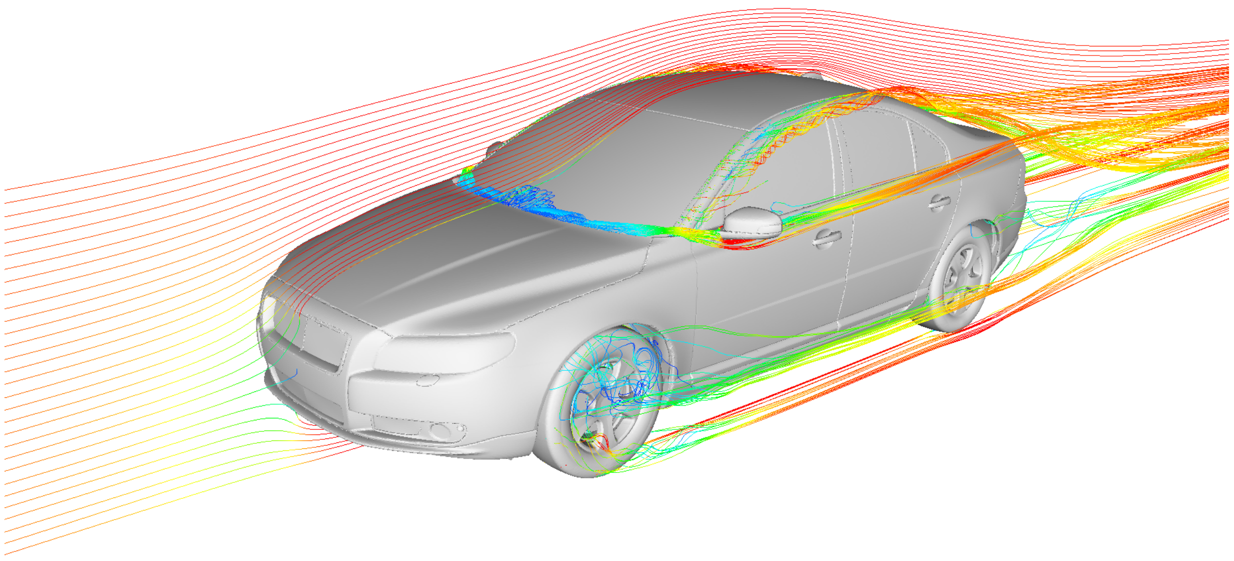
CHALMERS



OpenFOAM: A tool for predicting automotive relevant flow fields

Master's Thesis in Automotive Engineering

BASTIAN NEBENFÜHR

Division of Fluid Dynamics
Department of Applied Mechanics
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2010

Master's Thesis 2010:35

MASTER'S THESIS 2010:35

OpenFOAM: A tool for predicting automotive relevant flow fields

Master's Thesis in Automotive Engineering

BASTIAN NEBENFÜHR

Division of Fluid Dynamics
Department of Applied Mechanics
CHALMERS UNIVERSITY OF TECHNOLOGY

Göteborg, Sweden 2010

OpenFOAM: A tool for predicting automotive relevant flow fields
BASTIAN NEBENFÜHR

Master's Thesis 2010:35
ISSN 1652-8557
Division of Fluid Dynamics
Department of Applied Mechanics
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: + 46 (0)31-772 1000

Cover:
S80 with streamlines colored in terms of velocity

This document was typeset using LaTeX

Printed by Chalmers Reproservice
Göteborg, Sweden 2010

OpenFOAM: A tool for predicting automotive relevant flow fields

Bastian Nebenführ

Division of Fluid Dynamics
Department of Applied Mechanics
Chalmers University of Technology

Abstract

Turbulent flow is a common phenomenon in engineering problems such as ground vehicle or aircraft development. With the help of Computational Fluid Dynamics (CFD), it is possible to perform numerical simulations that enable the prediction of flow behavior for these applications. At Volvo Car Corporation (VCC), the commercial Finite Volume (FV) code Fluent is used. In contrast to commercial codes, there are also several open source CFD codes, such as OpenFOAM. Commercial, in this case, refers to the fact that licenses as well as support have to be purchased, whereas open source software is free of cost (support may not be for free, however). This thesis work aims to provide an evaluation of the results obtained with OpenFOAM and compare them to the results obtained with Fluent and experimental wind tunnel data.

Different cases are investigated in this work. Incompressible steady-state Reynolds Averaged Navier-Stokes (RANS) simulations are carried out in order to predict the lift and drag values of a generic vehicle model. In this case, the main focus is on methodology transparency, stability and accuracy. Incompressible Large Eddy Simulations (LES) are carried out in order to predict the aeroacoustic behavior of a generic Sports Utility Vehicle (SUV). Especially the flow induced noise due to the side mirror and the A-pillar is investigated. In the SUV case, the main interest is on lead time, robustness and quality of results. In the end, incompressible steady-state RANS simulations are carried out in order to predict the lift and drag force of a representative production vehicle. The treatment of so called baffles and the use of rotating wheels are of particular interest here.

For the first case, the results obtained with OpenFOAM correspond well with the Fluent reference data. The convergence rate and simulation time are comparable for both Fluent and OpenFOAM. In the aeroacoustics case, results that are similar to those obtained by Fluent are found. This was also the case for the detailed production vehicle, where an improved mesh quality was required.

Keywords: OpenFOAM, CFD, aero-acoustics, aerodynamics, LES, RANS

Göteborg June 2010
Bastian Nebenführ

Sammanfattning

Turbulent flöde är ett vanligt fenomen i tekniska problem, som t.ex. fordon eller flygplan utveckling. Med hjälp av Computational Fluid Dynamics (CFD) är det möjligt, att utföra numeriska simuleringar, som hjälper att förutsäga flödets beteende för dessa tillämpningar. På Volvo Car Corporation (VCC) används den kommersiella Finite Volym (FV) koden Fluent. Till skillnad från kommersiella koder finns också flera open source CFD koder, t.ex. OpenFOAM. Kommersiell betyder i detta fall att licenser samt support måste köpas, medan open source står för kostnadsfria programvaror (kostnad för användarstöd kan dock tillkomma). Detta examensarbete syftar till att ge en utvärdering av de resultat som erhållits med OpenFOAM mot de resultat som erhållits med Fluent och experimentella resultat från vind tunneln.

Olika fall undersöks i detta arbete. Inkompressibla stationära Reynolds Averaged Navier-Stokes (RANS) simuleringar utförs för att förutsäga lyft- och motståndskoefficienten av en generisk bilmodell. I detta fall är det största intresset metodens transparens, stabilitet och noggrannhet. Inkompressibla Large Eddy Simulations (LES) utförs i syfte att förutsäga aero-akustiska beteende av en generisk Sports Utility Vehicle (SUV). Speciellt flödes-inducerat buller orsakat av sidospegeln och A-stolpen undersöks. I detta fall är det största intresset ledtid, robusthet och kvalitet av resultat. Till slut utförs inkompressibla stationära RANS simuleringar för att förutsäga lyft- och motståndskoefficienten av ett representativt fordon. Speciellt behandlingen av så kallade bafflar och användningen av roterande hjul är av intresse här.

För det första fallet är resultaten med OpenFOAM i god överensstämmelse med Fluent referensdata. Konvergens och simulering tid är jämförbara för båda programmen. För det aero-akustik fallet får vi liknande resultat i jämförelse med Fluent. Även för det representativt fordon, vilket kräver en förbättrad nätkvalitet, liknande resultat hittas jämfört med Fluent.

Göteborg Juni 2010
Bastian Nebenführ

Nomenclature

Flow quantities

p	pressure
c	speed of sound

Greek symbols

ρ	density
ε	dissipation
ω	vorticity
μ	dynamic viscosity
ν	kinematic viscosity
Δ	LES length scale or increment
τ_{ij}	viscous stress tensor
ν_T	Smagorinsky eddy viscosity
$\tilde{\nu}$	viscosity-like variable in Spalart-Allmaras model
τ_w	wall shear stress
η	Kolmogorov length scale

Roman letters

v_i	instantaneous velocity
U_∞	free stream velocity
U	local velocity
u^*	friction velocity
x, y, z	Coordinate directions or distance in the according direction
D	Drag force
L	Lift force or characteristic length or wheel base
S_{ij}	stress strain rate
N	number of grid points
L_{int}	integral length scale
h	cell size
k	turbulent kinetic energy
A	projected area
t	time
P	Production term

Dimensionless quantities

C_s	Smagorinsky constant
Co	Courant number
C_d	Drag coefficient
C_l	Lift coefficient
C_p	Pressure coefficient
Re	Reynolds number
Ma	Mach number
$C_\mu, A_0, C_2, \sigma_k, \sigma_\varepsilon$	Constants in realizable $k-\varepsilon$ model
x^+, y^+, z^+	dimensionless wall distance

Subscripts

w	wall
t	turbulent
rms	Root Mean Square
∞	ambient condition
$front$	located in the front
$rear$	located in the rear
$total$	total quantity

Superscripts

$'$	fluctuation
$\bar{}$	time-average

Abbreviations

CFD	Computational Fluid Dynamics
BC	Boundary Condition
FVM	Finite Volume Method
FEA	Finite Element Analysis
CAA	Computational Aeroacoustics
VCC	Volvo Car Corporation
$OpenFOAM$	Open Source Field Operation And Manipulation
$RANS$	Reynolds Averaged Navier-Stokes
LES	Large Eddy Simulation
SGS	Sub-Grid Scale
DNS	Direct Numerical Simulation
CFL	Courant-Friedrichs-Lewy or Courant number
SUV	Sports Utility Vehicle
MRF	Multiple Reference Frame
RSM	Reynolds Stress Model
RMS	Root Mean Square
PSD	Power Spectral Density
GUI	Graphical User Interface
SD	Standard Deviation
URF	Under Relaxation Factor

Preface

This project evaluates the outcomes of the open source CFD code, OpenFOAM, against Fluent results for three different cases. The objective was to find out whether OpenFOAM might be able to replace Fluent at VCC for all (or at least, some) specific applications. Changing from Fluent to OpenFOAM would imply a great financial benefit for VCC. The work was carried out at the Fluid Dynamics Center of VCC during the period of February to June 2010 with Dr. Jonas Ask as industrial supervisor. The academic supervisor at the Division of Fluid Dynamics, Department of Applied Mechanics, Chalmers Technical University, Göteborg, Sweden, was Professor Lars Davidson.

Acknowledgements

First of all, I want to express my gratitude to my supervisor at VCC, Dr. Jonas Ask, for giving me the opportunity to work on such an interesting and challenging subject and helping with my first steps in using OpenFOAM. Thank you very much also for the never ending patience you had in explaining aspects of CFD and aeroacoustics, even though my questions might have been trivial sometimes.

I also want to thank my supervisor at the Division of Fluid Dynamics at Chalmers, Professor Lars Davidson, for being always supportive, which made the cooperation between industry and university very effective. Further, you left me all the freedom I wanted and needed to investigate many interesting things during these four months.

Furthermore, I am grateful to all my colleagues at the Fluid Dynamics Center at VCC, who were always very open minded and quickly integrated me into the group. It was also great that you were often speaking Swedish with me, which gave me the opportunity to improve also in this non-scientific field. To my Korean colleague and friend, "Chris", thank you for loads of entertainment during the breaks we had from work. It has been a great time, both in the office and in our spare time.

Thank you, assistant Professor Håkan Nilsson from Chalmers and Professor Hrvoje Jasak from Wikki Ltd., for quick and helpful email support. Without you, it would not have been possible to improve that quickly with OpenFOAM.

To all my friends, thank you for distracting me in my spare time from thinking of OpenFOAM and CFD.

Last, but not least, I want to thank my girlfriend Anke for always supporting me in the hard times I had during this project and for always believing in me.

Contents

Chapter 1

Introduction

Computational Fluid Dynamics (CFD) is widely used in the industry today. With its help, it is possible to numerically simulate and thus predict flow behavior. Two examples for applications of CFD in the industry are the aerodynamic development of airplanes or ground vehicles and weather forecasting.

1.1 Background

In order to be more environmentally friendly and fuel efficient, vehicles need to reduce aerodynamic resistance. Since wind tunnel experiments are expensive, time consuming and sometimes inefficient with regard to lead time, it is often beneficial to perform optimization processes with CFD simulations. A great advantage of CFD simulations in comparison to wind tunnel experiments is that changes are easy to implement and hence parameter studies can be carried out quite easily. Furthermore, parameter studies are more easily controlled, when using CFD. This increases the efficiency of the entire optimization process. However, in some cases, CFD simulations need to be validated through experimentation. Thus, wind tunnel experiments cannot be neglected entirely.

Another research field of great importance is referred to as aeroacoustics and this involves investigating the problem of undesirable noise created by air flows passing by the vehicle. Research concerning aeroacoustics has been carried out already since the 1950's. Most vehicles of that time produced considerably high noise levels, independent of model or manufacturer. This is one reason why the customers showed a high tolerance to this kind of noise. However, complaints have been raised when the noise made it impossible to have conversations in the vehicle or listen to the radio. It took until the 90's that some car manufacturers began investing in improved research equipment, such as aeroacoustic wind tunnels. This resulted in a considerable reduction of customer complaints for those manufacturers, whereas the complaints were kept at a constant level or were even increasing for the manufacturers that were not investing in aeroacoustic research. The rise in complaints can be explained by the customers experiencing a more silent car and thus not being satisfied with their old vehicles anymore. Therefore it can be expected that the remaining manufacturers will also invest in aeroacoustic research in the following years.

Traditionally, the aeroacoustics of a new vehicle were determined in an early phase of the development, based on either experience or empirical relations. The validity of the assumptions were controlled by measurements on prototypes or production vehicles. This means that any problems with the aeroacoustic behavior that were detected during this late phase of the vehicle development usually had to be solved in an expensive way.

Due to the quick increase in computational capacities, it is more and more possible to carry out aeroacoustic research on virtual models at an early stage of the development. This is referred to as Computational Aeroacoustics (CAA). Furthermore, the increased computational power allows accurate simulations of high resolved models with LES, which shows that there is a bright future for CAA in the automotive industry.

1.2 Purpose

The purpose of this project is to make a comparison between the two computational fluid dynamics codes Fluent and OpenFOAM. Fluent is a commercial code, which is currently used at the Fluid Dynamics Center of Volvo Car Corporation (VCC) for external and internal CFD simulations. Contrary to Fluent, OpenFOAM is an open source CFD code, which implies that there are no licensing costs involved and that the software can be downloaded from the internet for free. It goes without saying that it would be a great financial benefit for VCC, if all their simulations could be carried out with OpenFOAM instead of Fluent. Since the license of OpenFOAM is cost-free, it is especially interesting for parallel processing, which cannot be circumvented by VCC for performing high quality simulations involving massive mesh domains. If it were possible to change to OpenFOAM, the money currently spent on licensing costs could instead be invested in e.g. more computational power, which will make the research in the Fluid Dynamics Center of VCC more efficient.

1.3 Approach

Several different reference cases, primarily from the fields of aeroacoustics and aerodynamics, are evaluated using the open source code OpenFOAM. The aim is to reproduce or overcome the results obtained by the commercial code Fluent. The following cases are investigated in this thesis work:

- Steady-state incompressible Reynolds Averaged Navier-Stokes (RANS) simulations in order to predict the aerodynamic drag and lift coefficient of a generic production car model. The results obtained are compared to an identical Fluent reference case as well as to experimental data. In the simulations, a free stream velocity of 27.8 m/s (100 km/h) and a yaw angle of 0 degrees (no crosswind) are applied. For this case, the main focus is on methodology transparency, stability and accuracy.
- Transient, incompressible Large Eddy Simulation (LES) of the flow around a simplified Sport Utility Vehicle (SUV) in order to investigate the aeroacoustic behavior. The main focus is set to the flow around the A-pillar and the side-mirror, which are expected to be the most important sources of wind-rush noise due to strong separations and the vicinity to the driver. As in the first case, here the results are also compared to those of a Fluent reference case, as well as to experimental data. In the simulations, a free stream velocity of 39.0 m/s (140.4 km/h) and a yaw angle of 0 degrees (no crosswind) are applied. For this case the main focus is on lead time, robustness and quality of results.
- Steady-state incompressible RANS simulations in order to predict the aerodynamic drag and lift coefficient of a detailed production car model with the use of rotating boundaries. This allows the implementation of rotating wheels, which is supposed to increase the accuracy. The basic procedure of applying rotating boundary conditions in OpenFOAM and the treatment of so called baffles (see section 2.1.5) is the main focus of this investigation. In the simulations, a free stream velocity of 27.8 m/s (100 km/h) and a yaw angle of 0 degrees (no crosswind) are applied. Reference material of experiments as well as of Fluent results is available in order to enable an evaluation of the simulation outcomes.

1.4 Limitations

Due to the limited time-frame of only 20 weeks, this project is not aiming to evaluate all areas of the wide range of CFD applications at VCC, but mainly aerodynamics and aero-acoustics. Fields like climate control and heat management are not part of this thesis. Furthermore, there will be only limited pre-processing and little mesh generation involved in this work. Good meshing is essential in order to obtain reliable results, but since only reference cases are to be investigated, preferably the same meshes should be used to guarantee comparability. The evaluation process will be based on the already existing solvers and applications in their present state only, which means that no programming will be done during this project.

Chapter 2

Theoretical background

In this chapter, the theoretical background of fluid dynamics will be presented, as will the software used during this project.

2.1 Fluid dynamics

2.1.1 Turbulent flow

The external flow around a ground vehicle at cruising speed is either laminar or turbulent. In the laminar flow regime adjacent layers of fluid slide past each other in an orderly fashion and the flow is smooth. However, at cruising speed most of the external flow is turbulent. Turbulence is a chaotic state of a fluid with stochastic property changes, including rapid variation of pressure and velocity. Other important features of turbulent flow are increased diffusivity and three-dimensionality. Turbulent incompressible flow is fully described by the Navier-Stokes equations Eq. (2.1) and the continuity equation Eq. (2.2).

$$\rho \frac{\partial v_i}{\partial t} + \rho \frac{\partial v_i v_j}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \mu \frac{\partial^2 v_i}{\partial x_j \partial x_j} \tag{2.1}$$

$$\frac{\partial v_i}{\partial x_i} = 0 \tag{2.2}$$

A possibility to distinguish between laminar and turbulent flow is provided by the dimensionless Reynolds number in Eq. (2.3) as described by Reynolds in [1].

$$Re = \frac{U_\infty L}{\nu} \tag{2.3}$$

where U_∞ is the free stream velocity, L is a characteristic length and ν is the kinematic viscosity. Flow is considered as turbulent above the critical Reynolds number Re_{crit}. In turbulent flow, unsteady vortices appear and interact with each other. The vortices appearing have different sizes and are referred to as eddies. Large eddies provide progressively smaller ones with kinetic energy, until the energy is finally dissipated in the smallest eddies, the so called Kolmogorov scales. This progress is referred to as the energy cascade. Small eddies are dominated by viscous effects, whereas large eddies are dominated by inertia effects and viscous effects are negligible. For calculations, all properties of the flow are important, which is the reason why simulations are very complex. In order to resolve even the smallest eddies into detail, the cell-size of a computational mesh has to be of the same order of magnitude, which leads to an incredibly high demand on processing power.

