V \tag{2.32}$$

where:

$$\begin{cases} \dot{L}_S = \int_{S_0} \left(\tilde{T} \cdot \vec{n}\right) \cdot \vec{u} \, dS = \int_{S_0} [(-P\tilde{I} + \tilde{\tau}) \cdot \vec{n}] \cdot \vec{u} \, dS \\ \dot{L}_V = \int_{V_0} \rho\vec{f} \cdot \vec{u} \, dV \end{cases} \tag{2.33}$$

$$\begin{cases} \dot{Q}_S = -\int_{S_0} \vec{K} \cdot \vec{n} \, dS = \int_{S_0} \lambda\vec{\nabla}T \cdot \vec{n} \, dS \\ \dot{Q}_V = \int_{V_0} \rho\dot{q} \, dV \end{cases} \tag{2.34}$$

Having denoted by $\dot{q}$ the heat flux per unit mass and with $\vec{K}$ the heat exchange flow per unit area that enters the system through the outer surface, in accordance with the Fourier formulation.

$$\vec{K} = -\lambda \vec{\nabla} T \tag{2.35}$$

Then the first law of thermodynamics reported in eq. (2.29) can be rewritten as:

$$\frac{dE}{dt} = \frac{d}{dt}\int_V \rho\varepsilon \, dV = \dot{L}_S + \dot{L}_V + \dot{Q}_S + \dot{Q}_V \tag{2.36}$$

And substituting in this expression the eq. (2.30), (2.33) and (2.34), the equation of conservation of energy in integral form is ultimately obtained.

$$\begin{aligned} &\int_{V_0} \frac{d(\rho\varepsilon)}{dt} dV + \int_{S_0} \rho\varepsilon\vec{u} \cdot \vec{n} \, dS \\ &= -\int_{S_0} P(\tilde{I} \cdot \vec{n}) \cdot \vec{u} \, dS + \int_{S_0} (\tilde{\tau} \cdot \vec{n}) \cdot \vec{u} \, dS + \int_{V_0} \rho\vec{f} \cdot \vec{u} \, dV \\ &+ \int_{S_0} \lambda \vec{\nabla} T \cdot \vec{n} \, dS + \int_{V_0} \rho\dot{q} \, dV \end{aligned} \tag{2.37}$$

The expression just obtained states that the variation in the internal energy of a fluid particle (left member of the equation) depends on the sum of (right member of the equation, in order from left to right): normal actions, tangential actions, volume forces, surface heat power exchanges and volumetric thermal power exchanges. With the application of the Divergence Theorem, surface integrals are transformed into volume integrals.

$$\begin{aligned} &\int_{V_0} \left[\frac{d(\rho\varepsilon)}{dt} + \vec{\nabla} \cdot (\rho\varepsilon\vec{u})\right] dV \\ &= \int_{V_0} \left[-\vec{\nabla} \cdot (P\vec{u}) + \vec{\nabla} \cdot (\tilde{\tau}\vec{u}) + \rho\vec{f} \cdot \vec{u} + \vec{\nabla} \cdot \left(\lambda\vec{\nabla} T\right) + \rho\dot{q}\right] dV \end{aligned} \tag{2.38}$$

Finally, remembering that the choice of the control volume V_0 is totally arbitrary, it is possible to arrive at the differential form of the energy conservation equation.

$$\frac{d(\rho\varepsilon)}{dt} + \vec{\nabla} \cdot (\rho\varepsilon\vec{u}) = -\vec{\nabla} \cdot (P\vec{u}) + \vec{\nabla} \cdot (\tilde{\tau}\vec{u}) + \rho\vec{f} \cdot \vec{u} + \vec{\nabla} \cdot \left(\lambda\vec{\nabla}T\right) + \rho\dot{q} \qquad (2.39)$$

2.1.5 BERNOULLI'S PRINCIPLE

To arrive at the formulation of Bernoulli's Principle it is worth introducing the concept of material derivative. The material derivative, or substantial derivative, is a differential operator that describes the variation of a certain physical quantity of a material element subject to a macroscopic velocity field that changes in space and time. Below is the definition of a material derivative written for a generic function φ, subject to a velocity field $\vec{u}$.

$$\frac{D}{Dt}\varphi = \frac{\partial}{\partial t}\varphi + \vec{u} \cdot \nabla\varphi \qquad (2.40)$$

With this concept it is possible to rewrite the equation of conservation of momentum (eq. (2.26)) in a more compact manner.

$$\rho\frac{D\vec{u}}{Dt} = -\vec{\nabla}P + \vec{\nabla} \cdot \tilde{\tau} + \rho\vec{f} \qquad (2.41)$$

At this point, for $\tilde{\tau}$ (deviatory part of the stress tensor $\tilde{T}$, defined in eq. (2.23)) the following assumptions are made:

- depends only on the instantaneous value of $\vec{u}$ (eq. (2.1) and not from its history
- it does not depend on the orientation of the fluid (the fluid is isotropic).

With these hypotheses it is found that:

$$\tilde{\tau} = -\frac{2}{3}\mu\left(\vec{\nabla} \cdot \vec{u}\right)\tilde{I} + 2\mu E \qquad (2.42)$$

where with E has been denoted the modulus of elasticity of the fluid. Substituting this expression for $\tilde{\tau}$ within the eq. (2.41) it is obtained, following simple algebraic operations, in a more general form of the Navier-Stokes equation, valid for compressible flows.

$$\rho \frac{D\vec{u}}{Dt} = -\vec{\nabla}P + \frac{1}{3}\mu\vec{\nabla}\cdot\left(\vec{\nabla}\cdot\vec{u}\right) + \mu\nabla^2\vec{u} + \rho\vec{f} \tag{2.43}$$

Under the hypothesis of stationary flows $\left(\frac{\partial}{\partial t} = 0\right)$ and of inviscid fluid (μ = *0*) free of dissipations:

$$\vec{u}\cdot\vec{\nabla}\vec{u} = -\vec{\nabla}P + \rho\vec{f} \tag{2.44}$$

The integration of the expression just obtained allows to formulate the Bernoulli's Principle, valid precisely for stationary flows characterized by negligible viscous effects.

$$\frac{u^2}{2} + gz + \frac{P}{\rho} = cost \tag{2.45}$$

2.2. TURBULENT FLOW

A turbulent regime in fluid dynamics is a type of fluid motion in which the viscous forces are insufficient to resist the forces of inertia: the resulting fluid particles move in a chaotic way, rather than following orderly trajectories as in the laminar regime. This causes the production of unstable "vortices" of various sizes that interact with each other, as well as chaotic fluctuations in the velocity and pressure fields in general. The main difficulty in the study of turbulence is the simultaneous presence of a large number of swirling structures of different characteristic sizes. Moreover, all these characteristic structures interact with each other because of the nonlinear structure of the Navier-Stokes equations. All these peculiarities make the classic analytical approach difficult to apply. In 1922 Lewis F. Richardson introduced the concept of energy cascade [7], while in 1941 the first statistical theory of turbulence, elaborated by the Soviet mathematician and physicist Andrei N. Kolmogorov, was formulated. In recent decades, the study of turbulence has advanced significantly, both in terms of the technologies that can be used in experimental studies and, more importantly, in terms of the introduction of computer simulations, which allow researchers to study turbulent flows quantitatively and in detail through numerical integration of the Navier-Stokes equations.

It is very difficult to give a precise definition of turbulence. What can be done, however, is to list and briefly describe some of the main characteristics of a turbulent flow:

- Irregularity: turbulent flows are always highly irregular; thus, turbulence problems are normally treated statistically rather than deterministically.
- Diffusivity: another significant aspect of all turbulent flows is their diffusivity, which produces rapid mixing and enhanced rates of momentum, heat, and mass transfer [8]. Turbulent diffusion is usually parameterized by a turbulent diffusion coefficient. This coefficient of turbulent diffusion is defined in a phenomenological sense, by analogy with molecular diffusivities.
- Dissipation: viscous shear stresses perform deformation work which increases the internal energy of the fluid at the expense of kinetic energy of the turbulence. Turbulence needs a continuous supply of energy to make up for these viscous losses, otherwise it decays rapidly.
- Rotationality and three-dimensionality: turbulent flows have non-zero vorticity and are characterized by a strong three-dimensional vortex generation mechanism known as

vortex stretching. Vortex stretching is the core mechanism on which the turbulence energy cascade relies to establish and maintain identifiable structure function.

- Large Reynolds number: turbulence often originates as an instability of laminar flow if the Reynolds number becomes too large. For practical purposes, if the Reynolds Number is less than 2000, the flow is laminar. The accepted transition Reynolds number for flow in a circular pipe is *Re* = 2300. At Reynolds numbers between about 2000 and 4000 the flow is unstable as a result of the onset of turbulence. These flows are sometimes referred to as transitional flows. Lastly, if the Reynolds number is greater than 3500, the flow is turbulent.

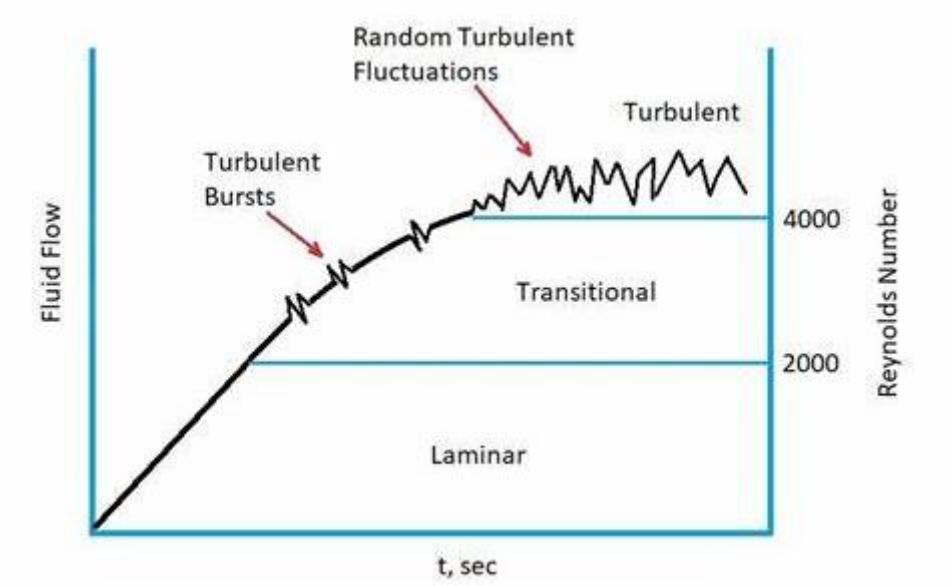

Figure 2.5: representation of the transition between laminar and turbulent flow.

- Kolmogorov length scales: the Kolmogorov scale is the smallest spatial scale where the kinetic energy coming from the upper scales (by the non-linear inertial term) is dissipated into thermal energy by viscosity. On such scales the flow is typically homogeneous and isotropic [9].
- Taylor microscales: the intermediate scales between the largest integral scale and the smaller Kolmogorov scale and make up the inertial interval. Taylor microscales do not correspond to dissipative phenomena, but to the transmission of energy from the largest to the smallest scale without dissipation.

In many applications, the efficiency of turbulence in moving and mixing fluids is critical. When separate fluid streams are brought together to mix, it's usually best if the mixing happens as

quickly as possible. Turbulence is also useful for mixing the momentum of the fluid. As a result, the wall shear stress (hence the drag) on aircraft wings and ship hulls is substantially higher than it would be if the flow were laminar. Similarly, compared to laminar flow, in turbulent flows heat and mass transfer rates at solid-fluid and liquid-gas interfaces are significantly higher.

2.2.1 TURBULENCE MODELING

As discussed in the previous chapter, the mechanism of turbulence in fluids is an extremely complex phenomenon to solve numerically and, in most cases, impossible to solve analytically. For this reason there is an interest, to derive a solution for flows with a high Reynolds number, in the implementation of calculation models able to describe in a sufficiently accurate way the turbulent phenomena that occur at large and small scales of motion. In the study of turbulent flows the ultimate objective is to obtain a tractable quantitative theory or model that can be used to calculate quantities of interests and practical relevance. Since there are no prospects of a simple analytic theory, the hope is to use the ever-increasing power of digital computers to achieve the objective of calculating the relevant properties of turbulent flows. Remembering that the Reynolds number is defined as:

$$Re = \frac{uL}{\nu} = \frac{\rho uL}{\mu} \tag{2.46}$$

It appears clear that it represents the ratio between inertial forces and viscous forces within a fluid. Therefore it is evident that the phenomenon of the energy cascade, which involves the creation of progressively smaller swirling structures at the lower scales of which the dissipation of energy takes place, occurs only in the presence of significant inertial terms (also called non-linear terms). The solution of a flow that generates small swirling spatial structures requires the adoption of grids that are sufficiently dense to adequately contain the variation in the quantity concerned. As a result, as the flow Reynolds number increases, the need to generate increasingly dense calculation grids (namely with progressively smaller spatial steps) increases and, ultimately, the computational cost of the calculation to be performed greatly increases. By calling ℓ_0 the characteristic length of the body around or inside which there is flow and η the Kolmogorov lengthscale, it can be shown that:

$$\frac{\eta}{\ell_0} \propto Re^{-\frac{3}{4}} \tag{2.47}$$

This means that for each length ℓ_0 the number of compute nodes needed to resolve the smaller scales η is proportional to $Re^{3/4}$, therefore, for a cube of volume $\ell_0{}^3$ are needed about $Re^{9/4}$ nodes.

Similarly, called τ_0 the macroscopic timescale and τ_η the *Kolmogorov* timescale, it is true that:

$$\frac{\tau_\eta}{\tau_0} \propto Re^{-\frac{1}{2}} \tag{2.48}$$

That is, the number of time-steps (the largest time-step cannot be greater than τ_η to effectively capture the dynamics of the scales of dimension η) is proportional to $Re^{1/2}$. In conclusion, it emerges that the number of operations to be performed for the numerical solution of the Navier-Stokes equations is of the order of $Re^{11/4} \sim Re^3$. Ultimately, it can be deduced that a direct numerical simulation (DNS) would require unsustainable calculation times. As things stand, the direct simulation of a flow at a Reynolds number of a few thousand already constitutes a challenge for modern supercomputers even if it is limited to a simplified geometry. Even taking Moore's law into account, if we then add the geometric complexities of industrial applications, and the inhomogeneities of the flow, we conclude that the direct simulation of turbulence, i.e. the solution of the Navier–Stokes equations without any model, does not constitute a possibility of investigation of practical problems not even in the coming decades.

To obtain a solution of the problem in acceptable calculation times it is common practice to establish a dimension of the vortex structures below which to proceed with a modeling of the phenomenon rather than with the direct numerical solution. The difference between the various calculation techniques consists precisely in the position of the "cut" in the cascade or, in other words, in which scales of motion one is willing to calculate and which to model.

Historically, many models have been proposed and many are currently in use. It is important to appreciate that there is a broad range of turbulent flows and a broad range of questions to be addressed. Consequently, it is useful and appropriate to have a broad range of models that vary in complexity, accuracy, and other attributes. The principal criteria that can be used to address different models are the: *level of description* (a higher level of description can provide a more complete characterization of the turbulence, leading to models of greater accuracy and wider applicability), *completeness* (a model is deemed complete if its constituent equations are free

from flow-dependent specifications), *cost and ease of use* (numerical methods are required to solve the model equations), *range of applicability* (limited by characteristics of flows and computational cost) and *accuracy* (the accuracy of a model can be determined by comparing model calculations with experimental measurements).

The suitability of a particular model for a particular turbulent-flow problem depends on a weighted combination of these criteria and the relative weighting of the importance of the various criteria depends significantly on the problem. Consequently, there is no one "best" model, but rather there is a range of models that can usefully be applied to the broad range of turbulent-flow problems.

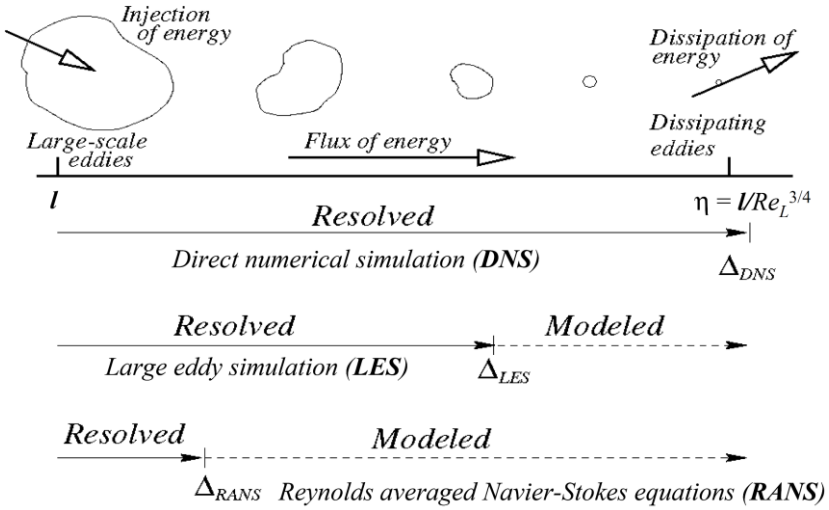

Figure 2.6: comparison of predictive methods, comparison of DNS, LES, and RANS models *[10]*.

The objective of turbulence modeling is to develop equations that will predict the time-averaged velocity, pressure, and temperature fields without calculating the complete turbulent flow pattern as a function of time. Turbulence models can be classified according to their computing cost, which is related to the number of scales modeled versus resolved (the more turbulent scales that are resolved, the finer the resolution of the simulation, and therefore the higher the computational cost). The computational cost is indeed very low if the majority or all of the turbulent scales are not modeled, but the accuracy suffers as a result.

However, from an engineering point of view, there is more interest in knowing the field of large scales, i.e. those scales that determine pressure fields and viscous stresses in the vicinity of the body. The oldest approach to turbulence modeling is the *Reynolds–averaged Navier–Stokes* (RANS) equations. The governing equations are solved as an ensemble, which adds new apparent stresses known as Reynolds stresses. This introduces a second order tensor of

unknowns, for which different models can provide various forms of closure. Most common approaches of RANS involve using an algebraic equation for the Reynolds stresses which include estimating the turbulent viscosity and, depending on the level of sophistication of the model, solving transport equations for determining the turbulent kinetic energy and dissipation [11]. The models available in this approach are often referred to by the number of transport equations associated with the method. The advantages of a RANS approach lie mainly in the simplicity of implementation and inexpensiveness of computation. The main disadvantages are attributable to the fact that an approach of this type requires energy production and dissipation to be equal instantly and locally, when in reality they are only statistically and globally equivalent; moreover, it is conceptually wrong to assume that everything that falls within the fluctuating contribution is attributable to turbulence.

Large eddy simulation (LES) is a technique in which the smallest scales of the flow are removed through a filtering operation, and their effect modeled using sub-grid scale models. This allows the turbulence's largest and most critical scales to be resolved while drastically decreasing the computing cost of the smallest scales. This method uses more computational resources than RANS methods, although it is much less expensive than DNS and it is very versatile.

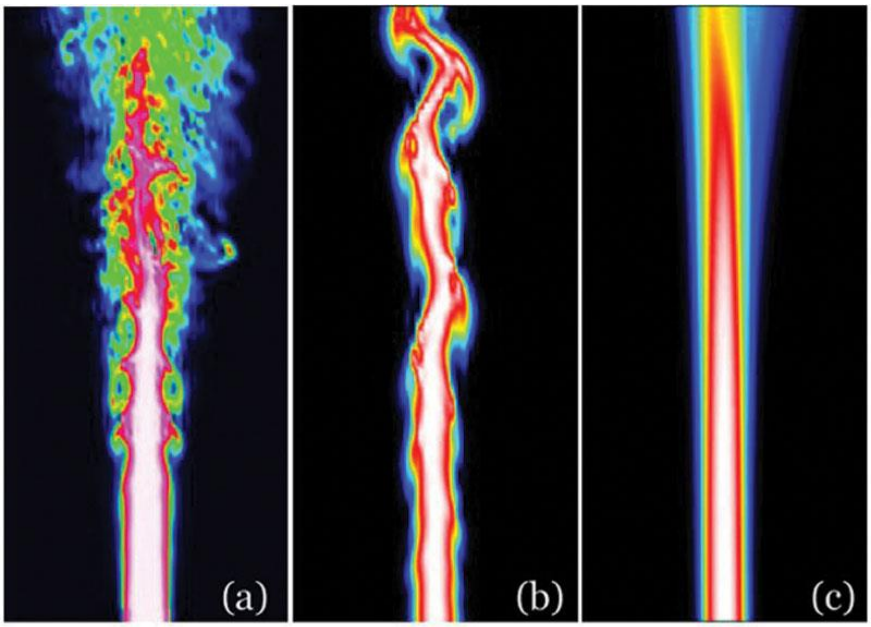

Figure 2.7: comparison between DNS (a), LES (b) and RANS (c) turbulence modeling.

To conclude, it can be said that a direct numerical simulation of a flow is the only method for accurately solving the entire range of vortex scales, although the absence of a turbulence model causes a prohibitive increase in the computational complexity of the calculation. A solution approach based on the LES equations allows to solve exactly the largest scales of the vortex

structures but still provides for a modeling of the sub-grid eddies. This feature makes this approach less expensive than a DNS simulation, but the computational complexity still remains too high to reasonably exploit these models for most practical applications. RANS equations solve flow by modeling the entire spectrum of turbulence scales, returning approximate but sufficiently significant results to make this approach the most widely used in industrial cases.

3. SOFTWARE AND IT TOOLS

The technology demonstrator shown in this work is based on a workflow that makes synergistic and combined use of various existing and widely established commercial software and tools on the market. These software are interfaced with each other by means of user-coded prototype tools developed specifically for the task of design exploration of road vehicles, but which are easily generalizable to adapt to any industrial application of research of optimal designs or analysis of specific shape variants. This chapter presents each of the main actors involved in the optimization process carried out, referring the reader to *chapter 5* for a more detailed description of the internal communication logic between the tools.

3.1. ANSYS WORKBENCH

ANSYS® Workbench™ is a workflow analysis platform that provides a single interface to all of the ANSYS tools, delivering to the users a software package that allows a more simple, schematic style approach to build simulation tasks. ANSYS Workbench is also built on a modular architecture that lets the users extend the functionality of such a numerical environment through add-in software components [12]. In particular, the ACT technology supplies internal mechanisms conceived to enable deeply integrated customizations of a Workbench application that can also make use of a connection with the inner libraries. In this work, ANSYS Workbench has been used to handle the shape parameters of the analyzed geometries and to create *Design of Experiment* (DOE) tables. In the next paragraph, the Optimal Space-Filling algorithm for locating sampling points is presented.

3.1.1. OPTIMAL SPACE-FILLING ALGORITHM

Design of Experiment is a technique used to scientifically determine the location of sampling points. The engineering literature contains a wide range of DOE algorithms or

approaches. They do, however, share some traits. They strive to find sampling points in such a way that the space of random input parameters is explored as efficiently as possible, or they try to gather the required information with the fewest possible sampling points. Sample points at efficient locations lower the number of sample points necessary while simultaneously improving the accuracy of the response surface derived from the results of the sampling points. Optimal Space-Filling design is a Latin Hypercube Sampling (LHS) that is extended with post-processing. It is initialized as LHS and then optimized several times, remaining a valid LHS (without points sharing rows or columns) while achieving a more uniform space distribution of points (maximizing the distance between points). Latin hypercube sampling (LHS) is a statistical method for generating a near-random sample of parameter values from a multidimensional distribution. When sampling a function with N variables, the range of each variable is partitioned into M equally probable intervals. The Latin hypercube requirements are then met by placing M sample points; this causes the number of divisions, M, to be equal for each variable. This sampling strategy does not require more samples for more dimensions (variables); this independence is one of the main advantages of this sampling scheme. Another benefit is that random samples can be taken one at a time while keeping track of which samples have already been taken. Possible disadvantages of the OSF algorithm are the fact that values closer to the edges and corners of the design space (extreme values) are normally not covered, and the lower quality of response prediction when too few design points are selected.

3.2. RBF MORPH

3.2.1. RADIAL BASIS FUNCTIONS

RBFs are a very powerful tool created for the interpolation of scattered data in the late ‘60s, but nowadays a rich collection of engineering problems can be faced using fast RBFs [13]. A system of RBFs is used to produce a solution for mesh morphing, once a list of source points and their displacements is defined (known values can be defined and retrieved everywhere in the space, no topology is required). This approach is valid both for surfaces shape changes and volume mesh smoothing. To define the shape changes, a system of radial basis functions that generate the nodal displacements based on a certain number of identified source points. The first step is to define the source points, or RBF points, on which the known user-defined displacements are applied. Then the RBF are used to interpolate the known displacements on the nodal positions using a principle of proximity: the displacement on the node is assigned based on the distance from the source points adjacent to the node itself. In this way, known displacement values can be defined anywhere, both inside and outside the domain, in arbitrary locations. In fact, a correspondence between the source points and the nodes of the mesh is not required. As a result, there is no mesh dependency and no control structures are required, thus the process is meshless. RBFs also allow to control the interpolation behavior of the function by choosing an appropriate base. From this point of view, RBF are very versatile and guarantee a good quality of the deformed mesh even for the most complex cases. The RBF technique is so fast and reliable that with parallel computing even large size models can be morphed in a reasonably short time, managing to process every kind of mesh element type (tetrahedral, hexahedral, polyhedral, prismatic, hexacore, non-conformal interfaces, etc.). The final goal of the technology is to perform parametric studies of component shapes and positions typical of the fluid-dynamic design, like design developments, multi-configuration studies, sensitivity analysis, design of experiment (DOE) and optimization processes.

The behavior of the function between points depends on the kind of the adopted radial function [14]. RBF’s can be classified based on the type of support they have, meaning that a function can be globally or locally supported, referring to the domain in which the chosen radial basis function is different than zero [15]. That choice also influences the computational cost and approach used for obtaining the RBF solution. In some cases, a polynomial corrector is used to ensure that the fit is solvable and unique. For the calculation of sought coefficients, a linear

system of order equal to the number of introduced source points must be solved. RBFs can be used both for interpolations and for regressions of data, that is, approximating a certain amount of data (especially useful in the presence of noisy data-sets), because not imposing the exact passage of the function in the source points but establishing a minimization of the error with respect to the passage can lead to a smoother and better manageable function. The scalar function at any arbitrary position inside or outside the domain (interpolation/extrapolation) is expressed by the sum of the radial contributions of each source point (RBF center) and a polynomial term once the unknown coefficients have been found. After the RBF fit, a closed form of the analytical solution that defines the displacement field at each point in the domain is obtained. This function will have on the source points the exact values imposed and will interpolate the values on the remaining nodes of the mesh with a trend dependent on the type of radial function used.

The general interpolation function, considering the presence of a polynomial correction ($h(\vec{x})$), can be written as follows:

$$s(\vec{x}) = \sum_{i=1}^{m} \gamma_i \varphi\left(\left\|\vec{x} - \vec{x}_{s_i}\right\|\right) + h(\vec{x}) \tag{3.1}$$

In this equation, *s* is the scalar function ($\mathbb{R}^n \rightarrow \mathbb{R}$) defined for an arbitrary sized variable $\vec{x}$ (the point at which the function is evaluated), φ is the so-called radial basis function, which is a scalar function of the Euclidean distance between each source point and the target point considered, γ_i is the weight of the radial function and *m* is the number of source points selected. The degree of the polynomial term depends on the kind of chosen basis. The weights of the radial functions γ_i and the polynomial coefficients β_i, unknowns of the system, can be retrieved by imposing the interpolation condition stating that the desired function values g_s are obtained at source points:

$$s\left(\vec{x}_{s_i}\right) = g_{s_i} \tag{3.2}$$

Because the order of the system is equal to the number of RBF source points, and the unknowns are both polynomial coefficients and RBF weights, an additional orthogonality constraint over the polynomial coefficients must be applied to make the system solvable:

$$\sum_{i=1}^{m} \gamma_i p(\vec{x}_{s_i}) = 0 \tag{3.3}$$

The eq. (3.3) represents the orthogonality condition of each coefficient of the polynomials *p* with a degree equal or less than that of polynomial *h*. If the RBF is conditionally positive definite, it can be demonstrated that a unique interpolant exists [16]. Moreover if the latter condition is respected and if the order is equal or less than 2 a linear polynomial can be used. Considering an *n-dimensiona*l space, the form of the polynomial function is the following:

$$h(\vec{x}) = \beta_1 + \beta_2 x_1 + \beta_3 x_2 + \ldots + \beta_{n+1} x_n \tag{3.4}$$

In matrix form, the problem to be solved takes the following form:

$$\begin{bmatrix} \widetilde{M} & \tilde{P}_s \\ \tilde{P}_s^T & \tilde{0} \end{bmatrix} \begin{Bmatrix} \vec{\gamma} \\ \vec{\beta} \end{Bmatrix} = \begin{Bmatrix} \vec{g} \\ \vec{0} \end{Bmatrix} \tag{3.5}$$

Where $\widetilde{M}$ is the interpolation matrix containing all the distances between RBF source points,

$$M_{i,j} = \varphi\left(\left\|\vec{x}_{s_i} - \vec{x}_{s_j}\right\|\right) \tag{3.6}$$

For all the $1 \leq i \leq m$ and $1 \leq j \leq m$. $\tilde{P}_s$ is a constraint matrix that arises balancing the polynomial contribution and contains a column of *ones* and the positions of source points in the other columns, where the j^{th} row is defined as:

$$P_{s_j} = \begin{bmatrix} 1 & x_{1_{s_j}} & x_{2_{s_j}} & \ldots & x_{i_{s_j}} & \ldots & x_{n_{s_j}} \end{bmatrix} \tag{3.7}$$

And finally $\vec{g}$ is the vector containing the *m* known values at RBF centers.
For a tridimensional case, the vector at a given point in space can be evaluated as:

$$\begin{cases} s^x(\vec{x}) = \sum_{i=1}^{m} \gamma_i^x \varphi\left(\left\|\vec{x} - \vec{x}_{s_i}\right\|\right) + \beta_1^x + \beta_2^x x + \beta_3^x y + \beta_4^x z \\ s^y(\vec{x}) = \sum_{i=1}^{m} \gamma_i^y \varphi\left(\left\|\vec{x} - \vec{x}_{s_i}\right\|\right) + \beta_1^y + \beta_2^y x + \beta_3^y y + \beta_4^y z \\ s^z(\vec{x}) = \sum_{i=1}^{m} \gamma_i^z \varphi\left(\left\|\vec{x} - \vec{x}_{s_i}\right\|\right) + \beta_1^z + \beta_2^z x + \beta_3^z y + \beta_4^z z \end{cases} \tag{3.8}$$

For a mesh morphing application, this task is generally carried out for each node at which the deformation must be computed. It's worth noting that mesh-morphing can have an impact on mesh quality, and the success of the morphing action is determined by the starting quality of the mesh as well as the extent and positions of local deformations. The distortion of the elements undergoing the most severe compression/stretching can be addressed with a judicious definition of the baseline mesh, keeping in mind the maximum predicted deformations, so that excellent mesh quality can be preserved even for very considerable shape adjustments.

Name	Abbreviation	$\phi(r)$
Polyharmonic spline	PHS	r^n, n odd $r^n \log(r)$, n even
Thin plate spline	TPS	$r^2 \log(r)$
Multiquadric biharmonics	MQB	$\sqrt{a^2 + (\epsilon r)^2}$
Inverse multiquadric biharmonics	IMQB	$\frac{1}{\sqrt{a^2+(\epsilon r)^2}}$
Quadric biharmonics	QB	$1 + (\epsilon r)^2$
Inverse quadric biharmonics	IQB	$\frac{1}{1+(\epsilon r)^2}$
Gaussian biharmonics	GS	$e^{-\epsilon r^2}$

Table 3.1: globally supported RBF *[17]*.

Name	Abbreviation	$\phi(\zeta)$
Wendland C^0	C0	$(1 - \epsilon\zeta)^2$
Wendland C^2	C2	$(1 - \epsilon\zeta)^4(4\epsilon\zeta + 1)$
Wendland C^4	C4	$(1 - \epsilon\zeta)^6(\frac{35}{3}\epsilon\zeta^2 + 6\epsilon\zeta + 1)$

Table 3.2: locally supported RBF, considering $\zeta = \frac{r}{R} = \frac{\left\|\vec{x} - \vec{x}_{s_i}\right\|}{R}$ *[17]*.

Numerical interpolation can be broken down into two stages: adaptation and evaluation. The centers are collected during the adaption phase, which is the most time-consuming in terms of calculation, and the suitable polynomial is defined based on the RBF chosen. The linear system

is then solved, yielding the polynomial coefficients and the base function's weight. Calculating the interpolated value at a given location is rather cheap at this point. This is referred to as the evaluation procedure. The major drawback for industrial applications is the large amount of memory required by RBF solutions. For example, using a double-precision format, an RBF problem with 1.000 centers requires 4 MB of memory, but a 100.000 source point problem requires 5 GB of memory [17]. However, there are several techniques that can be implemented to help lessen the problem. The distinction between sparse and dense matrices, for example, can be used. In fact, depending on the RBF type used, the $\widetilde{M}$ matrix can include a large number of null values. Sparse matrices can be easily compressed, resulting in decreased memory usage. Another option is the Partition of Unity (POU) approach, useful to break down the difficult main problem into a series of smaller, easier-to-solve subdomains. The global solution is obtained by combining each subdomain's solution with the others using appropriate weighting functions. The final subdomains will be of different sizes but will contain an equal number of centers, resulting in a faster resolution time. Finally, there are a number of strategies that can be utilized to reduce the time required for adaptation or evaluation, and hence the overall computation time. When there are too many source points, one method to speed up the adaptation time is to approximate the interpolation by discarding some centers and using an algorithm based on the distance between the points, which also allows for the gradual addition of new source points if the error committed is too high.

3.2.2. RBF MORPH ACT EXTENSION

RBF Morph ACT (Application customization Tool) is an extension powered by the high-performance numerical engine of RBF Morph with full support of ANSYS Mechanical. The already extensively proven RBF Morph technology integrated in ANSYS Fluent and available in Stand Alone format is able to perform fast mesh morphing using a meshless approach based on the state-of-the-art radial basis functions techniques. The use of such technology allows the CFD user to perform shape modifications compatible with the mesh topology, directly in the solving stage [18]. The software has a very intuitive graphical user interface (GUI) and is able to perform surface and volume mesh parametrization leaving unaltered the mesh topology, offering the potential for a fast shape change without the need for re-meshing of the computational domain. The RBF Morph solver has also the ability to perform advanced

morphing actions with parallel computing, allowing the user to obtain solutions faster and limiting the computational cost of every operation.
From the user's point of view, the morphing process can be reduced to the following activities:

- set-up: defines the source points on the surface of the geometry to be modified and on the volume around the geometry to control the propagation of the deformation;
- adaptation: the RBF problem is solved and stored;
- evaluation: the nodes to be moved are evaluated using the calculated solution.

When dealing with a parametrization for shape optimization purposes the property of adaptation can be a winning point since the solution can be solved once and then stored to be used when needed. Once the solution has been calculated, before the actual morphing of the geometry, the shape change can be viewed through a preview, and it is possible to check the quality of the mesh. The final quality of the deformed mesh depends on the RBF chosen and the set-up used.
A basic block of tools required by any modelling paradigm is composed by the geometric modifiers, a set of transformation algorithms that allow the user to apply the wanted modifications to the geometry. These modifiers should allow a level of control similar to CAD and enabling a persistent morphing with it if required, guaranteeing proper expressiveness to the user.

Translation modifier

The most basic modifier is translation. It's a geometric transformation that allows to move every selected point in a given direction by the same amount, making it a rigid transformation that's simple to perform by adding a constant vector to each point that has to be moved. Translation is a linear modifier, which means that the sequence in which numerous different displacements are applied has no impact on the final product.

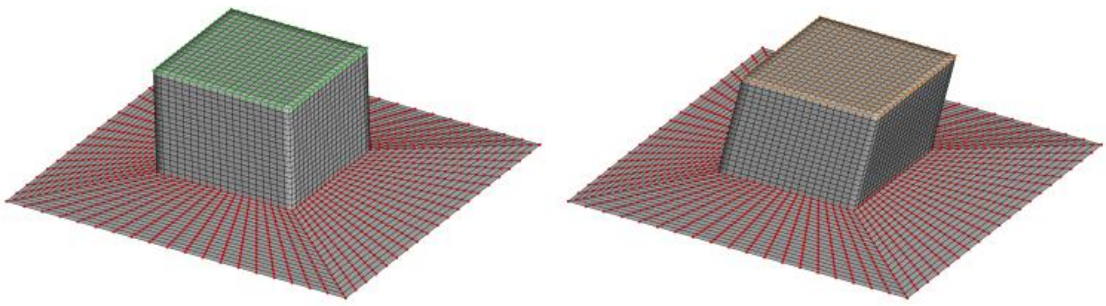

Figure 3.1: comparison of baseline (*left*) and deformed geometry (*right*) using a translation modifier.

Rotation modifier

Rotation is a non-linear geometric transformation that differs from translation in that it ensures that at least one fixed point in space is maintained at all times. In order to conveniently apply the modification to complex curved geometries, the user can give information relative to the axis of rotation and the desired angle, or by employing principal/relative reference systems. Due to the nonlinear nature of rotations, direct amplifications of the observed shape variations would result in deformed geometry. When dealing with amplified rotations, special attention must be taken to rectify the geometry using a scaling correction. Furthermore, to deal with rotations' non-commutativity property, a technique must be used to expressively apply them by combining multiple modifiers correctly.

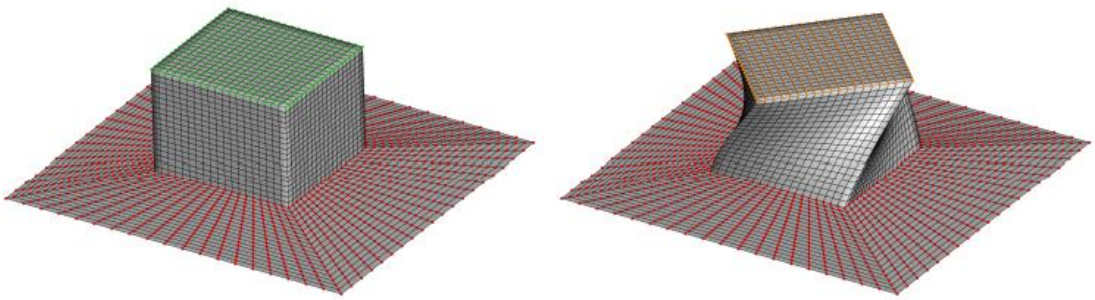

Figure 3.2: comparison of baseline (*left*) and deformed geometry (*right*) applying a rotation modifier.

Scaling modifier

Scaling is a linear transformation that allows you to shrink or dilate specified source points around a central point. It is possible to shrink or extend the geometry along the orthogonal axes by using a scaling factor with a negative or positive value. A unitary scaling preserves the positions of source points, yielding the same outcome as a zero translation.

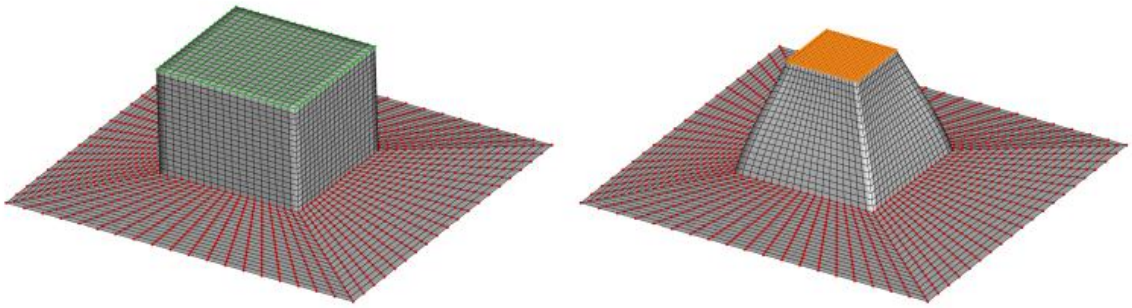

Figure 3.3: comparison of baseline (*left*) and deformed geometry (*right*) in case of a scaling modifier (the red points were fixed in space using a zero-displacement modifier.

Projection modifier

By defining a set of source points and target geometry, such as a mesh, this modifier projects the source points from their original location to the nearest point in the target geometry. As a result, it is possible to apply complex shape variations, such as CAD entities, or overlay parts of the mesh on a desired geometry.

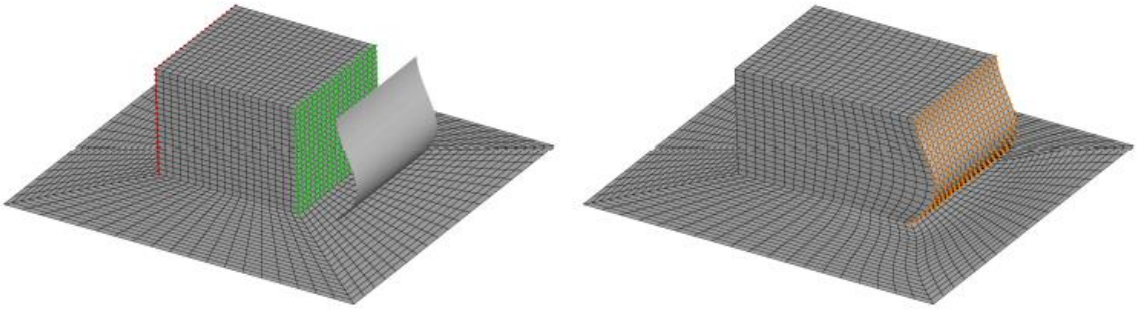

Figure 3.4: comparison of baseline (*left*) and deformed geometry (*right*) when a projection modifier is used.

Offset modifier

Another useful surface-based shape modifier is offset. Given a set of points on the grid it is possible to calculate the implicit surface on which they lie, evaluating for each point the normal to the surface. The points are moved along the normal direction.

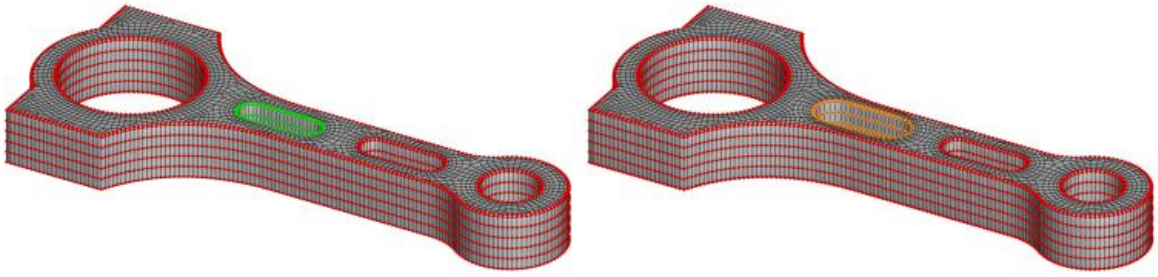

Figure 3.5: comparison of baseline (*left*) and deformed geometry (*right*) using the offset modifier for the resizing of the rod lightening hole.

For Translation, Scaling and Rotation a custom coordinate system to shorten the morphing set-up can be selected. Other available options are the order of the RBF used (linear or cubic) and the feature to apply the shape modification to the baseline mesh or to the already modified grid.

A distinction between hierarchical and single step methods can be conceived by distinguishing between source and target points, with source being the collection of centers determining the shape modification and target being the points interested by the interpolation. All mesh nodes are treated as target points in a conventional case, and displacements are calculated by assessing the field calculated on a single RBF problem defined by a collection of source points. On the other hand, while using the hierarchical approach, only a fraction of the points is employed as target, narrowing the problem to the ones that require more precise and local management. The final shape assumed by the target can be simply predicted by the user, avoiding long-distance interaction difficulties, and simplifying the set-up procedure. Using this strategy, the achieved target of a problem can be utilized to define the displacements of its source points in a new one, imposing hierarchically a field represented by precisely controlled source points on the new target. By changing points, edges, surfaces, or volumes to achieve the needed level of control, it is then possible to construct many connected problems with an increasing level of detail. When each RBF set-up is solved in cascade as a separate problem, it is possible to use alternative parameters for the solution, including the basis functions and thus the interpolation behavior, depending on the desired result. According to the ACT Extension of RBF Morph, the entire set-up, which is made up of several RBF problems that communicate with one another, can be viewed as a hierarchical tree structure in which each node transfers its target points to its parent, which is the node one level higher in the hierarchy on the same branch. The RBF problem is set up at a given node by blending the source nodes provided by its children. Each nested RBF child is called a Source, and each selection on which acts a shape modifier is called an RBF Target. The root problem, retrieving the whole set-up, is the more complex and the one that gives the final shape.

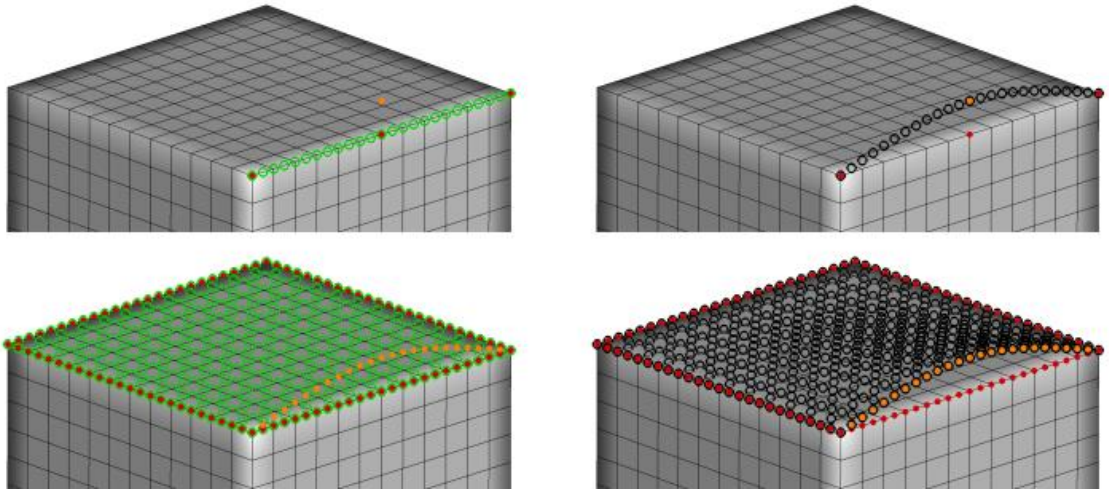

Figure 3.6: representation of a hierarchical set-up, showing how the target points of the first RBF problem (*top*) is employed to define the source of the second problem (*bottom*) *[17]*.

3.3. STAR-CCM+

Simcenter STAR-CCM+ is a commercial *Computational Aided Engineering* (CAE) simulation software by Siemens Digital Industries. Simcenter STAR-CCM+ allows designers to model and analyze a wide range of multidisciplinary engineering problems in both fluid and solid continuum mechanics involving fluid flow, heat transfer, stress, particle flow, electromagnetics, and other phenomena. The integrated environment includes, within a single user interface, everything from CAD, automated meshing, multiphysics CFD, sophisticated postprocessing, and design exploration. This allows engineers to efficiently explore the entire design space to make better design decisions faster.

The Simcenter STAR-CCM+ simulation environment includes all steps necessary for conducting engineering analyses, including: import and creation of geometries, mesh generation, solution of the governing equations, analysis of the results, automation of the simulation workflows for design exploration studies, connection to other CAE software for co-simulation analysis.

Simcenter STAR-CCM+ can simulate internal and external fluid flow for a variety of fluid types and flow regimes. It solves the mass, momentum, and energy conservation equations for general incompressible and compressible fluid flows. This chapter shows some of the theoretical foundations underlying the numerical methods implemented by the solver and used in the simulations conducted during the work.

3.3.1. NUMERICAL FLOW SOLUTION

Finite Volume Method

The numerical solution of a partial differential equation consists in finding the values of the dependent variable (i.e. φ) at specific points on which its distribution can be reconstructed on a domain of interest. These points are called grid elements (or nodes) and result from the discretization of the original geometry into a set of discrete, non-overlapping elements. This process is known as meshing. The nodes resulting from the meshing process are generally

placed in the center of gravity of the cells with which the domain is discretized or on the vertices of the same.

The Finite Volume Method is very popular in the discretization of differential problems in conservative form, as it allows to transform the set of partial differential equations into a system of linear algebraic equations. Nevertheless, the discretization procedure used in the Finite Volume Method is distinctive and involves two basic steps. In the first step, the partial differential equations are integrated and transformed into balance equations over an element. This involves changing the surface and volume integrals into discrete algebraic relations over elements and their surfaces using an integration quadrature of a specified order of accuracy. The result is a set of semi-discretized equations. In the second step, interpolation profiles are chosen to approximate the variation of the variables within the element and relate the surface values of the variables to their cell values and thus transform the algebraic relations into algebraic equations.

Differential and integral form of the general transport equation

Let φ be a generic scalar variable transported inside of a space-time domain. The differential equation that describes the evolution of this generic variable can be conveniently expressed through the following conservative form [19]:

$$\frac{\partial(\rho\varphi)}{\partial t} + div(\rho\varphi\vec{u}) = div(\Gamma\nabla\varphi) + S_{\varphi} \tag{3.9}$$

in which the various terms represent, from left to right: the rate of change of property φ in an elementary material particle (where ρ represents the particle density), the specific flow of φ coming out of the elementary particle, the specific increase in φ due to diffusion (with Γ diffusion coefficient), specific increase in φ due to generic sources.

Integrating equation (3.9) on a generic reference control volume of finite dimensions leads to the integral form of the general transport equation.

$$\int_{CV} \frac{\partial(\rho\varphi)}{\partial t} dV + \int_{CV} div(\rho\varphi\vec{u}) dV = \int_{CV} div(\Gamma\nabla\varphi)\, dV + \int_{CV} S_{\varphi}\, dV \tag{3.10}$$

This equation can then be reformulated by applying Gauss's Theorem to some of its terms.

$$\frac{\partial}{\partial t}\int_{CV}(\rho\varphi)dV + \int_{A}(\rho\varphi\vec{u})\cdot\vec{n}dA = \int_{A}(\Gamma\nabla\varphi)\cdot\vec{n}dA + \int_{CV}S_{\varphi}\,dV \qquad (3.11)$$

which is the basic form of the general transport equation normally used for the application of the finite volume method. The physical meaning of the various terms can be described as follows (again from left to right): accumulation rate of the property φ within the control volume; net balance of the convective flow of φ through the contour surface of the control volume (known as convective term), net balance of the diffusive flow of φ through the contour surface of the volume control (known as diffusive term), increased φ due to generic sources within the control volume.

Integrating on the generic finite time interval Δt, the (3.11) becomes:

$$\int_{\Delta t}\frac{\partial}{\partial t}\int_{CV}(\rho\varphi)dV + \int_{\Delta t}\int_{A}(\rho\varphi\vec{u})\cdot\vec{n}dA = \int_{\Delta t}\int_{A}(\Gamma\nabla\varphi)\cdot\vec{n}dA + \int_{\Delta t}\int_{CV}S_{\varphi}\,dV \qquad (3.12)$$

It is easy to identify, from left to right in the above equation: the transient term (which signifies the time rate of change of fluid property φ inside the control volume), the convective flux (which expresses the net rate of decrease of fluid property φ across the control volume boundaries due to convection), the diffusive flux (which corresponds to the net rate of increase of fluid property φ across the control volume boundaries due to diffusion) and the source term (which expresses the generation/destruction of fluid property φ inside the control volume).

Otherwise, in the hypothesis of a stationary problem, the transport equation is formulated as follows:

$$\int_{A}(\rho\varphi\vec{u})\cdot\vec{n}dA = \int_{A}(\Gamma\nabla\varphi)\cdot\vec{n}dA + \int_{CV}S_{\varphi}\,dV \qquad (3.13)$$

Discretization of government equations

The numerical solution of a fluid dynamic problem requires the spatial and temporal discretization of its government equations. Discretization is a mathematical process of transforming continuous equations into their discrete counterparts and is a fundamental process in making equations adequate for a numerical solution and its implementation for computer-

assisted digital computation. For simplicity, this paragraph presents the case of incompressible fluid on a two-dimensional domain, but the same considerations also apply to more general cases [20].

The discretization of the continuity equation (eq. (2.11)) is easy to derive and is obtained by integration on an arbitrary control volume followed by the application of the Divergence Theorem.

$$\frac{(\rho - \rho^0)\Delta x \Delta y}{\Delta t} + [(\rho u)_e - (\rho u)_w]\Delta y + [(\rho v)_n - (\rho v)_s]\Delta x = 0 \tag{3.14}$$

Continuing to consider an incompressible flow and assuming the absence of volume forces, the eq. (2.41) is reduced to:

$$\rho \frac{\partial \vec{u}}{\partial t} + \rho \vec{u} \cdot \vec{\nabla} \vec{u} = -\vec{\nabla} P + \mu \nabla^2 \vec{u} \tag{3.15}$$

which, when written in terms of velocity components, takes the form:

$$\rho \frac{\partial u_i}{\partial t} + \rho \frac{\partial u_i u_j}{\partial x_j} = -\frac{\partial P}{\partial x_j} + \mu \frac{\partial^2 u_i}{\partial x_j \partial x_j} \tag{3.16}$$

In the eq. (3.15) and (3.16) the dynamic viscosity coefficient of the fluid has been indicated with μ. Integrating again on an arbitrary control volume and applying the Gauss-Green Divergence Theorem, the convective terms that appear in the eq. (4.8) can be written in discretized form.

$$\rho \frac{\partial u_i}{\partial t} V + \rho \sum_{nbr} \left(u_i u_j n_j A\right)_{nbr} = -\sum_{nbr} (P n_i A)_{nbr} + \mu \sum_{nbr} \left(\frac{\partial u_i}{\partial x_j} n_j A\right)_{nbr} \tag{3.17}$$

where with the subscript *nbr* (*neighbor*) have been named the values on each of the faces that make up the considered volume. In the case of a two-dimensional cartesian grid, the eq. (3.17) can be expanded in the form shown below.

$$\rho\frac{\partial u_i}{\partial t}\Delta x\Delta y+\rho[(u_i u\Delta y)_e-(u_i u\Delta y)_w]+\rho[(u_i v\Delta x)_n-(u_i v\Delta x)_s]$$
$$=-[(Pn_i\Delta y)_e-(Pn_i\Delta y)_w+(Pn_i\Delta x)_n-(Pn_i\Delta x)_s] \tag{3.18}$$
$$+\ \mu\left[\left(\frac{\partial u_i}{\partial x}\Delta y\right)_e+\left(\frac{\partial u_i}{\partial x}\Delta y\right)_w+\left(\frac{\partial u_i}{\partial y}\Delta x\right)_n+\left(\frac{\partial u_i}{\partial y}\Delta x\right)_s\right]$$

The previous expression, in the presence of a *staggered* grid, is reported in the eq. (3.19) and (3.20) separating the components along the *x*-axis and along the *y*-axis.

$$\rho\frac{\partial u}{\partial t}\Delta x\Delta y+\rho[(uu\Delta y)_e-(uu\Delta y)_w]+\rho[(uv\Delta x)_n-(uv\Delta x)_s]$$
$$=-[(P\Delta y)_e-(P\Delta y)_w] \tag{3.19}$$
$$+\mu\left[\left(\frac{\partial u}{\partial x}\Delta y\right)_e+\left(\frac{\partial u}{\partial x}\Delta y\right)_w+\left(\frac{\partial u}{\partial y}\Delta x\right)_n+\left(\frac{\partial u}{\partial y}\Delta x\right)_s\right]$$

$$\rho\frac{\partial v}{\partial t}\Delta x\Delta y+\rho[(vu\Delta y)_e-(vu\Delta y)_w]+\rho[(vv\Delta x)_n-(vv\Delta x)_s]$$
$$=-[(P\Delta x)_n-(P\Delta x)_s] \tag{3.20}$$
$$+\mu\left[\left(\frac{\partial v}{\partial x}\Delta y\right)_e+\left(\frac{\partial v}{\partial x}\Delta y\right)_w+\left(\frac{\partial v}{\partial y}\Delta x\right)_n+\left(\frac{\partial v}{\partial y}\Delta x\right)_s\right]$$

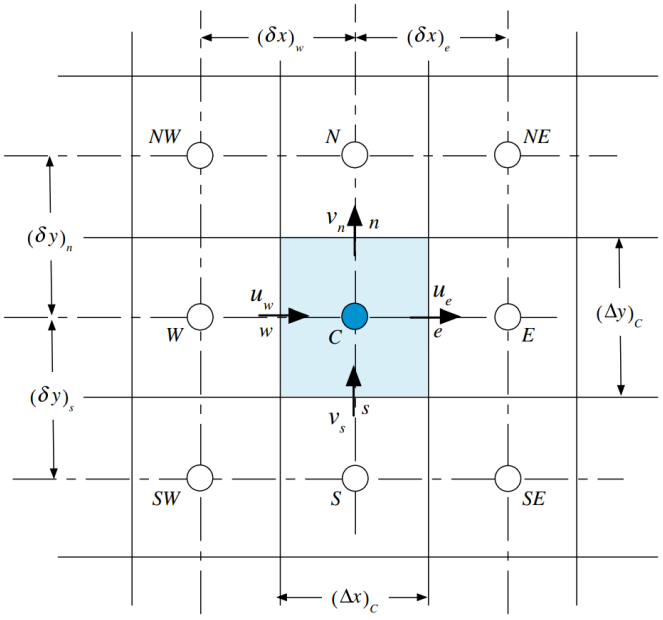

Figure 3.7: Two-dimensional cartesian grid element *[20]*.

For the above equations, expressions for the interface values of *u*, *v* and *P* and an expression that allows to approximate the derivatives in time and space as finite differences remain to be derived.

Time derivatives can be rewritten in discrete terms using the backward finite differences method. For both conservation equations of momentum (eq. (3.19) and (3.20)) the time derivative becomes:

$$\frac{\partial u_i}{\partial t} = \frac{u_i^n - u_i^{n-1}}{\Delta t} \tag{3.21}$$

where the temporal step has been indicated with Δt and the index of the temporal instant has been indicated with *n*.

The discretization of the spatial derivatives present in the diffusive terms takes place by evaluating the difference between velocity values located on the nodes adjacent to the interface to be evaluated.

$$\left(\frac{\partial u}{\partial x}\right)_e = \frac{u_E - u_P}{\Delta x} \tag{3.22}$$

$$\left(\frac{\partial u}{\partial x}\right)_w = \frac{u_P - u_W}{\Delta x} \tag{3.23}$$

$$\left(\frac{\partial u}{\partial y}\right)_n = \frac{u_N - u_P}{\Delta y} \tag{3.24}$$

$$\left(\frac{\partial u}{\partial y}\right)_s = \frac{u_P - u_S}{\Delta y} \tag{3.25}$$

Having interest in evaluating unknown variables at locations (interfaces) other than the control volume computational nodes, it is necessary to proceed to the formulation of a mathematical interpolation scheme. There are various types of interpolation formulae: *central difference, upwind, quadratic upwind, QUICK, hybrid, etc* [21].

As an example, let us compare *central differences* and *upwind* interpolation schemes: by applying the *upwind* scheme, the expressions of the interface values are derived according to eq. (3.26), while the same expressions treated with the interpolation scheme of the *centered differences* are derived as reported in eq. (3.27). The following equations are written for the horizontal velocity component *u* only but can be generalized for each component of the velocity vector.

$$upwind: \begin{cases} u_e = u_P \;\; ; \;\; u_w = u_W \;\; se \;\; F_e > 0 \\ u_e = u_E \;\; ; \;\; u_w = u_P \;\; se \;\; F_e < 0 \end{cases} \tag{3.26}$$

$$centered\ differences: \begin{cases} u_e = \frac{u_P + u_E}{2} \\ u_w = \frac{u_W + u_P}{2} \end{cases} \tag{3.27}$$

The decision on which interpolation method to employ is based on the Reynolds number associated with the flow:

$$Re = \frac{\rho u H}{\mu} \tag{3.28}$$

This dimensionless number represents a measure of the relationship between convection and diffusion. When the Reynolds number is small enough, the flow is poorly directional, and the method of *centered differences* works well; on the contrary, as *Re* grows, problems of instability begin to occur, and the solution tends to diverge. In practice, the *centered difference* method is only applied for limited Reynolds numbers, while for higher values the *upwind* interpolation scheme is preferable.

SIMPLE algorithm

The Simcenter STAR-CCM+ is a commercially available CFD package that includes geometry/computer-aided design (CAD) manipulation tools, a grid generator capable of generating different unstructured grid topologies (polyhedral, Cartesian, tetrahedral), various flow solvers, and post-processing tools. The flow solvers of STAR-CCM+ solve the RANS equation in finite-volume, cell-centered formulation. The Segregated Flow model invokes the

segregated solver which solves each of the momentum equations in turn, one for each dimension. The linkage between the momentum and continuity equations is achieved with a predictor-corrector approach. The complete formulation can be described as using a colocated variable arrangement and a pressure-velocity coupling combined with a SIMPLE-type algorithm.

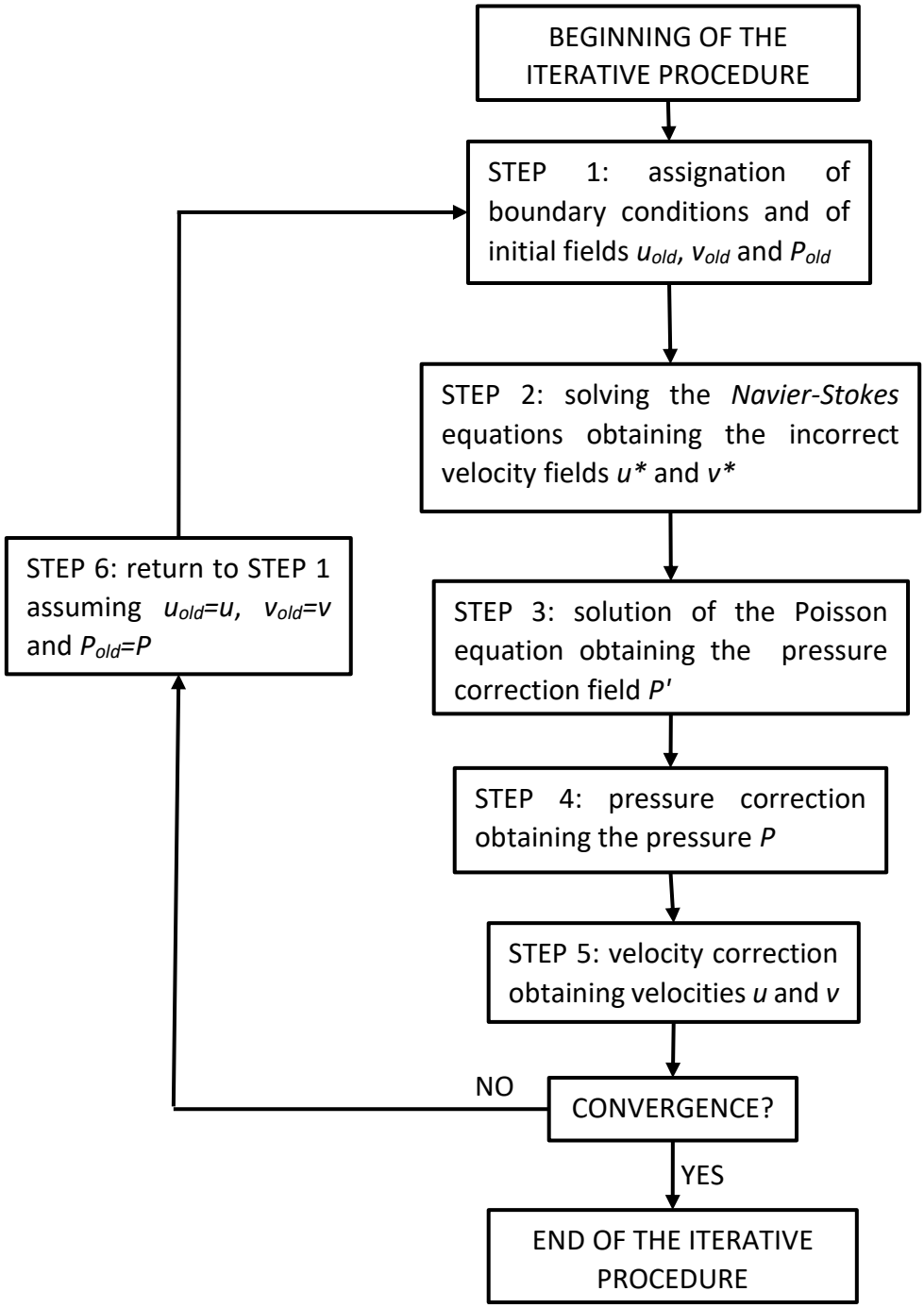

Figure 3.8: flowchart of the iterative process of the SIMPLE algorithm.

The SIMPLE algorithm, whose development is attributed to Prof. Suhas V. Patankar [19], is an iterative numerical method widely employed in computational fluid dynamics for solving the Navier-Stokes equations. In the algorithm, the solution to the fluid dynamics problem is derived iteratively, generating velocity and pressure fields that consecutively satisfy the equations of conservation of momentum and mass, progressively approaching the final solution with each iterative calculation cycle. The sequence of operations predicted by the algorithm is summarized in the flowchart shown in *fig. 3.8*.

Based on eq. (2.11), the single equation for the generic control volume identified around a velocity node *u* takes the following form:

$$a_e u_e = \sum_{nbr} a_{nbr}\, u_{nbr} + \frac{P_P - P_E}{\rho} A_e + b \tag{3.29}$$

Similarly, for a node of velocity *v* it is true that:

$$a_n v_n = \sum_{nbr} a_{nbr}\, v_{nbr} + \frac{P_P - P_N}{\rho} A_n + b \tag{3.30}$$

In the absence of volume forces, the preceding expressions are simplified by introducing the condition $b = 0$.

The eq. (3.29) and (3.30) are solvable only once the pressure field *P* is known, of which an initial distribution is assumed (estimated) in *STEP 1* of the algorithm procedure (*P**). A system of equations such as the following is ultimately reached.

$$\begin{cases} a_e u_e^* = \sum_{nbr} a_{nbr}\, u_{nbr}^* + \frac{P_P^* - P_E^*}{\rho} A_e \\ a_n v_n^* = \sum_{nbr} a_{nbr}\, v_{nbr}^* + \frac{P_P^* - P_N^*}{\rho} A_n \end{cases} \tag{3.31}$$

The solution of which is an incorrect velocity field (*u**,*v**) since it does not yet meet the continuity condition. To obtain the correct velocity and pressure fields, correction fields must be added to these incorrect values.

$$\begin{cases} P = P^* + P' \\ u = u^* + u' \\ v = v^* + v' \end{cases} \tag{3.32}$$

The link between velocity and pressure corrections is the equations system (3.33).

$$\begin{cases} a_e u'_e = \sum_{nbr} a_{nbr}\, u'_{nbr} + \dfrac{P'_P - P'_E}{\rho} A_e \\ a_n v'_n = \sum_{nbr} a_{nbr}\, v'_{nbr} + \dfrac{P'_P - P'_N}{\rho} A_n \end{cases} \tag{3.33}$$

Neglecting for the correction terms the contribution of neighboring nodes you get:

$$\begin{cases} a_e u'_e = \dfrac{P'_P - P'_E}{\rho} A_e \\ a_n v'_n = \dfrac{P'_P - P'_N}{\rho} A_n \end{cases} \tag{3.34}$$

Thus:

$$\begin{cases} u'_e = d_e (P'_P - P'_E) \\ v'_n = d_n (P'_P - P'_N) \end{cases} \tag{3.35}$$

Having indicated with d_e and d_n respectively the coefficients:

$$d_e = \frac{A_e}{\rho a_e} \tag{3.36}$$

$$d_n = \frac{A_n}{\rho a_n} \tag{3.37}$$

In conclusion, the correct velocity components are calculated as:

$$\begin{cases} u_e = u_e^* + d_e(P_P' - P_E') \\ v_n = v_n^* + d_n(P_P' - P_N') \end{cases} \tag{3.38}$$

The pressure corrections that appear in the preceding expressions are derived from the discretized continuity equation (eq. (2.6)).

$$a_P P_P' = a_E P_E' + a_W P_W' + a_N P_N' + a_S P_S' - b \tag{3.39}$$

Where:

$$a_E = d_e \Delta y \tag{3.40}$$

$$a_W = d_w \Delta y \tag{3.41}$$

$$a_N = d_n \Delta x \tag{3.42}$$

$$a_S = d_s \Delta x \tag{3.43}$$

$$a_P = a_E + a_W + a_N + a_S \tag{3.44}$$

$$b^* = (u_e^* - u_w^*)\Delta y + (v_n^* - v_s^*)\Delta x \tag{3.45}$$

SIMPLE-R algorithm

The approximation introduced in the derivation of the pressure field P' can lead to even very strong pressure corrections, making it necessary to proceed to a form of underrelaxation. This is because the pressure correction also preserves the weight of the velocity field correction. The SIMPLER algorithm, briefly illustrated in *fig. 3.9*, exploits the equation for P' for velocity correction only, deriving instead an additional expression for the correct pressure P.
Returning to consider the discrete equations starting from the conservation of the momentum along *x* and along *y*, SIMPLER provides for the preliminary calculation of the pseudo-velocity components, obtained from the eq. (3.29) and (3.30) neglecting pressure terms.

$$\hat{u}_e = \frac{\sum_{nbr} a_{nbr} u_{nbr}}{a_e} \tag{3.46}$$

$$\hat{v}_n = \frac{\sum_{nbr} a_{nbr} v_{nbr}}{a_n} \tag{3.47}$$

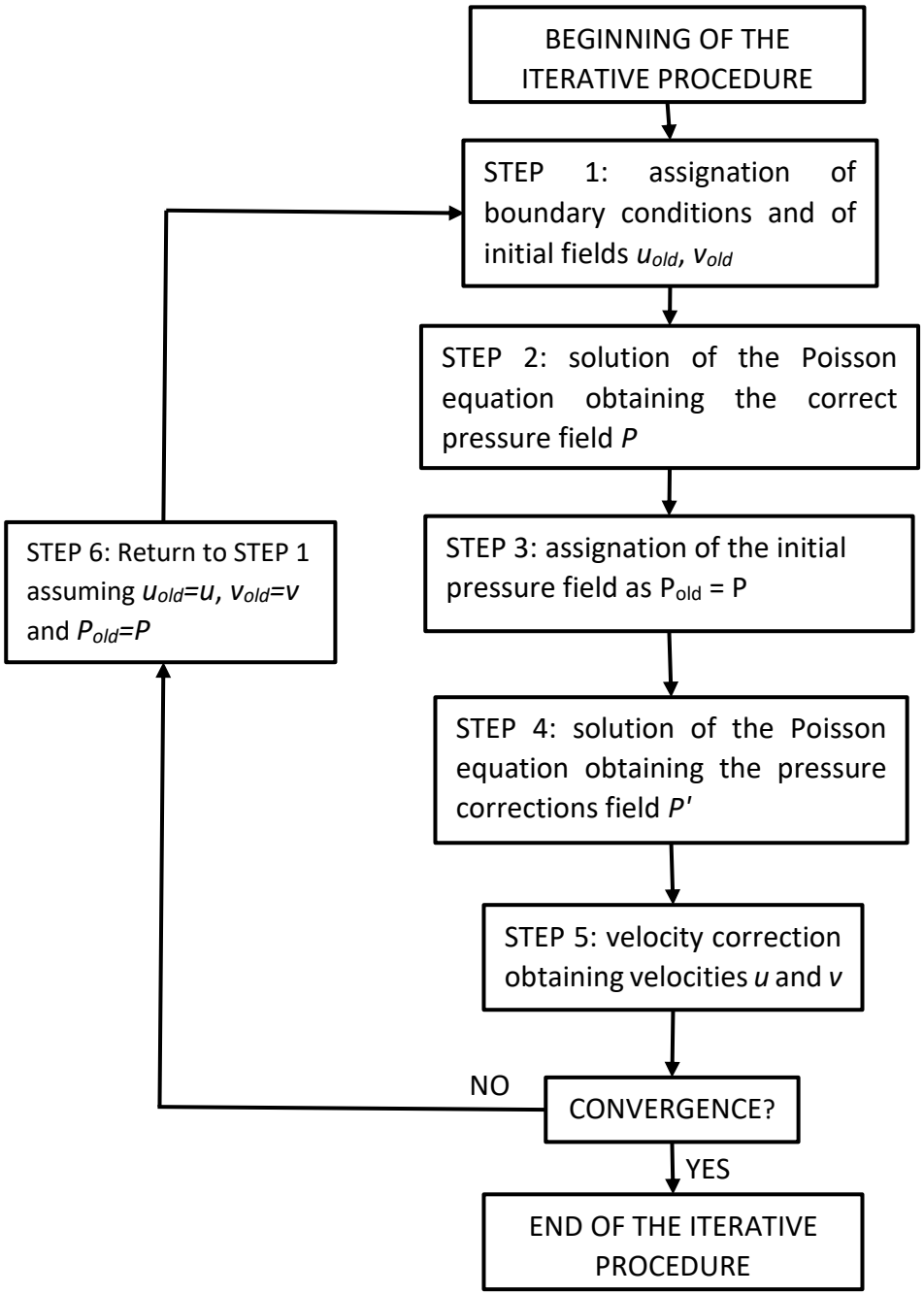

Figure 3.9: flowchart of the iterative process of the SIMPLER algorithm.