Another important number is the Mach number. It is a dimensionless number that represents the ratio of inertial forces to compression forces, but is usually given as defined in Eq. (2.4).

$$Ma = \frac{U_\infty}{c} \tag{2.4}$$

U_∞ is again the free stream velocity and c is the speed of sound. Another important application of the Mach number is that it can be used to distinguish between compressible and incompressible flow. Usually flows with a Mach number below 0.3 are considered as incompressible, since the compressibility effects have a small influence on the flow behavior.

2.1.2 Direct Numerical Simulation (DNS)

In DNS, the Navier-Stokes equations are solved directly without any turbulence model. This provides the most accurate results, but it is necessary to resolve all spatial and temporal scales into detail. The smallest scales to be resolved are the Kolmogorov scales (Eq. (2.5)) and the largest scales are of the integral length scale L_{int}.

$$\eta = \left(\frac{\nu^3}{\varepsilon}\right)^{\frac{1}{4}} \tag{2.5}$$

Further, the number of grid points N in one direction of the mesh with a cell size h has to fulfill the requirement $Nh > L_{int}$, so that the domain at least covers the largest scale. In order to be able to resolve the Kolmogorov scale, $h \leq \eta$, has to be fulfilled. Assuming that $\varepsilon \approx v_{3,RMS}/L_{int}$ and that the mesh should be three-dimensional (N^3), yields

$$N^3 \geq Re^{9/4} \tag{2.6}$$

where Re is the Reynolds number (compare Eq. (2.3)), based on v_{RMS}, the Root Mean Square (RMS) of the velocity, and L_{int}. Because of the requirement in Eq. (2.6), DNS is very extensive and therefore limited to low Reynolds numbers. Its main application is in research and academia. For normal engineering applications, either RANS or LES solutions are preferred.

2.1.3 Reynolds Averaged Navier-Stokes (RANS)

The RANS equations are a simplification of the instantaneous Navier-Stokes equations. In order to obtain the RANS equations, the instantaneous properties are split up into a mean (time-averaged) component and a fluctuating component. For example,

$$v_i = \bar{v}_i + v_i' \tag{2.7}$$

with $\bar{v}_i$ being the time-averaged component.

Now, one can replace the instantaneous properties in the Navier-Stokes equations by the Reynolds decomposition. The resulting equations are referred to as the Reynolds averaged Navier-Stokes equations or the Reynolds equations.

$$\rho\frac{\partial \bar{v}_i}{\partial t} + \rho\frac{\partial \bar{v}_i\bar{v}_j}{\partial x_j} = -\frac{\partial \bar{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\mu\frac{\partial \bar{v}_i}{\partial x_j} + \rho\overline{v_i'v_j'}\right) \tag{2.8}$$

In Eq. (2.8) $\rho\overline{v_i'v_j'}$ appears as a new term on the right hand side. This term is unknown and represents an additional stress tensor due to fluctuating velocities, referred to as the Reynolds stress tensor. Since the Reynolds stress tensor is unknown, it leads to the so called closure problem. There are ten unknowns, but only four equations (three components of Eq. (2.8) and the time-averaged continuity equation). Thus the Reynolds stresses have to be modeled in order to close the system of equations. Already in 1877, the first attempts of describing the turbulent stresses were undertaken by Boussinesq [2], who introduced the concept of the eddy viscosity. This concept became extremely successful and is widely known as the Boussinesq eddy-viscosity approximation or simply the Boussinesq approximation. According to Wilcox [3], there is a wide range of turbulence models available in four main categories:

1. Algebraic or Zero-Equation Models
2. One-Equation Models
3. Two-Equation Models
4. Reynolds Stress Models (RSM)

Algebraic turbulence models are the simplest of all and they use the Boussinesq approximation in order to compute the Reynolds stress tensor.

In one-equation models one transport equation of a turbulent property (usually the turbulent kinetic energy k) is solved. However, the one-equation Spalart-Allmaras model (Spalart and Allmaras [4]) is considered to be the most accurate and it solves a transport equation for a viscosity-like variable $\tilde{\nu}$.

Two-Equation models are the most commonly used turbulence models. They solve the transport equation for turbulent kinetic energy and an additional transport equation for a turbulence length scale or similar. The most famous two-equation models are the $k-\omega$ model, which was presented by Kolmogorov [5] in 1942, and the $k-\varepsilon$ model by Jones and Launder [6]. The latter can be seen as an industrial standard and also found great application in this work. Especially the implementation of the realizable $k-\varepsilon$ model is described later in section 2.2.1.

All models in the categories mentioned above are based on the Boussinesq approximation. In RSM, however, the eddy viscosity approach has been put aside and the Reynolds stresses are modeled separately.

2.1.4 Large Eddy Simulation (LES)

LES is another way of solving the Navier-Stokes equations. In 1970, Deardorff [7] published the first results of a Large Eddy Simulation and by now the technique has matured quite a lot. Not only the underlying theory advanced, but also the available computing power increased. In 1941, Kolmogorov published [8] his theory of self-similarity, which implies that the largest eddies of a flow are dependent on the flow geometry, whereas the small eddies are self-similar and more universal. Hence, in LES, the large energy containing eddies are calculated explicitly, while the eddies that are smaller than the filter width are implicitly treated by a so called Sub-Grid Scale (SGS) model. Assuming that the small scales are more homogeneous and less affected by the boundary conditions, one hopes that simpler models as compared to RANS can be sufficient (see Piomelli and Chasnov [9]). In order to distinguish between resolvable and sub-grid scales, some kind of filtering has to be applied to the Navier-Stokes equations. Popular filters in LES are, for example, the Fourier cutoff filter and the Gaussian filter. Since filters and their application are not in the scope of this project, the reader is kindly referred to literature, such as Ferziger [10], in order to get further information. However, the outcome of each filter is a scale Δ, which represents the smallest scale to be resolved.

In 1963, Smagorinsky [11] introduced the first SGS model. This model is an eddy-viscosity model. The sub-grid scale stresses τ_{ij} are modeled in the following way.

$$\tau_{ij} = 2\nu_T S_{ij} \tag{2.9}$$

where

$$S_{ij} = \frac{1}{2}\left(\frac{\partial \bar{v}_i}{\partial x_j} + \frac{\partial \bar{v}_j}{\partial x_i}\right) \tag{2.10}$$

is the resolved strain rate and

$$\nu_T = (C_S\Delta)^2\sqrt{S_{ij}S_{ij}} \tag{2.11}$$

the Smagorinsky eddy viscosity. C_S is referred to as the Smagorinsky constant and often is assigned a value between 0.1 and 0.2. The value depends on the flow and the geometry. There are even cases, in which 0 is used for the Smagorinsky constant (MiLES).

An important number for LES or other transient simulations is the CFL (Courant-Friedrichs-Lewy) number, which sometimes is also referred to as the Courant number. It can be seen as a stability criterion for transient simulations with explicit time-discretization.

$$Co > \frac{U\Delta t}{\Delta x} \tag{2.12}$$

In Eq. (2.12), U is the local velocity, Δt is the timestep size and Δx is the grid resolution. In order to have stability, the Courant number should commonly be below 1 for each cell. Since mostly the grid and the velocity are given parameters, the only possibility to manipulate the Courant number is the timestep size. This can be a very limiting constraint on the time necessary to perform a simulation. A physical explanation can be that the Courant number determines how many cells of the mesh a fluid element passes during one timestep.

2.1.5 Baffles

One of the difficulties in CFD are the so called baffles. A baffle is defined as an infinitesimally thin layer of material in a grid. The formation of baffles can be either intentional, such as a very thin spoiler lip on a detailed production car model, or unintended, such as artifacts from the meshing process itself. The difficulty with the existence of baffles is the fact that, since they are infinitesimal thin, it is hard for the solver to distinguish in which direction the surface normal vector has to point. In order to circumvent this problem, CFD codes are creating what is referred to as the shadow image of the baffle, where each of the shadows has the surface normal vectors pointing in opposite directions.

Figure 2.1 shows on the left hand side a very thin wall (representing the baffle) with normal vectors pointing in different directions. On the right hand side one can see two separate walls with distinct normal vectors.

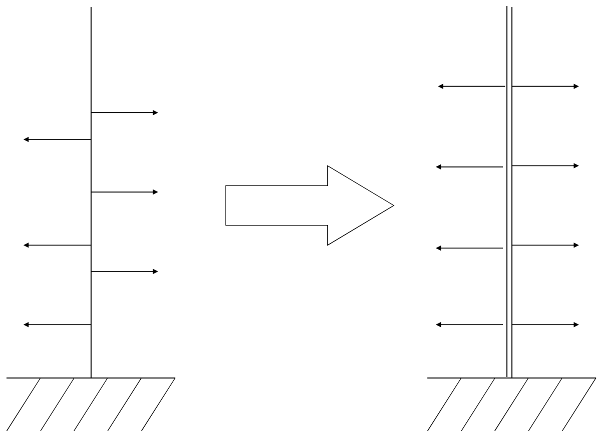


Figure 2.1: Conversion from infinitesimal thin surface into two surfaces with clear surface normal vectors

2.1.6 Non-Dimensional Coefficients

There are several different coefficients used in aerodynamics and this sections will cover the ones that were used during this work.

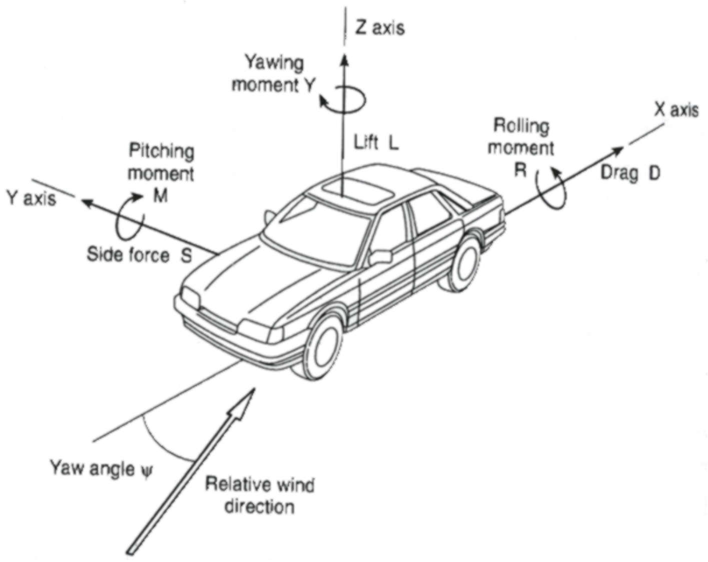


Figure 2.2: Forces and moments on a vehicle according to Barnard [12]

2.1.6.1 Drag coefficient

The drag coefficient is an important measure in automotive design, since improved fuel efficiency (at least at higher speeds) is a direct consequence of a low drag coefficient. Drag is specified as the resistance force of an object moving through a fluid environment, such as water or air. The drag force mainly consists of two different contributions, a pressure force due to the shape of the object and a viscous force due to surface friction. For airfoils, the drag force is dominated by viscous forces, whereas for bluff bodies, such as cars, the pressure forces are far more important.

$$C_d = \frac{D}{\frac{1}{2}\rho U_\infty^2 A} \tag{2.13}$$

Equation (2.13) shows how the drag coefficient is computed and that it is dimensionless. The drag force D (see Fig. 2.2) is divided by the dynamic pressure ($\frac{1}{2}\rho U_\infty^2$) multiplied with the projected area A in x-direction. Typical values for the drag coefficient currently range between 0.25 and 0.35, as seen online in [13]. Since the drag coefficient depends on the shape, it is easy to imagine that sedan vehicles usually are found at the lower end of the range, whereas SUVs dominate the upper end. Often the product C_dA is given instead of the drag coefficient itself.

2.1.6.2 Lift coefficient

The lift coefficient is defined in a similar way to the drag coefficient. As Fig. 2.2 shows, the lift force is the force working in direction of the z-axis. Knowing the lift force, one can compute the lift coefficient according to Eq. (2.14).

$$C_l = \frac{L}{\frac{1}{2}\rho U_\infty^2 A} \tag{2.14}$$

Out of convenience, the definition still contains the projected frontal area, whereas the lift force would actually be more related to the projected bottom area. Typical values for the lift coefficient are nowadays around 0 for well designed cars, according to Barnard [12]. One should be aware of the fact that for racing cars, large negative lift coefficients are normal, which are produced by a negative lift force. A negative lift force is usually referred to as down force and it is beneficial for racing cars, since it permits higher cornering speeds.

Usually the lift force is different for the front and the rear axle and consequently, the lift coefficient is also different. The front and rear contributions sum up to the total lift coefficient as follows.

$$C_{l,total} = C_{l,front} + C_{l,rear} \tag{2.15}$$

2.1.6.3 Pressure coefficient

Similar to the two coefficients previously described, the pressure coefficient is also a dimensionless quantity. It describes the relative pressure in a flow field, independent of the vehicle speed. The pressure coefficient is defined as shown below.

$$C_p = \frac{p - p_\infty}{\frac{1}{2}\rho_\infty U_\infty^2} \tag{2.16}$$

The subscript ∞ denotes free stream quantities. Thus, the pressure coefficient is, according to Eq. (2.16), the difference between the local static pressure and the free stream static pressure divided by the free stream dynamic pressure. Since the pressure coefficient is independent from the vehicle speed, it is much more convenient to use and easier to compare than the local static pressure.

2.1.7 Boundary layers

As an object moves through a fluid, or as a fluid moves past an object, the fluid particles near the object are disturbed and move around the object. The fluid right next to an object stick to the surface of the object and have zero velocity. These fluid particles will slow down the velocity of other fluid particles a little further away from the surface. This creates a thin layer of fluid near the surface in which the velocity changes from zero at the surface to the free stream value away from the surface and which is referred to as the boundary layer because it occurs on the boundary of the fluid. The boundary layer itself is split up into three different parts:

1. Laminar layer
2. Transition layer
3. Turbulent layer

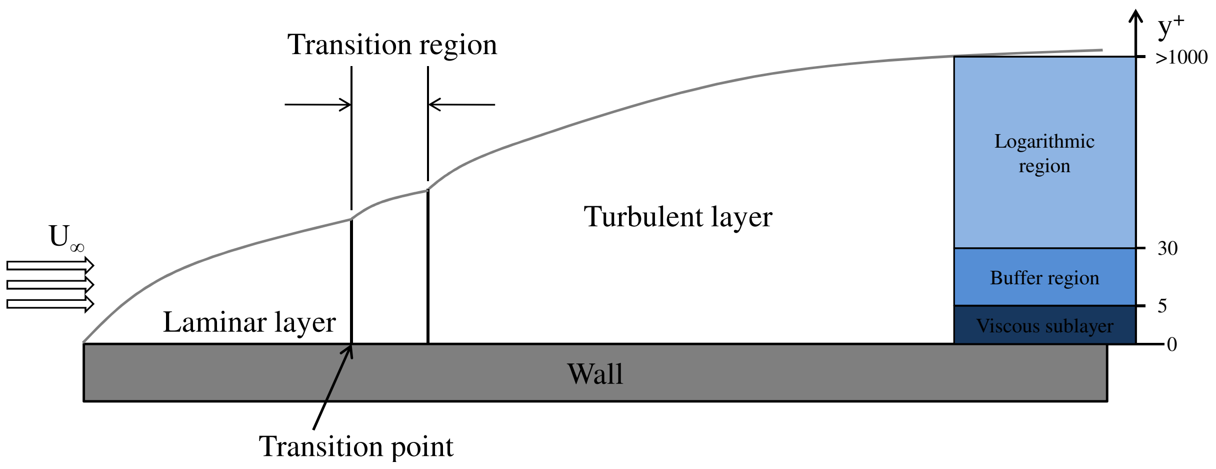


Figure 2.3: Different boundary layer regions

Near the leading edge of the surface that is exposed to the flow, the fluid flows smoothly without turbulent disturbances. In this region, the flow behaves rather like a stack of flat sheets of fluid, sliding next to each other, while the outer sheets are sliding faster than the inner ones. This is referred to as the laminar layer. Further downstream there will be a sudden change in the flow from laminar to turbulent, which is referred to as the transition region. After the transition region, the boundary layer is fully turbulent with turbulent motions of a very small scale. Close to the surface one will always find a very thin layer of flow, which is dominated by viscous diffusion and therefore referred to as the viscous sublayer. Further away from the wall, the so called logarithmic region follows, in which turbulent diffusion due to inertia dominates. In the buffer region, the viscous shear stresses are gradually replaced by turbulent shear stresses. Both shear stresses are equal at a y^+ value of around 11.

The quantity y^+ is referred to as the dimensionless wall distance and is defined as follows.

$$y^+ = \frac{u^* y}{\nu} \tag{2.17}$$

with $u^* = \sqrt{\frac{\tau_w}{\rho}}$ as the friction velocity, y as the distance to the wall and ν as the kinematic viscosity.

2.1.8 Wall functions

There are different approaches to model the flow very close to the walls in a domain, for example in the viscous sublayer (see section 2.1.7). One approach is to actually resolve the region in close proximity to the wall, as shown in Fig. 2.4. This makes it possible to calculate the flow all the way down to the wall. A disadvantage of this method is that in order to resolve the viscous sublayer properly, one needs to increase the resolution of the mesh significantly. This leads to an increased demand for computational power, which makes simulations very time consuming.