Therefore:

$$\begin{cases} u_e = \hat{u}_e + d_e(P_P - P_E) \\ v_n = \hat{v}_n + d_n(P_P - P_N) \end{cases} \tag{3.48}$$

Where the terms of correct pressure *P* appear, and not its corrections.
The pressure field to be inserted into the system of equations (3.48) is obtainable from the solution of the following system of linear equations (Poisson equations for pressure).

$$a_P P_P = a_E P_E + a_W P_W + a_N P_N + a_S P_S - b \tag{3.49}$$

Where:

$$a_E = d_e \Delta y \tag{3.50}$$

$$a_W = d_w \Delta y \tag{3.51}$$

$$a_N = d_n \Delta x \tag{3.52}$$

$$a_S = d_s \Delta x \tag{3.53}$$

$$a_P = a_E + a_W + a_N + a_S \tag{3.54}$$

$$\hat{b} = (\hat{u}_e - \hat{u}_w)\Delta y + (\hat{v}_n - \hat{v}_s)\Delta x \tag{3.55}$$

RANS turbulence model

The majority of engineering-relevant fluid flows have irregularly fluctuating flow quantities. These variations are frequently at such small scales and high frequencies that resolving them in time and space is prohibitively computationally expensive. It is less expensive to solve for averaged or filtered quantities and approximate the impact of minor fluctuating structures rather than solving for the exact governing equations of turbulent flows (as in Direct Numerical Simulation). Different techniques to modeling these structures are provided by turbulence models. The Reynolds-Averaged Navier-Stokes equations are time-averaged

equations of motion for fluid flow, and before deriving them it is appropriate to present the concepts of kinetic energy *k* and energy dissipation ε.
Starting from the balance equation of momentum for a viscous and incompressible flow,

$$\rho \frac{\partial \vec{u}}{\partial t} + \rho \vec{u} \cdot \nabla \vec{u} = -\nabla P + \rho \vec{f} + \mu \nabla^2 \vec{u} \tag{3.56}$$

it is possible to write an equation of the balance for the kinetic energy *k* in a volume of fluid *V* by multiplying the previous equation scalarly by *u*.

$$k = \frac{1}{2} \int_V \rho \vec{u} \cdot \vec{u} \, dV = \frac{1}{2} \rho \int_V |\vec{u}|^2 \, dV \tag{3.57}$$

Following some mathematical steps [22] an expression for the temporal evolution of kinetic energy in a fluid volume is obtained:

$$\frac{dk}{dt} = \int_V \rho \vec{f} \cdot \vec{u} \, dV - \int_V \rho \nu |\nabla \vec{u}|^2 \, dV \tag{3.58}$$

The last term of this equation contains the definition of the dissipation rate of kinetic energy per unit mass.

$$\varepsilon = \nu |\nabla \vec{u}|^2 \tag{3.59}$$

Therefore, the expression for the temporal variation of kinetic energy can be written in compact form by replacing the definition for ε.

$$\frac{dk}{dt} = \int_V \rho \vec{f} \cdot \vec{u} \, dV - \int_V \rho \varepsilon \, dV \tag{3.60}$$

The newly found relationship allows to see that in the expression for the variation of kinetic energy *k*, only the terms derived from volume forces and viscosity terms are included, while no trace of non-linear and pression terms is preserved. Because *k* is the kinetic energy of a fluid system integrated on *V*, it follows that the convective and pressure terms do not affect the

overall energy balance, but rather act on its transfer, whether from one point to another in space or through the many scales of kinetic energy. The other observation is related to the definition of ε which, being positive defined, indicates the continuous decrease in kinetic energy of the system due to this term. In other words, if in the relation (3.60) there were no term containing then the kinetic energy would inexorably decrease over time until a flow at rest is obtained. Conversely, if a flow $\vec{f}$ is statistically stationary then its kinetic energy is constant, and equation (3.60) implies that the energy input into the system in constant quantities over time (constant power) is dissipated at the same rate by the viscosity of the fluid.

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 in the instantaneous Navier-Stokes equations is decomposed into its mean, or averaged, value and its fluctuating component, according to the Reynolds hypothesis. For a generic variable φ this hypothesis translates into:

$$\varphi = \bar{\varphi} + \varphi' \tag{3.61}$$

Where it is indicated with $\bar{\varphi}$ the average value of φ and with φ' its fluctuating component. Substituting this expression (in the case when φ represents the velocity u and the pressure P) in the general equation of conservation of the momentum of a fluid yields the following form of the Navier-Stokes equations averaged according to Reynolds [22].

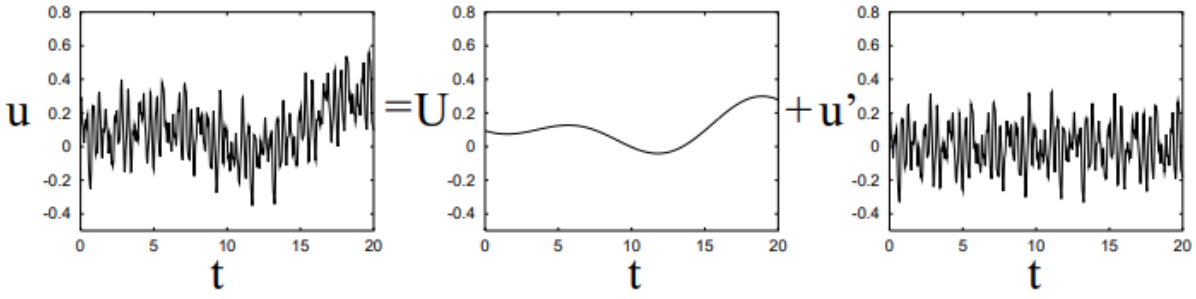

Figure 3.10: decomposition of a statistically non-stationary signal in its ensemble average part and fluctuating part *[22]*.

$$\frac{\partial \rho \vec{\bar{u}}}{\partial t} + \nabla \cdot \left(\rho \vec{\bar{u}} \vec{\bar{u}}\right) = -\nabla \cdot \bar{P}\tilde{I} + \nabla \cdot (\tilde{\bar{\tau}} + \tilde{\tau}_{RANS}) + \vec{f} \tag{3.62}$$

In which it is denoted with $\tilde{I}$ the identity tensor, $\tilde{\bar{\tau}}$ the mean stress tensor, and $\tilde{\tau}_{RANS}$ the Reynolds stress tensor. The equation just derived is essentially identical to that obtained in (2.26) with the difference that a further term now appears in the contribution of the stress tensor, represented precisely by the Reynolds stress tensor, which can be expressed in the following form:

$$\tilde{\tau}_{RANS} = -\rho \begin{pmatrix} \overline{u'u'} & \overline{u'v'} & \overline{u'w'} \\ \overline{v'u'} & \overline{v'v'} & \overline{v'w'} \\ \overline{w'u'} & \overline{w'v'} & \overline{w'w'} \end{pmatrix} + \frac{2}{3}\rho k\tilde{I} \tag{3.63}$$

Having indicated with *k* the turbulent kinetic energy. It is now evident that the problem is no longer of closed form, since the number of equations has remained the same, equal to 4 (considering the continuity equation), but the number of unknowns has risen to 13 (*u*, *v*, *w*, *P* and $\tilde{\tau}_{RANS}$). The challenge is thus to model in terms of the mean flow quantities, and hence provide closure of the governing equations. Two basic approaches are used in Simcenter STAR-CCM+: Eddy viscosity models and Reynolds stress transport models. The most common Eddy viscosity models are based on the Boussinesq approximation, which introduces the concept of a turbulent (eddy) viscosity μ_t allowing for a characterization of the stress tensor as a function of mean flow quantities. The eddy viscosity models in Simcenter STAR-CCM+ [23] solve additional transport equations for scalar quantities that enable the turbulent viscosity to be derived. Spalart-Allmaras, *k*-*ε* and *k*-*ω* are such models. Some considerations can be advanced on how to choose the most suitable turbulence model based on the application: Spalart-Allmaras is convenient when the separation, if it occurs, is contained (very common in external aerodynamic applications of aircraft vehicles, very unsuitable instead in case of strong recirculations or in the presence of natural convection); *k*-*ε* is a good compromise in terms of stability and computational cost, and is convenient in the presence of strong recirculations (with or without heat exchange); *k*-ω are similar to Spalart-Allmaras and are recommended for external aerodynamic applications. Reynolds Stress Transport Models are the most complex and recommended only in case of strong anisotropic turbulence and complex flows.

Standard k-ε model

The most frequent model used in computational fluid dynamics to simulate mean flow characteristics for turbulent flow conditions is the *k*-*ε* turbulence model. The model was created

with the goal of improving the mixing-length model and finding an alternative to algebraically prescribing turbulent length scales in particularly complex flows. The *k*-*ε* model is a two-equation model that uses two transport equations in the mathematical form of partial differential equations to give a general representation of turbulence. It solves one transport equation for the turbulent kinetic energy and one for the turbulent dissipation rate in order to determine the turbulent eddy viscosity. The formulation suggested by Jones and Launder [24], with coefficients suggested by Launder and Sharma, is typically called the "Standard" K-epsilon Model.

In these models, the turbulent kinetic energy *k* and its dissipation rate ε are used to create, by dimensional considerations, a scale of lengths,

$$\ell = \frac{k^{\frac{3}{2}}}{\varepsilon} \tag{3.64}$$

and time.

$$\mathcal{T} = \frac{k}{\varepsilon} \tag{3.65}$$

With these quantities it is possible to build a turbulent viscosity:

$$\nu_t = C_\nu \frac{k^2}{\varepsilon} \tag{3.66}$$

Where with C_ν is indicated a constant to be determined empirically.

For turbulent kinetic energy and its dissipation rate, the following two empirical correlations respectively apply [25].

$$\frac{\partial}{\partial t}(\rho k) + \nabla \cdot (\rho \vec{u} k) = \nabla \cdot \left(\mu_{eff,k} \nabla k\right) + P_k - \rho\varepsilon \tag{3.67}$$

$$\frac{\partial}{\partial t}(\rho \varepsilon) + \nabla \cdot (\rho \vec{u} \varepsilon) = \nabla \cdot \left(\mu_{eff,\varepsilon} \nabla \varepsilon\right) + C_{1_\varepsilon} \frac{\varepsilon}{k} P_k - C_{2_\varepsilon} \rho \frac{\varepsilon^2}{k} \tag{3.68}$$

Where:

$$\mu_{eff,k} = \mu + \frac{\mu}{\sigma_k} \tag{3.69}$$

$$\mu_{eff,\varepsilon} = \mu + \frac{\mu}{\sigma_\varepsilon} \tag{3.70}$$

$$P_k = -\rho\overline{u_i' u_j'}\frac{\partial u_j}{\partial x_i} \tag{3.71}$$

The formulation of the production term P_k depends on the K-Epsilon model variant.
The values of the model constants have been arrived at by numerous iterations of data fitting for a wide range of turbulent flows. A widely used set of values is $C_\nu = 0.09$, $C_{1_\varepsilon} = 1.44$, $C_{2_\varepsilon} = 1.92$, $\sigma_k = 1$ and $\sigma_\varepsilon = 1.3$.

Realizable k-ε model

The Realizable K-Epsilon model contains a new transport equation for the turbulent dissipation rate ε. Also, a variable damping function f_ν, expressed as a function of mean flow and turbulence properties, is applied to a critical coefficient of the model C_ν. This procedure lets the model satisfy certain mathematical constraints on the normal stresses consistent with the physics of turbulence (realizability). This concept of a damped C_ν is also consistent with experimental observations in boundary layers.
This model is substantially better than the Standard K-Epsilon model for many applications and can generally be relied upon to give answers that are at least as accurate. Both the standard and realizable models are available in Simcenter STAR-CCM+ with the option of using a *two-layer* approach, which enables them to be used with fine meshes that resolve the viscous sublayer.

Wall treatment

When a turbulent flow approaches a wall, the mean and fluctuation components of velocity, and hence *k*, disappear, resulting in huge gradients. Furthermore, the very high turbulent

stresses away from the wall diminish to magnitudes comparable to the viscous stresses in the near wall layer. As a result, a large number of grid points will be required to resolve the near wall layer. Low Reynolds number turbulence models can simulate the wall's dampening effects, but they do so at the cost of a high number of grid points. This is an inevitable expense that must be paid if precise flow solutions in the near wall region are needed.

The high Reynolds number turbulence approach, as illustrated by the conventional *k*-ε model, on the other hand, avoids the necessity to resolve the near wall layer by using wall functions. This method assumes and superimposes theoretical profiles between the boundary surface and the first near-wall node. Wall functions minimize the computational cost greatly when compared to the previous method. The fundamental drawback of this strategy is that the validity of these profiles can only be determined and justified in near-equilibrium boundary layers [25].

Considering first the region closest to the wall, another scale can be defined, which expresses the thickness of the viscous substrate, through the friction velocity (u_τ):

$$\delta_v = \nu \sqrt{\frac{\rho}{|\tau_w|}} = \frac{\nu}{u_\tau} \tag{3.72}$$

ν is the kinematic viscosity and $|\tau_w|$ is the magnitude of the wall shear stress. Thus, the distance from the wall can be expressed with the non-dimensional distance y^+ with respect to δ_v:

$$y^+ = \frac{y}{\delta_v} = \frac{u_\tau y}{\nu} \tag{3.73}$$

Similarly, based on other dimensional considerations, the wall-tangential velocity component u^+ of the velocity vector can be expressed as:

$$u^+ = \frac{u}{u_\tau} \tag{3.74}$$

Each of the sublayers can be modeled using different empirical approaches. The non-dimensional wall distance y^+ (eq. (3.73)) can be used to define the extents of the sublayers. The following plot shows the non-dimensional velocity u^+ (eq. (3.74)) as a function of y^+ across the three sublayers.

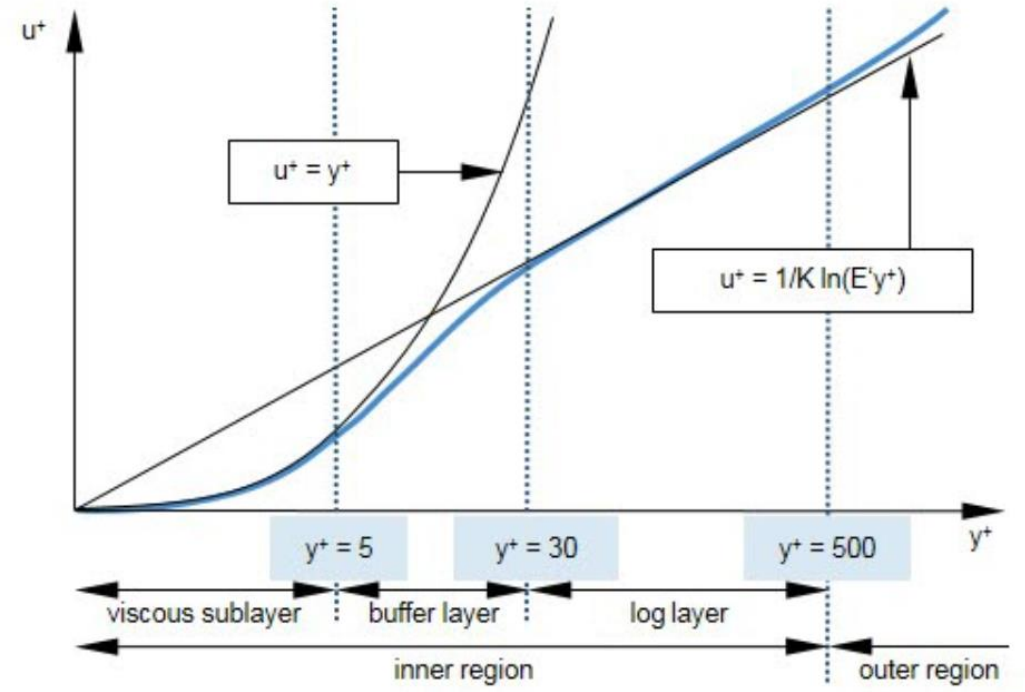

Figure 3.11: representation of the sublayers inside the inner region of the boundary layer *[23]*.

The boundary layer's inner region can be divided into three sublayers. The flow has different features in each of them:

- Viscous sublayer $(0 < y^+ < 5)$: the fluid layer in contact with the wall is dominated by viscous effects and is almost laminar. By adhesion hypothesis, in the vicinity of the wall the velocity is lower, and therefore the Reynolds number is lower and the swirling structures tend to not develop. The mean flow velocity only depends on the fluid density, viscosity, distance from the wall, and the wall shear stress. In the viscous substrate the stress can be considered constant (and equal to the wall stress), a hypothesis that determines the linear dependence of the velocity on the distance from the wall. For this region of the viscous sublayer the correlation between the normalized velocity and the normalized wall distance is linear $(u^+ = y^+)$.
- Log layer $(30 < y^+ < 200)$: away from the wall the inertial terms (and therefore the Reynolds stresses) prevail; the turbulent log layer is dominated equally by viscous and turbulent effects. In this region the correlation between normalized velocity and wall distance follows a logarithmic law of type $u^+ = \frac{1}{K}\ln(Ey^+)$, K being the von Karman's constant (K = 0.4187) and E being an integration constant that depends on the roughness of the wall.

- Buffer layer ($5 < y^+ < 30$): the buffer layer is a transitional layer between the viscous sublayer and the log layer. No mathematical correlation is perfectly representative of this subregion.

Turbulence is negligible in the viscous sublayer, and viscous effects are small in the inertial sublayer, but both effects are important in the buffer layer, with the maximum turbulent production occurring near $y^+ = 12$, with the location slightly dependent on the Reynolds number, making modeling of the flow in the buffer region extremely difficult [26].
STAR-CCM+ provides many types of wall treatment, but the most common and suitable for a wide range of near-wall mesh densities is the *two-layer all-y*$^+$ wall treatment, formulated with the desirable characteristic of producing reasonable answers for meshes of intermediate resolution, that is, when the wall-cell centroid falls within the buffer region of the boundary layer. Additionally, specific values (functions of wall distance) of the turbulence dissipation rate ε are imposed at the centroids of the near-wall cells.

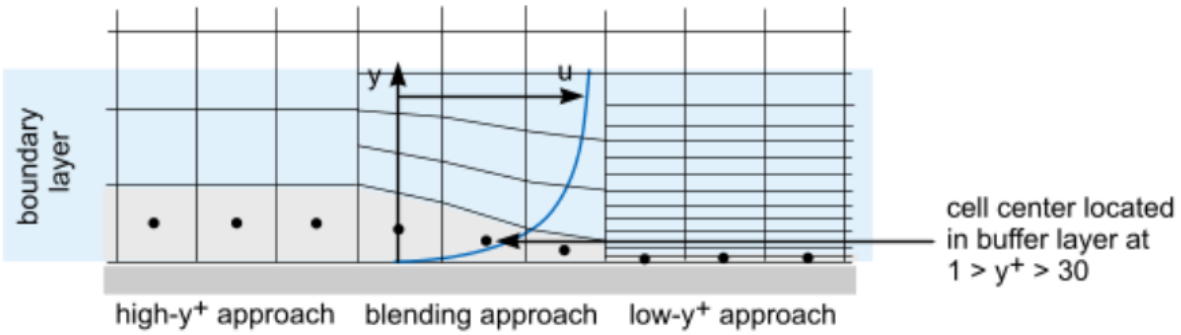

Figure 3.12: STAR-CCM+ *two-layer all-y*$^+$ wall treatment *[23]*.

Background mesh, overset mesh and mesh quality diagnostics

A mesh is a representation of a larger geometric domain by smaller discrete cells. This domain can include real-world geometry, its content, and its surrounding environment. A mesh divides space into elements (or cells or zones) over which the equations can be solved, allowing the solution to be approximated over the larger domain. Within a model, element boundaries may be limited to lie on internal or external boundaries. Higher-quality (better-shaped) elements have better numerical properties, where "better" is defined by the general governing equations and the specific solution to the model instance. Creating a mesh usually requires the creation of a suitable simulation domain. Internal flow, such as a flow in a pipe, and exterior flow, such as the flow around and through an automobile, both require distinct approaches for building the

simulation domain, however, capturing very complicated geometry, such as that found in cars and aircraft, can be really challenging.

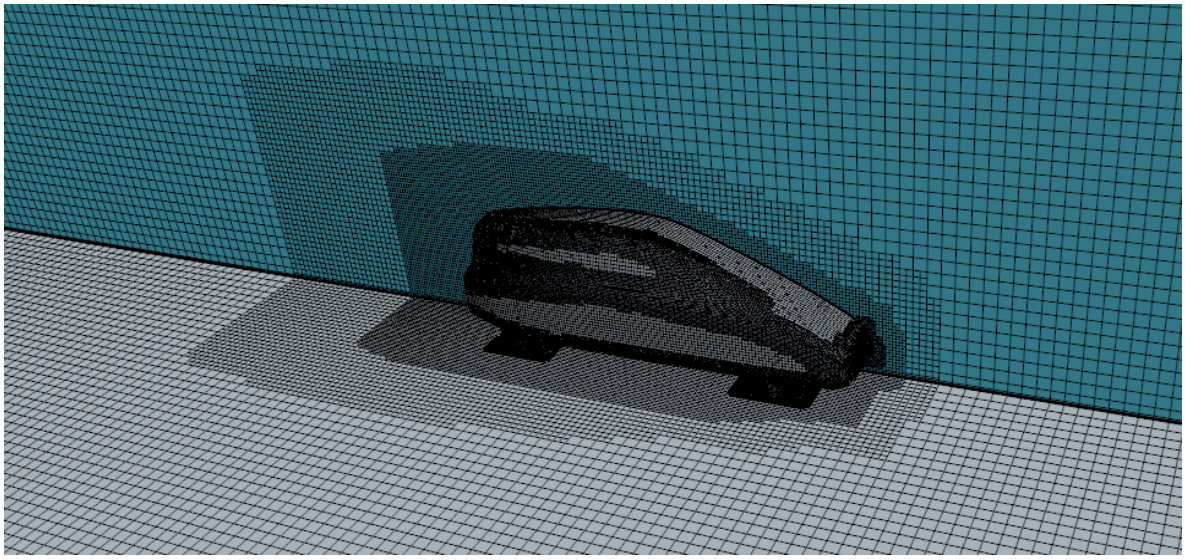

Figure 3.13: visualization of a volume mesh (with clearly visible wake refinement) around a vehicle, created inside STAR-CCM+.

Simcenter STAR-CCM+ solvers find solutions to physics equations at the locations defined by the mesh. For finite volume methods, Simcenter STAR-CCM+ computes values at cell centers. The surface and volume mesh can be generated automatically (an automatic mesh typically contains irregular mesh structures and is generated using tetrahedral, hexahedral, or polyhedral cells) or they can be user-guided.

Accuracy and speed are in tension. Although decreasing the mesh size improves accuracy, it also increases computational cost. Both discretization and solution errors affect accuracy. Even when equations are solved accurately, a given mesh is a discrete approximation of the space and can only yield an approximate solution due to discretization error. For PDEs, several iterations over the full mesh are required for solution error. The calculation is stopped before the equations are completely solved. The choice of mesh element type affects both discretization and solution error. The overall number of elements as well as the shape of individual elements affect accuracy. The number of iterations required relies on the local solution value and gradient relative to the form and size of local elements, and the speed of each iteration grows (linearly) with the number of elements. A mesh of poor quality may miss key elements such as the fluid flow boundary layer. The discretization error will be significant, and the rate of convergence will be slowed; the solution may not even converge. This is why it is so important to always make sure that the mesh generated meets certain important requirements

that determine its overall quality. The most important parameters that measure the quality of a calculation grid are:

- Skewness angle: is the angle that the normal of the face separating two adjacent cells forms with the straight line joining the two centroids of those cells and measures how much the geometric shape of a cell is dissimilar from the ideal shape. A well generated mesh exhibits cells each with skewness angle smaller than 85 degrees. Skewness angle equal to 0 indicates a perfectly orthogonal mesh.

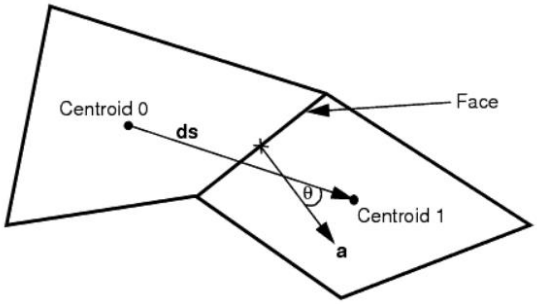

Figure 3.14: two-dimensional diagram of a face and the cell centroids on either side of the face *[23]*.

- Face validity: an element with good face validity is characterized by external face normals pointing in the opposite direction to the centroid of the cell; if the face validity is poor then one or more normals point towards the centroid of the cell. This parameter has to be as close to 1 as possible, if less than 0.5 indicates the presence of negative volume cells and should be avoided.

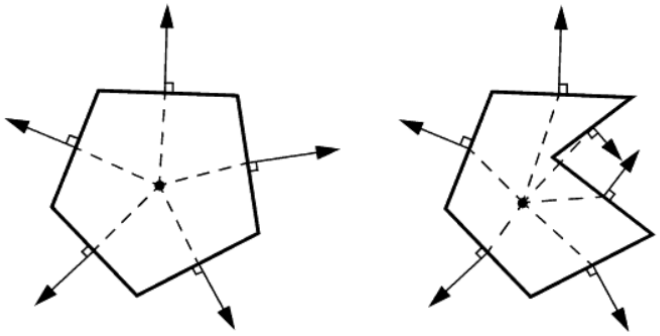

Figure 3.15: good cell validity (*left*) and bad cell validity (*right*).

- Cell quality metric: an element with highly non-orthogonal faces has a low cell quality. If the cell quality is close to 1 it is of high quality, if close to 0 it is of low quality, generally with a value below 10^{-5} it is considered of poor quality.

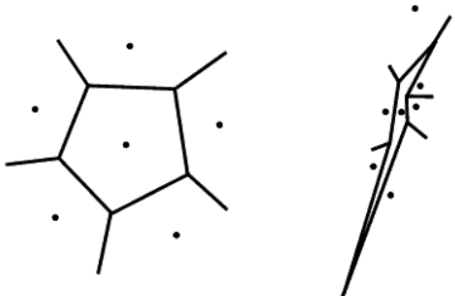

Figure 3.16: good cell quality (*left*) and bad cell quality (*right*).

- Volume change metric: is the ratio of the volume of the cell to the volume of the larger neighboring cell. To be acceptable, a cell must have volume change greater than 0.01.

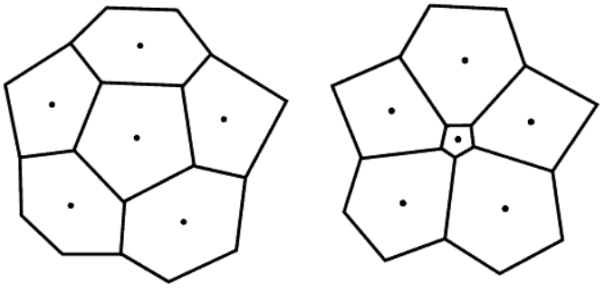

Figure 3.17: good volume change metric (*left*) and bad volume change metric (*right*).

- Chevron Quality indicator: if the line joining the centroids of two adjacent cells does not pass through the common face, for those cells the Chevron indicator is 1, while the chevron indicator is 0 for cells where the opposite situation occurs.

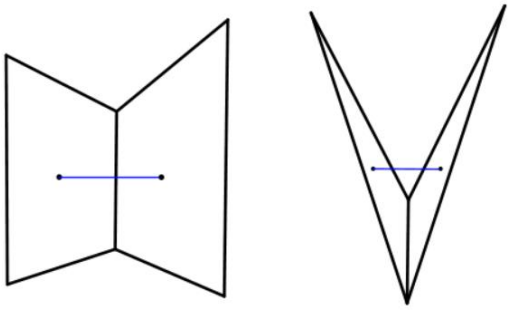

Figure 3.18: normal cells (*left*) and Chevron cells (*right*).

- Aspect ratio: is the ratio of the maximum size to the minimum size of an element. Best meshing practices suggest that this number must be smaller than 5.

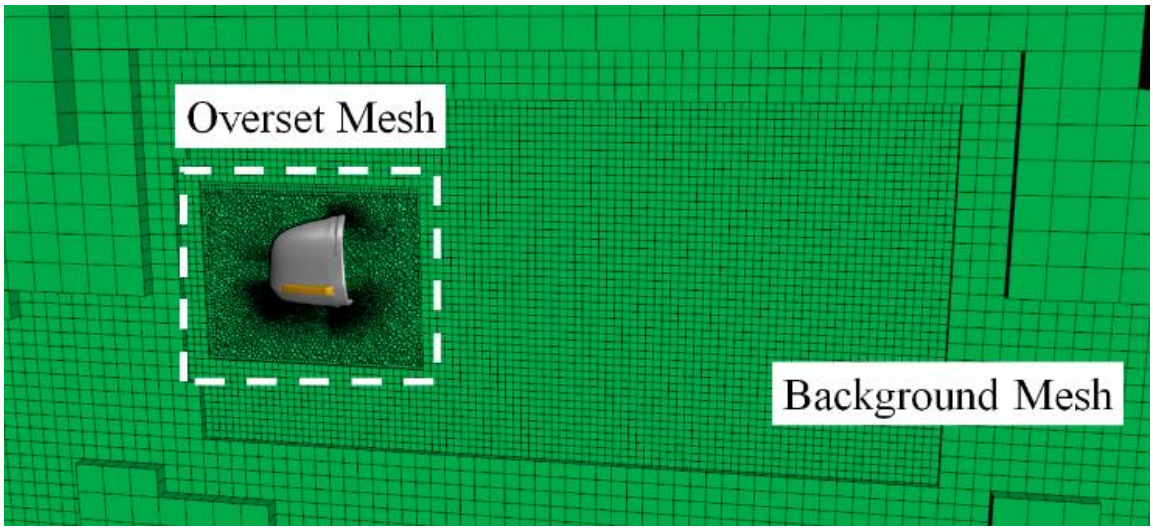

Figure 3.19: overset mesh region overlapped to the background mesh *[27]*.

Overset meshes are used to discretize a computational domain by overlapping many separate meshes in an arbitrary way. They're particularly effective in situations involving large motions (such as those involving bodies that are close together or have intersecting parts), as well as optimization/parametric studies in which a geometry can be encased in an overset zone and moved around, orienting identical bodies in different relative positions or immerse different bodies in the same environment. A typical overset simulation has a background region enclosing the entire solution domain and one overset region that surrounds a body. However, as will be shown in the next paragraph, using the overset mesh approach for a simulation involving the modeling of fluid film demands certain requirements to be met.

3.3.2. MULTIPHASE FLOW

In fluid mechanics, multiphase flow is the simultaneous flow of materials with two or more thermodynamic phases within the same system where distinct interfaces exist between the phases. There are two types of topologies: dispersed flows and segregated flows. The former is made up of discrete particles, droplets, or bubbles spread throughout a continuous phase, whereas the latter is made up of two or more continuous fluid streams separated by surfaces. Simcenter STAR-CCM+ provides many distinct models to meet the requirements of these two categories of flow. The Fluid Film model predicts the dynamic characteristics of wall films using boundary layer approximations and assumed velocity and temperature profiles across the

depth of the film. Film transport is predicted using thin shells that lie across the surface of solid walls on which the film is formed.

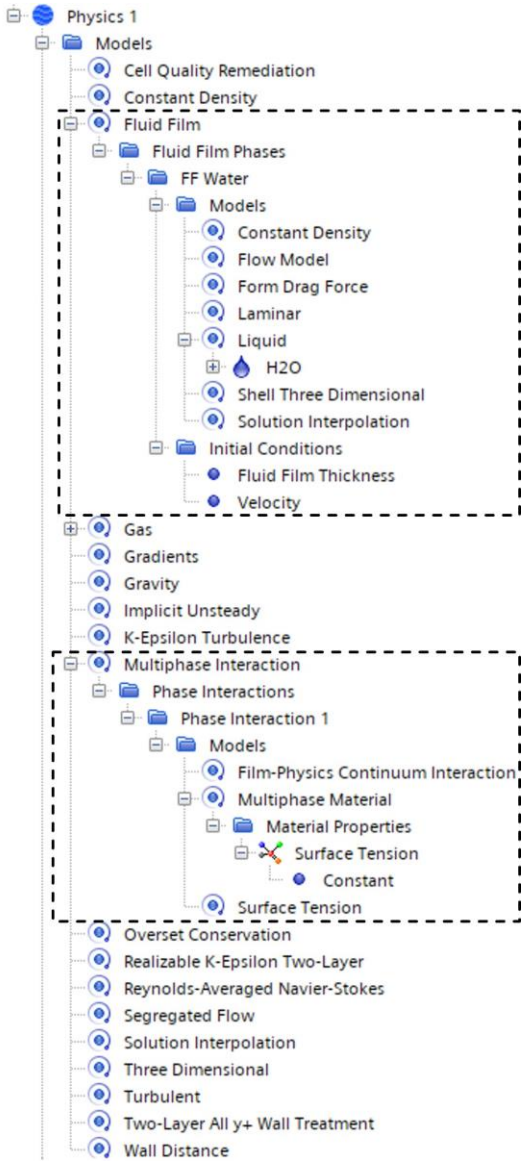

Figure 3.20: physics simulation tree with detail of Fluid Film model and Multiphase Interaction set up.

Modeling fluid film

In engineering practice, problems involving a thin film of fluid on solid boundaries are very prominent. Complex interactions between the fluid film and the surrounding environment might develop depending on the conditions. Further mathematical modeling is required to capture the intricacies of the created flows. Multiphase flows, where several fluids flow in the domain of interest, play an important role in variety of industrial applications and due to the presence of an interface over which there is a leap in fluid-fluid characteristics and the

interchange of mass, momentum, and heat between the phases, numerical simulations of such flows need to handle more complexity than single phase flow simulations. The Simcenter STAR-CCM+ two Eulerian Multiphase (EMP) Fluid Film model provides a mathematical description of the behavior of such films, being successfully used for modelling dispersed flows as well as multi-scale flow situations.

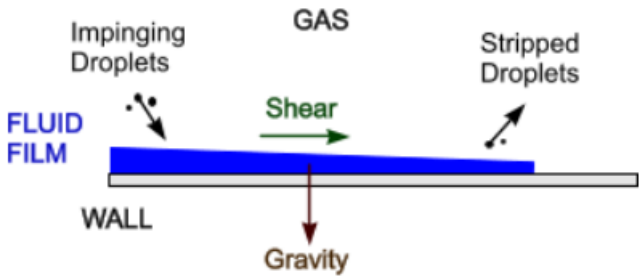

Figure 3.21: fluid film formation and stripping process from a wall-gas interface *[23]*.

A fluid film can be formed and removed as a result of many different phenomena. It is possible to initialize a layer of fluid film on the surface and specify its properties, or one can proceed by specifying an inlet from which the fluid film is emitted and an outlet from which it is removed. Other mechanisms of fluid accumulation on surfaces are droplet impingement (droplets carried by the air collide with surfaces, forming a liquid layer) and condensation from the gas phase. The fluid film can also be reduced in thickness or be removed through the following mechanisms: wave stripping (wave instabilities of the fluid film strips away fluid droplets), edge stripping (fluid droplets are stripped away from the surface because of sharp edges), evaporation and boiling. Similarly, the processes of generating and removing the fluid film from the surface can be modeled through the concepts of mass sources (and sinks). In the simulations carried out in this work, it was decided to generate the fluid film layer with an inlet-outlet mechanism.

The fluid film model solves mass, momentum, energy, species, and volume fraction transport equations. The model assumes that the film is thin enough to be approximated by the laminar boundary layer approximation and also assumes that the velocity profile across the film thickness follows a parabolic trend. The assumption of laminar boundary layer is violated if turbulence occurs in the film.

The continuity equation that governs mass conservation within the fluid film is as follows:

$$\frac{\partial}{\partial t}\int_V \rho_f \, dV + \int_A \rho_f \, \vec{v}_f \cdot \vec{n} dA = \int_V \frac{S_u}{h_f} dV \tag{3.75}$$

Where ρ_f and $\vec{v}_f$ are respectively the density and velocity of the fluid film, the quantity S_u represents the source term of the mass of fluid film per unit area and h_f symbolizes the thickness of the fluid film. The eq. (3.75) is used for the calculation of h_f.

The equation for the conservation of momentum of the film is expressed as shown below.

$$\begin{aligned} &\frac{\partial}{\partial t}\int_V \rho_f \vec{v}_f \, dV + \int_A \rho_f \, \vec{v}_f \otimes \vec{v}_f \cdot \vec{n} dA \\ &= \int_A \tilde{T}_f \cdot \vec{n} \, dA - \int_A P_f \, dA + \int_V \vec{f}_b + \frac{\vec{S}_m}{h_f} dV \end{aligned} \tag{3.76}$$

Having indicated with $\tilde{T}_f$ the viscous stress tensor within the film, P_f the pressure, $\vec{f}_b$ the body force vector (related to the surrounding fluid) and with $\vec{S}_m$ the momentum source corresponding to the mass source S_u.

Assuming that the normal components of the viscous and convective terms are negligible, the pressure distribution within the fluid film is obtained from eq. (3.77) as:

$$P_f(\xi) = p_{int} - \vec{S}_m \cdot \vec{n} - \rho_f \vec{f}_b \cdot \vec{n}\big(h_f - \xi\big) + \int_\xi^{h_f} \frac{d}{dt}\big(\rho_f \vec{v}_f \cdot \vec{n}\big) \, d\xi \tag{3.77}$$

Being ξ the local coordinate normal to the wall and $\vec{n}$ the wall surface unit vector pointing towards the film.

For a multi-component film, the mass conservation equation for the i^{th} species is:

$$\frac{\partial}{\partial t}\int_V \rho_f y_{i,f} \, dV + \int_A \rho_f \vec{v}_f y_{i,f} \cdot \vec{n} \, dA = \int_A \left(\frac{\mu_f}{\sigma}\right) \nabla y_{i,f} \cdot \vec{n} \, dA + \int_V \frac{S_{i,u}}{h_f} dV \tag{3.78}$$

Where σ is the molecular Schmidt number, $y_{i,f}$ and $S_{i,u}$ are the mass fraction and mass source (or sink) of the i^{th} species respectively.

A fluid film is not modeled directly within a regular Simcenter STAR-CCM+ region. Instead, Simcenter STAR-CCM+ uses a shell region, a surface domain in space that is effectively a two-

dimensional one cell thick region with edges as boundaries, that represents the space within which the fluid film flows. An interface connects one shell region to a region or another shell region. When a shell region is built, an interface between the original fluid region and the shell region is automatically created at the boundary.

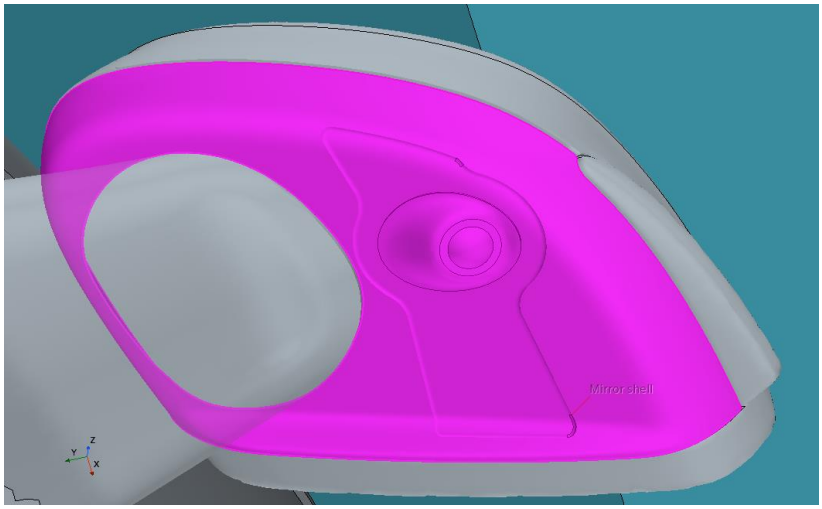

Figure 3.22: shell region created on the underside of the side-view mirror of a Volvo vehicle.

The strategies used by fluid film solvers are similar to those used by Segregated Flow solvers. Differences arises from the fact that these solvers deal with film equations on surfaces and film control volumes are dependent on the film distribution.

4. VEHICLE AERODYNAMICS

Aerodynamics is defined as the branch of fluid dynamics that studies the dynamics of gases, mainly air, around solid bodies. With regard to automotive applications, the concept of Aerodynamics therefore refers to the study of the behavior of the air around and through the structure of the vehicle. For low vehicle speeds the airflow around the vehicle mainly affects acceleration while for high vehicle speeds, it affects fuel consumption and vehicle management. The interaction of the air flow with the vehicle can be of three types: air flux around the vehicle, air flux through the engine compartment, air flux through the transmission system and inside the engine [28]. While the first two are closely related, the third requires independent study. The optimization of the aerodynamics of the vehicle, i.e. the optimization of these three types of interaction, is aimed at reducing fuel consumption, ensuring greater and comfort of the vehicle (noise reduction, reduction of mud deposition on the vehicle, ventilation/cooling of the passenger compartment) and improve driving conditions. Here the study of the flow outside the vehicle that generates forces and moments that greatly affect the performance of the vehicle and its directional stability will be addressed.

For a moving land vehicle, the effects of air viscosity are detectable only in the boundary layer, adjacent to the body of the vehicle. Outside the boundary layer the air can be treated as an inviscid fluid, which exerts a pressure force on the boundary layer itself. When the air reaches the rear of the vehicle it detaches. The existence of the boundary layer depends on the value of the Reynolds number: the formation of the boundary layer occurs for values of *Re* greater than 104. The Reynolds number is influenced by the characteristic length and speed of the vehicle and the kinematic viscosity of the fluid. The motion of the fluid around the vehicle depends on the shape of the vehicle and the Reynolds number.

Another important aerodynamic phenomenon is the formation of the vehicle's wake: when the air detaches from the rear of the vehicle, it generates the formation of a large turbulent region of low pressure known as the wake. The wake contributes to the formation of pressure drag, which reduces the performance of the vehicle. The influence of aerodynamic drag increases considerably with the speed of the vehicle, becoming more impactful on performance.

The aerodynamics of vehicles on the road differ from the aerodynamics of aircraft for a few different factors: the shape of the vehicle is less tapered than that of an aircraft, the car is in close contact with the ground and does not travel in free air, the operating speeds are much

lower, the ground vehicle has less degree of freedom than an aircraft and its movement is less influenced by aerodynamic forces.

The principles of aerodynamics are the basis of the design of vehicles on the road: the evolution that the shape of cars has undergone over the years has made it possible to improve the aerodynamics of the vehicle. While at the beginning of the twentieth century the shape of the vehicles resembled that of a boat or an airship, in the ‘20s the Hungarian Paul Jaray brought back to the automotive field the experience gained in the aeronautical field. Jaray formulated design principles for the aerodynamics of vehicles on the road, which were collected in a patent issued in 1927 [29].

Figure 4.1: boat shaped road vehicle (1900-1920).

Jaray's patent spread rapidly and the design he promoted became more mainstream. Mercedes, Opel, Maybach, and numerous other brands, mainly German, made models using the Jaray patent, which then became a real formula. The biggest challenge of those times was the "Streamlining", which led to several possible forms for the vehicle. In this period much attention was paid to the phenomenon of separation of the boundary layer.

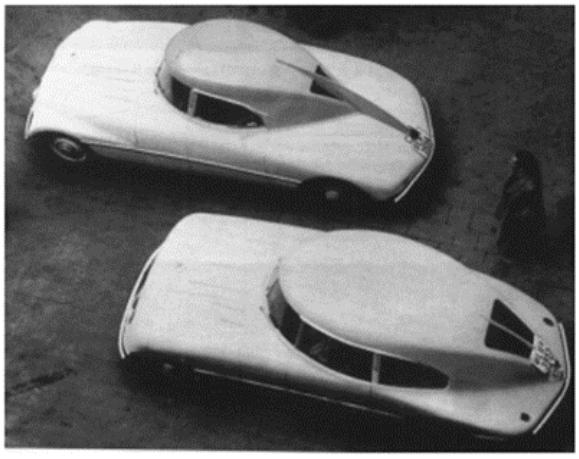

Figure 4.2: Jaray’s patented model.

Between the 70s and the 90s much attention was paid to the optimization of details, conducted through the use of wind tunnels. New models such as the sedan and SUVs were designed and manufactured. Since the 90s, engineers have mainly focused on optimizing the basic shape of the vehicle using new tools such as CFD. The main objective of this period was to make the body of the vehicle more streamlined and characterized by the minimum value of the coefficient of aerodynamic drag.

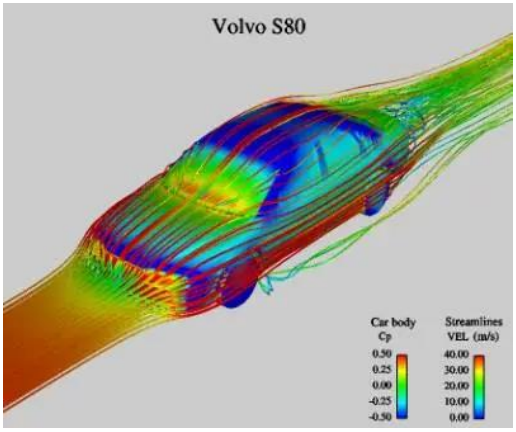

Figure 4.3: path lines around Volvo S80 using CFD.

The aerodynamics of on-road vehicles influence numerous factors: from the car's energy efficiency related to aerodynamic drag, to the vehicle's environmental impact in terms of air pollution and noise. Good prediction of the aerodynamic characteristics of a vehicle on the road is a complicated task and requires the combination of both experimental and numerical techniques. The experimental and numerical tools currently available are mainly used for the reduction of drag force. However, there are many other aerodynamic phenomena such as lateral wind stability, instabilities due to tunnel passages, on platforms or instabilities related to the transit of other vehicles and aeroacoustics, which require more sophisticated approaches for flow prediction [29]. In recent years there have been several advances in both experimental and numerical techniques used for the study of aerodynamics. As for numerical methods, time-dependent simulations have been introduced. In general, both approaches are currently used to improve vehicle characteristics by controlling flow pattern or optimizing shape. The objectives behind the aerodynamic study conducted by the designers are the improvement of the vehicle's performance, the stabilization of the car in windy conditions, the improvement of comfort and visibility of the driver. Although the aerodynamic study incorporates several aspects, the present thesis work is focused on the analysis of the interaction of the flow with the external body of the vehicle in order to reduce the resistance of the car under normal working conditions.

4.1. WIND TUNNELS

Although the ideal methods for the aerodynamic study of a vehicle are the one that measure performance when the car is directly on the road, with which the rotation of the wheels, the flow in the underbody and the natural atmospheric wind are considered, these are hindered by several problems. The main problem lies in measuring the drag separately from the rolling resistance, friction resistance and mechanical losses of the vehicle. Moreover, the equipment required to carry out an analysis on the road involves many practical problems. The only two alternative methods for conducting aerodynamic evaluations are computational fluid dynamics (CFD) and experimental wind tunnel tests.

The wind tunnel is a laboratory tool used to study the flow of a fluid (air for the problem under consideration) around a body, simulating its interaction in a plausible way. The measurements conducted in the wind tunnel concern the local and global speeds, pressure, temperature, and forces exerted by the fluid on the body. In wind tunnels the stationary body is hit by a flow at the speed required for the specific object to be tested. For economic reasons often the models used are scale reproductions. The greatest difficulty of this approach is linked to the inconsistency of the Reynolds numbers on the real application and on the scaled model. The measured forces are not simply multiples of the forces that would be measured with the original scale, due to the diversity of the Reynolds number. So, keeping in mind the equation that describes this parameter, the methods used to solve this drawback are to increase the speed of the flow or use pressurized wind tunnels, to increase the density of the air and therefore keep the Reynolds number constant. Wind tunnels are divided into two main categories: open-circuit tunnels and closed-circuit tunnels. A further classification distinguishes closed-loop tunnels as a function of the flow rate in the test chamber or as a function of the Mach number. Thus, there are: incompressible subsonic galleries (if the Mach is between 0 and 0.3), compressible subsonic galleries (if the Mach is between 0.3 and 0.8), transonic galleries (if the Mach is between 1.2 and 5), hypersonic galleries (if the Mach of the flow is greater than 5).

Open-circuit wind tunnel

Open-circuit tunnels are typically made using convergent-divergent nozzles. The entrance to the wind tunnel is represented by a circular or rectangular section in which devices are placed

for the quality control of the incoming flow. It follows a converging duct that ends at the initial point of the test chamber, a region with a constant section representing the nozzle throat. In the test chamber is placed the prototype to be subjected to the aerodynamic study. The prototype will then be subjected to speeds of the order of 50-70 m/s. Subsequently, the fluid passes through a divergent section in which fans connected to an electric motor are positioned. The fans transfer the kinetic energy generated by the motor to the fluid which is then recalled in the test section. These components are followed by another diffuser aimed at compressing the fluid and an air outlet section to the external environment. The position of the fans downstream of the test section is due, in addition to the need to provide kinetic energy to the flow, to the fact that the fans generate vortices and turbulence downstream that could alter the measurements made in the throat section. The disadvantages for this type of configuration are the noise and loss of kinetic energy at the discharge of the fluid into the environment. The main negative factor, however, is that the test chamber is closed and therefore the pressure inside it is lower than the pressure outside. Therefore, the test chamber must be perfectly sealed to avoid infiltration of fluid from the outside which would cause appreciable undesired alterations of the measurements [30].

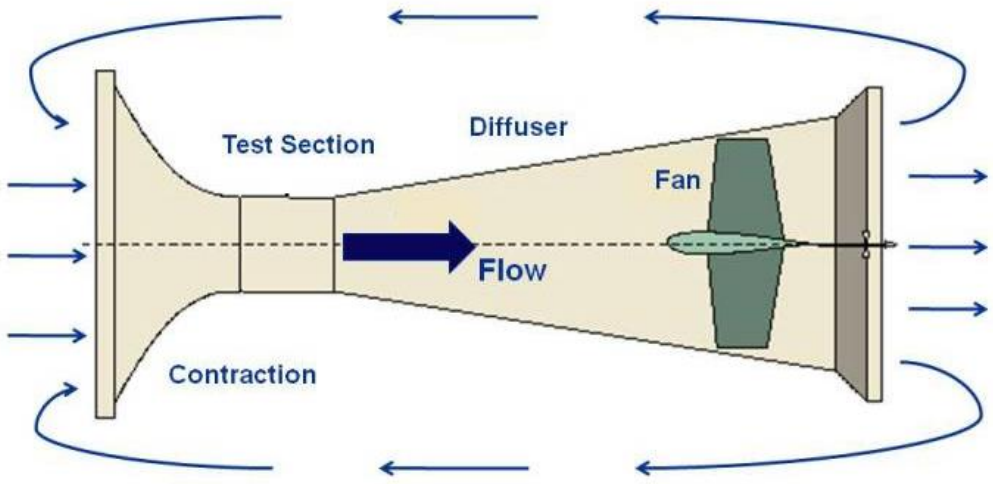

Figure 4.4: open-circuit wind tunnel *[30]*.

Despite several decades of development, the wind tunnel still presents some problems and approximations, because of which the flow around the body is different from the real condition due to various factors, including blocking effects, non-parallel flow, and interaction with the support structures. A CFD analysis can be used with the same approach to simulate a wind tunnel model.

Closed-circuit wind tunnel

Closed-circuit tunnels are equipped with the same components as open-circuit tunnels. The only difference is that the flow is not expelled outside but is recirculated internally. The advantages compared to the open solution lie in the possibility of modifying the characteristics of the flow in terms of pressure, temperature, humidity, and viscosity and in being able to use an open or semi-open test chamber, ensuring a considerable simplification in the positioning of the prototypes to be subjected to experimental tests. However, closed-loop tunnels must be equipped with a heat exchanger and radiators capable of cooling the fluid because the flow undergoes considerable heating that could alter the aerodynamic measurements.

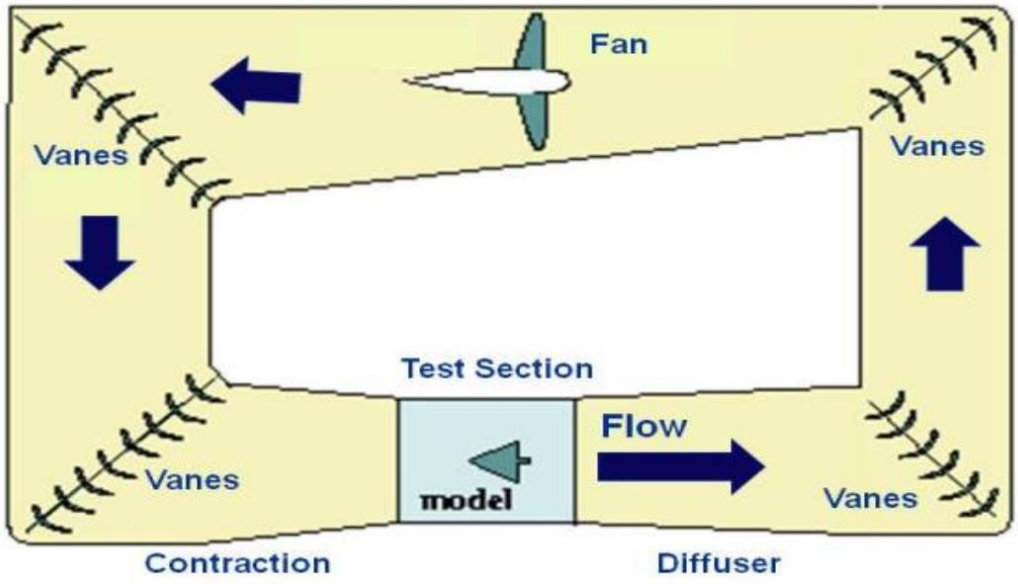

Figure 4.5: closed-circuit wind tunnel *[30]*.

4.2 AERODYNAMIC FORCES ON A VEHICLE

To deal with aerodynamic phenomena it is not possible to disregard one of the most important principles of fluid dynamics represented by Bernoulli's theorem. Bernoulli's equation (eq. (2.45)) represents a simplified model of inviscid flow of an incompressible fluid in steady and irrotational motion. Bernoulli's equation allows to correlate two fundamental quantities in the field of aerodynamics: the speed of a fluid and its pressure. Imposing the above boundary

conditions, Bernoulli's principle states that the sum of kinetic energy, pressure energy and potential energy along a current line is a constant quantity. Considering two points at the same height and thus imposing the negligible nature of the potential contribution, the eq. (2.45) can be simplified as:

$$P + \frac{1}{2}\rho u^2 = const \tag{4.1}$$

in which the first term expresses a local static pressure, and the second term expresses a pressure of a dynamic nature, that is, linked to the speed of the fluid. It can be emphasized that while the static pressure of a fluid is the pressure exerted by it in resting conditions, the dynamic pressure is a measure of the kinetic energy of the fluid per unit volume, which represents the pressure exerted by a fluid due to its kinetic energy. From the relationship of eq. (4.1) it is immediately deducible how as the speed of the fluid increases, its pressure decreases (and vice versa) so that the sum of the two contributions is kept constant. This sum represents a total pressure, which is constant along a flow line (fluid vein of air flow running through a car):

$$P_{stat} + P_{dyn} = P_{tot} = const \tag{4.2}$$

Moreover, at a narrowing of the passage section, by virtue of the constancy of the volumetric flow rate (as established by the continuity equation under the hypothesis of flow stationarity and incompressibility), the speed of the fluid increases, inevitably generating a decrease in static pressure.

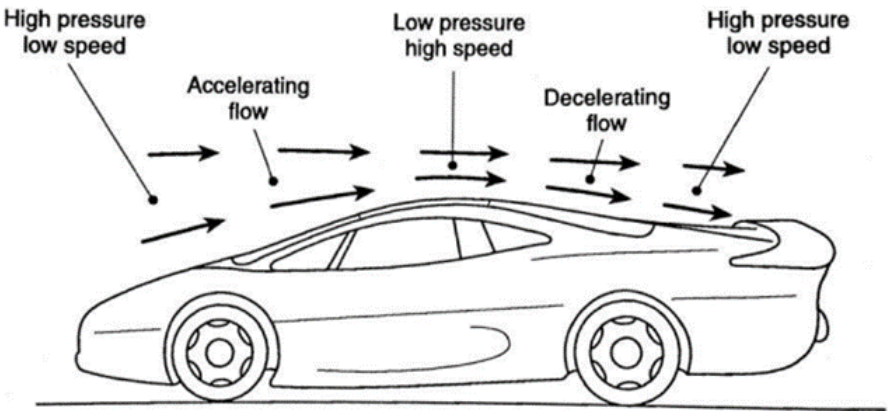

Figure 4.6: trends of static pressure and velocity of air around the vehicle body.

Away from the vehicle the flow has an undisturbed speed. In the impact with the car, this speed is canceled on a point called stagnation point. The flow follows the profile of the vehicle accelerating until it reaches its maximum speed at the maximum thickness of the vehicle. After that the flow decelerates. The trend of the static pressure of the air is exactly mirrored to the trend of its speed.

The pressure coefficient is an important parameter to be evaluated in an aerodynamic study because it allows to describe the pressure trend along the vehicle profile. During the modeling phase of the vehicle it is extremely useful to evaluate any pressure changes due to geometric anomalies of the car body. The pressure coefficient (C_P) is expressed by the following relation:

$$C_P = \frac{P - P_\infty}{\frac{1}{2}\rho u_\infty^2} = 1 - \left(\frac{u}{u_\infty}\right)^2 \tag{4.3}$$

Where P and u are respectively the local static pressure and the local velocity, P_∞ and u_∞ are respectively the static pressure and the velocity of the undisturbed current, i.e. the fluid (upstream or downstream) away from the vehicle. The value of the pressure coefficient can take values ranging from 0 to 1:

- is equal to 0 at any point in the flow field sufficiently far from the vehicle so as not to be altered by the vehicle;
- is equal to 1 at the points where the flow is at rest, called stagnation points;
- is between 0 and 1 if the air flow is characterized by a velocity gradually lower than that of the vehicle, that generates areas of overpressure;
- is negative in areas where the static air pressure is lower than the undisturbed pressure. This occurs at the points of circumvention of the car body, at which there is a very marked curvature of the profile, which leads to a strong acceleration of the flow.

When a solid body is hit by a flow, the pressure actions (normal to its surface) and the friction forces (tangent to its surface) that fluid threads exert on it, contribute to a single force called aerodynamic resultant. This force can be broken down into 3 components (*fig. 4.7*): lift force (acting along the *z*-axis), drag force (acting along the *x*-axis), deviance (due to lateral winds or gusts of wind along the *y*-axis) [31]. Only the first two components will be specifically analyzed here since the numerical simulations of interest are conducted for a vehicle under normal conditions.

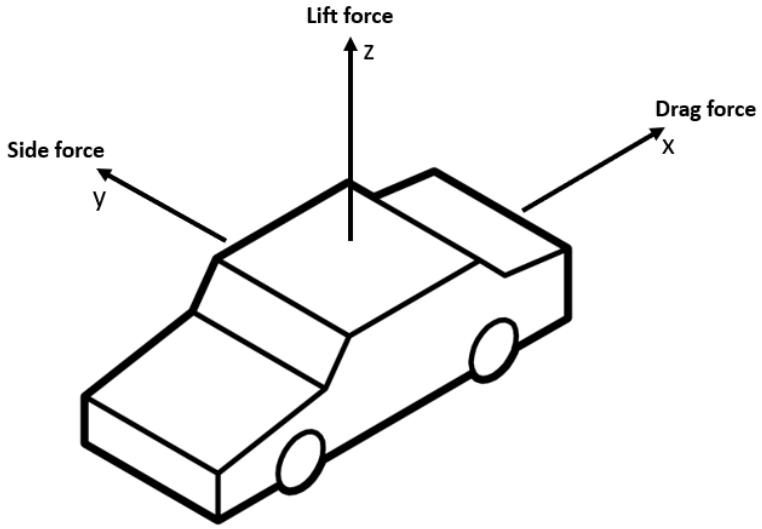

Figure 4.7: aerodynamic forces acting on a car body.

In the automotive sector, aerodynamic resistance is defined as the force that the air exerts in a horizontal direction by opposing the advancement of the vehicle. The aerodynamic resistance is proportional to the speed of a laminar flow and to the square of the speed of a turbulent flow. The expression of the aerodynamic drag force is as follows:

$$D = \frac{1}{2}\rho u_{\infty}^{2} A C_D \tag{4.4}$$

Where ρ is the density of the air, u is the velocity of the flow, A is the reference surface (frontal area of the vehicle), and C_D is the drag coefficient. This relationship shows that aerodynamic drag is proportional to the square of the velocity. The drag coefficient is a function of both the shape of the vehicle and the Reynolds number and is expressible through the relation obtained from the eq. (4.4).

$$C_D = \frac{D}{\frac{1}{2}\rho u_{\infty}^{2} A} \tag{4.5}$$

The drag coefficient regulates the aerodynamic efficiency of the vehicle, i.e. fuel consumption and speed performance. As far as a vehicle is concerned, aerodynamic resistance can be traced back to four types: skin friction drag, pressure drag, vortex-induced drag and interference drag [28].

The skin friction drag is a function of the Reynolds number. In fact, the skin friction drag is due to the effect of viscous forces in the boundary layer, the origin of which is linked to the non-ideality of the fluid. The thickness δ of the boundary layer of the latter quantifies its effect: the greater the thickness and the greater the resistance that meets the air near the body of the vehicle. This results is a decrease in the penetrability of the vehicle into the air. In a laminar boundary layer there is a lower resistance to flow than in a turbulent boundary layer. The laminar boundary layer develops at the leading edge of the vehicle profile. At a certain distance from the leading edge the laminar boundary becomes unstable undergoing a transition to a turbulent regime that causes a sudden increase in thickness. The goal is therefore to make the body of the car in such a way as to delay as much as possible the transition from one regime to another. In general, the contribution of skin friction drag, being linked to the presence of the boundary layer in which the viscous effects of the fluid are not negligible, is greater the more delayed is the detachment of the flow.

Pressure drag is the resistance linked to the trend of pressures on the body of the vehicle due to the presence of the boundary layer. At the point of anterior stagnation, the point from which the boundary layer develops, the flow has zero velocity and maximum pressure. In circumventing the body the air accelerates and according to Bernoulli's theorem the pressure decreases to a minimum value, which is reached in the thickest section of the body. Then follows a pressure recovery zone that should theoretically extend to the point of rear stagnation. But due to the progressive increase in the thickness of the boundary layer along the body, it is possible that the reunion of the flow to another stagnation point will not occur. This involves the separation of the flow at the rear of the vehicle which results in the creation of vortices, which contribute to the resistance to advancement. Therefore, because of the lack of complete recovery of the pressure, a downstream pressure lower than the upstream will be detected, producing a greater resistance to the advancement of the vehicle. The pressure or shape resistance is greater the larger the front area of the vehicle and the thickness of the boundary layer and it is the primarily responsible for the formation of the vehicle's wake.

Vortex-induced drag is generated by lift or downforce. The lifting or crushing of the vehicle occurs as a result of the presence of a pressure differential between the lower and upper surface of the vehicle. Since, in general, a flow naturally evolves in the direction of mitigation of any differential, the presence of a certain ΔP between the two parts of the vehicle generates a tangential vorticity between the two surfaces that diverts the flow lines (*fig. 4.8*). The energy needed to generate these fluid structures, called *C-pillar vortices*, translates into greater resistance.

Interference drag is due to the fact that the individual components of a car have their own resistance. When the various components are combined to form the vehicle, the total resistance of the vehicle is always greater than the resistance of the individual parts. It is due to imperfections of manufactures on the body, joints, connecting elements and the coupling of the fairing and wheel fairings to the body.
Among the various contributions to the resistance to advancement for a vehicle on the road the most impactful is represented by the pressure drag, followed by friction drag.

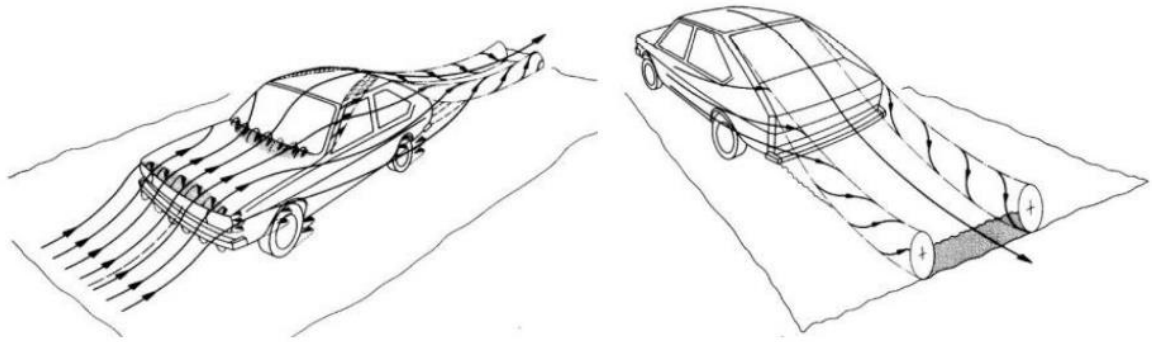

Figure 4.8: C-pillar vortices on a car body *[28]*.

Lift is defined as the aerodynamic force that acts in a vertical direction with respect to the direction of the relative wind. Such a force is responsible for the take-off of an aircraft. The shape of the wings connected to the central part of the aircraft, in the take-off configuration, guarantees a greater passage of air above the airfoil rather than below. This generates a strong depression in the lower part of the wing. Therefore, the airfoil develops an upward direct force that allows the take-off. In road vehicles, this force worsens stability at high speeds, by virtue of a higher top speed. Downforce involves the opposite phenomenon and in vehicles on the road it represents that vertical force that, through the tires, exerts a crushing force towards the ground. The greater the load that acts vertically on the tires, the greater the grip of the vehicle. Lift and downforce are primarily responsible for the stability of the vehicle.

$$L = \frac{1}{2}\rho u_{\infty}^{2} A C_L \tag{4.6}$$

Where C_L is the lift coefficient. As it was done for the eq. (4.5), the lift coefficient can be easily derived from the above expression.

$$C_L = \frac{L}{\frac{1}{2}\rho u_\infty^2 A} \tag{4.7}$$

Methods to reduce drag

There are many ways a designer can improve the performance of a vehicle by acting on its geometry. The most common methods to reduce the drag on a car body are the use of vortex generators, diffuser, rear fairing (fastback) and streamlining [28]. A vortex generator (VG) is an aerodynamic device made out of a tiny vane that is commonly attached to a lifting surface (or airfoil, like an airplane wing) or a wind turbine rotor blade. VGs can also be mounted to parts of an aerodynamic vehicle, such as the fuselage of an airplane or the body of an automobile. When the airfoil or body moves in relation to the air, the VG creates a vortex, which delays local flow separation and aerodynamic stalling by removing some of the slow-moving boundary layer in contact with the airfoil surface, thus improving the effectiveness of flaps, elevators, ailerons, and rudders.

One of the aerodynamic features that contributes to the ground effect is the diffuser. Under the moving automobile body, the diffuser creates a low-pressure zone. A racing car's diffuser accelerates and reduces the pressure of the airflow beneath the car, resulting in a pressure difference between the car's upper and lower surfaces. When a car is moving, the air flow passing through the lower part of the front end accelerates, but when it reaches the diffuser, it encounters a low-pressure zone and expands back to normal speed, providing more downforce and reducing resistance than a wing. This means greater grip given by the aerodynamic downforce, a trick that allows the car to go through a curve at a higher speed. The downforce produced makes it possible to improve the traction force of the tires.

A fastback is a car body style whose roofline slopes continuously down at the back. It is a form of back for an automobile body consisting of a single convex curve from the top to the rear bumper. Fastbacks succeed in the task of reducing the drag on a vehicle by avoiding the boundary layer separation.

A body is said to be streamlined if a conscious effort is made to align its shape with the anticipated streamlines in the flow. Streamlined bodies such as race cars and airplanes appear to be contoured and sleek. Otherwise, a body (such as a building) tends to block the flow and is said to be bluff or blunt. Usually it is much easier to force a streamlined body through a fluid, and thus streamlining has been of great importance in the design of vehicles and airplanes.

5. AUTOMATED DESIGN EXPLORATION WORKFLOW

In this thesis has been presented so far, according to the methodology proposed in this work, the technological tools and software essential to perform a complete parametric analysis on a geometric model of interest. ANSYS, in the role of RBF Morph hosting platform, firstly performs the task of product integration, putting at the service of the morphing tool the GUI of ANSYS Mechanical through its ACT (*Application Customization Technology*) features; secondly, it provides the tools, algorithms and methods necessary to meticulously explore a design space setting up a design of experiment (DOE) table. The commercial morpher RBF Morph, through a vastly intuitive GUI, lets the user set-up a customized geometry parametrization and thus generate each shape variation of the geometry based on the prepared DOE table. STAR-CCM+ provides the working environment in which the high-fidelity CFD numerical simulations decided during the parametric analysis preparation phase are performed. The key feature that allows STAR-CCM+ to receive input morphing instructions is its API (*Application Programming Interface*) functionality: thanks to the writing of custom user-defined functions (UDF) it is possible to manage input and output information from the solver. The UDF created allow to appropriately modify the geometries through the update of the coordinates of the grid vertices.

The prototype demonstration tool developed in this work makes use of each of the aforementioned software and their APIs, integrating and interfacing them with each other by means of C++ user coded applications, and automating the entire process through a low-level scripting of the instructions provided to the machine that performs the operations. This procedure achieves the objective performing a fully automated parametric CFD analysis while integrating disjointed software solutions and proving the feasibility of the workflow by testing it with practical applications in the automotive sector. The entire implemented workflow is represented schematically in *fig. 5.1*.

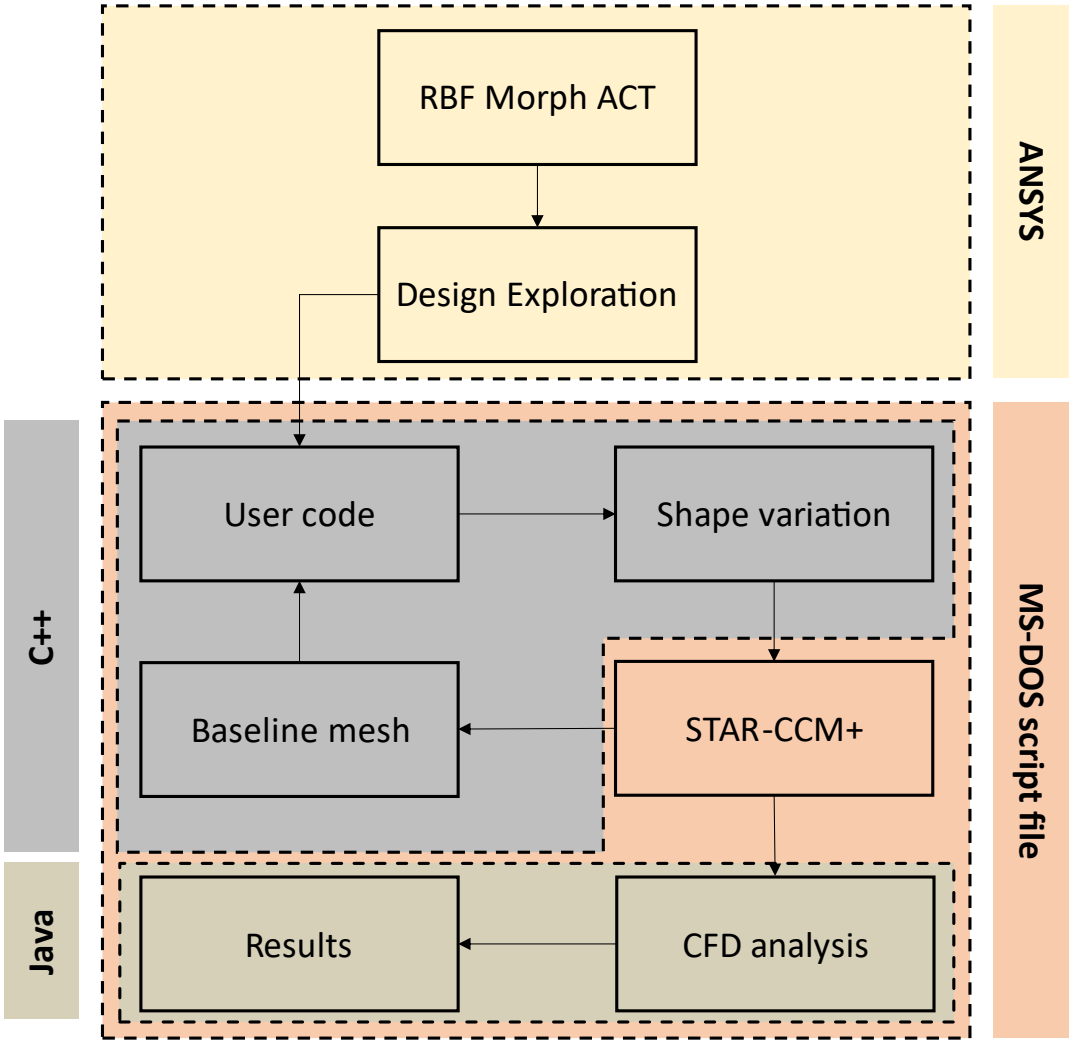

Figure 5.1: flowchart of the automated design exploration workflow.