Another method to treat the flow close to the wall are the so called wall functions. Wall functions were originally developed for flat plates with small or no pressure gradients. In order to use this approach, the mesh at the wall has to be rather coarse ($y^+ \geq 30$), so that the viscous sublayer is not resolved and wall functions can be applied instead (Fig. 2.5). Wall functions are semi empirical formulas that describe how the flow in the viscous sublayer behaves under normal conditions. Wall functions provide sufficient accuracy for most high-Reynolds number flows, but may show problems in complex flows involving massive separation and reattachment. Also, the wall function approach shows deficiencies when it comes to geometries with strong curvatures. However, the approach to use wall functions is very popular, especially in industry, since it decreases the necessary number of computational cells in a mesh and thus simulations can be carried out in less time.

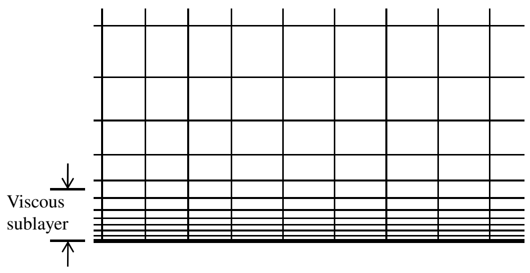


Figure 2.4: Resolved near wall region

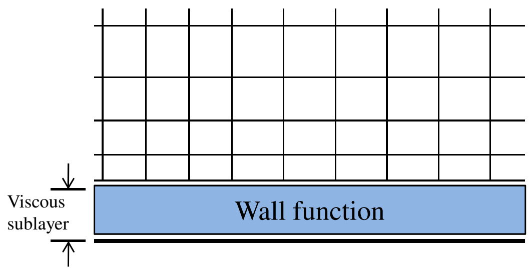


Figure 2.5: Applied wall functions

A third attempt of near-wall treatment is a combination of both methods described above. At some points, the mesh is fine enough to resolve the flow and at some it is suitable to use wall functions. In OpenFOAM, the y^+ value for the first grid point is calculated and the result is used to decide whether to apply wall functions or not. Wall functions are applied if the first grid point is located in the logarithmic region, more precisely, if the first grid point has a y^+ above 30 (see Fig. 2.3).

2.2 Software

This section describes the software packages that were mainly used during the project.

2.2.1 OpenFOAM

OpenFOAM stands for Open Source Field Operation And Manipulation (OpenFOAM) and is an open source software package for CFD, which is produced by a commercial company, OpenCFD. The software package can be seen as a C++ library that provides two kinds of applications, solvers and utilities. Solvers are designed to solve a specific problem in continuum mechanics, whereas utilities are performing data manipulation tasks. For users with knowledge in programming, it is possible to add new solvers and utilities or modify the existing ones in order to fit the desired purpose.

Since it is licensed under GNU General Public Licence, it is free of charge and thus very interesting for academic and industrial use with largely parallel applications.

2.2.1.1 File structure

In order to work with OpenFOAM, the user needs to be familiar with the file structure, since there is no Graphical User Interface (GUI). During this project work, mainly RANS simulations were carried out, thus the following explanations are given for the file structure of a RANS case. However, for LES, the file structure would be very similar. Figure 2.6 aims to explain the general structure.

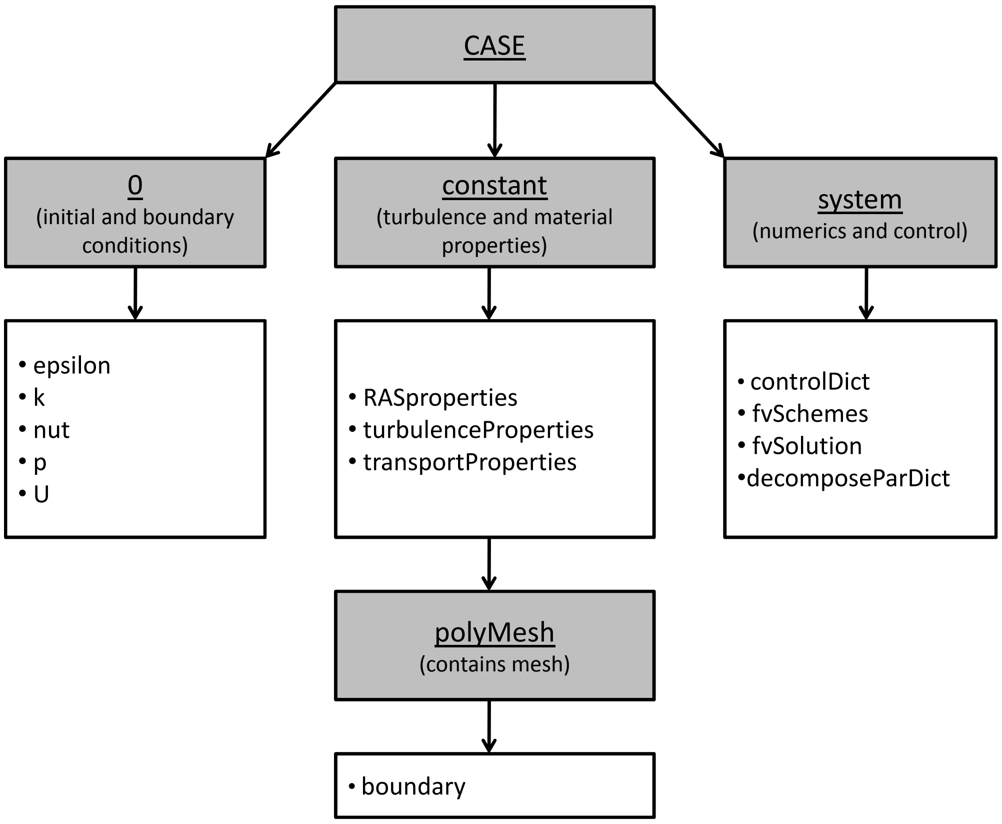


Figure 2.6: File structure of OpenFOAM

The content of all the required files is given in Appendix A.

In the parent folder of the case, one will find three subdirectories, namely *0*, *constant* and *system*. The *0* directory contains files in which the initial conditions for the different variables and surface patches can be specified. Examining, for example, the file *p* (Appendix A.1) in which the pressure field is treated, shows that there are three main entries to be done.

In *dimensions*, one specifies the dimensions of the variable: in this case pressure divided by density (pressure in OpenFOAM is always divided by density for the incompressible solvers!), which yields $[m^2/s^2]$. The different possible units in this entry are shown in Table 2.1:

Table 2.1: Possible units for the dimensions entry

1	Mass [kg]
2	Length [m]
3	Time [s]
4	Temperature [K]
5	Quantity [kgmol]
6	Current [A]
7	Luminous intensity [cd]

internalField specifies the internal field as a scalar (*uniform*) or a vector (*nonuniform*).

boundaryField specifies the boundary field. The boundary and initial conditions for the different surface patches are set here.

In the *constant* directory, turbulence and material properties are specified. The file *RASproperties* contains the option to choose a RANS turbulence model or to change to laminar flow, if desired. *transportProperties* contains the possibility to specify the value of the kinematic viscosity ν. Editing the file *turbulenceProperties* enables changes in the constants of the different turbulence models. As shown in Fig. 2.6, there is a yet another subfolder in the *constant* directory. This subfolder contains all information about the mesh and the only file to be edited is the file *boundary*. In this file, the user has to set the physical type of the different surface patches (*wall*, *symmetryPlane*, *patch*).

There is one final directory to be explained. The *system* directory allows the user to change the general control properties, such as start and end time, timestep size or output interval by editing the *controlDict* file. In the file *fvSchemes*, one can specify different discretization schemes for time and space. Finally, there is the file *fvSolution*, where the user is able to influence which solver should be used to solve for the different quantities. Furthermore, one can specify the Under Relaxation Factors (URF) and if correctors for non-orthogonality should be used.

More detailed information, especially regarding the different possibilities for adjustment, can be found in the OpenFOAM User Guide [14].

2.2.1.2 Solvers

There is a great variety of solvers available for many different applications, ranging from chemical reactions over turbulent flows to solid dynamics and electromagnetics. However, here only the solvers that were used in conjunction with this work are presented.

potentialFoam
The *potentialFoam* solver is, as the name already suggests, a basic solver for potential flow. It was during this project mainly used to generate a converged initial field for RANS or LES simulations.

simpleFoam
SimpleFoam is a RANS solver for incompressible steady-state simulations of turbulent flow. The pressure-velocity coupling is solved using the SIMPLE algorithm, see OpenFOAM Uses Guide [14]. Since this solver is a RANS solver (see section 2.1.3), this solver needs the specification of a turbulence model. During this thesis work the Spalart-Allmaras model and the realizable $k-\varepsilon$ model were used.

pimpleFoam
The *pimpleFoam* solver is a transient solver, which is capable of large timesteps, since the pressure-velocity coupling is solved using the PIMPLE (merged PISO-SIMPLE) algorithm, see OpenFOAM Uses Guide [14].

2.2.1.3 Realizable $k-\varepsilon$ model

There are a lot of different turbulence models implemented in OpenFOAM, amongst others also the $k-\varepsilon$ models. In general there are different versions of the $k-\varepsilon$ model, such as the standard $k-\varepsilon$ model or the RNG $k-\varepsilon$ model. However, during this thesis only the so called realizable $k-\varepsilon$ model was applied.

The standard $k-\varepsilon$ model is widely used in CFD today, but it has some deficiencies. Problems appear especially for flows with a high mean shear rate or a massive separation, since the eddy viscosity is overpredicted in these cases. Furthermore, the turbulent length scale, given by the ε equation of the standard model, is sometimes not appropriate. In order to improve performance of the standard model, even for complex flows, the realizable $k-\varepsilon$ model is formulated. The implementation of the realizable $k-\varepsilon$ model in OpenFOAM is according to Shabbir et al [15]. According to Marzouk and Huckaby[16], Eq. (2.18) and Eq. (2.19) are the transport equations for turbulent kinetic energy and dissipation, respectively. Due to simplicity, only the most important quantities in these equations are elaborated on in more detail.

$$\frac{\partial \rho k}{\partial t} + \frac{\partial \rho k U_j}{\partial x_j} = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] + P - \rho\varepsilon \tag{2.18}$$

with $P = \tau_{ij}\frac{U_i}{x_j}$ as the production of turbulent kinetic energy due to the mean velocity gradients.

$$\frac{\partial \rho \varepsilon}{\partial t} + \frac{\partial \rho \varepsilon U_j}{\partial x_j} = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_\varepsilon}\right)\frac{\partial \varepsilon}{\partial x_j}\right] + \rho C_1 S\varepsilon - \rho C_2 \frac{\varepsilon^2}{k + \sqrt{\nu\varepsilon}} \tag{2.19}$$

with $C_1 = max\left[0.43, \frac{\eta}{\eta+5}\right]$ and $\eta = \frac{k}{\varepsilon}\sqrt{2S_{ij}S_{ij}}$.

μ_t is referred to as the turbulent viscosity and is defined in the following way:

$$\mu_t = \rho C_\mu \frac{k^2}{\varepsilon} \tag{2.20}$$

In contrast to the standard $k-\varepsilon$ model, where C_μ is defined as a constant, it is replaced by a function in the realizable $k-\varepsilon$ model.

$$C_\mu = \frac{1}{A_0 + A_S(U^* \frac{k}{\varepsilon})} \tag{2.21}$$

The coefficients of the realizable $k-\varepsilon$ model, which are used in OpenFOAM, are presented in Table 2.2.

Table 2.2: Coefficients of the realizable $k - \varepsilon$ model in OpenFOAM

A_0	C_2	σ_k	σ_ε
4.0	1.9	1.0	1.2

2.2.2 Other software

Harpoon
Harpoon is a commercial software package for the fully automatic meshing of complex geometries, which is currently used at VCC. The software produces meshes with structures that are hex-dominant. This means that the main type of cells in the mesh will be hexahedral. In other words, the cells have six faces. One famous example of a hexahedron is the cube, which has six squared faces. In a hexahedral mesh, however, the cells do not necessarily need to be cubes. They can be any highly skewed version of this shape, even though such cells are not desired in a mesh, because they reduce the accuracy and the performance or, in the worst case, they prevent the mesh from converging at all. With Harpoon, it is also possible to refine a mesh in any desired way and you can rid the surface of bumps and caves as well as delete highly skewed cells. During this project, Harpoon was used to create new meshes or to increase the quality of previous ones.

Fluent
Fluent is a commercial CFD code and the tool of choice at VCC. Similar to OpenFOAM, Fluent has a wide range of engineering applications, such as aerodynamics and aero-acoustics, as well as multiphase and reacting flow. During this thesis project, Fluent was used in order to produce reference data, concerning accuracy and simulation speed.

EnSight
EnSight is a visualization and post processing tool for applications in structural analysis, such as CFD and Finite Element Analysis (FEA). At VCC, it is used for analyzing and visualizing the results of simulations with Fluent or OpenFOAM. During this thesis work all visualization tasks were performed using EnSight.

MATLAB
MATLAB is a commercial software package and one of the world's leading numerical computing environments. Its operations are mainly based on calculations conducted with matrices, so it is rather evident how the name is derived: MATrix LABoratory. During this thesis, MATLAB was used in order to assess and visualize the simulation outputs.

Chapter 3

Methodology

This section aims to explain the setup of the different cases dealt with in this work.

3.1 Simulation setup

During this project, different cases were considered and different approaches were used in order to achieve the given goals. However, the simulation setup is the same for all the cases and hence this procedure is described here rather than in the respective subsections.

The first step required to begin a new simulation is to input the actual mesh into the file structure of OpenFOAM. There are several different ways to accomplish this.

- Exporting the mesh from the third party meshing tool directly into OpenFOAM. In the case of VCC this meshing tool is the software Harpoon. Exporting the mesh directly is advantageous, since one can already set boundary conditions for the different patches and the process is straight forward and therefore easy to understand.
- Exporting the mesh from the meshing tool into another file format (like Fluent case $*$.cas) and then transforming this file with the OpenFOAM utilities (in this case *fluent3DMeshToFoam*) into an OpenFOAM mesh. This technique is a little more elaborate than the first, but it also offers some advantages. It is, for example, easier to store the mesh, since only one file is obtained instead of the divided files of an OpenFOAM mesh. Furthermore, this file can be used directly in Fluent as well.
- Using the meshing tool that comes with OpenFOAM (*snappyHexMesh*) and getting

an OpenFOAM mesh directly. This technique is, of course, the most straight forward option, but since there is no experience with *snappyHexMesh* available at VCC, it is not considered in this thesis.

In this thesis, the focus is on the first two techniques, mainly because the respective Fluent cases were available already. A possibility to check the quality of the mesh is given by the utility *checkMesh*.

Once the mesh is imported into the OpenFOAM file structure and the mesh has passed the *checkMesh* test, one has to set the boundary conditions in the *0* folder. Depending on the method to be used (RANS, LES), one has to set the boundary conditions in several files accordingly.

Secondly, one has to specify the numerical method and, if necessary, the turbulence or SGS model. These specifications are carried out by manipulating the files in the *constant* folder. In the same folder, it is possible to set the fluid properties and decide between laminar and turbulent flow.

Lastly, the user has to treat the files stored in the *system* folder. Here it is possible to set the controls for the simulation, such as number of timesteps, timestep size and result output frequency (*controlDict*). In the same file one has the possibility to enable functions that are used during the simulation. These functions can, for example, determine the forces and moments on certain patches of the geometry or probe a field at defined locations. In addition, the time averaging of field variables and the calculation of their RMS values can be enabled here. The numerical schemes that should be used can be set in the file *fvSchemes*. The file *fvSolution* manages the solvers for the different transport equations (pressure, momentum) and the under relaxation.

With the settings from above, it is possible to run a simulation. However, engineering problems are usually quite complex and require a lot of computational power, so that parallel computing is state of the art. In order to prepare the case for parallel computing, manipulations of the file *decomposeParDict* in the *system* folder have to be completed. Here, one states the desired number of nodes and the method used for distributing the mesh between them. The actual mesh decomposition process is initiated by typing the command *decomposepar*. This process will produce an individual folder for each processor used.

After running a simulation, the flow results are distributed to all the different processors. In order to work with them, it is necessary to merge them together. This can be done using the command *reconstructPar*.

Post-processing at VCC is done with the software EnSight and exporting OpenFOAM results into EnSight is done using the command *foamToEnsight*.

3.2 VRAK

This section explains the VRAK case and its specific setup.

3.2.1 Geometry

The geometry of the VRAK case is a simple, vehicle-like geometry in a wind tunnel. Examining Fig. 3.1, one can see that the vehicle is simplified and lacks, for example, side mirrors, door handles or detailed wheels. Also, there is only a flat underbody in this model. Due to the fact that the geometry is symmetric about the $y = 0$ plane, only half of the vehicle is used in simulations, which will result in a decreased demand for computational power. The geometry is a so called closed front model. This means that the geometry is actually tight for the fluid and thus there is also no mesh on the inside of the vehicle. Open front models, in comparison, are much more complex and have a much higher demand for computational power. For example in simulations of the engine cooling an open front model is essential.

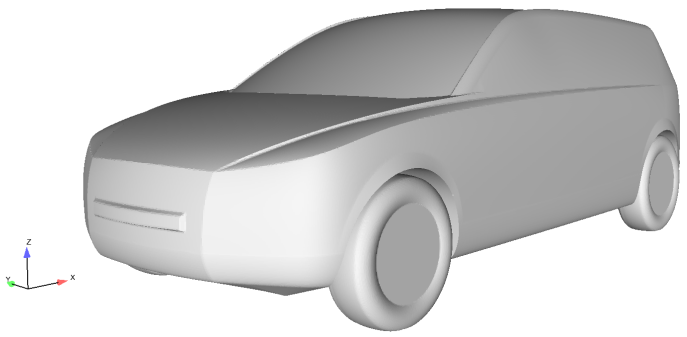


Figure 3.1: Generic production car model used for steady-state drag and lift investigations

Figures 3.2 and 3.3 show the mesh of the wind tunnel and the position of the vehicle. The coordinate system is defined as follows: the x-axis follows the streamwise direction, the y-axis points out of the domain and originates at the symmetry plane of the vehicle, and the z-axis is normal to the ground surface. In the mesh there are $12.2 \cdot 10^6$, mainly hexahedral, cells. The overall domain stretches out over 50 $[m]$ in streamwise direction, 10 $[m]$ in vertical direction and 4.75 $[m]$ in lateral direction for half a vehicle. Inside the domain, the front bumper of the vehicle is located 14.2 $[m]$ downstream of the inlet and the rear bumper 31.7 $[m]$ upstream of the outlet.