The first step of the design exploration process takes place within RBF Morph ACT Extension. The software is integrated in the ANSYS Mechanical interface sharing with it the interaction logics such as the selection tools and named selections, and is located in the mechanical three inside the project as shown in *fig. 5.2*.

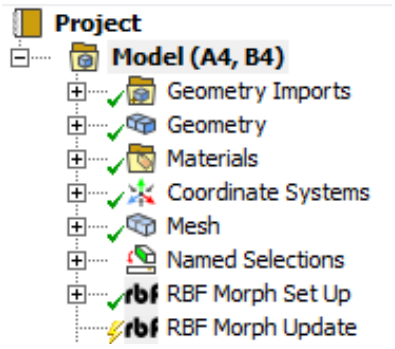

Figure 5.2: RBF Morph ACT integration inside ANSYS Mechanical tree.

The modeling logic for the RBF Morph ACT Extension is hierarchical, allowing for the prescription of displacements at mesh nodes using appropriate shape modifiers (as discussed in chapter 3.2.2). The hierarchical working logic implemented in the morphing tool considers the employment of several children to manage complex mesh alterations. Each child in the RBF Morph tree is a shape modifier that acts on its selection and propagates it to its father.

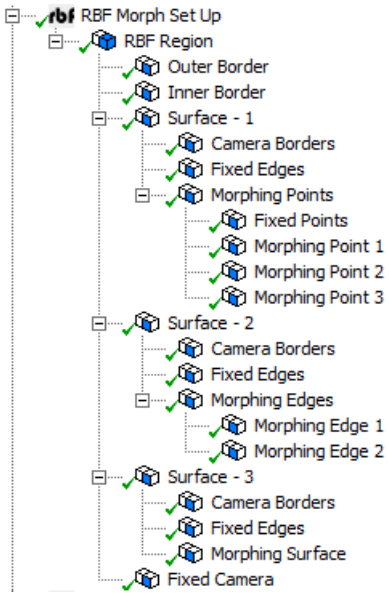

Figure 5.3: expansion of the RBF Morph set-up tree.

Before parameterizations are applied to geometry, a support mesh is generated. For each RBF Source the resulting behavior is interactively previewed for checking the modifications and, once the set-up is completed, each shape modifiers gets parametrized and sent to ANSYS Workbench.

Definition	
Transformation	Translation
Translation Definition	By Coordinate System
Delta x	0 m
Delta y	0 m
Delta z	0 m
Coordinate System	Coordinate System - control points 1 & 2

Figure 5.4: parametrization of the RBF shape modifiers.

Here, a DOE table is generated using an Optimal Space-Filling algorithm in order to obtain as much information as possible on the behavior of the entire parametric space, establishing a maximum of 25 automatically generated shape variants. The range of interest of each shape parameter has been decided by evaluating the mesh in each of the extremes of validity and

making sure that the mesh remains valid and that it still maintains, following the morphing action, an acceptable quality with respect to the requirements on skewness, face validity, cell quality, volume change, etc. At this point, for each shape variant, a file (*.pts*) containing all the morphing data (number, coordinates, and displacements of source points) is generated and exported.

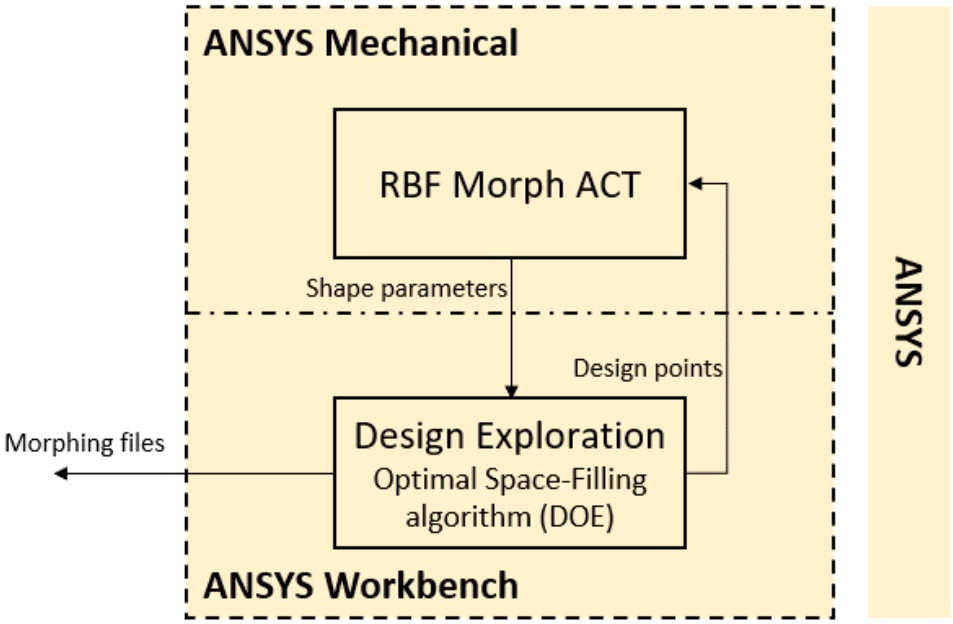

Figure 5.5: detailed diagram of the operations performed inside ANSYS environment.

Once all the morphing files have been generated and exported, to complete the process of updating and analyzing the new geometry, it is necessary to transfer the information contained in them to the STAR-CCM+ fluid dynamic numerical solver. To do this, an adequate calculation grid must first be created on the baseline model, and clearly a CFD study must be performed on the initial model (the results of which are to be used as a benchmark for subsequent simulations). Then, the grid must be exported from the solver, which means that the coordinates of all the vertices that make up the volume mesh must be exported. The coordinates of the vertices are then subjected to the morphing action making calls to functions (*Purge*, *Solve*, *PrintOutput*, *Morph*) contained in the RBF libraries. To accomplish this, a C++ code has been developed, the basic steps of which are described in more detail below:

1. **reading of data stored in the morphing files**: the coordinates and displacements of the RBF centers (referred to the supporting mesh) are read by the code and stored in a dynamic array;

2. **purging of nonessential source points**: the code makes a call to the *Purge* function, which checks the source points stored in the array and discards those that are below a certain minimum distance from the nearby points (so as to not unnecessarily burden the calculation);
3. **encapsulation volume generation**: to ensure a good quality of the morphing action and avoid distant vertices from being affected by the shape changes applied to the surface of the geometry, through the *Encap* function a grid of additional zero displacement source points is generated around the geometry, spaced apart from each other at equal distance;
4. **RBF problem solution**: Source point coordinates and displacements are provided as arguments to the *Solve* function, which solves the RBF problem;
5. **reading of data stored in the baseline mesh file**: the coordinates of the vertices (referred to the baseline mesh built inside of STAR-CCM+) are also read and memorized in a dynamic array;
6. **mesh morphing**: the baseline mesh coordinates are fed to the Morph function and the mesh morphing is finally achieved;
7. **writing to files of the new coordinates of the mesh**: at the end of the calculation, the coordinates (*x*, *y*, *z*) resulting from the morphing process are stored on a new file.

The encapsulation operation performed in *step 3* involves creating a box around the geometry to be morphed, defined by the user by providing the spatial coordinates of an origin (box edge) and the three dimensions of the box in the *x*, *y*, and *z* directions. Once the spatial position of the encapsulation volume is defined, the *Encap* function places a user-defined number of zero displacement source points on the external surface of the box.

The operation performed in *step 5* of the C++ code is made possible by a previous export of the coordinates of the volume mesh generated within STAR-CCM+. This is done by means of the software's API features, which allows the user to visit the nodal positions of all the vertices of the volume mesh and act on them in various ways through user code. User code allows Simcenter STAR-CCM+ to be customized with functions written in a compiled language. User code takes the form of one or more user libraries that are attached to Simcenter STAR-CCM+, each of which contains one or more user functions and a library registration function. When a user library has been attached, its user functions are available for use and are provided in drop-down lists for any operation that needs them. In STAR-CCM+, user functions and library registration functions can be coded in any language, as long as that language is able to bind like C functions or Fortran subroutines. Therefore, C++ was used to write user-defined functions

(UDFs) for the purpose of exporting the coordinates required to perform the morphing. These functions were then compiled and linked together generating a dynamic linked library (DLL). Finally, the DLL was loaded into STAR-CCM+ and the UDFs were made accessible to the simulation file. At the end of the morphing phase, the same method based on UDFs is employed to transfer the morphed coordinates back to STAR-CCM+.

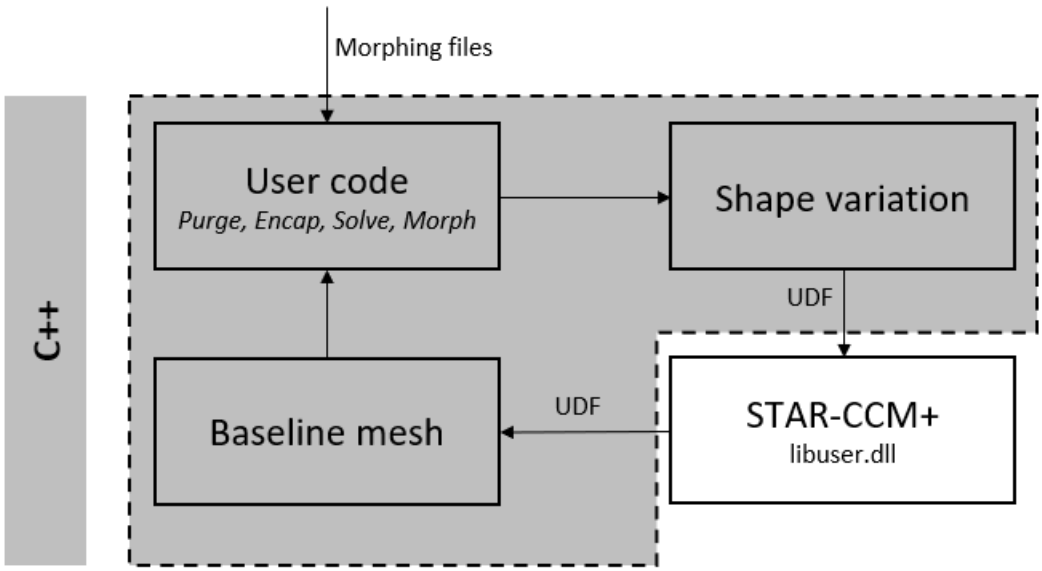

Figure 5.6: detailed schematic of the operations performed by C++ user code.

Each C++ code was compiled using Microsoft Visual Studio Community 2022. When compiling the user-defined functions and linking them together to create the DLL, the x64 Native Tools Command Prompt has been crucial, since dynamic libraries need to be compiled in 64 bits and at a low level to be correctly recognized by STAR-CCM+. A more detailed guide on how to create and use UDFs inside STAR-CCM+ is given in chapter 8.2.

When the numerical solver receives the morphing instructions, the geometry can be updated by moving each of the nodes that make up the mesh. To perform this morphing operation internally to STAR-CCM+, as described above, a UDF is used. Because the solver interprets this update of the nodal points of the mesh as if it were a movement of the model, STAR-CCM+ requires the simulation to be set to *Unsteady* mode before executing the morphing. Therefore, the updating of the numerical model takes place through a fictitious time step, operated with the foresight to keep all the solvers off, avoiding advancing the solution. This functionality is called *User-Defined Vertex Motion* and allows to set a displacement or velocity for every vertex position in the simulation. It is typically intended for use in cases where the other motions cannot give the precise control that is required over the mesh movement.

After updating the model, the simulation file is almost ready to launch the CFD analysis. A custom Java Macro called “CFD_set_up.java” is then activated. The macro performs the following tasks:

1. edit the settings to prepare the simulation to iterate;
2. run the simulation;
3. end the iterations when the simulation stopping criteria are met;
4. export the results.

Java macros are a valuable tool to automate repetitive tasks and to work more efficiently. They can be recorded through the GUI of STAR-CCM+, but most of the times a manipulation of the code is recommended to customize and extend their functionalities.

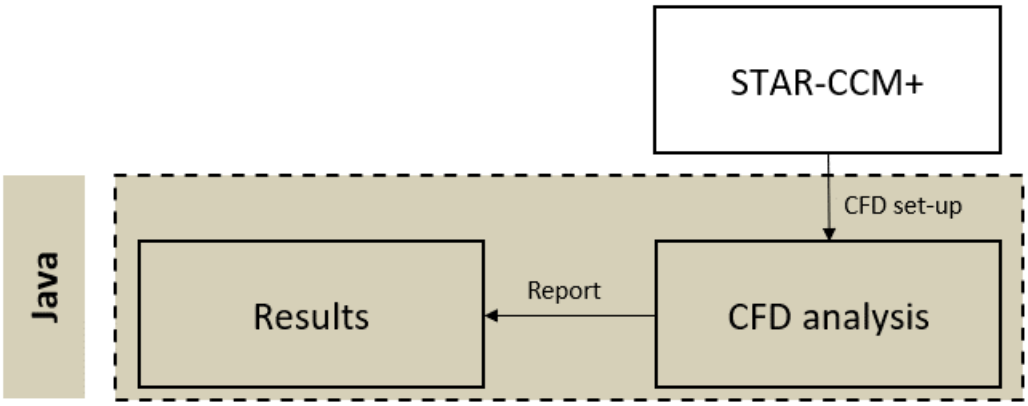

Figure 5.7: detailed schematic of the operations carried out by the Java Macro.

The set of operations described so far, starting from reading the morphing files and ending exporting the results of the fluid dynamics simulation from STAR-CCM+, is enclosed in a script described by a single batch file in MS-DOS, able to automate the entire parametric analysis through a series of commands to be executed by the command-line interpreter of a Microsoft Windows system. The entire parametric analysis is then performed in batch mode, and upon its completion, the log file of the procedure is memorized and stored. The log contains all the information on the actions performed during the execution of the workflow and in addition to allowing the user to monitor the progress of the analysis in real time, it is an excellent debugging resource.

6. APPLICATIONS

This chapter shows the results obtained applying the workflow previously presented. The implemented procedure was put to the test on two technological applications, both of which were primarily demonstration. The initial application involves optimizing the aerodynamics of the ASMO (*Aerodynamics Studien Model*) idealized car body shape, which is publicly available, with the goal of lowering the drag coefficient (C_D). The second application, developed in conjunction with Volvo Cars and RBF Morph, is a design investigation of a detail located around the lens of a camera mounted below a side-view mirror of a Volvo vehicle. The goal of the study is to reduce the thickness of the fluid film layer deposited on the camera in adverse weather and soiling environments.

6.1. AERODYNAMICS STUDIEN MODEL (ASMO)

External car aerodynamics research is critical for overall car efficiency and ride stability and is a crucial component of successful automotive design. The flow over car geometries shows three dimensional and unsteady turbulent characteristics. Around the bluff body, vortex shedding, flow reattachment, and recirculation bubbles can also be detected. These phenomena have a significant impact on the lift and drag coefficients, which are critical for ride stability and energy efficiency. In this chapter, The aerodynamic of the ASMO geometry is studied and optimized.

6.1.1. BASELINE MODEL

ASMO geometry is a model created by Daimler-Benz in the ‘90s to investigate low drag bodies in automotive aerodynamics and testing of CFD codes with a geometry not related to

the development of Mercedes cars. Wind tunnel experiments were made by both Daimler-Benz and Volvo [32]. The shape consists of a square-back rear, smooth surfaces, a boat tail, an underbody diffuser, and no pressure-induced boundary layer separation. The geometry (presented in *fig. 6.1* and *fig. 6.2*) was studied by performing a stationary fluid dynamics simulation of the flow around the model placed in a wind tunnel (*fig. 6.3*), the dimensions of which were chosen to ensure that the most relevant fluid dynamic effects and fluid structures are captured inside it. To accommodate the vehicle inside, the best practice states that it is necessary to leave a minimum space ranging from two to four vehicle sizes upstream and from six to eight measures downstream. Accordingly, the wind tunnel has a length of 8 *m*, a horizontal width of 2 *m*, and develops in height of 1.4 *m*.

Figure 6.1: ASMO (Aerodynamics Studien Model) idealized car body shape.

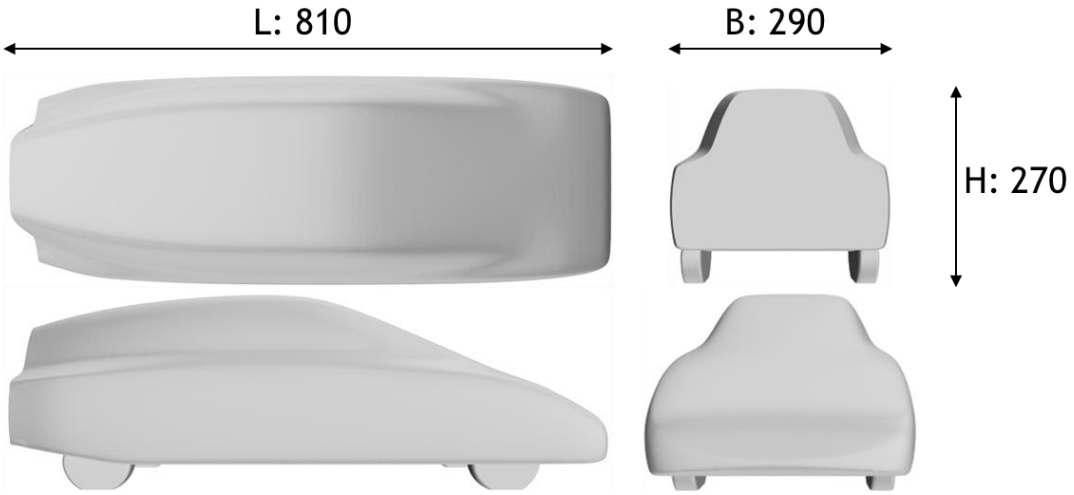

Figure 6.2: ASMO idealized car body shape geometrical projections; measures are expressed in millimeters.

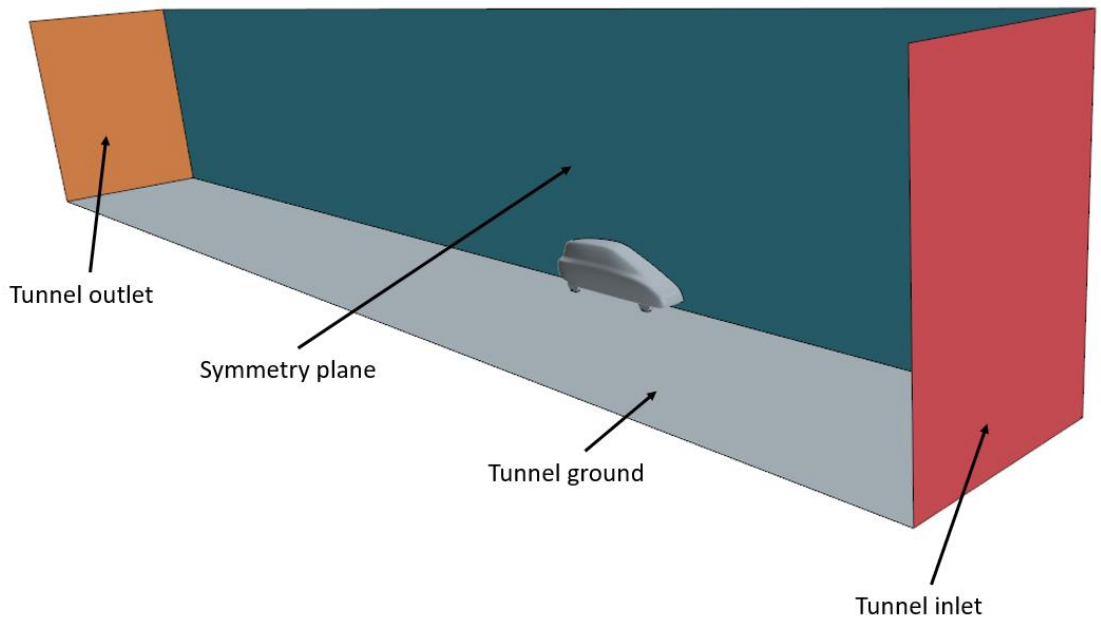

Figure 6.3: wind tunnel geometry and boundary conditions.

The mesh on the model was made with 1,107,966 hexahedral elements and 1,293,064 vertices, taking advantage of the bilateral symmetry of the vehicle. The calculation grid is denser near the surface of the vehicle and less dense elsewhere. This allows to grasp with more accuracy the phenomena that occur near the model rather than far from it, and to limit the overall number of elements of the mesh, reducing the computational cost required for CFD analysis. To capture the wake of the vehicle, a mesh refinement has been created spanning approximately 60% of the car length and spreading with a 20° angle. At the entrance to the wind tunnel, the boundary condition was set with a inlet velocity of the fluid (air) of 50 *m/s*, while at the outlet a zero-reference pressure (0 *Pa*) was set. No-slip conditions are set for the bottom and car surfaces. The air evolving in the tunnel was modeled as an ideal gas ($\rho = 1.225\ kg/m^3$), and since the maximum velocities reached are well below the Mach threshold of 0.3, the fluid was reasonably considered incompressible. The numerical flux solver has been set by selecting a two-layer realizable *k*-ε turbulence model with second-order upwind convection scheme and a segregated flow solver has been employed. The steady state simulation has been terminated when the residuals (continuity, momentum, energy, turbulent kinetic energy and turbulent dissipation rate) reached convergence. The aerodynamic drag and downforce on the vehicle were calculated and the respective coefficients were derived. Numerical values obtained for the drag coefficient (C_D) forces and lift forces (C_L) are 0.143 and -0.035. *Fig . 6.5* displays the pressure and velocity fields at the end of the CFD analysis. In *fig. 6.6*, the pressure coefficient is shown.

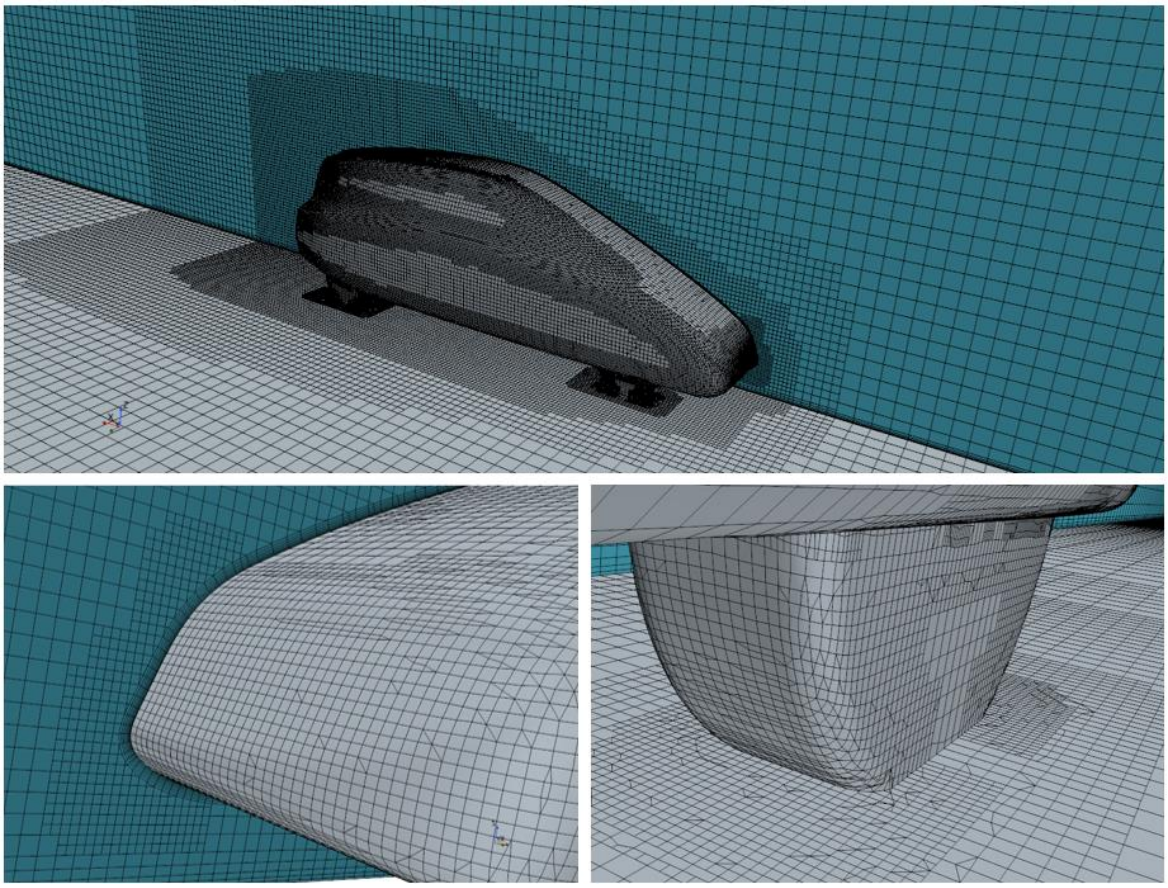

Figure 6.4: mesh build around the model (*top*) with detail of thin prism layers close to the surface (*bottom left*) and refinement in the proximity of the wheels (*bottom right*).

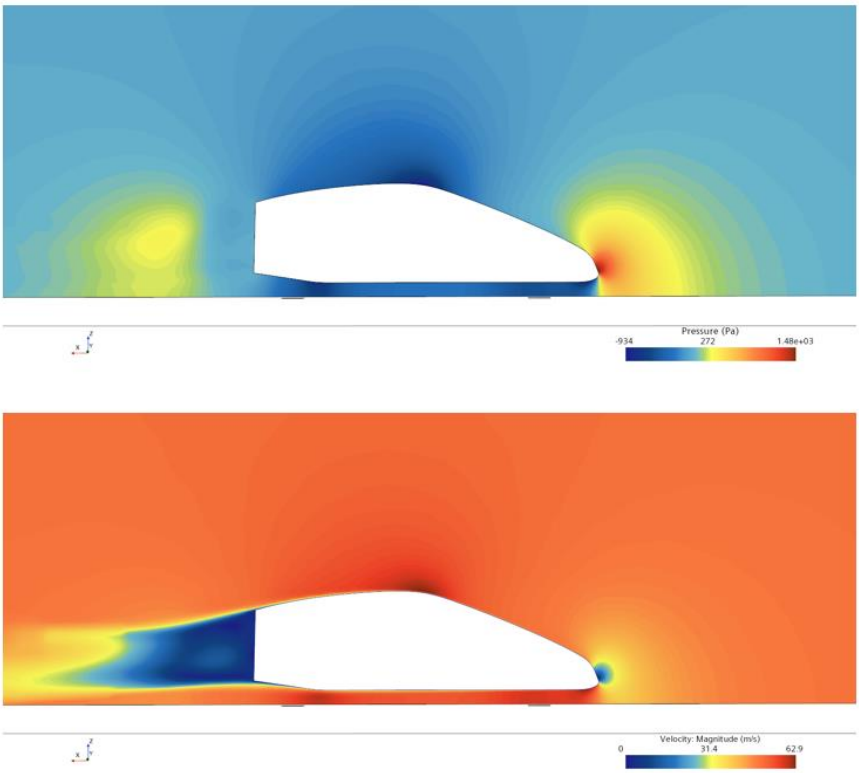

Figure 6.5: flow field pressure (*top*) and velocity magnitude (*bottom*) distributions.

It can be observed that the values of C_D and C_L obtained in the CFD analysis are in good agreement with the drag coefficients obtained from experimental wind tunnel tests (Volvo and Daimler Benz), and with the drag and lift coefficients obtained from other previous studies in the literature (Aljure et al. (2014) [33], Tsubokura et al. (2009) [32] and Perzon et al. (2000) [34]). The drag coefficient of 0.143, calculated as the sum of the pressure contribution and the friction contribution, is only slightly lower than those measured by Volvo (C_D = 0.158) and Daimler Benz (C_D = 0.153). This underestimation of the coefficient ranging from 6.5% to 9.5%, is probably due to the choice of mesh, which in some points of the geometry could benefit from a refinement (i.e. around the wheels). The value of the lift coefficient appears equally consistent with the results of Aljure (-0.023 ≤ C_L ≤ -0.058 obtained by applying several different turbulence models [33]). The roof and the under body pressure coefficient along the symmetry plane of the vehicle confirms the previous considerations.

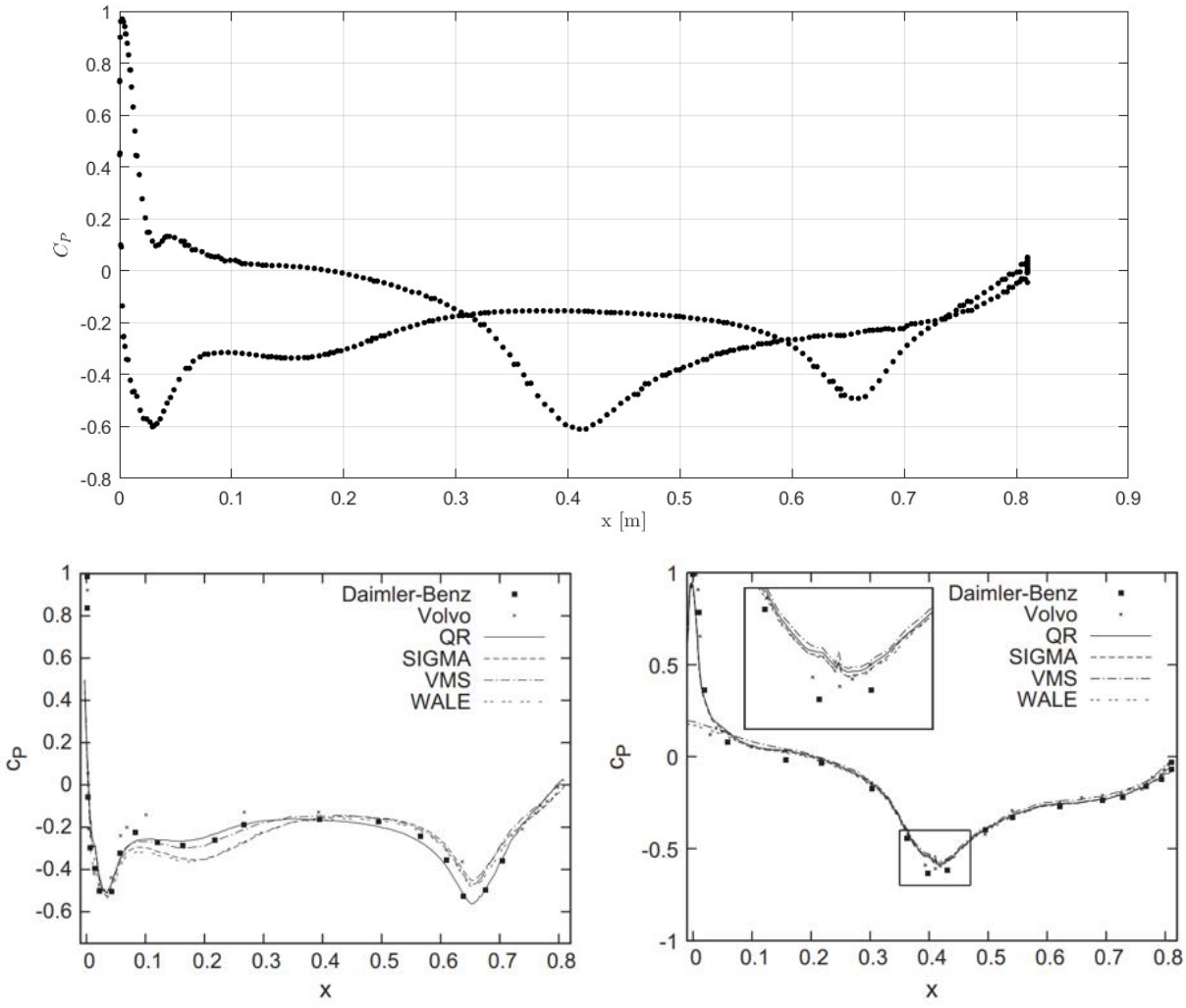

Figure 6.6: pressure coefficient for car roof and under-body (*top*); pressure coefficient in the under-body (*bottom left*) and in the roof (*bottom right*) from Aljure et al. [33].

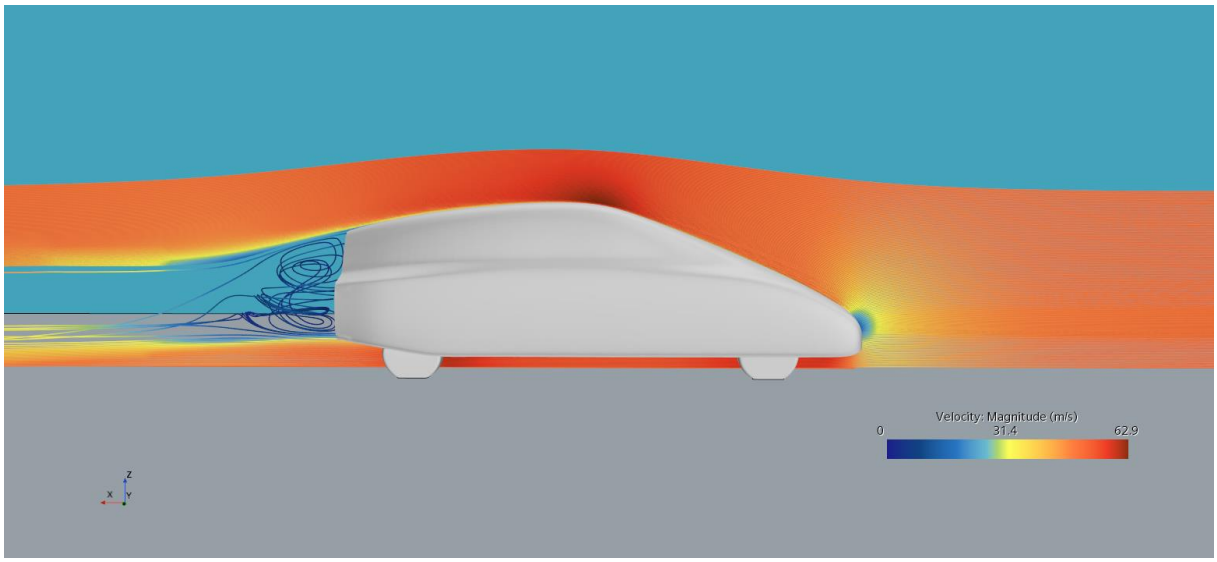

Figure 6.7: streamlines on the symmetry line of the ASMO vehicle.

6.1.2. SHAPE PARAMETRIZATION

The way the ASMO geometry is made, it is already an extremely streamlined shape in itself. However, an improvement in its aerodynamic performance is definitely possible by altering some aspects of the geometry. The parameterization of the shape was carried out using the mesh morphing software RBF Morph, introducing three shape parameters [35] [36](*fig. 6.8*):

- Boat Tail;
- Roof Drop;
- Front Spoiler.

Figure 6.8: visualization of the selected shape parameters.

The most sensitive aspect of the shape parameterization is the design of the shape parameters using RBF Morph's graphical user interface, which entails defining a set of rules for generating source points and extracting information from the mesh itself.

Boat Tail parameter

This parameter consists of a variation of the lower rear end of the vehicle, acting on the width of the back-rear. The variation in shape obtained with amplifications and reductions of this parameter was obtained by rigidly translating RBF centers located on the left and right edge of the back of the car. This parameter was varied between a minimum of -0.01 *m* and a maximum of +0.02 *m*. In *fig. 6.9* the implementation of the shape parameter is shown through a preview display in RBF Morph, highlighting in red the source points in the initial positions and in blue the source points following the morphing action.