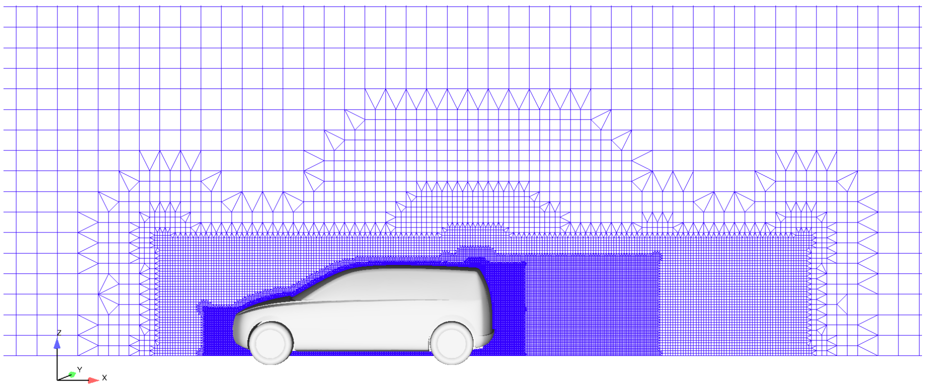


Figure 3.2: Computational grid for the VRAK case at the symmetry plane of the vehicle. One can see the different refinement regions.

By investigating the mesh, one can see that the mesh is refined in the regions of interest. Naturally, the refinements occur around the vehicle, but special refinements are made for the base of the vehicle. Having a refined rear end is important in order to study the wake flow behind the vehicle, which will be of vital importance for the drag prediction. In order to obtain correct lift values, the correct representation of the flow separation is important. Since separation is expected to occur at the top end of the hatch, this region is more refined (see Fig. 3.3).

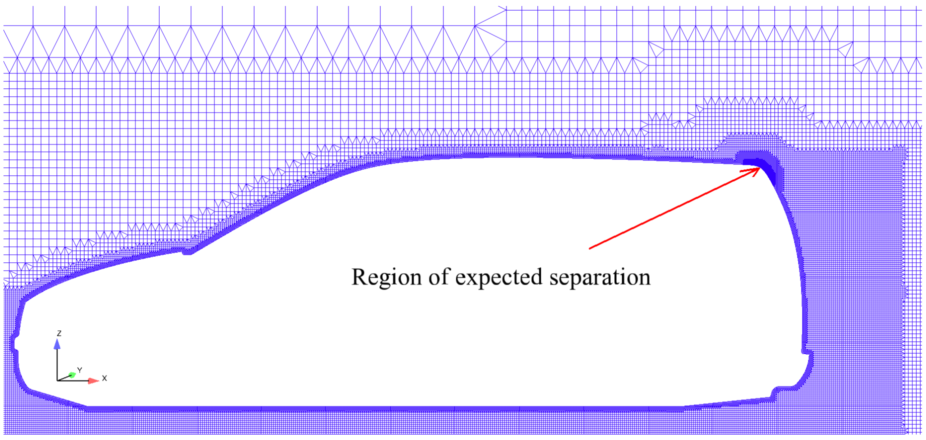


Figure 3.3: Detailed view of the computational grid for the VRAK case at the symmetry plane of the vehicle with refinement regions

The wheelbase of the model is $L = 2.65$ $[m]$, the free stream velocity is $U_\infty = 100$ $[km/h] = 27.8$ $[m/s]$ and the kinematic viscosity is $\nu = 1.5 \cdot 10^{-5}$ $[m^2/s]$. This yields, according to Eq. (2.3), a Reynolds number of $Re \approx 5 \cdot 10^6$. Calculating the Mach number as in Eq. (2.4), yields: $Ma \approx 0.081$. Thus, the incompressibility criterion is satisfied for this case.

3.2.2 Approach

The approach to running a simulation of the VRAK case is simple and straight forward. After setting up the case, as described in section 3.1, one has to initialize the flow field. This can be done with the potential flow solver of OpenFOAM, called *potentialFoam*. *PotentialFoam* will converge the pressure and the velocity field in one iteration, which yields a suitable set of initial conditions for the RANS solver. After running *potentialFoam*, the user has to adjust the settings in the *system* folder for the new solver.

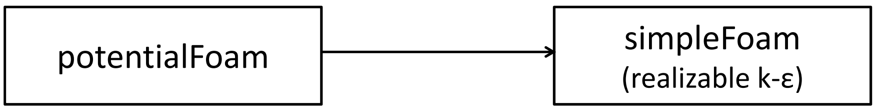


Figure 3.4: The approach to setup a new simulation for the VRAK case

Once all new settings are specified, a RANS simulation can be initialized, using the solver *simpleFoam*. This approach is the same as for a simulation of the VRAK case with Fluent.

3.2.3 Solver settings

The VRAK case is solved in steady state, which means that one assumes that there are no time dependencies affecting the solution. This is why the time derivative (first term) in Eq. (2.8) is not retained. A runtime of 3000 iterations is specified for each simulation.

Physical boundaries are modeled with the *wall* boundary condition; the farfield is treated with the *symmetry* condition. The Boundary Conditions (BCs) at the inlet and outlet are set as follows: at the inlet, a fixed velocity of 27.8 $[m/s]$ in streamwise direction (x-direction) is specified. The velocities in y- and z-direction are set to 0 $[m/s]$ and the initial values for the internal velocity field are set in the same way. The inlet pressure is set to *zeroGradient*, which assigns the normal gradient of pressure to be zero for the inlet. At the outlet, the *zeroGradient* BC is specified for the velocity and the pressure is given the fixed value of 0 $[m^2/s^2]$. In order to circumvent the development of a boundary layer at the floor, which would significantly influence the flow around and particularly under the vehicle, the floor of the virtual wind tunnel is assigned to move with the same speed as the air (27.8 $[m/s]$). Initial values for turbulent kinetic energy k and dissipation ε are set to 0.00116 $[m^2/s^2]$ and $3.4 \cdot 10^{-5}$ $[m^2/s^3]$ for the inlet, respectively.

For the spatial discretization, the following schemes are used. The *Gauss linear* scheme is used for both the discretization of the gradient of momentum as well as pressure. In order to guarantee comparability to Fluent settings, the convection scheme for the momentum discretization is specified to be of second order and thus the *Gauss linearUpwind* scheme was chosen. For the turbulent quantities a simple first order upwind scheme is perceived to deliver sufficient results. As a scheme for all the diffusion terms, *Gauss linear limited 0.5* is used.

Solving for the pressure is done using the *Generalized geometric-Algebraic Multi Grid (GAMG)* solver with a relative tolerance of 0.1 on an absolute tolerance of $1 \cdot 10^{-6}$. Momentum, turbulent kinetic energy and dissipation are solved with a *smoothSolver* and a relative tolerance of 0.1. As stated earlier, the coupling between velocity and pressure is solved using the SIMPLE algorithm. Since the mesh was found to be of good quality, it is not necessary to use any correctors for non-orthogonality.

Finally, the under relaxation factors, used are shown in Table 3.1 below.

Table 3.1: Under relaxation factors for the VRAK case

Property	*pressure*	*momentum*	*turb.kin.energy*	*dissipation*
URF	0.3	0.8	0.7	0.7

All simulations of the VRAK case are carried out with OpenFOAM version 1.6 in single precision.

3.3 XC90

This section explains the aero-acoustics case and its specific setup.

3.3.1 Geometry

As one can see in Fig. 3.5, the vehicle model used in this case is the Volvo XC90. Also are the definitions of some important vehicle parts shown in the figure, which will be mentioned throughout this thesis. In comparison to the real production vehicle, this model is reduced in its level of detail, especially with regard to the wheels and the underbody. These details are expected to have a low impact on the flow above the belt line, which is of main interest in this case, and thus can be neglected in order to save computational power.

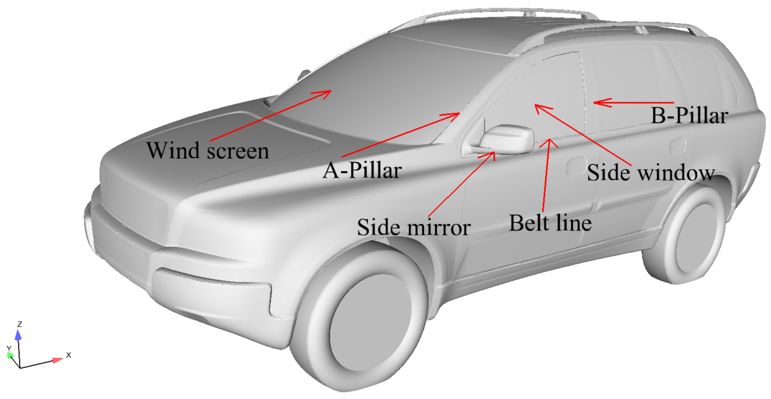


Figure 3.5: Geometry used for the transient aero-acoustics predictions including names of specific parts

There are different meshes used in the evaluation of this case. The first ones are meshes created with the meshing software ICEM, which is the reason why they will be referred to as ICEM1 and ICEM2 in the following. They are the same meshes as described by Ask and Davidson [17] (meshes GSM1I and GSM2I) and are summarized in Table 3.2. There are also two meshes used that were created with Harpoon (named Harpoon1 and Harpoon2 in the following). The mesh Harpoon1 can be studied in Fig. 3.6, whereas the mesh Harpoon2 is the same as studied by Ask and Davidson in [17], where the mesh is referred to as GSM3H. Different meshes are used in order to assure comparability to the Fluent reference case and to study the influence of the meshing method as well as the mesh resolution.

It can be observed that this case is not optimized to predict the drag or lift, since the wake region behind the vehicle is represented in a rather coarse way. Most refinement is done above the belt line from the front end of the vehicle until the B-pillar. This is due to the fact that the separated flow around the side mirror and the A-pillar is of most importance here.

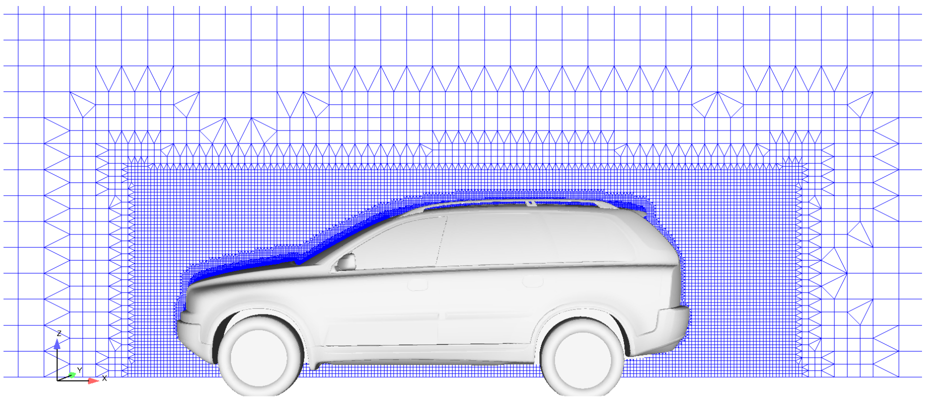


Figure 3.6: Computational grid for the aero-acoustics case at the symmetry plane of the vehicle

Figure 3.7 shows the mesh Harpoon1 for the side window region of the XC90. One can see how the mesh resolution is reduced quite abruptly from 2 $[mm]$ cell size to 8 $[mm]$ cell size in the middle of the side window. The effect of this will be explained later. The darkest regions in the figure (A-pillar, side mirror) contain very small cells (0.5 $[mm]$), so that it is not possible to distinguish between them here.

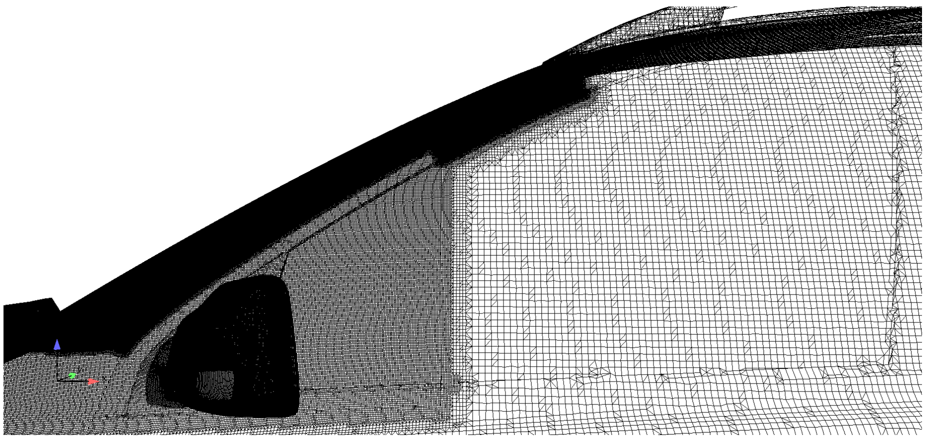


Figure 3.7: Mesh for the XC90 in the region of interest

As previously stated in section 3.2, the symmetry of the model is exploited and thus only half a vehicle is used with a closed front. The definition of the coordinate system is the same as in the VRAK case and the domain stretches out for 44.5 $[m]$ in streamwise direction, where the front bumper of the vehicle is located 15.5 $[m]$ downstream of the inlet and the rear bumper 24.2 $[m]$ upstream of the outlet. In the vertical direction, the farfield is 10 $[m]$ above ground and in the lateral direction the farfield is 4.75 $[m]$ from the symmetry plane at $y = 0$. The mesh Harpoon1 is dominated by hexahedral elements and contains a total of $17.7 \cdot 10^6$ cells.

In the following, Table 3.2 sums up the four meshes used, according to Ask and Davidson[17].

Table 3.2: Different meshes for the XC90

Mesh	# of prism layers	Cell size mirror wake	Cell size A-pillar	Number of cells
$ICEM1$	2	4 mm	8 mm	$6.6 \cdot 10^6$
$ICEM2$	2	4 mm	4 mm	$8 \cdot 10^6$
$Harpoon1$	0	> 8 mm	0.5 mm	$17.7 \cdot 10^6$
$Harpoon2$	0	2.5 mm	2.5 mm	$19.7 \cdot 10^6$

There are wind tunnel measurements of the dynamic pressure on the side window and the velocity magnitude in the side mirror wake available for the XC90, which are already given by Ask and Davidson in [17].

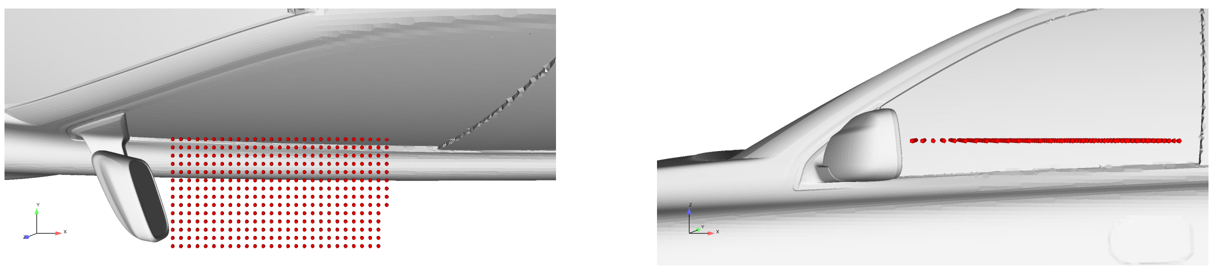

(a) top view (b) side view

Figure 3.8: Velocity probe locations

The measurements of the velocity magnitude downstream the side mirror were carried out in the acoustic wind tunnel at Ford Merkenich in Germany, with a 14-hole probe. Figure 3.8 shows the locations of the measurement points, where the spacing between the probes is 2 [cm] in stream- and spanwise direction. In total there are 14 rakes of measurement points, with the first rake being the closest one to the side window and the last being the one furthest away from it.

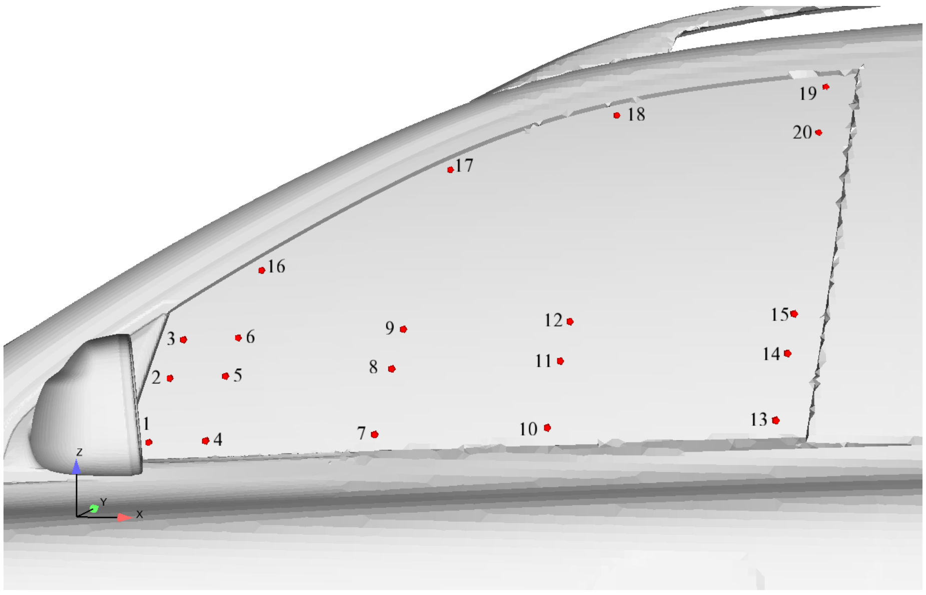


Figure 3.9: Locations for pressure probes on the side window

A total of 20 microphones on the inside of the side window were used to measure the dynamic pressure. The probes were connected by a 3 $[mm]$ hole in the window with the external flow and the positions are shown in Fig. 3.9.

For this model, the Reynolds number is $Re \approx 8 \cdot 10^6$, with a wheelbase of $L = 3$ $[m]$, a free stream velocity of $U_\infty = 140.4$ $[km/h] = 39$ $[m/s]$ and a kinematic viscosity of $\nu = 1.5 \cdot 10^{-5}$ $[m^2/s]$. The Mach number yields: $Ma \approx 0.114$. Therefore this case can be treated as incompressible flow.