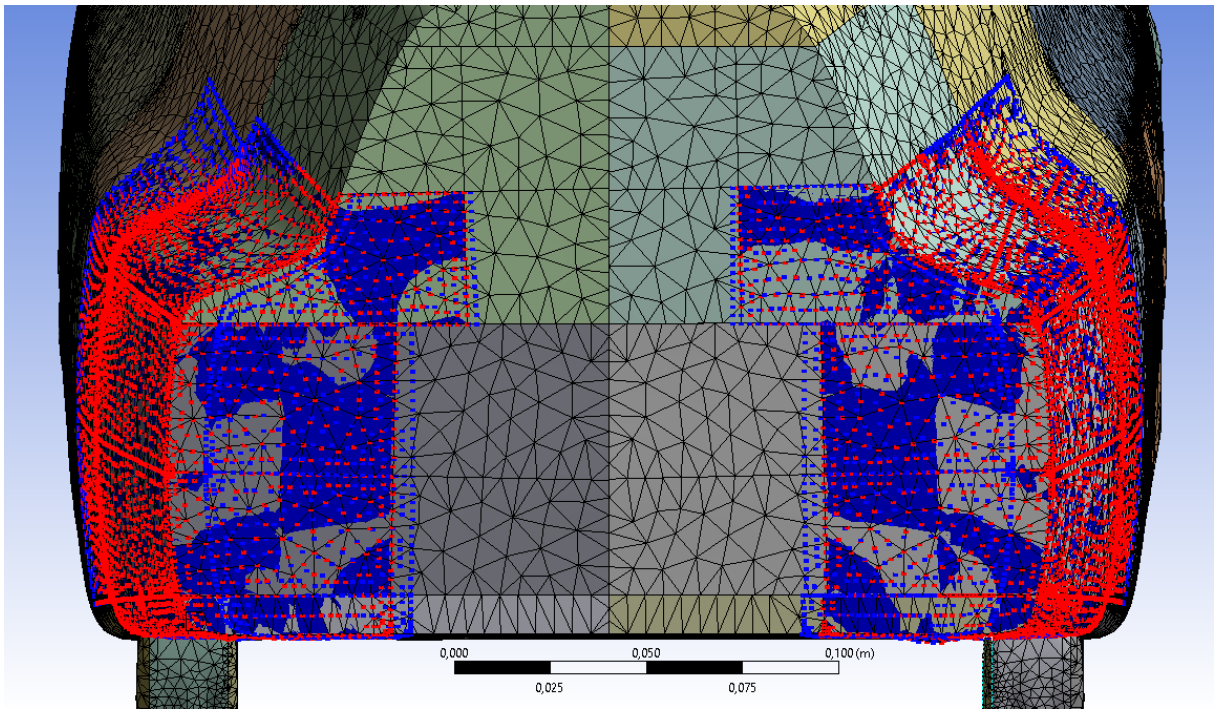

Figure 6.9: preview of the Boat Tail morphing on the ASMO body shape.

	Lower limit [m]	**Upper limit [m]**
Boat Tail (left side)	-0.01	+0.02
Boat Tail (right side)	-0.02	+0.01

Table 6.1: lower and upper limits of the Boat Tail parameter.

Roof Drop parameter

As in the previous case, this parameter acts on the rear end of the vehicle. The Roof Drop parameter controls the inclination of the rear roof, moving up or down a set of RBF centers as shown in *Fig. 6.10*. The range of variation of this parameter, as shown in *table 6.2*, is between a minimum of -0.02 *m* and a maximum of + 0.01 *m*.

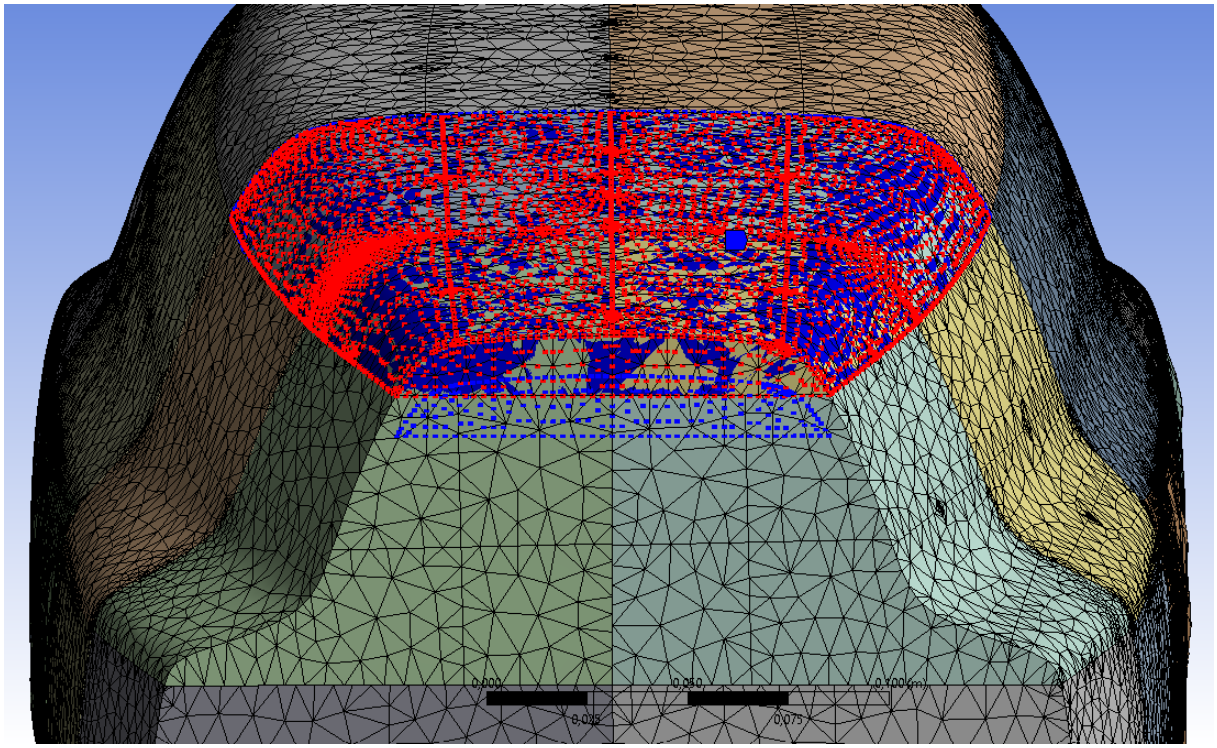

Figure 6.10: preview of the Roof Drop morphing on the ASMO body shape.

	Lower limit [m]	**Upper limit [m]**
Roof Drop	-0.02	+0.01

Table 6.2: lower and upper limits of the Roof Drop parameter.

Front Spoiler parameter

The shape parameter named Front Spoiler controls the shape of the front section of the vehicle model, moving the RBF centers along a direction oriented with an angle of 20° with respect to the transverse direction of the vehicle. The lower and upper limits for the variation of this parameter are respectively -0.0025 *m* and + 0.02 *m*. *Fig. 6.11* displays the RBF set-up on the ASMO model for this final shape parameter.

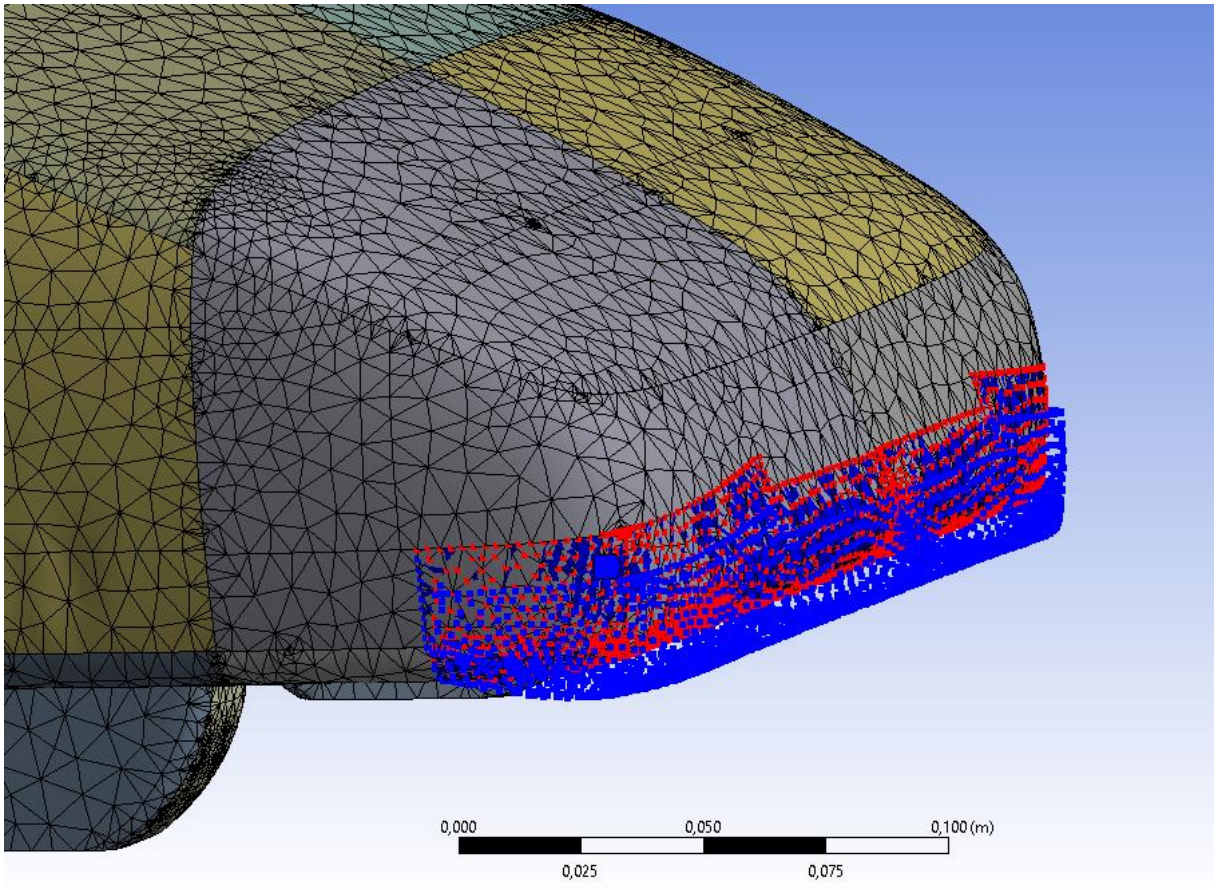

Figure 6.11: preview of the Front Spoiler morphing on the ASMO body shape.

	Lower limit [m]	**Upper limit [m]**
Front Spoiler	-0.0025	+0.02

Table 6.3: lower and upper limits of the Front Spoiler parameter.

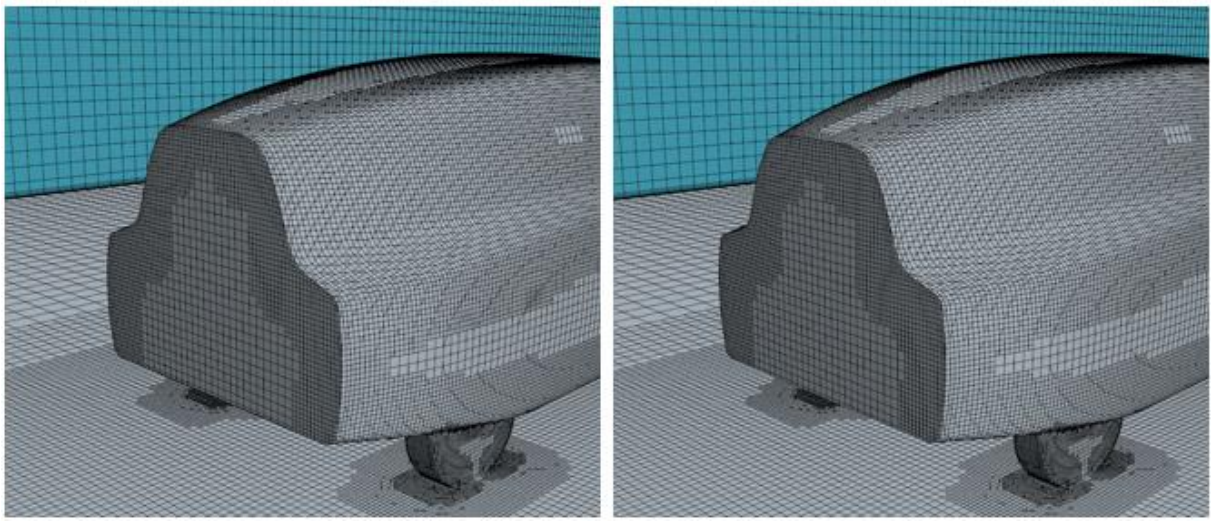

Figure 6.12: comparison between baseline (*left*) and generic morphed geometry (*right*) with focus on the rear end of the vehicle.

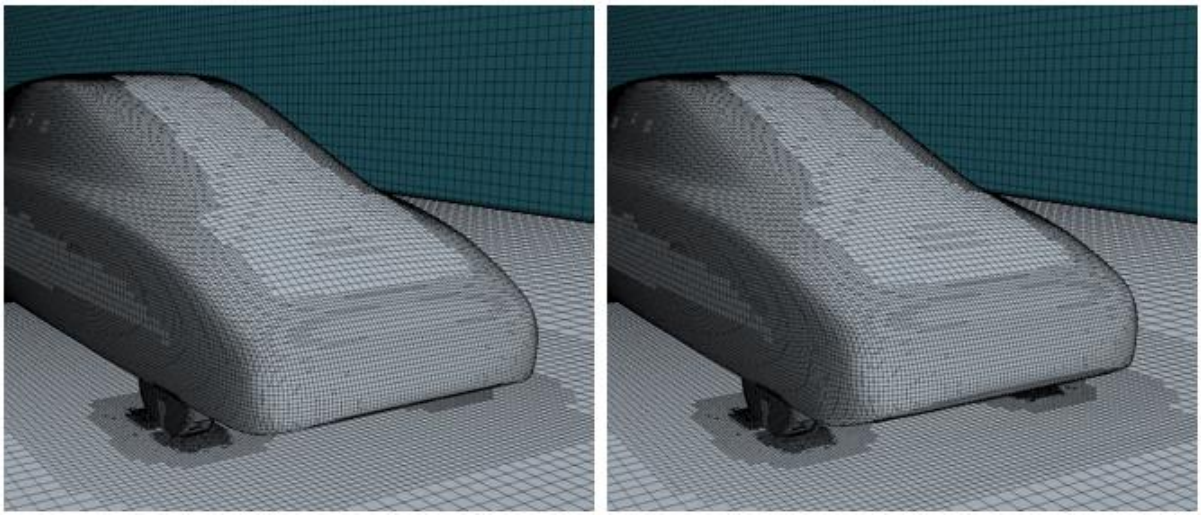

Figure 6.13: comparison between baseline (*left*) and generic morphed geometry (*right*) with focus on the front of the vehicle.

Each of the shape modifiers listed and described above was parameterized and sent to the ANSYS Workbench. Based on the variability ranges set for each parameter and applying an Optimal Space-Filling algorithm to sample the design space, a DOE table (*table 6.4*) was created. The table was generated by choosing a number of shape variants equal to 25, which when added to the baseline geometry amount to a total of 26 shapes evaluated at the end of the parametric analysis.

Design Number	Boat Tail (left) [mm]	Boat Tail (right) [mm]	Roof Drop [mm]	Front Spoiler [mm]
1	18,20	-18,20	-8,60	6,95
2	11,00	-11,00	-17,00	9,65
3	8,60	-8,60	4,60	17,75
4	12,20	-12,20	9,40	10,55
5	-2,20	2,20	-1,40	18,65
6	0,20	-0,20	1,00	-2,05
7	1,40	-1,40	-19,40	3,35
8	-1,00	1,00	8,20	13,25
9	7,40	-7,40	-9,80	19,55
10	2,60	-2,60	-18,20	14,15
11	9,80	-9,80	5,80	0,65
12	17,00	-17,00	-13,40	15,95
13	6,20	-6,20	2,20	7,85
14	13,40	-13,40	-15,80	1,55
15	-4,60	4,60	-11,00	-0,25
16	14,60	-14,60	-5,00	-1,15
17	5,00	-5,00	-7,40	2,45
18	19,40	-19,40	3,40	6,05
19	3,80	-3,80	-6,20	11,45
20	-5,80	5,80	-12,20	16,85
21	-3,40	3,40	7,00	4,25
22	-7,00	7,00	-14,60	8,75
23	15,80	-15,80	-2,60	15,05
24	-8,20	8,20	-3,80	5,15
25	-9,40	9,40	-0,20	12,35

Table 6.4: DOE for the parametric analysis on the ASMO car body shape.

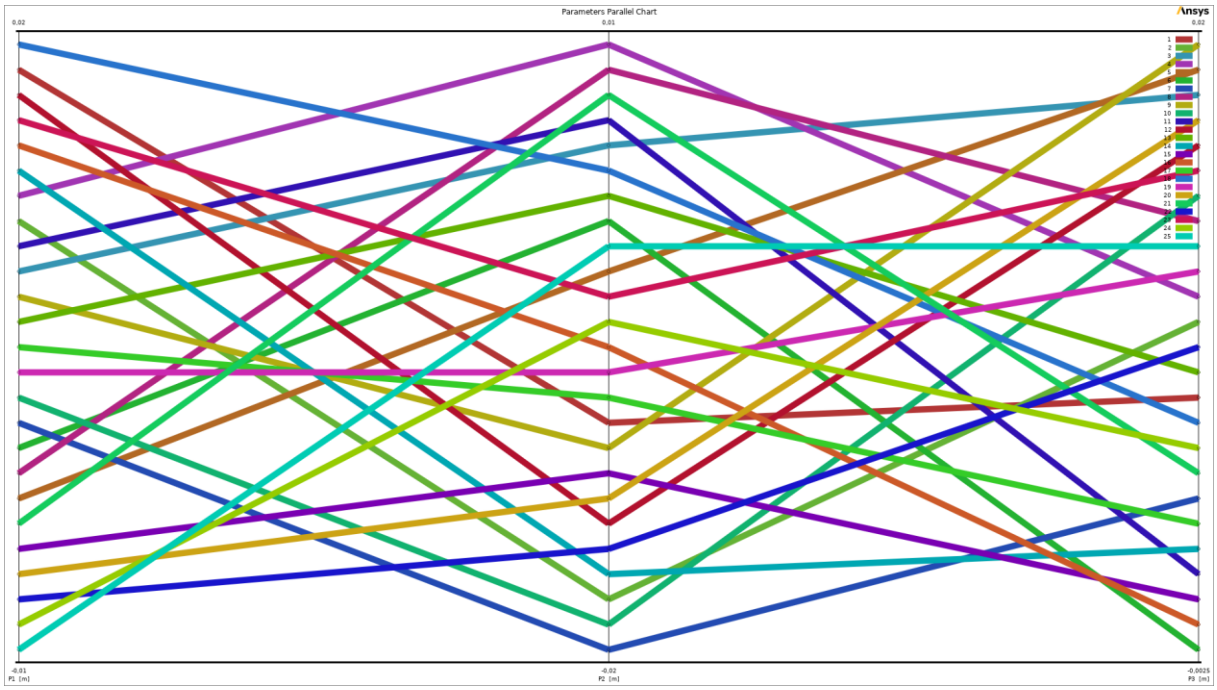

Figure 6.14: parallel chart for the three parameters.

6.1.3. RESULTS

Prepared the design exploration table and exported the morphing files for each combination of the shape parameters from RBF Morph, the parametric analysis was started by launching the batch file that contains the instructions in the form of command-lines for the operating system of the computer. The script, looping on every shape variant, runs and stores the results of all the 25 simulations. All simulations are computed on an 8 GB RAM HP machine, with an Intel® Core™ i7-8550U processor, composed by 4 independent central processing units operating at a base frequency of 1.80 GHz. The numerical results of greatest interest, such as the aerodynamic forces acting on the vehicle (downforce, drag force) and the respective coefficients (C_L, C_D) are presented in *table 6.5*.

Design Number	Downforce [N]	Drag [N]	C_L [-]	C_D [-]
0	3,27975	13,47044	0,03477	0,14280
1	-0,31402	13,05597	-0,00332	0,13819
2	-1,42447	13,17700	-0,01507	0,13940
3	5,08376	13,45192	0,05370	0,14210
4	1,85017	13,37583	0,01960	0,14173
5	4,74168	13,71682	0,05008	0,14487
6	2,93289	13,64141	0,03108	0,14455
7	-1,65695	13,66147	-0,01755	0,14471
8	6,72995	13,89578	0,07117	0,14694
9	1,18513	13,18660	0,01251	0,13922
10	-0,48814	13,53902	-0,00516	0,14310
11	4,11472	13,52637	0,04362	0,14338
12	-0,78821	13,06499	-0,00833	0,13803
13	3,85430	13,35891	0,04080	0,14140
14	-2,20308	13,14323	-0,02334	0,13926
15	0,40245	14,01567	0,00427	0,14856
16	0,20156	13,12589	0,00214	0,13912
17	0,80800	13,28668	0,00856	0,14078
18	3,07086	13,54295	0,03251	0,14339
19	2,24509	13,32411	0,02374	0,14092
20	1,82183	14,14167	0,01924	0,14938
21	5,78925	14,01153	0,06133	0,14843
22	0,74567	14,31430	0,00789	0,15146
23	2,10025	13,17041	0,02220	0,13919
24	3,36276	14,35760	0,03561	0,15204
25	5,12882	14,52660	0,05423	0,15361

Table 6.5: numerical results at the end of the parametric analysis.

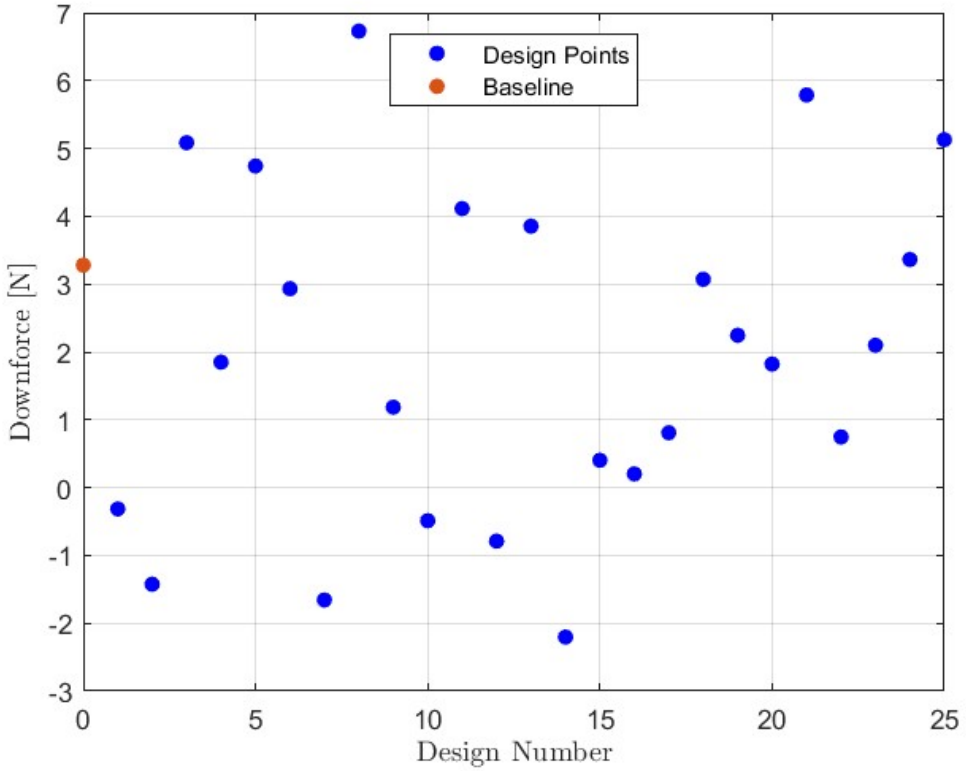

Figure 6.15: downforce of each design point.

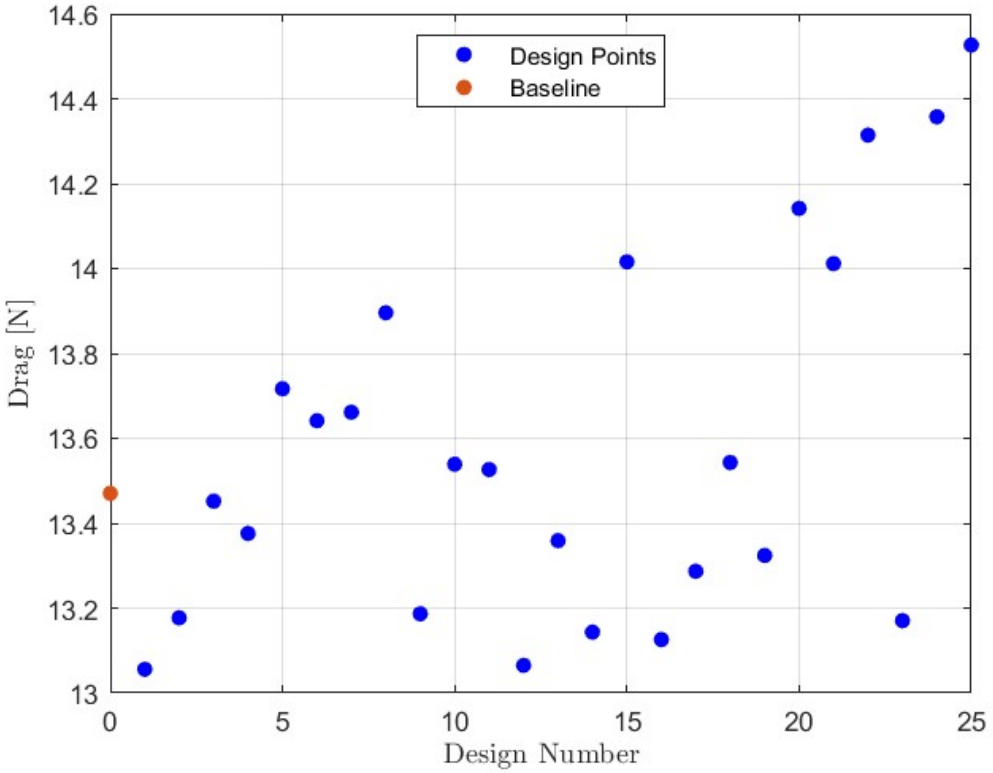

Figure 6.16: drag force of each design point.

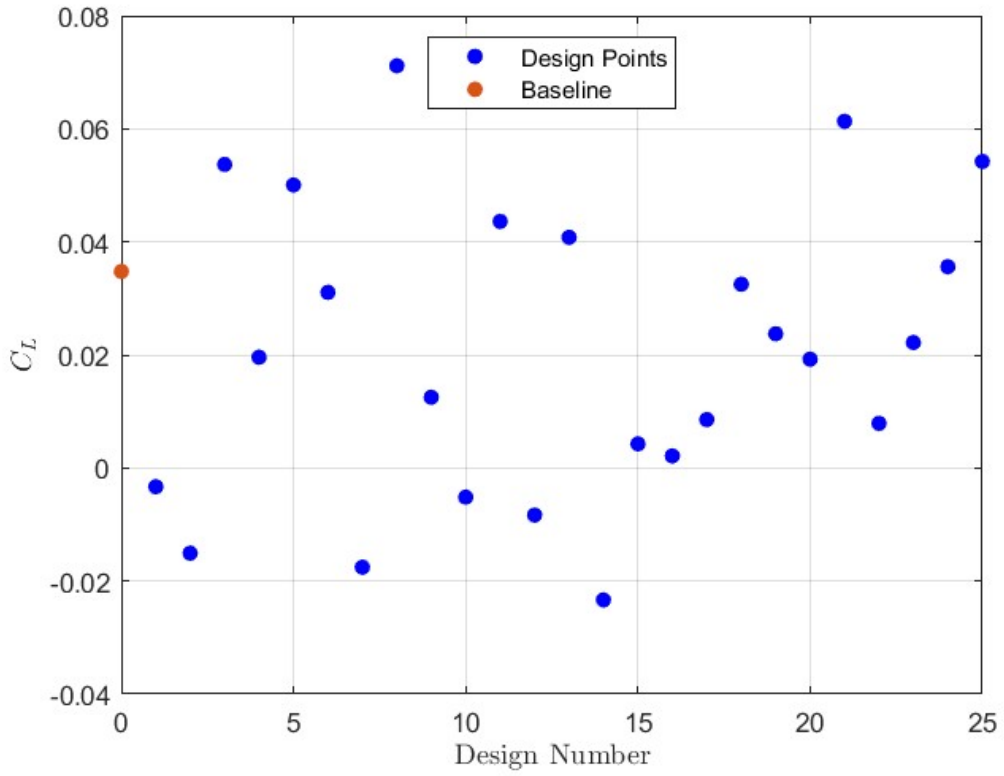

Figure 6.17: lift (downforce) coefficient of each design point.

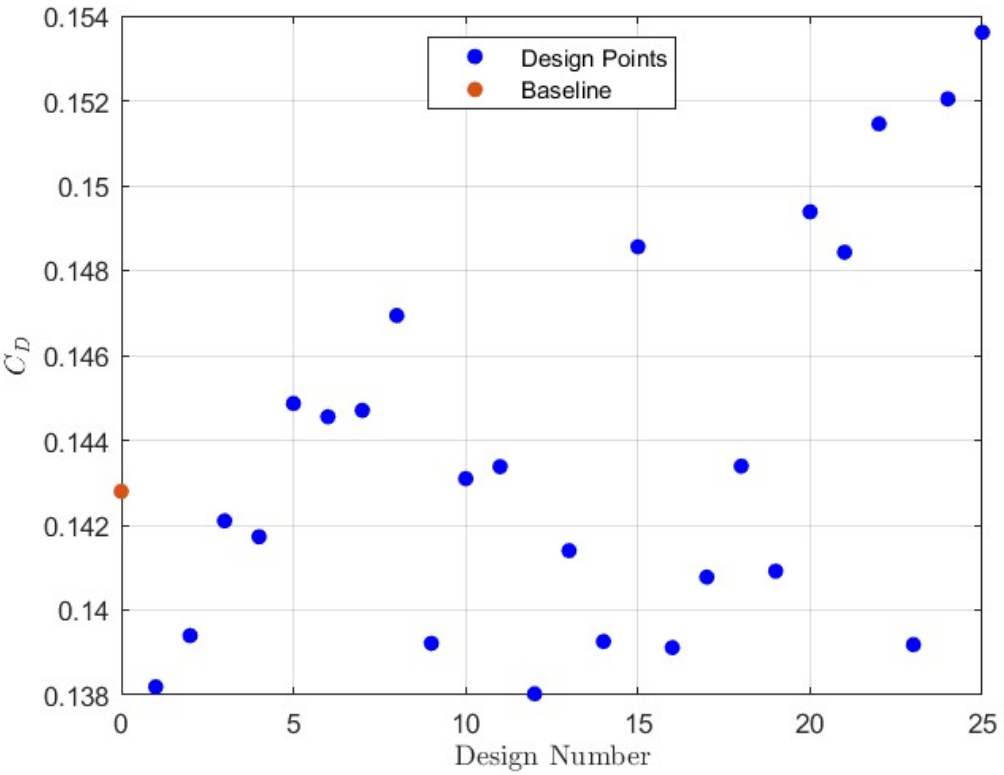

Figure 6.18: drag coefficient of each design point.

Although the total number of geometries studied in the parametric analysis of the vehicle is low, in a comparison between the values of lift coefficient and drag coefficient obtained for each shape variant is already possible to glimpse a hint of a Pareto front, which emerges due to the conflicting nature of the two objectives with respect to the selected design variables. Evidently, it can be observed that the C_L is maximized whilst the C_D is minimized and vice versa. *Fig. 6.19* shows the Pareto front that joins the non-dominated points of the design space. It is clear from this figure that choosing appropriate value for the parameters of the car shape for obtaining a better value of one objective would cause a worse value of another objective. However, if the set of decision variables is selected based on each of the Pareto sets, it will lead to the best possible combination of those two objectives. In other words, if any other pair of decision variables is chosen, the corresponding values of the pair of objectives, i.e. C_L and C_D, will locate a point inferior to the corresponding Pareto front. Such inferior area in the space of the two objectives is in fact top/right side of *fig. 6.19*. Clearly, there are some significant optimal design facts between the two objective functions which have been discovered by the design exploration. Such design facts would be decisive to the designer to switch from one optimal solution to another for achieving different trade-off requirements of the objectives.

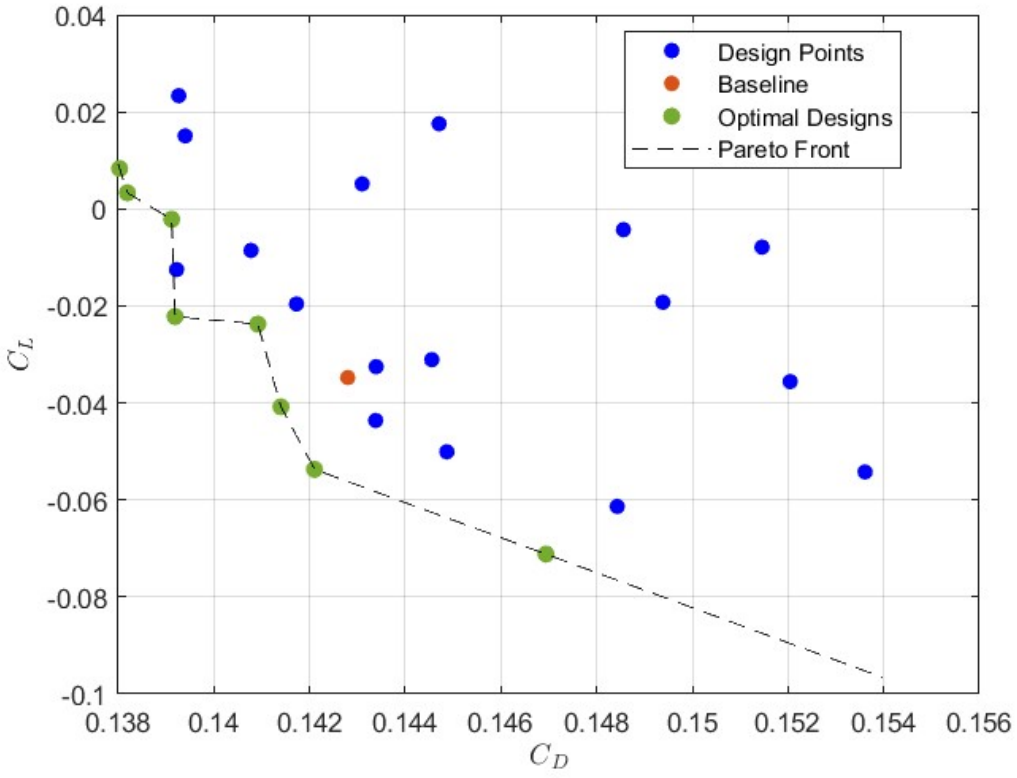

Figure 6.19: comparison of the two objectives (lift and drag).

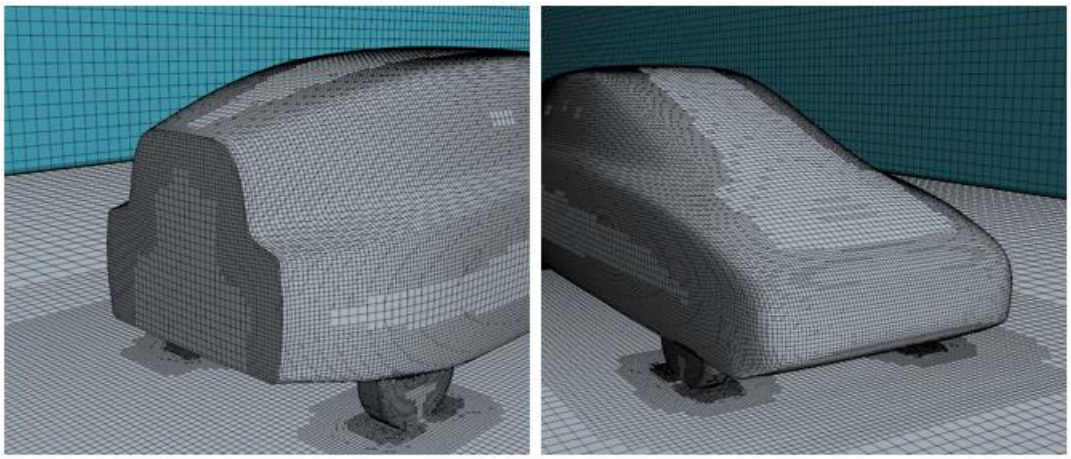

Figure 6.20: lowest drag configuration (design number 12).

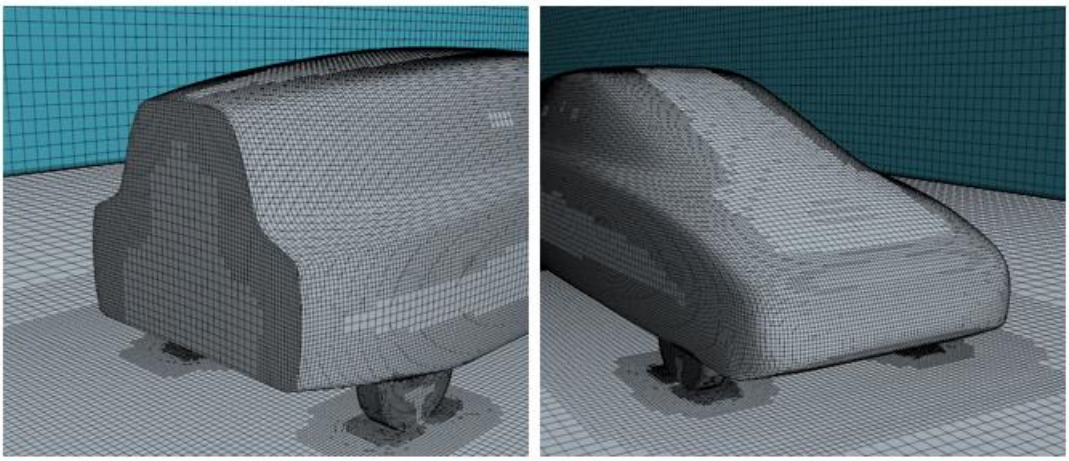

Figure 6.21: highest downforce configuration (design number 8).

6.2. VOLVO CAR SIDE-VIEW MIRROR

As a next-generation transportation system, automated driving technology is being investigated and developed for goals such as lowering traffic accidents and driving loads. Public road demonstration tests are currently being carried out mostly in Europe, the United States, and Asia. In the commercial launch of automated driving technology, solving technical challenges in diverse situations is a critical issue. A self-driving vehicle recognizes objects in its environment, makes judgments, and manages its steering and acceleration autonomously [37]. Autonomous vehicles (AVs) rely on a variety of sensor technologies to evaluate their

surroundings and make logical decisions based on the data gathered, just like humans do. Perception systems (sensors onboard AVs) provide enough information to enable autonomous transportation and mobility under optimal operating conditions. In actuality, there are still a number of issues that might obstruct the operation of AVs sensors and, as a result, reduce their effectiveness under more realistic situations that occur in the real world. Because the road infrastructure is constructed for human visual sensors, cameras will continue to be one of the most important sensors. In the first generation of systems, a single camera was utilized, but more recently, multiple cameras have been used to deliver full coverage around the car to handle more complex driving conditions. In adverse environmental conditions such as rain, fog, snow, and other forms of severe weather, the quality of computer vision algorithms degrades dramatically. This is aggravated when the camera lens is exposed to rain or to freezing conditions in the winter. Furthermore, external cameras are exposed to mud and dust. As a result, it's critical to detect when the camera lens is soiled so that the system is aware that the vision algorithms will suffer serious degradation. In this chapter, the developed workflow is tested for the design exploration of an industrial case study in collaboration with Volvo Cars and RBF Morph. The study carried out is aimed at appropriately modifying the shape of a geometric detail located around a camera mounted on the underside of a side-view mirror in order to minimize the thickness of fluid film that is deposited on the lens of the camera.

6.2.1. BASELINE MODEL

The geometry of the Volvo side-view mirror, with the camera mounted beneath, is attached to the generic DrivAer car model, a generic car model developed at the Institute of Aerodynamics and Fluid Mechanics at the Technische Universität München to facilitate aerodynamic investigations of passenger vehicles. The complete simulation on Volvo geometry can be divided into two parts: a first part in which the flow field around the car is solved in stationary conditions and air as the only fluid, and a second part in which, on the basis of the stationary results obtained in the first, a multiphase transient simulation (air / fluid film) is carried out. Since the model is very complex, the simulations to be performed have a moderately high computational cost and considering that the computing power available for this work is rather modest, the results reported in this paragraph are presented more for demonstration purposes of the feasibility of the workflow than for their practical value.

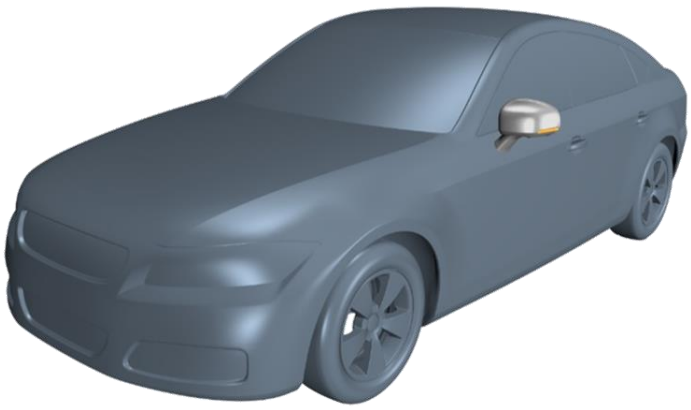

Figure 6.22: Volvo side mirror attached to the DrivAer car model.

Figure 6.23: Volvo side-view mirror (*top*) and detail of the camera lens (*bottom*).

For this model of simulation, since we are not interested in most of the fluid dynamic effects that occur near the surface of the DrivAer vehicle, except for the region of space that circumscribes the side mirror, the dimensions of the wind tunnel have been chosen with greater flexibility: the wind tunnel (*fig. 6.24*) has a width of 6.5 *m*, a length of 27.5 *m* and a height of 5.2 *m*.

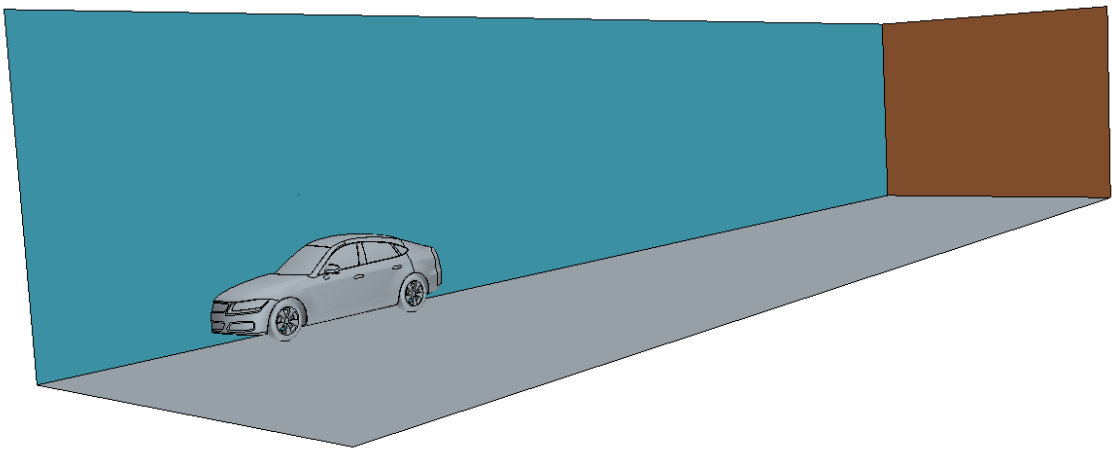

Figure 6.24: wind tunnel geometry

The creation of the calculation grid on the model is one of the most delicate phases of the simulation set-up, as it is necessary to concentrate a huge number of cells in the direct vicinity of the mirror to better grasp the physical phenomena that is important to evaluate. For this reason, around the side mirror, a region of overset mesh was placed. On the contrary, since as previously said there is no interest in the wake effects of the vehicle (and to reduce the computational complexity of the calculation), the mesh has not been refined behind the DrivAer model, saving a significant number of elements. The resulting mesh consists of only 906,734 trimmed and polyhedral cells, with prismatic cell layers next to the boundary surface to help capturing the viscous boundary layer accurately.

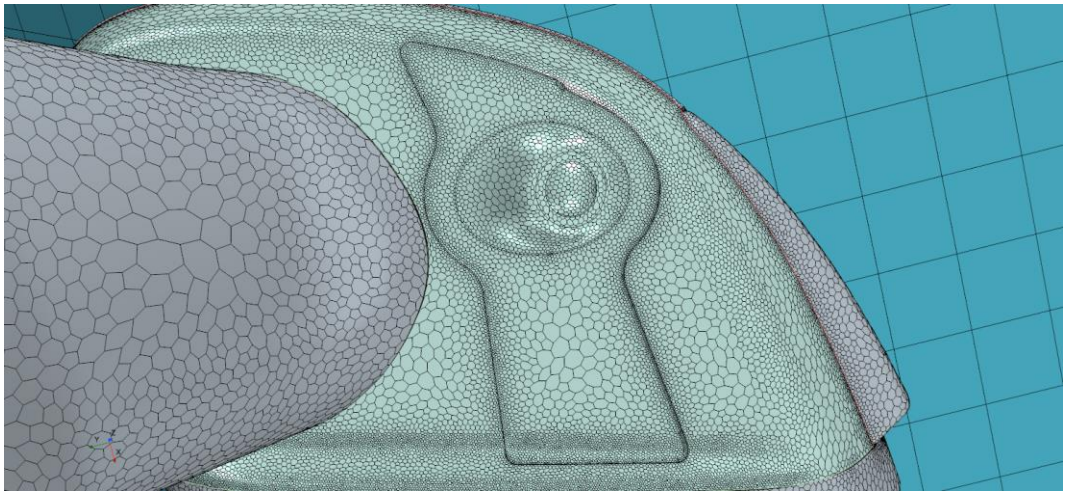

Figure 6.25: meshing of the underside of the side-view mirror.

The boundary conditions for this simulation are a 40 *mph* inflow velocity at the entrance of the wind tunnel, a 0 *Pa* outlet pressure at the exit, and a no-slip condition on the ground surface

and the vehicle surface. Both the steady and the unsteady solutions were calculated employing a realizable *k-ε* turbulence model with a two-layer all-y^+ wall treatment. The second part of the simulation, which solves the multiphase non-stationary case, involves establishing additional boundary conditions. These include the definition of a shell region (region of space affected by the multiphase air/fluid film solution model), and the definition of a fluid film inlet/outlet system through the shell region. Initial fluid film thickness on the shell region is set to 0.1 *mm*. Water flows out of the inlet boundary with a 0.5 *mm* thickness and a velocity of 4.36 *mm/s*. As it flows down the shell region, the water is affected by both air flow around the vehicle and gravity, forming a liquid film in the process. In some cases the film thickness can become very large. This behavior is typically physical (for example, the liquid film is pushed into a corner and it cannot escape). A very large film thickness (much larger than the thickness of the neighbor volume cell) can cause numerical instabilities in the simulation. Setting a maximum film thickness property ensures that the film thickness does not exceed an appropriate maximum value at any location. When the specified thickness of 1 *mm* is reached, any extra fluid is removed from the film and is lost from the simulation. In this scenario, mass conservation is not explicitly satisfied in the simulation.

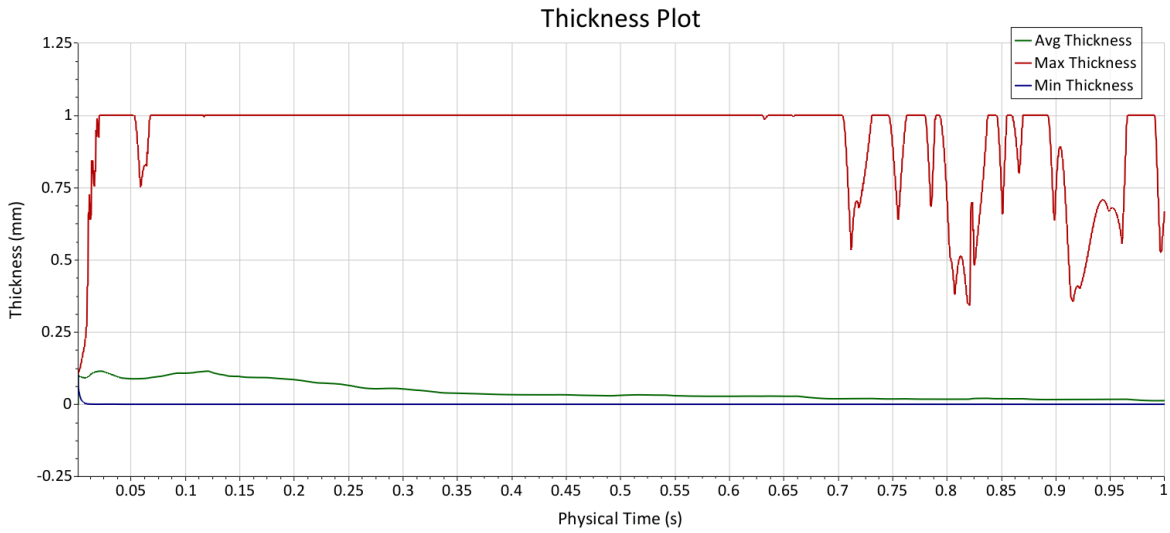

Figure 6.26: visualization of the minimum, maximum and average film thickness during the transient simulation.

The simulation is run using the implicit unsteady solver for a simulated time of 1 second with a time step size set to 1 *ms*. *Fig. 6.27* and *fig. 6.278*display the inlet and outlet boundaries selected for the side-view mirror. At the end of the unsteady simulation, mean fluid film thickness is evaluated in the camera lens (*fig. 6.29*).

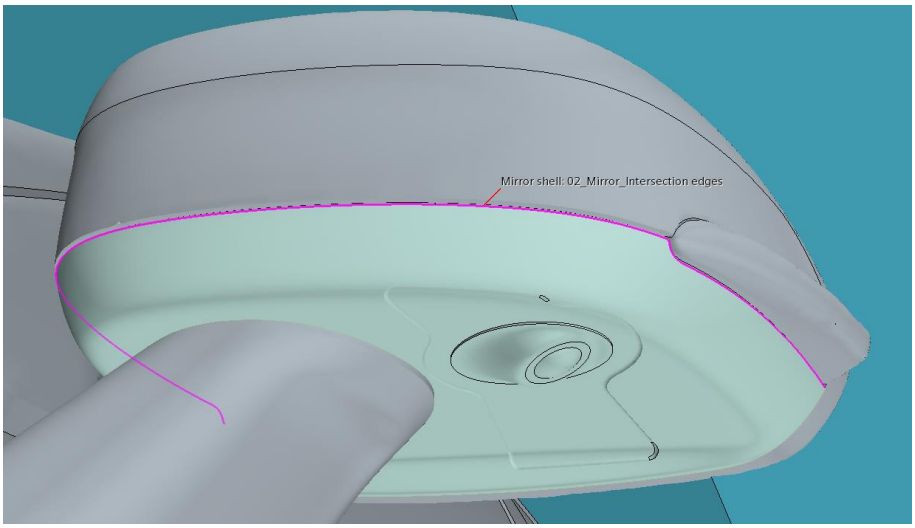

Figure 6.27: fluid film inlet boundaries.

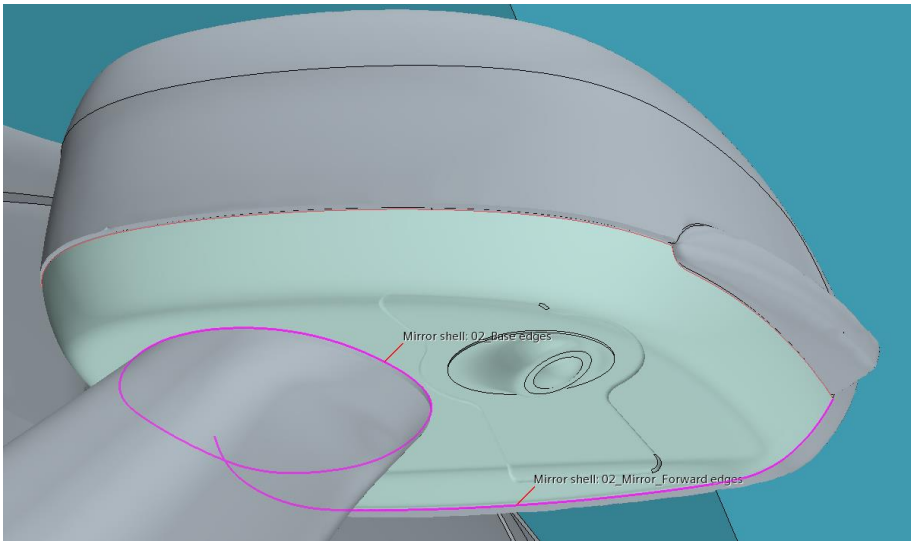

Figure 6.28: fluid film outlet boundaries.

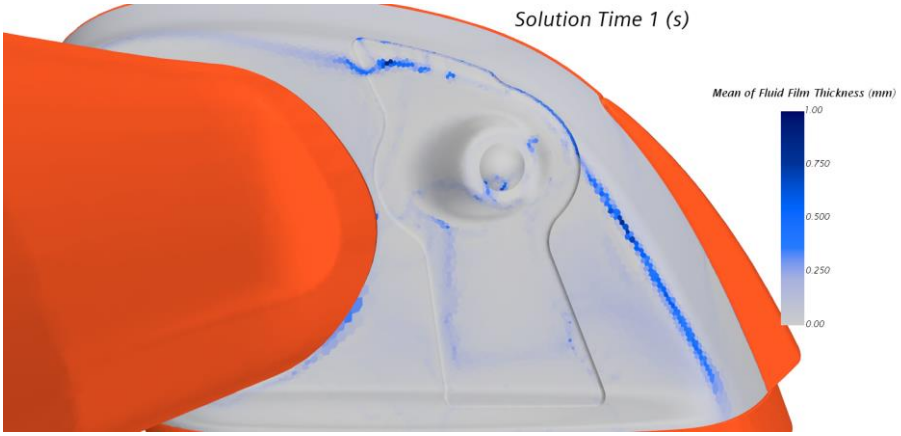

Figure 6.29: average fluid film thickness deposition on the camera lens.

6.2.2. SHAPE PARAMETRIZATION

As in the previous case, the parameterization of the Volvo geometry was performed using the graphical interface of ANSYS Mechanical, taking advantage of the functionality of RBF Morph ACT Extension. The shape was modified introducing 6 shape parameters, each one controlling a specific detail of the geometry. To allow a better agreement between the parts that are morphed and those that are kept fixed, a buffer region has generally been interposed. The critical aspect to be considered in the parameterization and modification of the geometry is that the region of the surface occupied by the camera must remain fixed in space, therefore careful attention was paid to making sure that the variations in shape obtained with the parameters introduced did not involve movements of the camera.

P_1, P_2 and P_3 parameters

These parameters act on the positions of the outer edges of the target geometry. RBF centers were rigidly translated to achieve the shape variation obtained with amplifications and decreases of the parameters. *Fig. 6.30* shows a preview of a possible combination for all three parameters (original source points are colored in red and morphed points are represented in blue), whilst *table 6.6* summarizes the selected variation ranges for each of them.

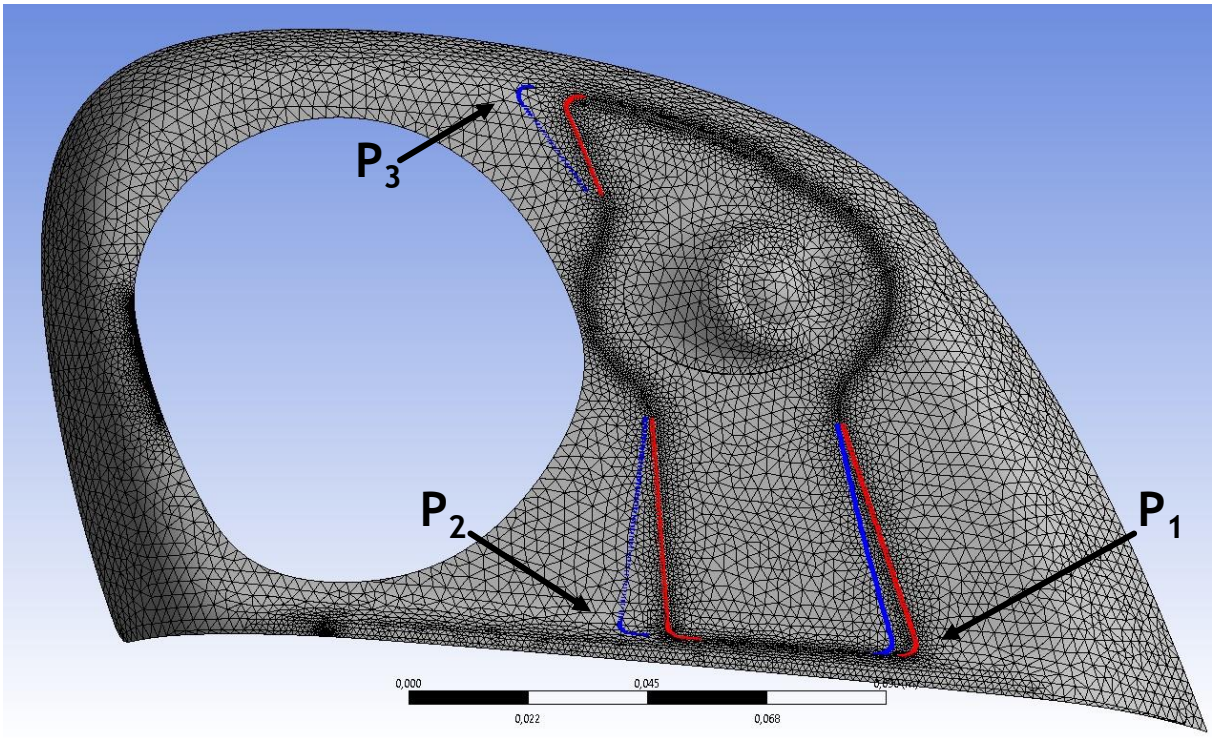

Figure 6.30: shape parametrization (P_1, P_2 and P_3) of the geometry.

	Lower limit [m]	Upper limit [m]
P_1	-0.005	+0.010
P_2	0.000	+0.015
P_3	0.000	+0.015

Table 6.6: lower and upper limits of P_1, P_2 and P_3.

E_1, E_2 parameters

These parameters were introduced to control the position of the rounded edges located on the left and on the right of the camera. A comparison between original shape (red points) and a generic morphed shape (blue points) is presented in *fig. 6.31*.

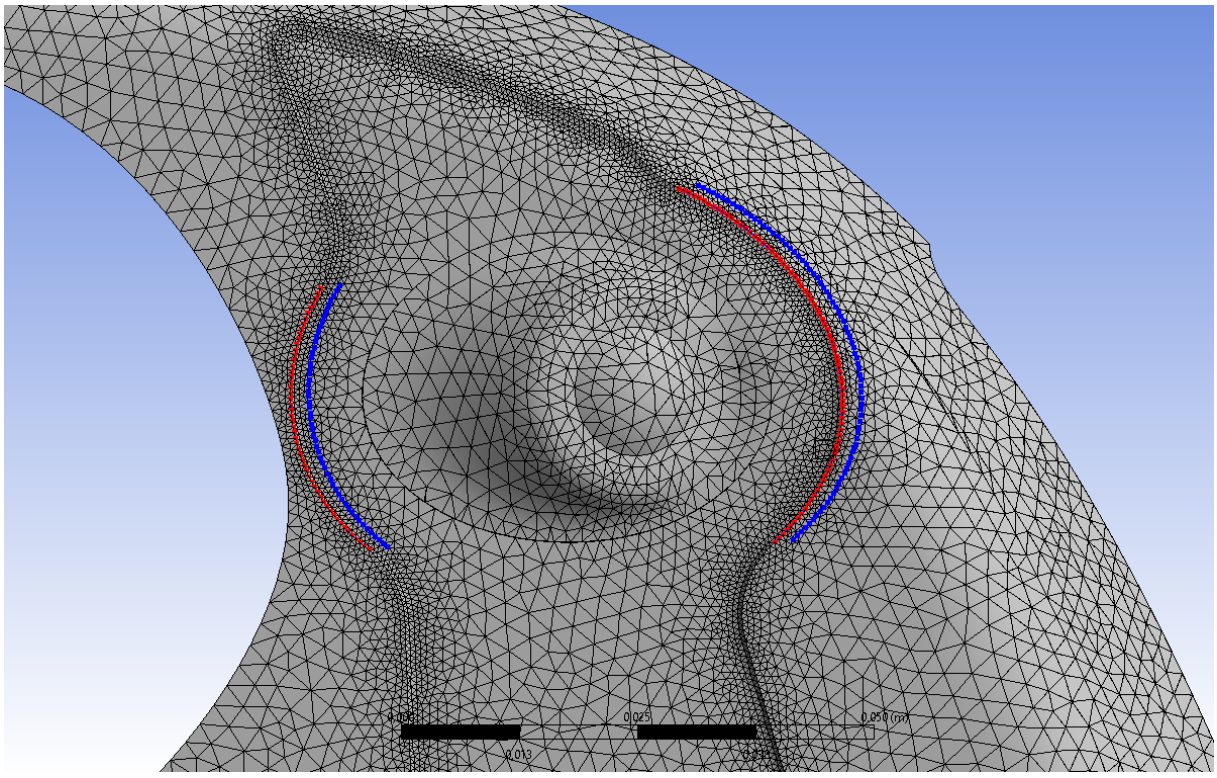

Figure 6.31: shape parametrization (E_1 and E_2) of the geometry.

	Lower limit [m]	Upper limit [m]
E_1	-0.002	+0.000
E_2	-0.002	+0.004

Table 6.7: lower and upper limits of E_1 and E_2.

Offset parameter

This parameter controls the depth of the surface surrounding the camera. In order to avoid geometric distortions due to too aggressive morphing, for this parameter it was decided to limit the range of variation to fractions of a millimeter (*table 6.8*).

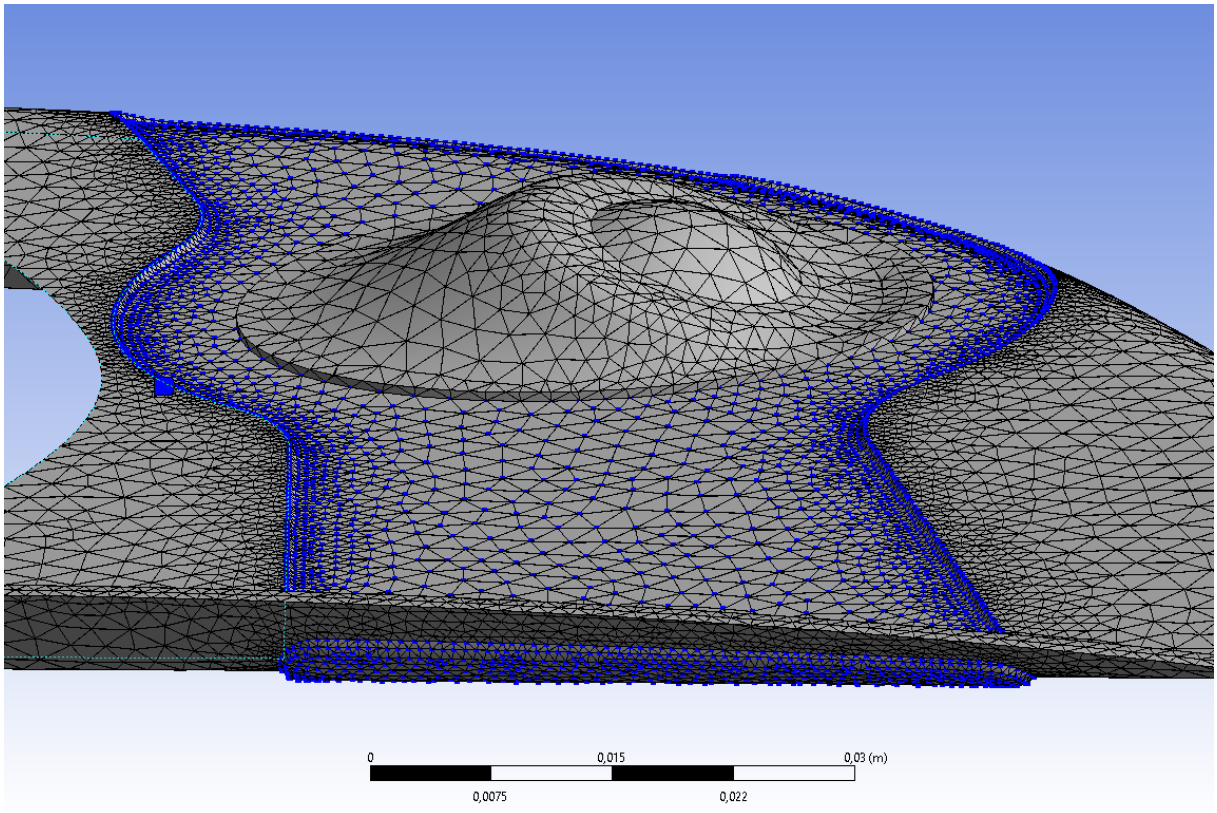

Figure 6.32: Offset of the surface around the camera.

	Lower limit [m]	**Upper limit [m]**
Offset	0.0000	+0.0005

Table 6.8: lower and upper limits of the Offset parameter.

In this case the design space was explored by generating a DOE table through the Optimal-Space-Filling algorithm of ANSYS Workbench, choosing again to evaluate 25 different combinations of design parameters, but with the difference that now the number of parameters in 6, compared to the 3 of the previous application. It is worth noting that with such a number of shape parameters to be studied, a number of 25 combinations does not allow, generally, a good evaluation of the design space. As mentioned above, the parametric analysis conducted has more the purpose of demonstrating the feasibility of the proposed workflow than of

indicating a solution to the optimization problem. A comparison between the baseline geometry and a generic morphed geometry after the parametrization procedure is displayed in *fig. 6.33*.

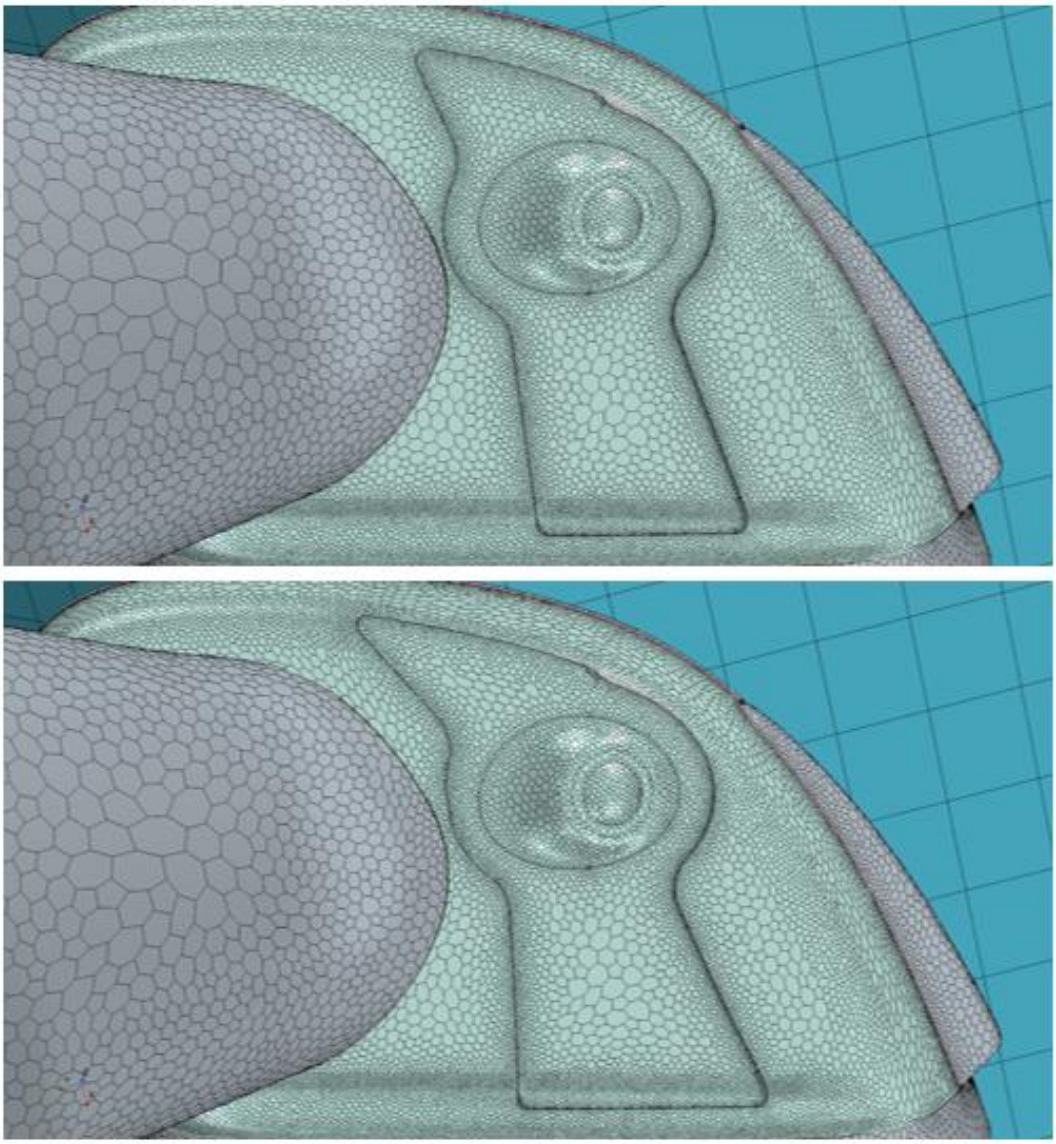

Figure 6.33: comparison between baseline geometry (*top*) and generic modified shape (*bottom*).

Design Number	**P_1 [mm]**	**P_2 [mm]**	**P_3 [mm]**	**E_1 [mm]**	**E_2 [mm]**	**Offset [mm]**
1	0,70	14,70	3,90	-0,36	1,72	0,37
2	-4,10	2,10	6,90	-0,44	0,52	0,41
3	6,10	5,70	5,70	-0,04	2,92	0,39
4	1,30	8,70	14,70	-0,68	2,68	0,45
5	-1,70	1,50	2,10	-1,64	-0,20	0,19

6	9,70	12,90	9,30	-0,60	0,28	0,13
7	-1,10	7,50	2,70	-0,28	0,76	0,03
8	-0,50	14,10	10,50	-0,92	3,16	0,09
9	8,50	10,50	9,90	-1,48	3,64	0,31
10	7,30	12,30	0,30	-1,56	1,00	0,25
11	5,50	3,90	7,50	-1,08	-1,88	0,05
12	4,90	2,70	4,50	-1,72	1,24	0,47
13	-2,90	13,50	5,10	-1,32	-1,16	0,17
14	3,10	5,10	12,90	-0,12	1,96	0,07
15	-4,70	4,50	13,50	-1,16	0,04	0,15
16	7,90	0,90	11,70	-0,76	-0,44	0,33
17	-3,50	9,90	8,10	-1,80	1,48	0,43
18	4,30	9,30	14,10	-1,88	-0,68	0,21
19	2,50	8,10	6,30	-1,96	2,20	0,01
20	-2,30	6,90	1,50	-1,00	3,88	0,27
21	0,10	11,10	12,30	-0,20	-1,40	0,29
22	3,70	6,30	0,90	-0,52	-1,64	0,35
23	6,70	11,70	8,70	-1,24	-0,92	0,49
24	9,10	3,30	3,30	-0,84	2,44	0,11
25	1,90	0,30	11,10	-1,40	3,40	0,23

Table 6.9: DOE for the parametric analysis on the Volvo side-view mirror geometry.