3.3.2 Approach

The simulation setup for the XC90 is more complex than that for the VRAK case. Initially *potentialFoam* is used to converge the pressure and the velocity field in order to obtain a good starting point for the following simulation. After the initialization, a steady state RANS simulation with *simpleFoam* is carried out for 1000 iterations in order to converge the fields even more and to get a smooth starting point for the transient simulation. For the RANS simulation, the Spalart-Allmaras one-equation turbulence model is used. The transient calculations are then carried out using the solver *pimpleFoam*. Here it is important to have the additional boundary condition file *nuSgs* in the according starting folder. After 0.04 seconds of realtime that are carried out in order to get rid of the first, strong fluctuations in the solution, the output of statistics is enabled and the case is run for a time span of another 0.4 seconds realtime.

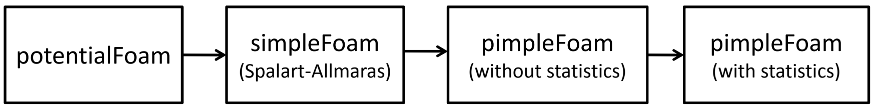

potentialFoam
simpleFoam (Spalart-Allmaras)
pimpleFoam (without statistics)
pimpleFoam (with statistics)

Figure 3.10: The approach to setup a new simulation for the XC90 aero-acoustics case

This procedure is the same, if Fluent is used to compute the case.

3.3.3 Solver settings

Since the source of noise, which is investigated in this project, emerges from pressure fluctuations, the instantaneous flow quantities are of interest. A steady state simulation only provides time averaged quantities and thus it is not possible to investigate these properties without solving the time derivative in Eq. (2.8). Due to this and because of the increased demand in computational accuracy, the simulations are carried out using LES. For the time discretization the Euler backward scheme is used, which is of second order accuracy. The timestep size is set to $\Delta t = 2 \cdot 10^{-5}$ $[s]$.

The boundary conditions are set as *wall* for all the actual walls, such as the vehicle surfaces and the wind tunnel floor. For the farfield, the *symmetry* condition is applied. At the inlet as well as initial value for the internal flow field, a fixed value of 39 $[m/s]$ is set for the velocity in streamwise direction. The velocity in the spanwise and vertical directions is set to 0 $[m/s]$. Furthermore, the pressure is given the *zeroGradient* BC at the inlet. At the outlet a fixed pressure of 0 $[m^2/s^2]$ and a *zeroGradient* velocity BC are assigned. The wind tunnel floor is modeled as a moving wall with a streamwise velocity of 39 $[m/s]$.

For the spatial discretization of gradient terms the *Gauss linear* scheme is used. Divergence terms are solved by the *Gauss upwind* scheme, except for the momentum, which is solved with the second order upwind scheme (*Gauss linearUpwind*). All the diffusion terms are solved with the help of the *linear limited 0.5* scheme.

The pressure-velocity coupling is solved with the help of a merged PISO-SIMPLE algorithm, called PIMPLE. The PIMPLE algorithm allows larger Courant numbers (or a larger timestep size on the same mesh) as compared with a normal PISO algorithm. Using a PISO algorithm, pressure and momentum have to converge during one subiteration for each timestep. A difference between the PISO and the PIMPLE algorithm is that one can specify more than one subiteration, which means that there is a larger margin for pressure and momentum to converge. In this case, four subiterations are used, where each of the subiterations has another three inner iterations in order to converge the pressure. During the simulation the Courant number reaches a maximum around 17 and a mean value of 0.0033. Solving for the pressure is done with the *GAMG* solver with a relative tolerance of 0.01 on an absolute tolerance of $1 \cdot 10^{-6}$. For the momentum, the *Preconditioned Bi-Conjugate Gradient (PBiCG)* solver is used.

Another difference between the PIMPLE algorithm and the PISO algorithm is the ability to apply under relaxation factors. The ones used in this case are shown in Table 3.3.

Table 3.3: Under relaxation factors for the aero-acoustics case

Property	*pressure*	*momentum*
URF	0.5	0.7

The aero-acoustics calculations were performed in double precision with OpenFOAM version 1.6.x.

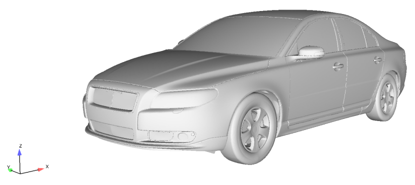
3.4 S80

This section explains the aerodynamic case concerning the S80 and its specific setup.

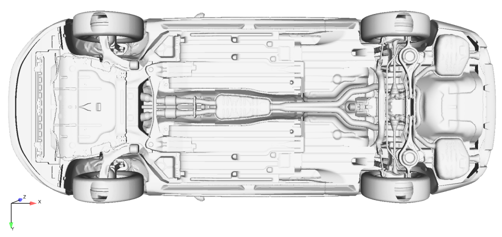
3.4.1 Geometry

The geometry for this case consists of a detailed model of the Volvo S80, as shown in Fig. 3.11(a). Included in this model are many features, such as door handles and a spoiler lip. Furthermore, the underbody of the vehicle is very detailed (see Fig. 3.11(b)) and side view mirrors are included. Since the function of the rotating boundary condition in OpenFOAM is supposed to be validated in this part of the thesis, it also contains very detailed wheels with actual rims and brake discs. External aerodynamics are investigated with this geometry, which is why the model is once again a closed front model. Also, the vehicle is symmetric and thus only half of it is simulated.

Not only the rotating boundary condition should be tested, but also the treatment of baffles (explained in section 2.1.5) in OpenFOAM should be addressed. Therefore, there are two intentional baffles in this geometry. First of all, there is a small spoiler lip, which is placed in the front of the vehicle's underbody and follows the vehicles front end shape in a semicircle. The second intentional baffle is a small deflector panel on the underbody. It is placed downstream of the spoiler lip in the center of the vehicle in the spanwise direction. The spoiler lip can be seen in Fig. 3.12, whereas the second baffle is hard to identify in



(a) Geometry of the S80



(b) Detailed underbody of the model

Figure 3.11: Geometry of the S80 containing all the details

the same figure. Due to the complexity of the model, numerous unintended baffles were created during the meshing process.

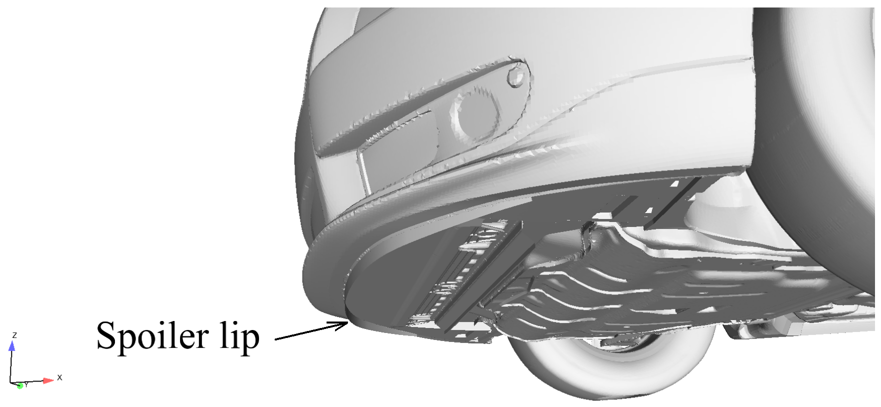


Figure 3.12: Spoiler lip at the front end of the vehicle

Figure 3.13 shows the mesh at the symmetry plane of the vehicle. One can see the similarities to the mesh in the VRAK case (Fig. 3.2). This is due to the fact that, in both cases, aerodynamics must be investigated and it is of great importance to represent the wake flow properly.

The coordinate system is defined in the same way as for the other two cases. In the streamwise direction, the virtual wind tunnel has a length of 50 $[m]$, with the front bumper of the vehicle 15.2 $[m]$ downstream of the inlet and the rear bumper 30 $[m]$ upstream of the outlet. In the vertical and lateral directions, the farfield is located 10 $[m]$ above ground and 4.75 $[m]$ from the symmetry plane, respectively. There is a total of $35.5 \cdot 10^6$, mainly hexahedral cells in the mesh. It is not the same mesh as used for simulations with Fluent. The biggest cells in the domain have a side length of 40 $[mm]$ and they gradually become smaller, down to a size of 2.5 $[mm]$, in the region of interest.

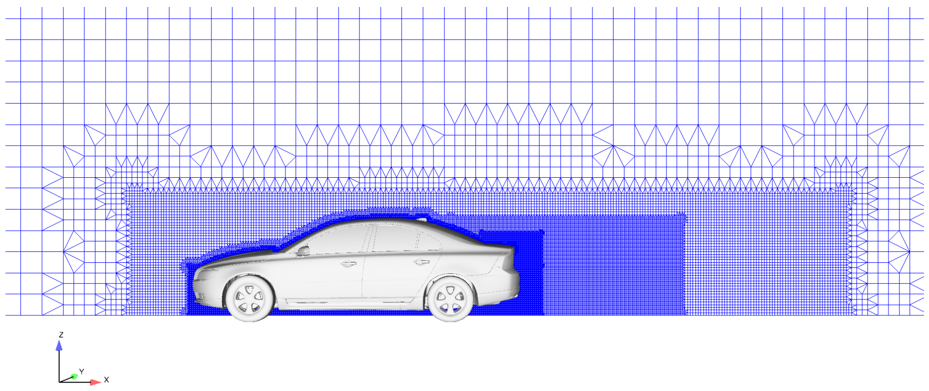


Figure 3.13: Mesh for the S80 at the vehicles symmetry plane

In this case the Reynolds number is $Re \approx 3.5 \cdot 10^6$, with a wheelbase of $L = 2.8$ $[m]$, a free stream velocity of $U_\infty = 100$ $[km/h] = 27.8$ $[m/s]$ and a kinematic viscosity of $\nu = 1.5 \cdot 10^{-5}$ $[m^2/s]$. The Mach number yields: $Ma \approx 0.081$. Thus it is an allowable assumption that this case can be treated as incompressible flow.

3.4.2 Approach

As compared to the VRAK case the approach to simulating the S80 is much more complex. In order to speed up the simulation, first the *applyBoundaryLayer* function is used and a boundary layer of 1 $[cm]$ thickness is created at all walls. Then the flow field gets initialized with a potential flow simulation. Using this as a starting point, a *simpleFoam* simulation is initialized, but only for 200 iterations. During this part of the simulation, the convection discretization for momentum is set to the first order upwind scheme. This results in a stable beginning of the simulation, since usually in the beginning the fluctuations in the properties are very high and one might run into problems with divergence. After 200 iterations, the actual simulation with a second order *linearUpwind* scheme is initiated. Figure 3.14 shows the approach in a schematic view.

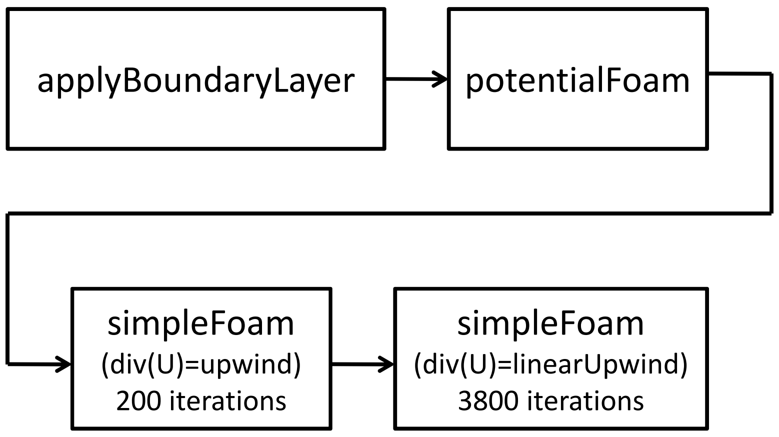


Figure 3.14: The approach to setup a new simulation for the S80 case

3.4.3 Solver settings

As with the VRAK case, aerodynamics is the focus here and thus a steady state simulation is sufficient. During the simulation, a total of 4000 iterations are performed, where the first 200 are carried out using a first order convection scheme for the momentum. According to Fig. 3.14, 3800 iterations are then performed using the second order *linearUpwind* scheme. The under relaxation is changed simultaneously with the scheme in order to avoid stability problems. Table 3.4 shows the under relaxation factors used in the two steps of the simulation.

Table 3.4: Under relaxation factors for the S80

Property	*pressure*	*momentum*	*turb.kin.energy*	*dissipation*
$URF - 1storder$	0.3	0.8	0.3	0.3
$URF - 2ndorder$	0.2	0.7	0.3	0.3

The physical boundaries of the domain are treated as *wall* at the solid boundaries and as *symmetry* at the farfield. Inlet and outlet are modeled as velocity inlet and pressure outlet, respectively. At the inlet, a constant velocity of 27.8 $[m/s]$ in streamwise direction is set. The velocities in lateral and vertical directions are set to 0 $[m/s]$. For the pressure at the inlet, a *zeroGradient* boundary condition is assigned. At the outlet, the velocity gradient in all directions is set to 0 $[m/s]$. The pressure at the outlet is given a fixed value of 0 $[m^2/s^2]$. Initial values for turbulent kinetic energy k and dissipation ε are set to 0.00116 $[m^2/s^2]$ and $3.4 \cdot 10^{-5}$ $[m^2/s^3]$ for the inlet. The wind tunnel floor is assigned a moving wall boundary condition with a velocity of 27.8 $[m/s]$ in streamwise direction. Since the impact of rotating wheels will be studied in this case, the wheels are given a rotating boundary condition. They spin with an angular velocity of -85.5 $[s^{-1}]$. It is important to have a negative value here, since otherwise the direction of motion of the wheels would be incorrect.

As previously mentioned, the convection discretization of momentum is done with the first order *upwind* scheme at the start of the simulation. It is changed to the second order *linearUpwind* after 200 iterations. Gradient terms are discretized using the *Gauss linear* scheme. *Gauss linear limited 0.5* is used as a scheme for all the diffusion terms.

The SIMPLE algorithm is used in order to solve the velocity pressure coupling. Solving for pressure is done with the help of the *GAMG* solver with a relative tolerance of 0.05 on an absolute tolerance of $1 \cdot 10^{-6}$. For the momentum, as well as turbulent kinetic energy and dissipation, the *PBiCG* solver is used. The mesh was found to have a high non-orthogonality and thus six correctors are applied.

Furthermore, all simulations of the S80 are run with OpenFOAM version 1.6.x and in double precision.

Chapter 4

Results and discussion

In this section, the outcome of this project work for all the different cases is presented.

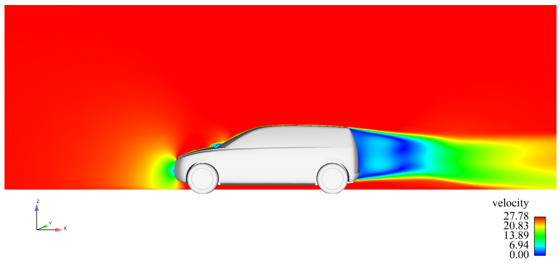
4.1 VRAK

4.1.1 Flow results

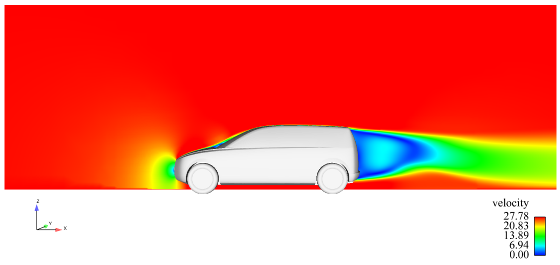
In order to carry out a valid evaluation between the performance of Fluent and OpenFOAM, one needs to compare the general flow results. The velocity field is shown in Figs 4.1 and 4.2, once on the symmetry plane of the vehicle and once on a cut plane normal to the z-axis. The two figures on the right hand side are obtained by a Fluent simulation, whereas the figures on the left hand side are obtained by OpenFOAM. Figure 4.1 shows that both solvers predict a similar wake behavior. However, Fluent predicts a stronger contraction of the wake. With the OpenFOAM solution, one finds a region of low velocity near the plenum, which is not realistic and cannot be found in the Fluent solution. This is deemed as an artifact of the symmetry boundary condition, since the effect diminishes quite rapidly for cut planes further into the domain.

Investigating the wake flow from above (Fig. 4.2) shows again a similar appearance, even though OpenFOAM provides a narrower wake than Fluent. Furthermore, the velocity field obtained by OpenFOAM contains more structures, which indicates a transient simulation rather than a steady state one.

Figures 4.3 and 4.4 show the VRAK model colored in terms of the pressure coefficient.

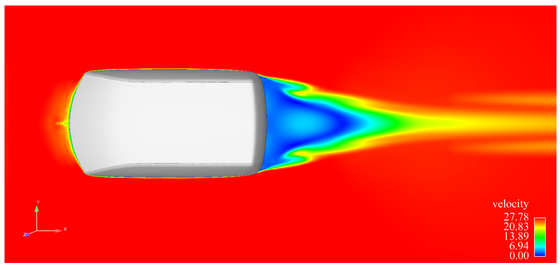


(a) OpenFOAM

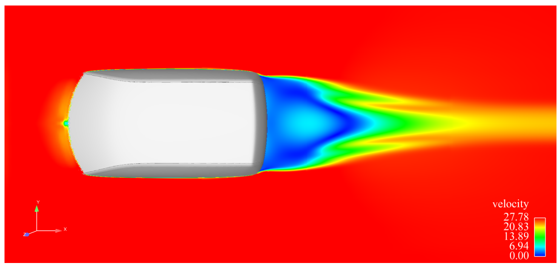


(b) Fluent

Figure 4.1: Velocity field on symmetry plane



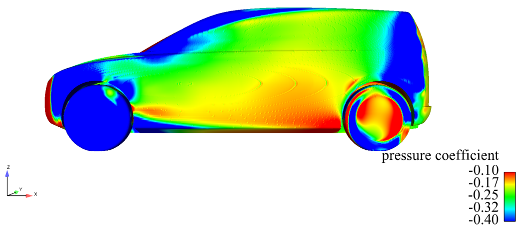
(a) OpenFOAM



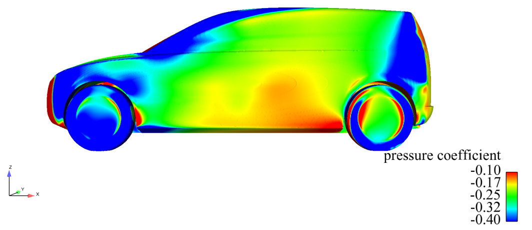
(b) Fluent

Figure 4.2: Z-Cut of the velocity field

The side view reveals no severe differences between OpenFOAM and Fluent. Only under keen observation can differences be found. The front wheel shows a stronger separation (low relative pressure) in the OpenFOAM solution. The red spot on the wheel house downstream of the front wheel in Fig. 4.3(b) is caused by reattachment of the flow on the wheel. This phenomenon cannot be seen in wind tunnel tests and OpenFOAM does predict a more separated flow. It is believed that OpenFOAM gives the more reasonable flow behavior for this case.



(a) OpenFOAM



(b) Fluent

Figure 4.3: Pressure coefficient C_p in side view

The pressure coefficient at the base of the vehicle is compared in Fig. 4.4. It is obvious that the pressure recovery predicted by OpenFOAM is mainly focused on one spot in the center of the vehicle's rear. In contrast, Fluent gains back the pressure over a wider area. Furthermore, the OpenFOAM solution shows more small structures than the one obtained by Fluent. Another obvious difference is again related to the wheels. Fluent predicts that the flow is attached on the backside of the wheel and OpenFOAM shows a separated flow behind the wheel instead. Again, it is believed that OpenFOAM provides a more reasonable result in this instance.

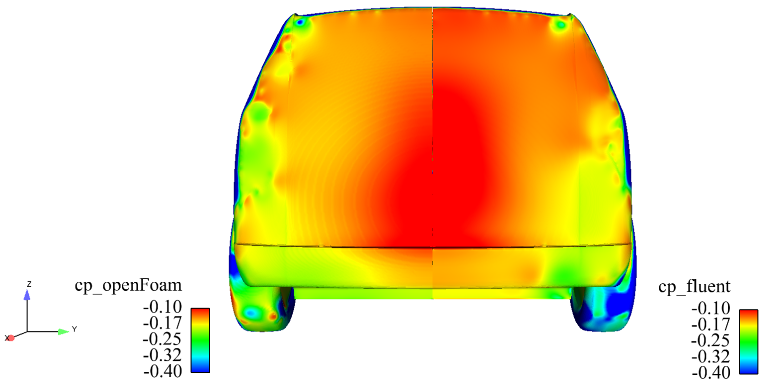


Figure 4.4: Comparison of the pressure coefficient C_p at the base of the model, obtained by OpenFOAM (left half) and Fluent (right half)

4.1.2 Drag and lift evaluation

For aerodynamic simulations, it is important to obtain accurate results for the drag and lift values. In this section the drag and lift obtained by OpenFOAM are presented and evaluated against the Fluent results. Figure 4.5 shows the product of drag coefficient and projection area (C_dA) in relation to the number of iterations. The black line represents the experimental value. The turquoise and red lines stand for the Fluent and the OpenFOAM results, respectively. Obviously, the OpenFOAM solution increases very quickly in the beginning to values high above those of Fluent. At around 450 iterations, a high convergence rate can be observed (the curve falls rapidly). However, after that, the drag coefficient increases slightly and finally decreases to similar values as Fluent at around 800 iterations. For the rest of the simulation, the drag coefficient fluctuates around a value approximately 3% lower than the experimental value. The mean value of the drag coefficient taken over the last 1000 iterations is $C_dA = 0.6343$ with a standard deviation of $SD(C_dA) = 0.0085$, whereas the mean value of Fluent is $C_dA = 0.6156$. Both Fluent and OpenFOAM converge approximately at the same iteration ($\approx$ 1000 iterations), with OpenFOAM being closer to the experimental value.

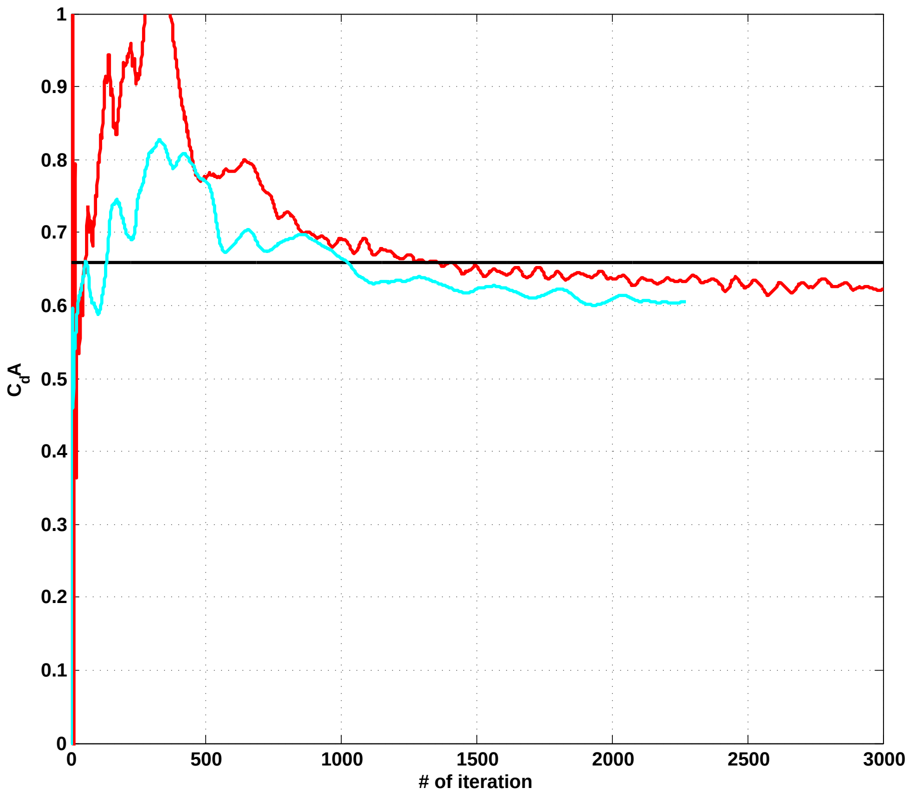


Figure 4.5: Drag values of the VRAK case; black line: experimental value; red line: OpenFOAM values; turquoise line: Fluent values

Figure 4.6 helps to illustrate the behavior of the lift coefficient C_l over the number of iterations. As mentioned above, the black line represents the experimental mean value. Unfortunately, only a single value for the lift coefficient is available for the Fluent simulation. The value is a snapshot of the last iteration of the simulation. The red, blue and green curves are obtained by OpenFOAM and are the total lift, rear lift and front lift, respectively. In order to make the distinction between front and rear lift, the moment around the front axle (y-direction) is used. One can see that the total lift value is oscillating quite much and is underpredicted by OpenFOAM. In comparison, the snapshot of the Fluent simulation overpredicts the experimental value. Furthermore, it is shown by the figure that there is positive lift at the rear axle and negative lift at the front axle. Negative lift can be interpreted as a force pressing the vehicle towards the ground, instead of lifting it. The lift value starts fluctuating around a stable value after approximately 750 iterations. The mean value of the total lift coefficient over the last 1500 iterations is $C_l = 0.326$, with a standard deviation of $SD(C_l) = 0.009$.

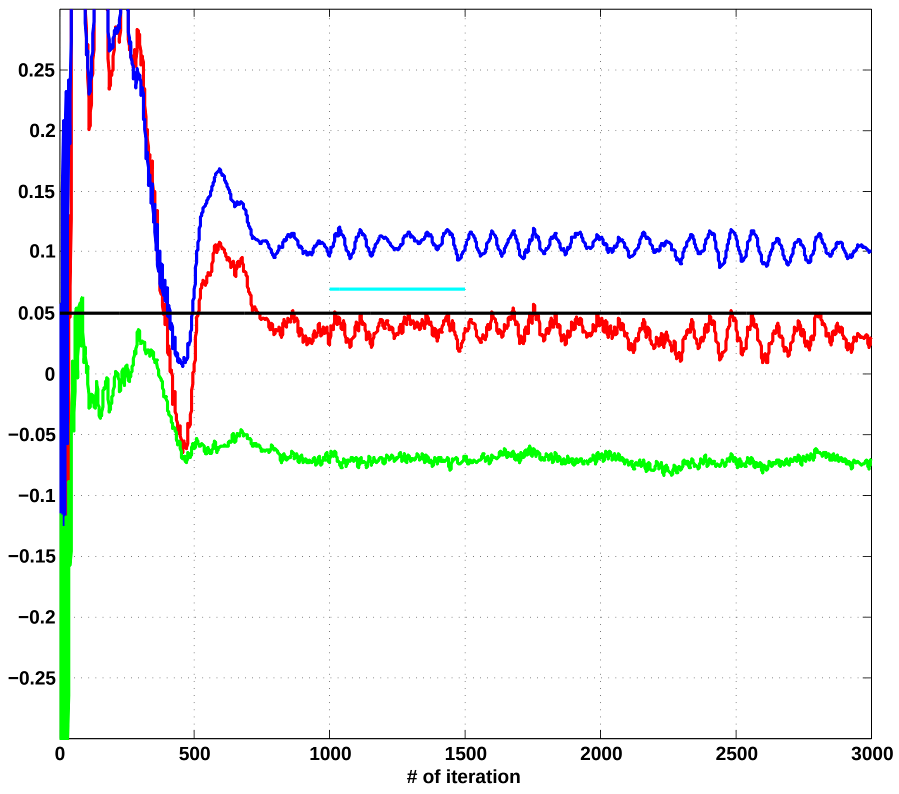


Figure 4.6: Drag values of the VRAK case; black line: experimental value; red/blue/green lines: OpenFOAM values; turquoise line: Fluent value - red line: total lift, blue line: rear lift, green line: front lift

In the following, Table 4.1 sums up all obtained and used values.

Table 4.1: Summary of drag and lift results for the VRAK case

Property	*OpenFOAM*	*Fluent*	*Experiment*
C_d	0.302	0.295	0.315
C_dA	0.6298	0.6156	0.658
$SD(C_dA)$	0.0085	0.0101	—
$C_{l,total}$	0.0326	0.069	0.05
$SD(C_{l,total})$	0.009	—	—
$C_{l,front}$	-0.0719	—	—
$C_{l,rear}$	0.1045	—	—
Δ	0.1764	—	—

4.1.3 Influencing factors on convergence and simulation time

It was one main goal of the VRAK study to reduce the simulation time in OpenFOAM so that it would be similar to the simulation in Fluent. This can be achieved either by reducing the number of iterations required to produce a converged result or by reducing the time necessary for each iteration. The factors that were found to be most influential will be discussed now.

In general, it is advantageous to have a short calculation time per iteration. There are many possibilities in OpenFOAM to influence the iteration time. A great improvement can be found using no correction for non-orthogonality. This is still a natural result, since the non-orthogonality corrector is performing an additional sweep for the pressure. Thus, solving for the pressure takes twice as long as it would without non-orthogonality correction. Also, the specification of the relative tolerance for the different quantities has a great influence on iteration time. This is due to the fact that the solver does not need to converge anymore until the appointed absolute tolerance is reached and therefore less time is needed. Performing simulations in single precision instead of double precision yields another improvement of about 40% in iteration time. Further improvement is achieved by using the *applyBoundaryLayer* utility prior to a simulation. However, this utility was not used, since it was found to produce non-satisfactory results. Figure 4.7 shows approximately how much the different factors help to reduce the total iteration time.

Initial settings lead to an iteration time of around 10 seconds. Assuming this as a starting point, a reduction of about 70% in iteration time can be achieved. In other words, the current iteration time for the VRAK case is about 3 seconds on 40 CPUs.

It is also advantageous to achieve convergence as fast as possible, since the number of iterations required for a valid result is fewer. The greatest influence on convergence behavior is related to the under relaxation factors. These factors are specified for each quantity

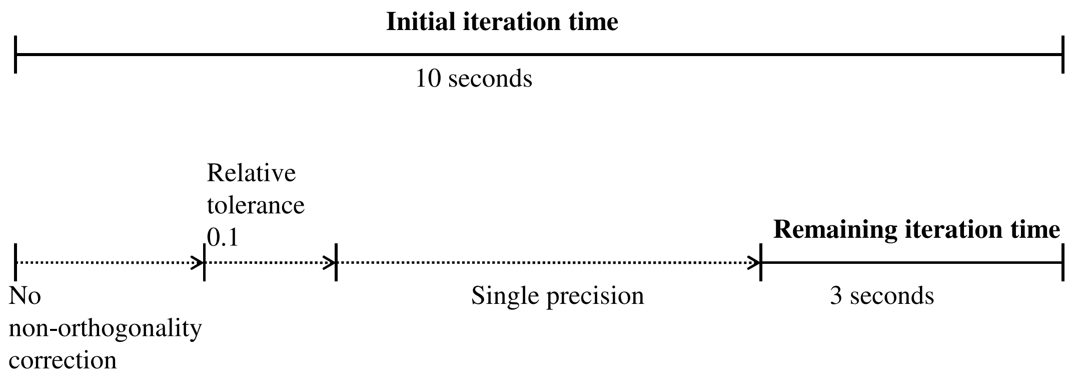


Figure 4.7: Influence on the iteration time

one is solving for (in this case: pressure, momentum, turbulence quantities). By using the factors according to Table 3.1, the simulation had already reached convergence after about 1000 iterations as stated in section 4.1.2, instead of after 2000 - 3000 iterations in previous settings.

4.1.4 Discretization schemes

Even though decent convergence speed drag values could be obtained, an analysis of the lift value, the base pressure distribution and the general flow behavior revealed great problems in the results. As shown in Fig. 4.8, separations at the rear of the car were not predicted in a satisfactory manner. That is to say that the separation from the rear of the vehicle occurred too late or, with other words, the flow stayed attached to the surface for too long. This yields incorrect predictions of the wake flow behind the vehicle and since the C_l value depends strongly on the correct prediction of separations, no satisfactory result can be expected.

A first approach to eliminating these problems involved looking deeper into the spatial discretization schemes – especially the divergence schemes. It was assumed that the discretization of the turbulence variables would not have a strong impact on the result and thus the main focus was on the momentum discretization. The scheme used in the momentum-equations when obtaining the unrealistic results was the *linearUpwind* scheme with faceLimited *Gauss linear* as a gradient scheme with a blending factor of 0.5. The *linearUpwind* scheme is of second order accuracy, but it blends to a first order upwind scheme with the factor specified. Factor 0 represents a pure first order scheme, whereas factor 1 stands for a pure second order scheme.

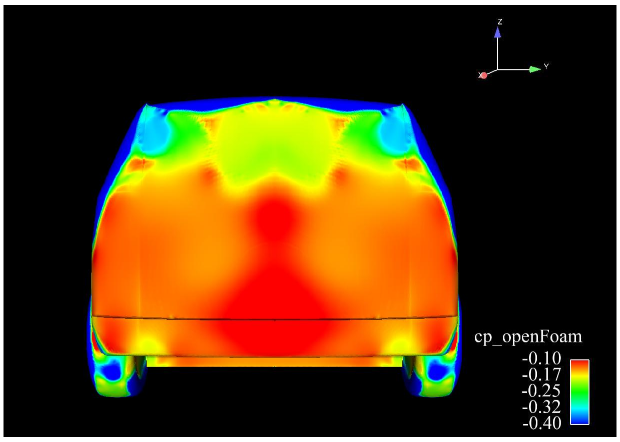


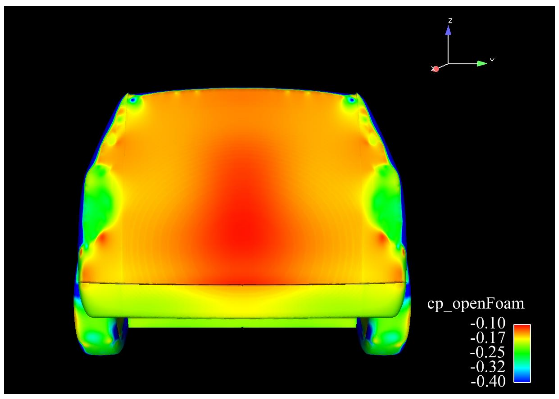
Figure 4.8: Non satisfactory base pressure distribution

Since a second order upwind scheme was also used in Fluent for momentum convection, the *linearUpwind* scheme was kept, but instead of the faceLimited version, a cellLimited version was used. The change showed an immediate impact on the results. In order to find the optimal blending factor, a sweep of different factors between 0 and 1 was performed (see Table 4.2). The results for this sweep are shown in Fig. 4.9 below.

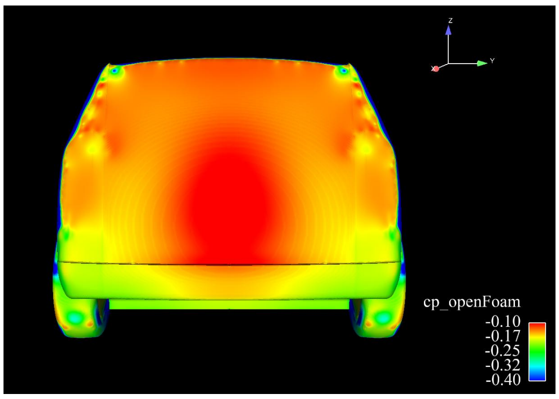
Table 4.2: Sweep of blending factors for the VRAK case

Run #	*Blending Factor*
1	0.25
2	0.5
3	0.6
4	0.75
5	1

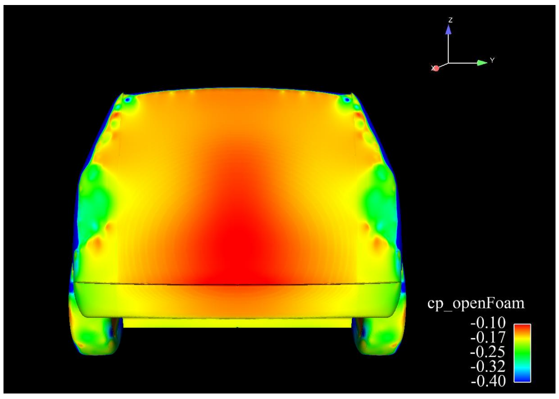
Based on the different solutions, the decision was made to continue with a blending factor of 0.75 (Fig. 4.9(d)). It shows the best agreement with the Fluent solution in Fig. 4.9(f).