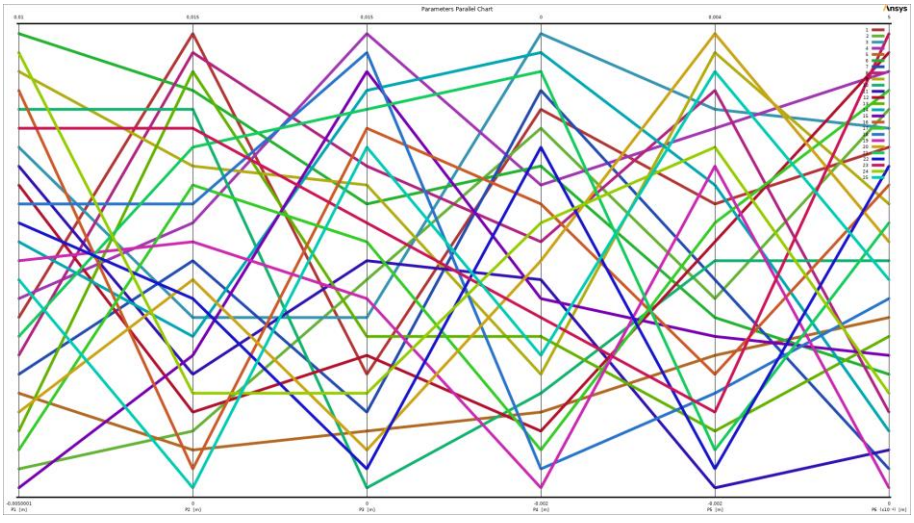

Figure 6.34: parallel chart for the 6 shape parameters.

6.2.3. RESULTS

The parametric analysis is performed following the same methodology as the previous one, with the difference that for these simulations CFD analysis the computational expense is much higher. All simulations are computed on a 128 GB RAM workstation, with two Intel® Xeon® E5-2680 v2 processors, composed by 20 independent central processing units operating at a base frequency of 2.80 GHz. The most relevant performance indicator, the mean fluid film thickness deposited on the camera lens, is shown for each simulation in *fig. 6.35*. It can be observed that 2 of the 25 simulations performed (number 10 and number 19), at the end of the parametric analysis, failed to execute. An inspection of the log file reveals that those two geometries, following the morphing action based on the shape parameter values decided during the creation of the DOE, returned a volume mesh within STAR-CCM+ with one or more negative volume cells, and therefore the simulation did not start.

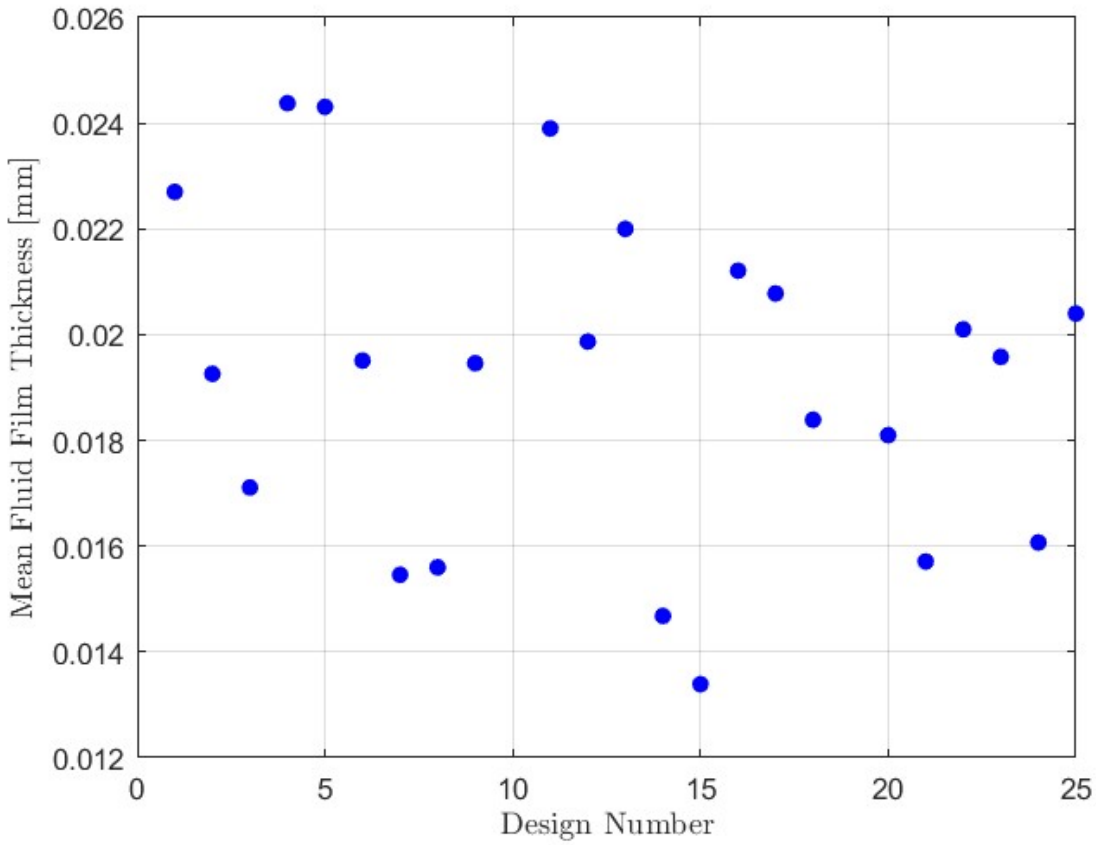

Figure 6.35: Mean of fluid film thickness for each design point.

7. CONCLUSIONS

The aerodynamic of road vehicles can be explored and improved by using computational fluid dynamic (CFD) tools. These tools provide quite accurate results within reasonable time scales. Results of simulations are generally not grid independent and not model independent, however, once these limitations are properly understood, these tools may be used for the design exploration and optimization of many industrial applications. Computationally based designed experiments and optimization studies have historically been limited by the cost and time associated with configuration changes and simulation times. The majority of computational studies in the traditional processes have also tended to be sequential in nature. The innovative and now consolidated parameterization and mesh-morphing techniques based on state-of-the-art fast Radial Basis Functions (RBF) obviate the need for the more time-consuming and often troublesome traditional CAD-based parameterization approaches.

The work carried out and presented in this thesis was done in a context of numerical aerodynamic calculation based on existing tools and software, in collaboration with Volvo and RBF Morph. An advanced technological prototype of an orchestrator, based on an automatic workflow combining disjointed software and CAE tools, was developed and put to test with two practical industrial applications. In the first one, the aerodynamics of the ASMO idealized car body shape was studied, then the geometry shape was parametrized and optimized. The shape parameters introduced (Boat Tail, Roof Drop and Front Spoiler) made it possible to evaluate the performance of the vehicle corresponding to various possible designs. It emerged from the parametric analysis conducted on the vehicle that the initial baseline solution can be improved and optimized according to two objectives: aerodynamic resistance (through a minimization of drag force) and stability / grip on the road (through a maximization of downforce. The results of the CFD analysis performed on the baseline model, even with a coarse grid, are in accordance with the values obtained by other numerical evaluations in the literature (Aljure et al. (2014) [33]) and with the experimental results obtained by experimental tests in Volvo wind tunnels (Tsubokura et al. (2009) [32]). The CFD analysis returned optimal combinations of the shape parameters to generate a vehicle geometry that presents a reduction in the drag coefficient compared to the initial geometry by about 3.34%, going from 0.143 to 0.138. Considering that the initial geometry of ASMO already has a remarkably streamlined shape, such a reduction in the drag coefficient is still a significant result. The lift coefficient has

been improved from a value of -0.035 in the baseline geometry to a value of -0.071 in the corresponding optimized geometry. Even with a relatively small set of shape variants studied it was possible to obtain a rudimentary Pareto front, pointing to a set of trade-off solutions to the problem of vehicle shape optimization. In the second application of the workflow a multiphase CFD simulation was carried out on a side-view mirror of a Volvo car. The study was performed with the objective of finding a solution to a shape optimization problem involving the minimization of the fluid film thickness deposited on the lens of a camera. Again, the workflow proved to be effective in the implementation of a fast and reliable prototype of orchestrator, automating the exploration of many different design configurations and thus achieving the objective of supporting the designer in the process of shape optimization and gaining of a much deeper understanding of the design space around a given exterior theme.

The workflow presented also embodies a sophisticated and automatic calculation methodology with enormous applicability potential and was developed as a support demonstrator for the RBF Morph technology, acting as a pilot for the new Stand-Alone version of the commercial morpher, currently under development and which will soon be released on the market.

8. APPENDIX

This appendix contains the user codes that are part of the orchestrator prototype. The codes were written to facilitate the management of complex tasks and automate the workflow.

8.1. CALLING RBF LIBRARIES AND EXECUTING MESH MORPHING

The following C++ code performs the task of: reading morphing data from files generated in RBF Morph ACT, calling functions in the RBF Morph libraries to solve the RBF problem, morphing the baseline mesh, and saving the coordinates of the new grid to a new file.

```
#define _CRT_SECURE_NO_DEPRECATE
#include "RBF_FGP.h"
#include <string>
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <iostream>
#include <fstream>
#include <vector>
#include <cmath>
using namespace std;


vector<vector<double>> encap(double x0, double y0, double z0, double Lx,
double Ly, double Lz, int Nx, int Ny, int Nz)
{
    double delta_x = Lx / (Nx-1);
    double delta_y = Ly / (Ny-1);
    double delta_z = Lz / (Nz-1);

    vector<vector<double>> r;
    vector<double> xyz;
    int i, j, k;

    k = 0;
    for (int i = 0; i < Nx; i++)
    {
```

```
    for (int j = 0; j < Ny; j++)
    {
        xyz.push_back(x0 + i*delta_x);
        xyz.push_back(y0 + j*delta_y);
        xyz.push_back(z0 + k*delta_z);
        r.push_back(xyz);
        xyz.clear();
    }
}

k = Nz-1;
for (int i = 0; i < Nx; i++)
{
    for (int j = 0; j < Ny; j++)
    {
        xyz.push_back(x0 + i*delta_x);
        xyz.push_back(y0 + j*delta_y);
        xyz.push_back(z0 + k*delta_z);
        r.push_back(xyz);
        xyz.clear();
    }
}

j = 0;
for (int i = 0; i < Nx; i++)
{
    for (int k = 0; k < Nz; k++)
    {
        xyz.push_back(x0 + i*delta_x);
        xyz.push_back(y0 + j*delta_y);
        xyz.push_back(z0 + k*delta_z);
        r.push_back(xyz);
        xyz.clear();
    }
}

j = Ny-1;
for (int i = 0; i < Nx; i++)
{
    for (int k = 0; k < Nz; k++)
    {
        xyz.push_back(x0 + i*delta_x);
        xyz.push_back(y0 + j*delta_y);
        xyz.push_back(z0 + k*delta_z);
        r.push_back(xyz);
        xyz.clear();
    }
}

i = 0;
for (int j = 0; j < Ny; j++)
{
    for (int k = 0; k < Nz; k++)
    {
        xyz.push_back(x0 + i*delta_x);
        xyz.push_back(y0 + j*delta_y);
        xyz.push_back(z0 + k*delta_z);
        r.push_back(xyz);
```

```
            xyz.clear();
        }
    }

    i = Nx-1;
    for (int j = 0; j < Ny; j++)
    {
        for (int k = 0; k < Nz; k++)
        {
            xyz.push_back(x0 + i*delta_x);
            xyz.push_back(y0 + j*delta_y);
            xyz.push_back(z0 + k*delta_z);
            r.push_back(xyz);
            xyz.clear();
        }
    }

    const char file_name_box[100] = "C:\\ASMO_VOLVO\\C++_box.txt";

    fstream file;
    file.open(file_name_box, ios::out | ios::binary);
    if (!file)
    {
        cout << "Error: " << file_name_box << " could not be opened" << endl;
    }
    else
    {
        for (int i = 0; i < r.size(); i++)
        {
            for (int j = 0; j < r[i].size(); j++)
            {
                file << r[i][j] << "\t";
            }
            file << 0 << "\t" << 0 << "\t" << 0 << "\t"  << "\n";
        }
        file.close();
    }

    cout << "Encapsulation volume created." << endl;
    return r;
}


int main(int argc, char *argv[])
{
    cout << "\n=======================\n" << endl;
    cout << "Solving linear system... " << endl;

    const int file_name_size = 1024;
    const int max_line = 2048;


    string file_name_in = argv[1];
    const char file_name_p[file_name_size] =
"C:\\ASMO_VOLVO\\tmp\\temp_p.txt";


    int nol = 0;
```

```
string line;
ifstream file(file_name_in);

while (getline(file, line))
{
    nol++;
}
cout << "Number of source points: " << nol << endl;
file.close();

ifstream in(file_name_in, ios::in | ios::binary);
ofstream out(file_name_p, ios::out | ios::binary);

for (char c; in.get(c); out.put(c))
{
    if (c == ',')
    {
        c = '.';
    }
}

in.close(); out.close();


const char file_name[file_name_size] = "C:\\ASMO_VOLVO\\tmp\\temp.txt";
char buffer[max_line];
int delete_line = 1;

FILE *temp_1, *temp_2;
temp_1 = fopen(file_name_p, "r");
temp_2 = fopen(file_name, "w");

if (temp_1 == NULL || temp_2 == NULL)
{
    cout << "Error: one or more files could not be opened" << endl;
    return 1;
}

bool keep_reading = true;
int current_line = 1;

do
{
    fgets(buffer, max_line, temp_1);
    if (feof(temp_1))
    {
        keep_reading = false;
    }
    else if (current_line != delete_line)
    {
        fputs(buffer, temp_2);
    }
    current_line++;
} while (keep_reading);

fclose(temp_1); fclose(temp_2);

// VOLVO
```

```
    double x0 = 2.450;
    double y0 = -1.100;
    double z0 = 1.050;
    double Lx = 0.350;
    double Ly = 0.310;
    double Lz = 0.270;
    int Nx = 20;
    int Ny = 20;
    int Nz = 20;
/*
    // ASMO (BOAT TAIL)
    double x0 = 0.60;
    double y0 = -0.15;
    double z0 = 0.00;
    double Lx = 0.25;
    double Ly = 0.30;
    double Lz = 0.25;
    int Nx = 20;
    int Ny = 20;
    int Nz = 20;

    // ASMO (BOAT TAIL + LONG ROOF DROP)
    double x0 = 0.40;
    double y0 = -0.15;
    double z0 = 0.00;
    double Lx = 0.45;
    double Ly = 0.30;
    double Lz = 0.28;
    int Nx = 20;
    int Ny = 20;
    int Nz = 20;

    // ASMO (BOAT TAIL + LONG ROOF DROP + FRONT SPOILER)
    double x0 = -0.10;
    double y0 = -0.20;
    double z0 = 0.00;
    double Lx = 1.00;
    double Ly = 0.40;
    double Lz = 0.30;
    int Nx = 20;
    int Ny = 20;
    int Nz = 20;
*/

    encap(x0, y0, z0, Lx, Ly, Lz, Nx, Ny, Nz);


    const char file_name_box[100] = "C:\\ASMO_VOLVO\\C++_box.txt";

    int nol_box = 0;
    string line_box;
    ifstream file_box(file_name_box);

    while (getline(file_box, line_box))
    {
        nol_box++;
    }
```

```
    cout << "Number of source points on the encapsulation volume: " << nol_box
<< endl;
    file_box.close();

    int sp_n = nol + nol_box;
    cout << "Total number of source points: " << sp_n << endl;

    double* x = new double[sp_n];
    double* y = new double[sp_n];
    double* z = new double[sp_n];
    double* dx = new double[sp_n];
    double* dy = new double[sp_n];
    double* dz = new double[sp_n];
    double* gx = new double[sp_n];
    double* gy = new double[sp_n];
    double* gz = new double[sp_n];
    double* lx = new double[sp_n];
    double* ly = new double[sp_n];
    double* lz = new double[sp_n];

    int counter = 1;
    float x_i, y_i, z_i, dx_i, dy_i, dz_i;


    FILE* f = fopen(file_name, "r");
    FILE* f_box = fopen(file_name_box, "r");
    if (f == NULL || f_box == NULL)
    {
        cout << "Error: one or more files could not be opened" << endl;
        exit(1);
    }

    for (int i = 0; i < nol; i++)
    {
        fscanf_s(f, "%f %f %f %f %f %f %*i %*i %*c %*c", &x_i, &y_i, &z_i,
&dx_i, &dy_i, &dz_i);
        x[i] = x_i; y[i] = y_i; z[i] = z_i;
        gx[i] = x_i; gy[i] = y_i; gz[i] = z_i;
        dx[i] = dx_i; dy[i] = dy_i; dz[i] = dz_i;
        counter++;
    }

    fclose(f);

    for (int i = nol; i < (nol + nol_box); i++)
    {
        fscanf_s(f_box, "%f %f %f %f %f %f", &x_i, &y_i, &z_i, &dx_i, &dy_i,
&dz_i);
        x[i] = x_i; y[i] = y_i; z[i] = z_i;
        gx[i] = x_i; gy[i] = y_i; gz[i] = z_i;
        dx[i] = dx_i; dy[i] = dy_i; dz[i] = dz_i;
        counter++;
    }

    fclose(f_box);

    if (sp_n != (counter-1))
    {
```

```
        cout << "Error: incorrect data reading" << endl;
        exit(1);
    }


    int omp_cores = 0;
    bool use_gpu = false;
    bool precompute = false;
    bool single_prec = true;
    int rbf_exp = 1;
    int q = 25;
    double tolerance = 1e-5;
    int max_iter = 100;
    double dist_dupl = 1e-8;
    double Cpar = 0;

    double a[3];
    double max_err;
    int iter;


    clock_t t;
    t = clock();
    int temp = sp_n;
    sp_n = Purge(sp_n, x, y, z, dx, dy, dz, dist_dupl);
    t = clock() - t;
    printf("Purging (%g distance, in %.3f seconds):\n  Number of source points
excluded: %d\n  New number of source points: %d\n", dist_dupl, ((float)t) /
CLOCKS_PER_SEC, temp - sp_n, sp_n);


    t = clock();
    int rc = Solve(sp_n, q, tolerance, max_iter, precompute, omp_cores,
use_gpu, single_prec, rbf_exp, Cpar, x, y, z, dx, dy, dz, lx, ly, lz, a,
&max_err, &iter, NULL);
    if (rc != 0)
    {
        cout << "Error: Solve function" << endl;
        getchar();
        exit(1);
    }
    t = clock() - t;
    printf("Solving (%.3f seconds)\n", ((float)t) / CLOCKS_PER_SEC);
    // PrintOutput(sp_n);
    printf("ALPHA:    %g    %g    %g\n", a[0], a[1], a[2]);

    cout << "Solving linear system: done." << endl;
    cout << "\n=======================\n" << endl;
    cout << "Morphing process started... " << endl;

    const char file_name_star[file_name_size] =
"C:\\ASMO_VOLVO\\STAR_vertex_coord.txt";

    int size = 0;
    string line_star;
    ifstream file_star(file_name_star);

    while (getline(file_star, line_star))
```

```
{
    size++;
}
cout << "Number of vertices to be moved: " << size << endl;
file_star.close();

int mp_n = size;

double *x_m = new double[mp_n];
double *y_m = new double[mp_n];
double *z_m = new double[mp_n];
double *dx_m = new double[mp_n];
double *dy_m = new double[mp_n];
double *dz_m = new double[mp_n];

int counter_star = 0;

FILE *f_star = fopen(file_name_star, "r");
if (f_star == NULL)
{
    cout << "Error: " << file_name_star << " could not be opened" << endl;
    exit(1);
}
else
{
    for (int i = 0; i < size; i++)
    {
        fscanf_s(f_star, "%f %f %f", &x_i, &y_i, &z_i);
        x_m[i] = x_i; y_m[i] = y_i; z_m[i] = z_i;
        dx_m[i] = 0.0; dy_m[i] = 0.0; dz_m[i] = 0.0;
        counter_star++;
    }
}

fclose(f_star);


t = clock();
int rc_m = Morph(mp_n, x_m, y_m, z_m, dx_m, dy_m, dz_m);
if(rc_m!=0) {printf("ERRORE Morph  %d   \n", rc_m); getchar(); exit(1);}
t = clock() - t;
printf("Morphing (%.3f seconds)\n", ((float)t) / CLOCKS_PER_SEC);
cout << "Morphing process: done.\n" << endl;


double *result_x = new double[mp_n];
double *result_y = new double[mp_n];
double *result_z = new double[mp_n];

for (int i = 0; i < size; i++)
{
    result_x[i] = (x_m[i] + dx_m[i]);
    result_y[i] = (y_m[i] + dy_m[i]);
    result_z[i] = (z_m[i] + dz_m[i]);
}
```

```
    const char file_name_result[file_name_size] =
"C:\\ASMO_VOLVO\\C++_vertex_coord.txt";

    t = clock();
    fstream file_result;
    file_result.open(file_name_result, ios::out | ios::binary);
    if (!file_result)
    {
        cout << "Error: " << file_name_result << " could not be opened" <<
endl;
    }
    else
    {
        for (int i = 0; i < size; i++)
        {
            file_result << result_x[i] << " " << result_y[i] << " " <<
result_z[i] << "\n";
        }
        file_result.close();
    }

    cout << "New file created: " << file_name_result << endl;
    t = clock() - t;
    printf("Writing to file (%.3f seconds)\n", ((float)t) / CLOCKS_PER_SEC);


    delete[] x;
    delete[] y;
    delete[] z;
    delete[] dx;
    delete[] dy;
    delete[] dz;
    delete[] gx;
    delete[] gy;
    delete[] gz;
    delete[] lx;
    delete[] ly;
    delete[] lz;

    delete[] x_m;
    delete[] y_m;
    delete[] z_m;
    delete[] dx_m;
    delete[] dy_m;
    delete[] dz_m;

}
```

8.2. REGISTERING A USER LIBRARY ON STAR-CCM+

The following summarizes the procedure from writing user-defined functions (UDFs) to creating and registering a library populated by these functions on the STAR-CCM+ software. The instructions in this document are valid for users of Windows operating system but are also easily extended to Linux users; moreover, reference is made to a user code written in C/C++, but the same functionality can be obtained by programming in the Fortran language.

1. *Interface between STAR-CCM+ and user code*

In the working directory, create the file "*uclib.h"* defined as follows. This header file acts as an interface between the STAR-CCM+ software and user-defined functions.

```
#ifndef UCLIB_H
#define UCLIB_H
#ifdef DOUBLE_PRECISION
typedef double Real;
#else
typedef float Real;
#endif
typedef double CoordReal;

#ifdef __cplusplus
extern "C" {
#endif
#if defined(WIN32) || defined(_WINDOWS) || defined(_WINNT)
# define USERFUNCTION_EXPORT __declspec(dllexport)
# define USERFUNCTION_IMPORT __declspec(dllimport)
#else
# define USERFUNCTION_EXPORT
# define USERFUNCTION_IMPORT
#endif

      extern void USERFUNCTION_IMPORT ucarg(void*, char*,const char*, int);
      extern void USERFUNCTION_IMPORT ucfunc(void*, char*,const char*);
      extern void USERFUNCTION_IMPORT ucfunction(void*, char*, char*, int,
...);

      void USERFUNCTION_EXPORT uclib();

#ifdef __cplusplus
}
```

```
#endif
#endif
```

---

2. *User-defined function*

In the same working directory, add the code containing land user function that acts on the coordinates of the mesh vertices. The file "*vertex_exporting.cpp*" exports the coordinates of the vertices of the mesh built inside the numerical solver; "*vertex_motion.cpp*" updates the mesh by reading the new coordinates from files following the morph operation.

```
#include "uclib.h"
#include "RBF_FGP.h"
#include <math.h>
#include <iostream>
#include <string>
#include <fstream>
#include <vector>
#include <cstdlib>
using namespace std;


void USERFUNCTION_EXPORT
vertex_exporting(CoordReal(*result)[3], int size, CoordReal(*vpos)[3])
{
    cout << "Exporting vertex coordinates... ";
    const char file_name_out[100] = "C:\\ASMO_VOLVO\\STAR_vertex_coord.txt";
    fstream file;
    file.open(file_name_out, ios::out | ios::binary);
    if (!file)
    {
        cout << "Error: " << file_name_out << " could not be opened." << endl;
    }
    else
    {
        for (int i = 0; i < size; i++)
        {
            file << vpos[i][0] << " " << vpos[i][1] << " " << vpos[i][2] <<
"\n";
        }
        file.close();
    }
    cout << "Done." << endl;
    cout << "New file created: " << file_name_out << endl;

    for (int i = 0; i < size; i++)
    {
        result[i][0] = (vpos[i][0]);
        result[i][1] = (vpos[i][1]);
        result[i][2] = (vpos[i][2]);
```

```
    }
}


void USERFUNCTION_EXPORT
vertex_motion(CoordReal(*result)[3], int size, CoordReal(*vpos)[3])
{
    cout << "Updating vertex coordinates... ";
    const char file_name[100] = "C:\\ASMO_VOLVO\\C++_vertex_coord.txt";
    FILE *f = fopen(file_name, "r");
    if (f == NULL)
    {
        cout << "Error: " << file_name << " could not be opened." << endl;
        exit(1);
    }
    else
    {
        float x_i, y_i, z_i;
        for (int i = 0; i < size; i++)
        {
            fscanf_s(f, "%f %f %f", &x_i, &y_i, &z_i);
            result[i][0] = x_i;
            result[i][1] = y_i;
            result[i][2] = z_i;
        }
    }
    fclose(f);
    cout << "Done." << endl;
}
```

---

3. <u>User-functions (UDFs) and library (DLL) registration</u>

With the following function (*uclib.cpp*), the user-defined function (*UDF*) is registered in the dynamic link library (*DLL*). The file indicates the input and output parameters of the function and the type of output returned by it.

```
#include "uclib.h"

void zeroGradT(Real*, int, int(*)[2], Real*);
void initVelocity(Real(*)[3], int, CoordReal(*)[3]);
void sutherlandViscosity(Real*, int, Real*);
void vertex_exporting(CoordReal(*)[3], int, CoordReal(*)[3]);
void vertex_motion(CoordReal(*)[3], int, CoordReal(*)[3]);

void USERFUNCTION_EXPORT uclib()
{
    /* Register user functions here */
```

```
    ucfunc(zeroGradT, "BoundaryProfile", "Zero Gradient Temperature");
    ucarg(zeroGradT, "Face", "FaceCellIndex", sizeof(int[2]));
    ucarg(zeroGradT, "Cell", "Temperature", sizeof(Real));

    ucfunc(initVelocity, "RegionProfile", "Initial Velocity");
    ucarg(initVelocity, "Cell", "Centroid", sizeof(CoordReal[3]));

    ucfunc(sutherlandViscosity, "ScalarFieldFunction", "Sutherland Viscosity");
    ucarg(sutherlandViscosity, "Cell", "Temperature", sizeof(Real));

    ucfunc(vertex_exporting, "RegionProfile", "Vertex Exporting");
    ucarg(vertex_exporting, "Vertex", "Coord", sizeof(CoordReal[3]));

    ucfunc(vertex_motion, "RegionProfile", "Vertex Motion");
    ucarg(vertex_motion, "Vertex", "Coord", sizeof(CoordReal[3]));

}
```

---

4. *Creating the dynamic library*

To create a dynamic linked library on STAR-CCM+, open the x64 Native Tool Command Prompt for VS 2022 and enter a series of commands following the instructions below.

1. Access the working directory:
   > cd [PATH]
   where [PATH] represents the address of the directory.
2. Compile the source codes by generating the respective object files:
   > cl /MD /D_WINDOWS -c *.cpp
   to enable double-precision include the command /DDOUBLE_PRECISION.
   (i.e.: cl /MD /D_WINDOWS /DDOUBLE_PRECISION -c uclib.cpp vertex_exporting.cpp vertex_motion.cpp)
3. Link the newly created object files to generate the dynamic library, making use of the UserFunctions.lib file located in the STAR-CCM+ software installation directory:
   > link -dll /out:libuser.dll *.obj [PATH]\UserFunctions.lib
   (i.e.: link -dll /out:libuser.dll uclib.obj vertex_exporting.obj vertex_motion.obj "C:\Program Files\Siemens\16.02.009-R8\STAR-CCM+16.02.009-R8\star\lib\win64\intel20.1vc14.2-r8\lib\UserFunctions.lib")
   The "*libuser.dll*" library is now accessible from *Tools->UserCode* section of STAR-CCM+. After changes to the functions contained in the library, repeat the steps from step 2.

8.3. MS-DOS script

In this paragraph the DOS shell script executed by the DOS command interpreter is presented. This batch file automates the entire workflow.

```
@echo off

set LOGFILE=batch_ASMO_log.log
call :LOG > %LOGFILE%
exit /B

:LOG

setlocal enableextensions enabledelayedexpansion

del /s /q "C:\ASMO_VOLVO\txt\list_ASMO.txt"

cd C:\ASMO_VOLVO\txt
for %%i in (C:\ASMO_VOLVO\txt\ASMO\*.*) do echo %%i >> list_ASMO.txt
echo\
echo ============================================================================
echo ============================================================================
echo\

cd "C:\ASMO_VOLVO\src"
cl /MD /D_WINDOWS /DDOUBLE_PRECISION -c uclib.cpp UDF.cpp
link      -dll      /out:libuser.dll      uclib.obj      UDF.obj      "C:\Program
Files\Siemens\16.02.009-R8\STAR-CCM+16.02.009-
R8\star\lib\win64\intel20.1vc14.2-r8\lib\UserFunctions.lib"
"C:\ASMO_VOLVO\src\RBF_FGP.lib"

echo\
echo ============================================================================
echo\
echo CFD ANALYSIS: ASMO
echo\

set /a count = 0
for %%x in (C:\ASMO_VOLVO\txt\ASMO\*.pts) do set /a count += 1

set /a x = 1
for /f "tokens=* delims= " %%i in ('type "C:\ASMO_VOLVO\txt\list_ASMO.txt"') do
(

      echo\
      echo ============================================================================
      echo\
      echo Simulation number !x! out of %count%
      echo Morphing file being evaluated: %%i
      echo\
      echo ============================================================================
```

```
	cd "C:\ASMO_VOLVO\C++\Morphing_STAR-CCM+\x64\Debug"
	Morphing_STAR-CCM+.exe %%i

	starccm+ -batch step "C:\ASMO_VOLVO\sim\ASMO\ASMO.sim"

	starccm+ -np 18 -batch "C:\ASMO_VOLVO\sim\ASMO\CFD_set_up_ASMO.java"
"C:\ASMO_VOLVO\sim\ASMO\ASMO@00501.sim"

	set /a x += 1

)
echo Batch process completed.

endlocal
```

9. BIBLIOGRAPHY

[1] M. Roser, "Technological Change - Advances in computational technology," 12 March 2022. [Online]. Available: https://ourworldindata.org/technological-change#advances-in-computational-technology.

[2] IEA, "Net Zero by 2050," International Energy Agency, 2021.

[3] I. Burch, "Survey of Global Activity to Phase-Out Internal Combustion Engine Vehicles," 2020.

[4] S. McBain, "Electric Vehicles," International Energy Agency, November 2021. [Online]. Available: https://www.iea.org/reports/electric-vehicles. [Accessed 18 May 2022].

[5] R. Irle, "Global EV sales for 2021," 2021. [Online]. Available: https://www.ev-volumes.com/country/total-world-plug-in-vehicle-volumes/. [Accessed 18 May 2022].

[6] M. Uricar, "Soiling Detection on Automotive Surround-View Cameras," March 2019. [Online]. Available: https://arxiv-export1.library.cornell.edu/pdf/1905.01492. [Accessed 20 May 2022].

[7] L. F. richardson, Weather prediction by numerical process, Cambridge: Cambridge University Press, 1922.

[8] J. H. Ferziger, Computational Methods for Fluid Dynamics, Berlin: Heidelberg Springer Berlin, 2002.

[9] U. Frisch, Turbulence: The Legacy of A. N. Kolmogorov, Cambridge University Press, 1995.

[10] A. Bakker, "Lectures on Applied Computational Fluid Dynamics," 2008. [Online]. Available: https://www.bakker.org/Lectures-Applied-CFD.pdf. [Accessed May 2022].

[11] B. E. Launder and D. B. Spalding, "Computer Methods in Applied Mechanics and Engineering," in *The Numerical Computation of Turbulent Flows*, 1974, pp. 269-289.

[12] I. Ansys, DesignXplorer User's Guide, Canonsburg, 2022.

[13] M. E. Biancolini, Fast Radial Basis Functions for Engineering Applications, Rome: Springer International Publishing AG, 2017.

[14] C. Groth, A. Chiappa and M. Biancolini, "Shape optimization using structural adjoint and RBF mesh morphing," *International Conference on Stress Analysis*, 2017.

[15] G. Fasshauer, "Meshfree approximation method with MATLAB," *World Scientific Pub Co Inc*, 2009.

[16] C. Micchelli, "Interpolation of scattered data: Distance matrices and conditionally positive definite functions," *Constructive Approximation,* vol. 2, no. 1, pp. 11-22, 1986.

[17] C. Groth, *Adjoint-based shape optimization workflows using RBF,* Rome, 2016.

[18] E. Costa, C. Groth, M. E. Biancolini, F. Giorgetti and A. Chiappa, "Structural optimization of an automotive," in *International CAE Conference*, Rome, 2015.

[19] S. V. Patankar, Numerical Heat Transfer and Fluid Flow, New York: McGraw-Hill Book Company, 1980.

[20] H. K. Versteeg and W. Malalasekera, An Introduction to Computational Fluid Dynamics, Harlow: Pearson Prentice Hall, 2007.

[21] G. Falcucci and V. K. Krastev, *Appunti de lcorso di Gasdinamica dei Procesi Industriali: soluzione numerica di flussi incomprimibili,* Rome, 2021.

[22] R. Verzicco, "significato fisico dei termini delle equazioni di," in *Appunti sulla Turbolenza*, 2007, pp. 18-20.

[23] Siemens, *Simcenter STAR-CCM+ User Guide,* Siemens, 2020.

[24] W. Jones and B. Launder, The Prediction of Laminarization with a Two-Equation Model of Turbulence, International Journal of Heat and Mass Transfer, 1972.

[25] F. Moukalled, L. Mangani and M. Darwish, The Finite Volume Method in Computational Fluid Dynamics, An Advanced Introduction with OpenFoam and Matlab, Saint Martin d'Hères Cedex, France: Springer, 2016.

[26] A. Sahay and K. Sreenivasan, "The wall-normal position in pipe and channel flows at which viscous and turbulent shear stresses are equal," *Physics of Fluids,* vol. 11, no. 10, 1999.

[27] A. Godfrey, P. Altmann, M. Johannesson, F. Ross and T. Virdung, "Automated Design Optimization of Side View Mirror Geometries for Improved Autonomous Sensor and Vehicle Soiling Performance," in *SAE International Digital Summit,* 2021.

[28] S. R. Ahmed, H.-J. Emmelmann, K.-D. Emmenthal, H. Flegl, W. Gengenbach, H. Götz, W.-H. Hucho, D. Hummel, G. A. Necati, R. Piatek and M. Rauser, Aerodynamics of Road Vehicles: From Fluid Mechanics to Vehicle Engineering, Würzburg: Butterworth-Heinemann, 1987.

[29] S. L. Vellala, "Shape Optimization of a Car Body for Drag Reduction and to Increase Downforce," 2016.

[30] N. Hall, "National Aeronautics and Space Administration," NASA, 13 May 2021. [Online]. Available: https://www.grc.nasa.gov/WWW/k-12/airplane/tunoret.html. [Accessed 1 June 2022].

[31] M. Khaled, F. Harambat, A. Yamine and H. Peerhossaini, "Aerodynamic forces on a simplified car body - towards innovative designs for car drag reduction," in *ASME 2010*

3rd Joint US-European Fluids Engineering Summer Meeting and 8th International Conference on Nanochannels, Microchannels, and Minichannels, Montreal, 2010.

[32] M. Tsubokura, T. Kobayashi, T. Nakashima, T. Nouzawa, T. Nakamura, H. Zhang, K. Onishi and N. Oshima, "Computational visualization of unsteady flow around vehicles using high performance computing," *Computers & Fluids,* vol. 38, no. 5, pp. 981-990, 2009.

[33] D. Aljure, O. Lehmkuhl, I. Rodriguez and A. Oliva, "Flow and turbulent structures around simplified car models," *Computers & Fluids,* vol. 96, pp. 122-135, 2014.

[34] S. Perzon, D. Aronson and S. Brahim, "Multigrid heat transfer calculations using different iterative schemes," *SAE Technical Paper,* 2000.

[35] M. E. Biancolini, "Il metodo 50:50:50. Ottimizzazione aerodinamica della Volvo XC60," *ATA Ingegneria dell'Autoveicolo,* vol. 65, no. 7, pp. 36-47, 2012.

[36] R. L. Lietz, "Vehicle Aerodynamic Shape Optimization," *SAE International,* 2011.

[37] K. Yoneda, N. Suganuma, R. Yanese and M. Aldibaja, "Automated driving recognition technologies for adverse weather conditions," *IATSS International Association of Traffic and Safety Sciences,* vol. 43, pp. 253-262, 2019.