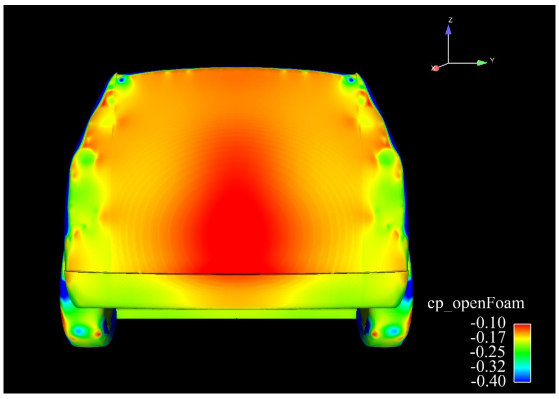
(a) OpenFOAM cellLimited 0.25



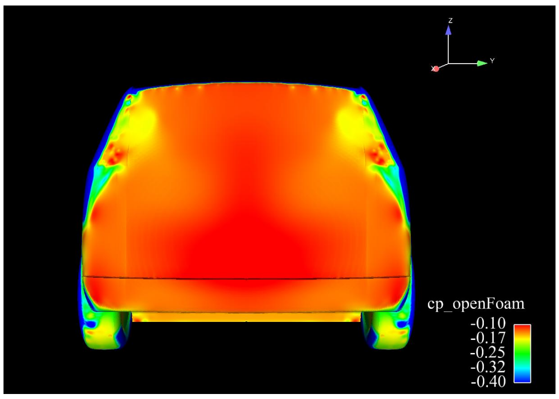
(b) OpenFOAM cellLimited 0.5



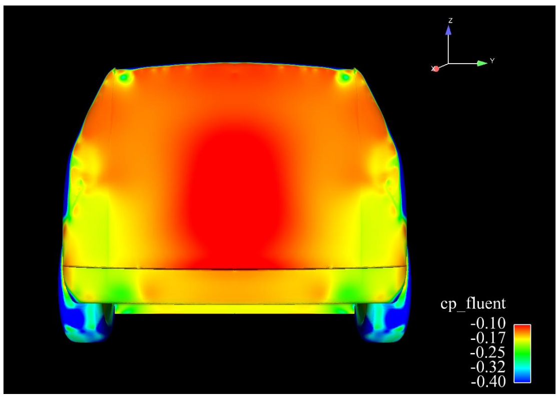
(c) OpenFOAM cellLimited 0.6



(d) OpenFOAM cellLimited 0.75



(e) OpenFOAM cellLimited 1



(f) Fluent

Figure 4.9: Comparison of the pressure coefficient C_p on the base of the VRAK model for the blending factor sweep

4.2 XC90

4.2.1 Flow results

From the front end of the hood air becomes accelerated and is directed to the windscreen by the little bend on the hood. Therefore, the flow is split up into one vortical structure inside the plenum and a boundary layer flow towards the roof and the A-pillar. The vortical flow in the plenum leaves the plenum at the lower end of the A-pillar and is directed beneath the side mirror foot. The boundary layer flow of the windscreen will separate at the A-pillar and turn into a spiral vortex that trails for a long distance downstream.

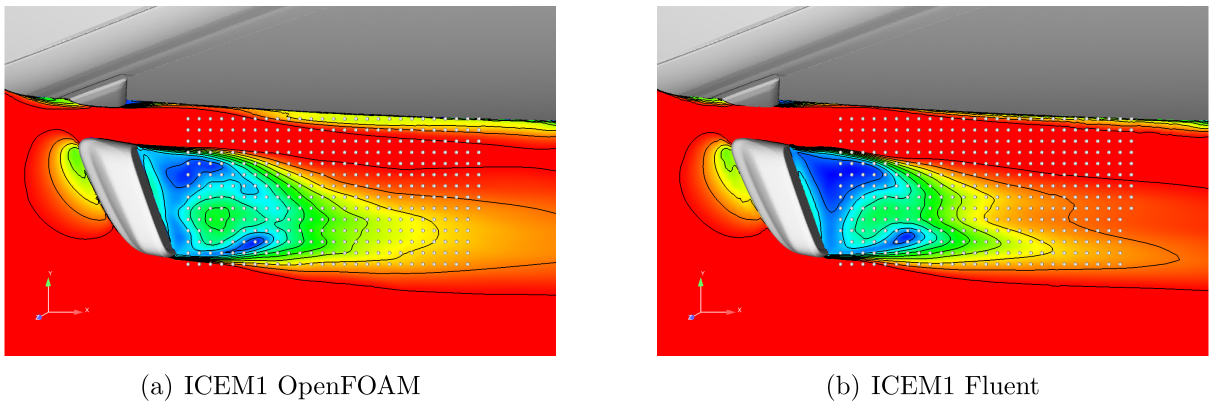
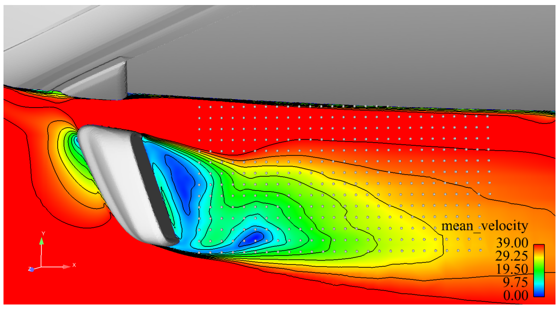

(a) ICEM1 OpenFOAM (b) ICEM1 Fluent

Figure 4.10: Mean velocity for the mesh ICEM1 in the side mirror region including measurement points

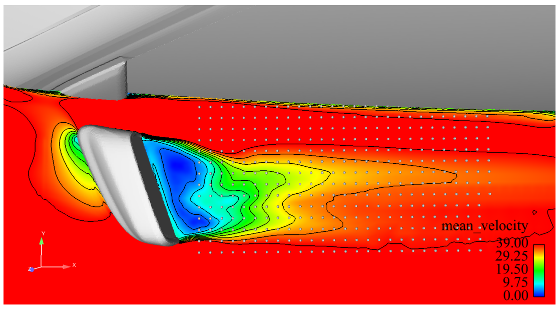
Figure 4.10 compares the flow in the side mirror wake of mesh ICEM1 for OpenFOAM and Fluent. One can see that both codes yield similar results for the same mesh. The white dots in the figure mark the locations of the 14-hole probe.

Investigating Fig. 4.11, it becomes obvious that OpenFOAM and Fluent predict quite different wake flows for the mesh Harpoon2. Fluent provides a narrower wake as compared to OpenFOAM. The reason for this can be found in the different prediction of the separation zone. OpenFOAM predicts that the flow will separate from the side mirror early upstream, whereas for Fluent the flow stays attached until it reaches the trailing edge of the mirror before it separates. However, in comparison with the Flow results of ICEM1, OpenFOAM captures the trend in a better way.

Along rake one, Fig. 4.12(a), the mean velocity magnitude shows a decreasing tendency, which is due to the decreasing space between the window and the probes. This trend is shown by the OpenFOAM simulations as well. Harpoon1 starts to break the trend after



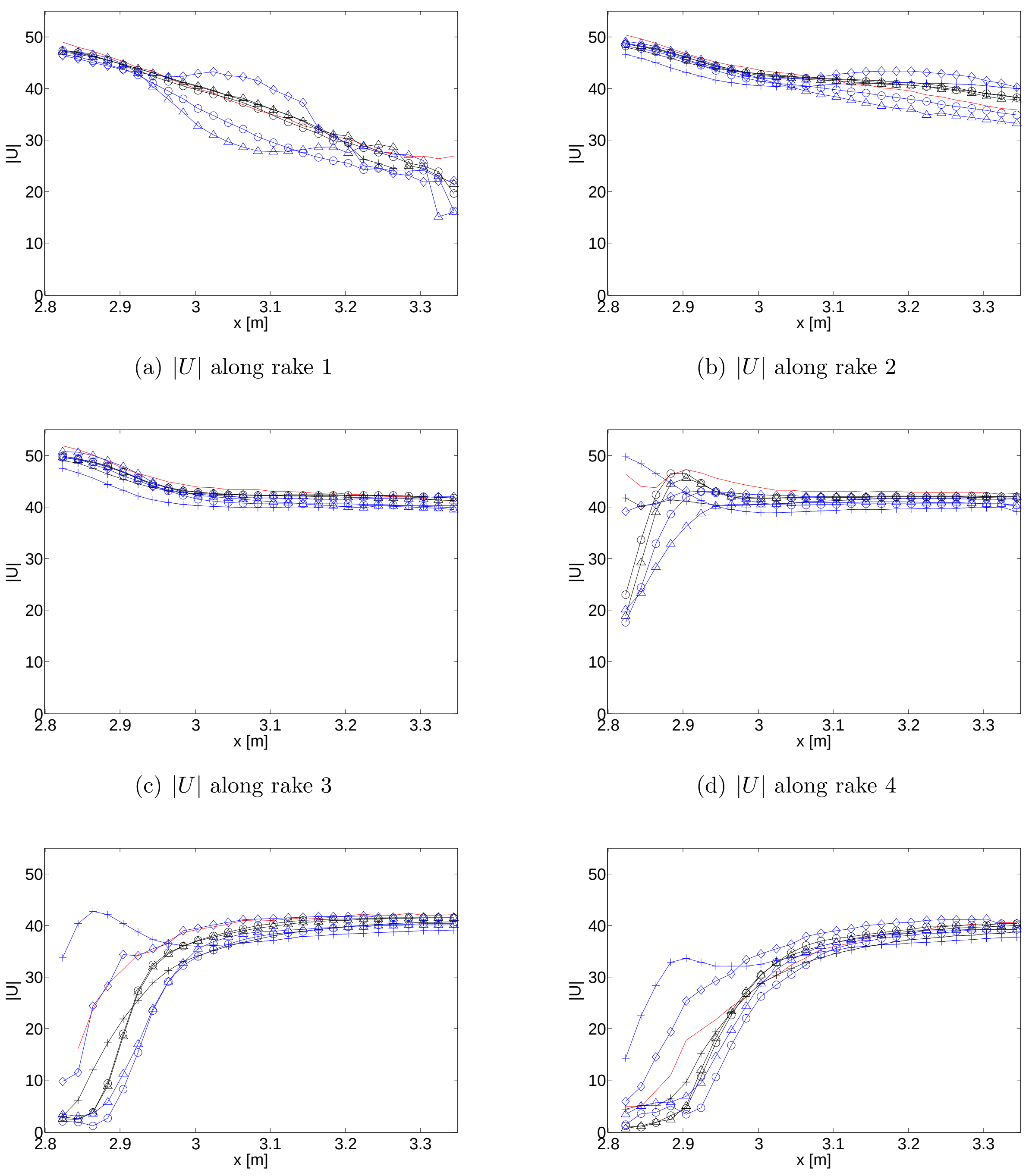
(a) Harpoon2 OpenFOAM



(b) Harpoon2 Fluent

Figure 4.11: Mean velocity for the mesh Harpoon2 in the side mirror region including measurement points

the eighth measurement point, which can be explained with a rapidly decreasing mesh resolution from this region onward. ICEM1 and ICEM2 were run without the moving ground BC and the dip in the curves is explained by the influence of the boundary layer found on the tunnel floor (compare with Fig. 4.10(a)). All simulations show a decreasing mean velocity magnitude for the entire rake, whereas the measurements break this trend. An increased velocity is measured for the locations furthest downstream. This effect might be caused by the 14-hole probe itself. Since it is located in close proximity to the wall, the flow becomes accelerated between the probe and the wall. For rakes two and three (and in the case of Harpoon2 also for rake four) a jet-like flow is created between the mirror head and the base cover. This can be seen in the first few points of Fig. 4.12(b), (c) and (d). The OpenFOAM simulations for these rakes are in good agreement with the measurements and the Fluent simulations. Rake four shows that the results are very sensitive to the location of the probes. A slightly different placement of the recirculation zone can drastically change the result, since very sharp velocity gradients can be found. In rake five, OpenFOAM strongly overpredicts the velocity magnitude for the first seven points of Harpoon2. This occurs, because the jet-like region is spread wider for this simulation. The mesh with the lowest resolution in the region downstream the side mirror, Harpoon1, catches the measured velocity magnitude field almost perfectly for rake five. The same overprediction as in rake five can be seen for Harpoon2 in the OpenFOAM solution of rake six. Harpoon1, however, catches the tendency of the measurements the best, even though the values are slightly overpredicted. Judging from Fig. 4.13(a) and (b), the trend from rake six continues for the two Harpoon meshes in OpenFOAM. For rake eight, OpenFOAM predicts perfectly matching flow fields from the 10th probe downstream using the two ICEM meshes. Along rake nine, ICEM1 agrees very well with the measured values, as shown in Fig. 4.13(c). For all the rakes from six to 10, Harpoon1 and Harpoon2 were able to catch the correct tendency, but with overpredicted levels. Rake 11 and 12 show an almost perfect match between the OpenFOAM solution on Harpoon1 and the measurements. In general, it is fair to say that OpenFOAM and Fluent behave similarly on the ICEM meshes.



(a) $|U|$ along rake 1

(b) $|U|$ along rake 2

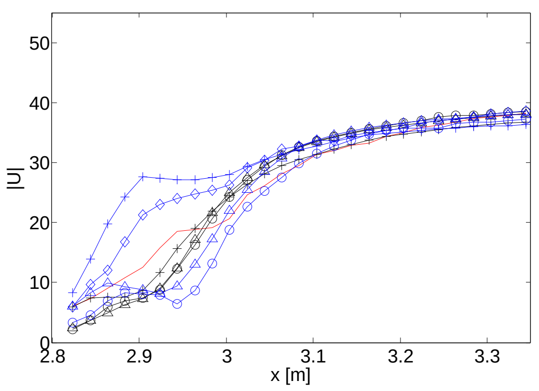
(c) $|U|$ along rake 3

(d) $|U|$ along rake 4

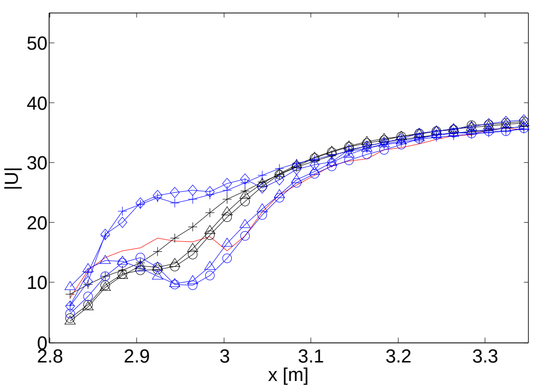
(e) $|U|$ along rake 5

(f) $|U|$ along rake 6

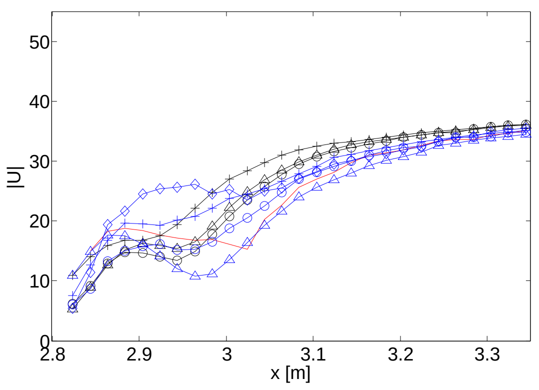
Figure 4.12: Mean velocity magnitude along rake 1 to 6 for OpenFOAM (blue line), Fluent (black line) and Measurement (red line), where ICEM1 ($\triangle$), ICEM2 ($\circ$), Harpoon1 ($\diamond$), Harpoon2 ($+$), Measurement ($-$)



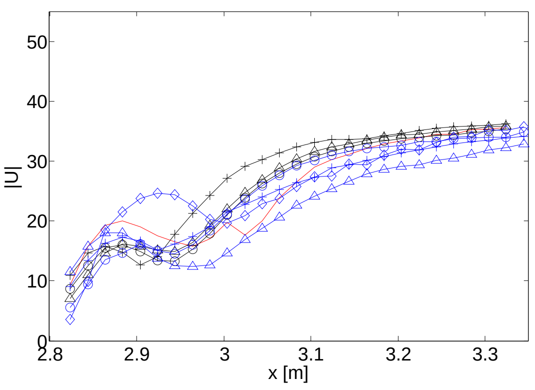
(a) $|U|$ along rake 7



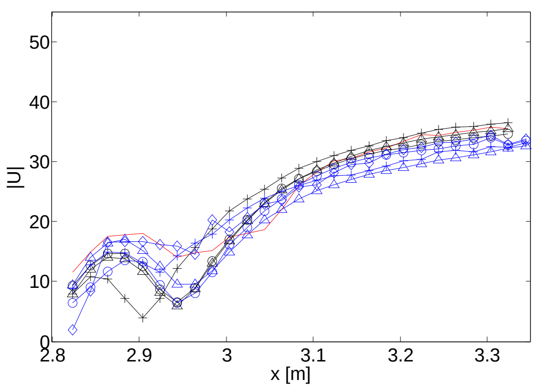
(b) $|U|$ along rake 8



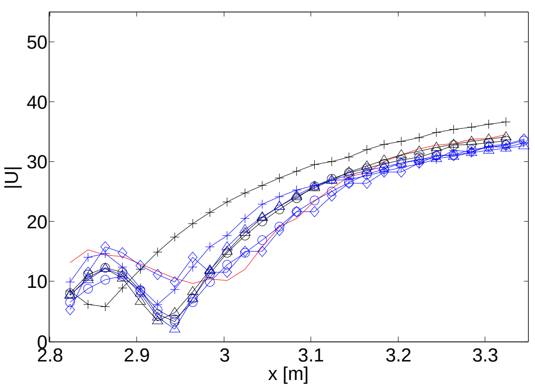
(c) $|U|$ along rake 9



(d) $|U|$ along rake 10



(e) $|U|$ along rake 11

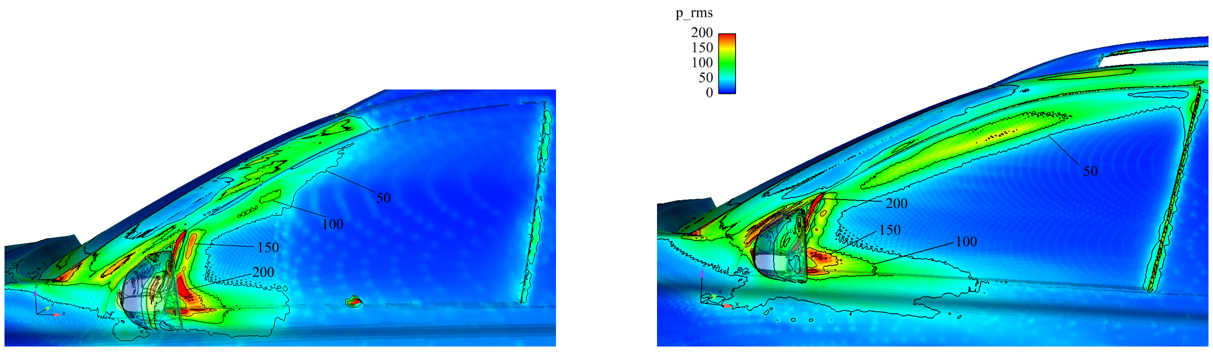


(f) $|U|$ along rake 12

Figure 4.13: Mean velocity magnitude along rake 7 to 12 for OpenFOAM (blue line), Fluent (black line) and Measurement (red line), where ICEM1 ($\triangle$), ICEM2 ($\circ$), Harpoon1 ($\diamond$), Harpoon2 ($+$), Measurement ($-$)

4.2.2 Aeroacoustic results

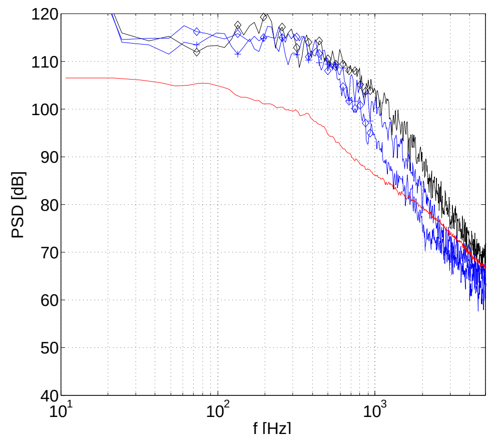
Figure 4.14 illustrates the distribution of the pressure fluctuations in the region of interest. It is shown that the highest levels appear at the root of the A-pillar, as well as downstream the base cover and mirror foot. Excitation is also found in the separated region close to the door frame due to the A-pillar trailing vortex. By comparing Figs. 4.14(a) and 3.7, it is evident that the mesh resolution is of great importance. The cell size is increased dramatically from the center of the side window downstream and one can see that all information is lost due to this. One difficulty with the results, as shown in Fig. 4.14, is the fact that a small displacement of the excited region can have a high impact on the PSD spectra, due to the sharp gradients.



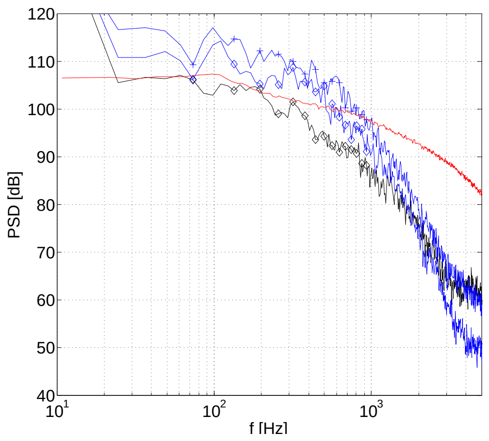
(a) Harpoon1 (b) Harpoon2

Figure 4.14: Contour plot of p_{rms} in the side window region for OpenFOAM on Harpoon1 (a) and Harpoon2 (b)

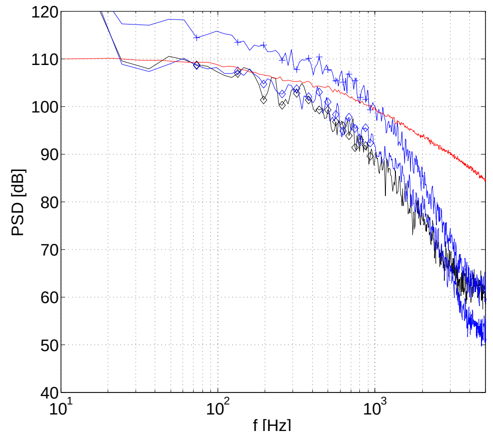
In Fig. 4.15, the PSD of the fluctuating pressure is given for the first six measurement positions (see Fig. 3.9 for the locations of the pressure probes). Both OpenFOAM and Fluent clearly overpredict the intensities for both Harpoon meshes at the first measurement position. However, Harpoon2 yields similar results in Fluent and OpenFOAM. At position two, Fig. 4.15(b), Fluent underpredicts the measured signal on almost the entire spectrum for mesh Harpoon2. OpenFOAM provides intensities that are in good agreement up to 700 $[Hz]$ on the same mesh, whereas for Harpoon1 the measurements are overpredicted. Both solvers yield almost the same result for position three on Harpoon2. The measurements are well represented up to a frequency of 500 $[Hz]$, but decay rapidly after that. Figure 4.15(d) displays the PSD of the RMS pressure at position four, which is the lowest point of the second column of probes, a little downstream from the first column. Here, both codes overpredict the excitation levels slightly, but follow the trend up to 2000 $[Hz]$. When compared with each other, Fluent and OpenFOAM show almost identical results on both the Harpoon2 and Harpoon1 meshes. Fluent captures the levels at position five well up to 1000 $[Hz]$, whereas OpenFOAM gives excessively high levels for both meshes. As seen in Fig. 4.15(f), Fluent predicts rapidly decaying levels from about 100 $[Hz]$. OpenFOAM, in contrast, captures the measurements on both meshes up to frequencies of about 1000 $[Hz]$. In the following, at positions seven, eight and nine, there are no sharp gradients anymore. This is why both codes are capable to predict matching results as compared to the measurements on both meshes up to a frequency of around 1000 $[Hz]$. For position 10 up to position 15, the resolution of Harpoon1 is decreased dramatically. The effect of this can be studied in Figs. 4.16(d)-(f) and 4.17(a)-(c). Due to the increased cell size, it is not possible any longer to resolve sufficient small structures and thus the influence of the higher frequencies gets lost. However, for Harpoon2, OpenFOAM as well as Fluent predict the measurements well up to at least 1000 $[Hz]$ at the positions 10, 11, 12 and 13. Figure 4.17(b)-(c) shows that OpenFOAM has difficulties to predict the measurements at position 14 and 15, whereas Fluent still provides satisfactory agreement on the same mesh. Position 16, which is located close to the door frame (compare Fig. 3.9), shows a clear overprediction for both meshes in OpenFOAM, whereas Fluent captures the trend perfectly up to 700 $[Hz]$.



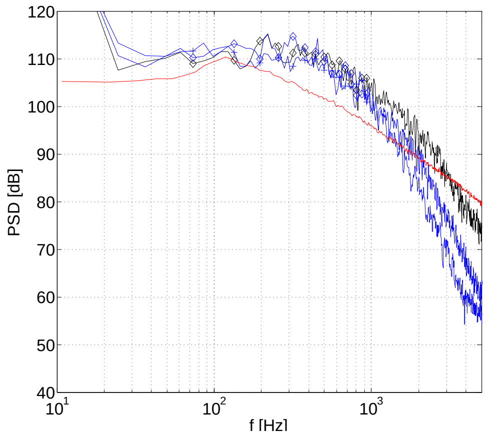
(a) Position 1



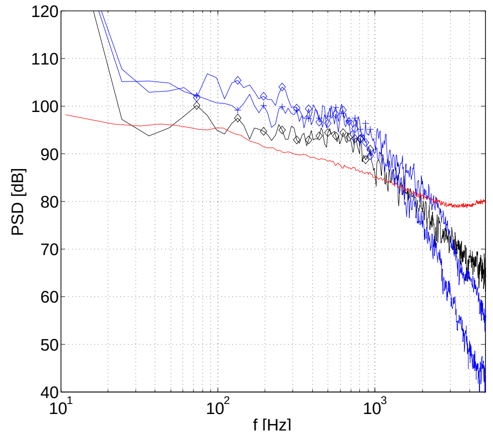
(b) Position 2



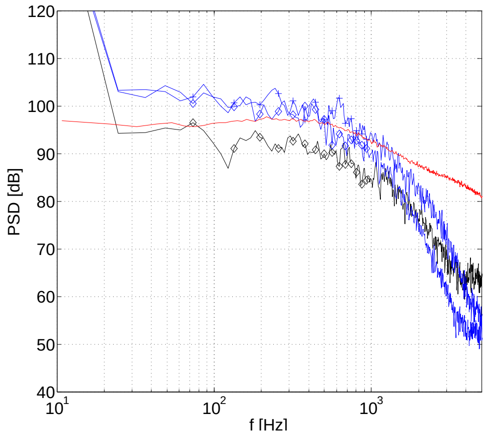
(c) Position 3



(d) Position 4

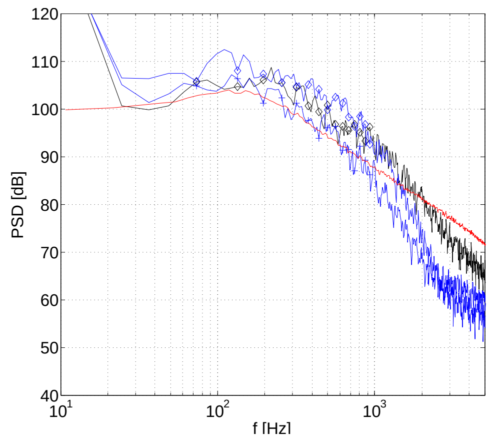


(e) Position 5

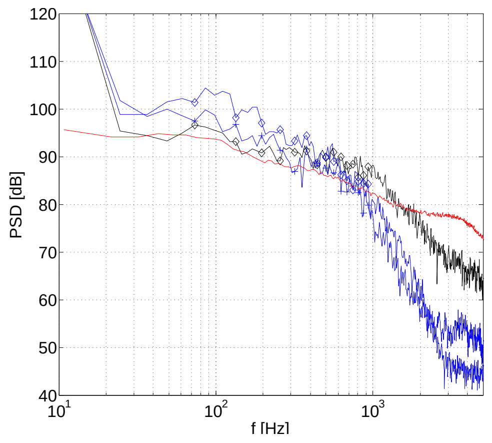


(f) Position 6

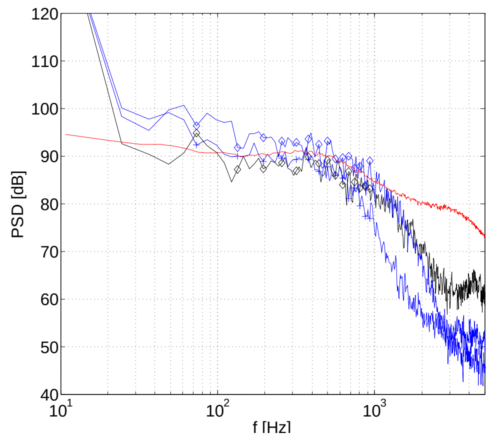
Figure 4.15: Power Spectral Density (PSD) of the pressure fluctuations for OpenFOAM (blue curves), Fluent (black curves) and Measurement (red curves), where Harpoon1 ($\diamond$), Harpoon2 ($+$), Measurement ($-$)



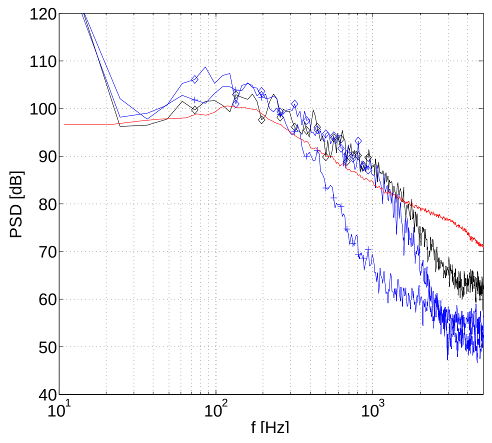
(a) Position 7



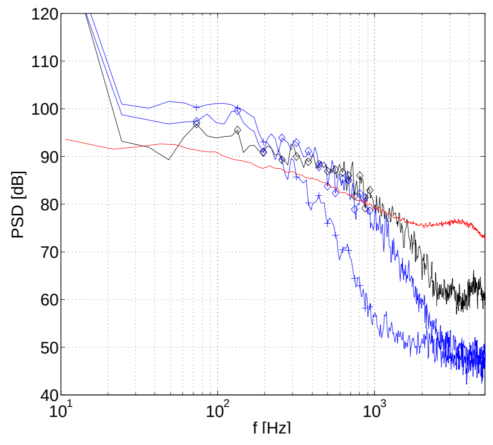
(b) Position 8



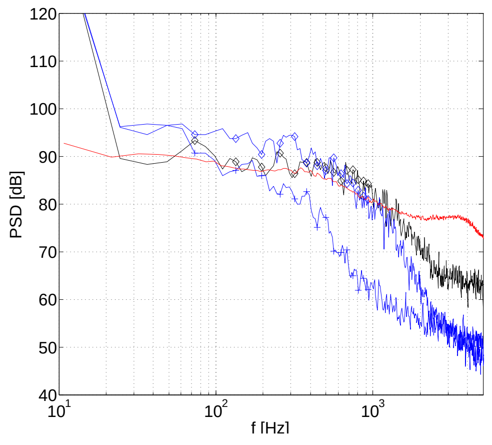
(c) Position 9



(d) Position 10

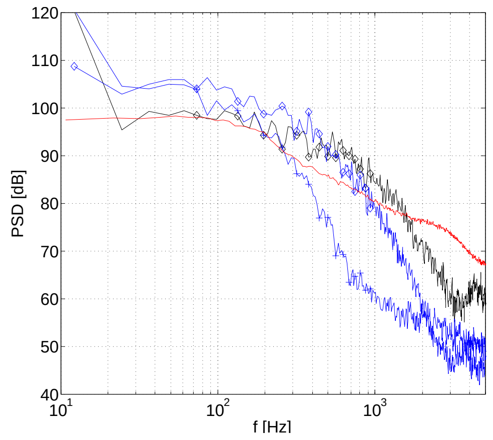


(e) Position 11

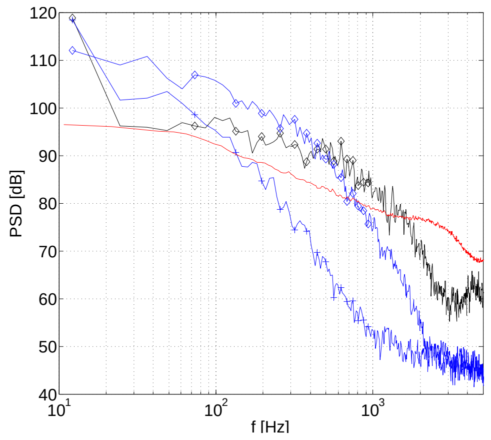


(f) Position 12

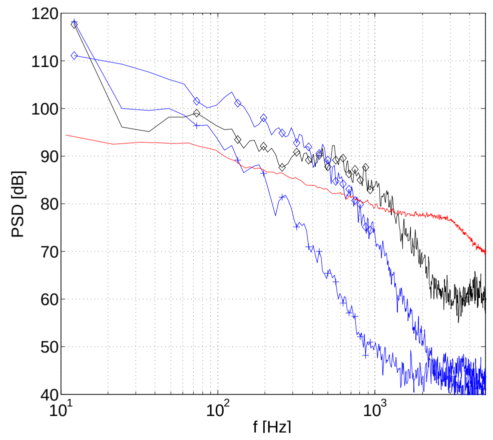
Figure 4.16: Power Spectral Density (PSD) of the pressure fluctuations for OpenFOAM (blue curves), Fluent (black curves) and Measurement (red curves), where Harpoon1 ($\diamond$), Harpoon2 ($+$), Measurement ($-$)



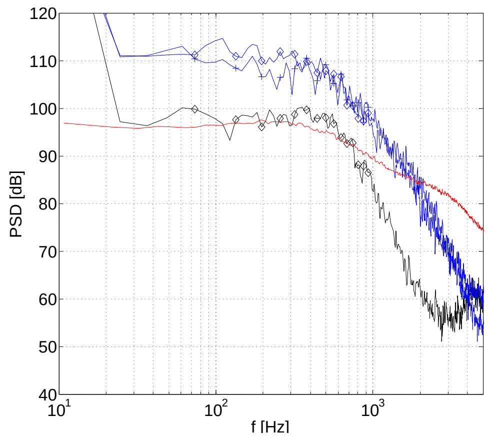
(a) Position 13



(b) Position 14



(c) Position 15

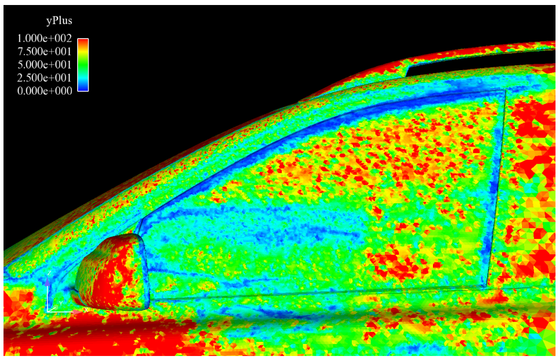


(d) Position 16

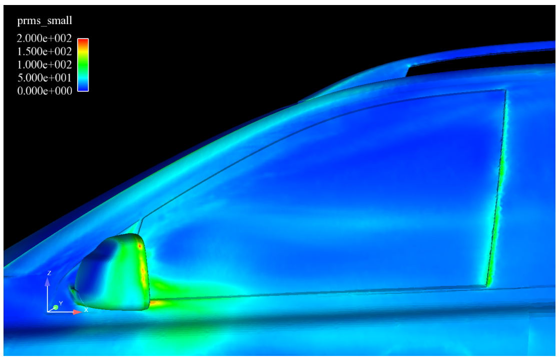
Figure 4.17: Power Spectral Density (PSD) of the pressure fluctuations for OpenFOAM (blue curves), Fluent (black curves) and Measurement (red curves), where Harpoon1 (+), Harpoon2 ($\diamond$), Measurement ($-$)

4.2.3 Mesh dependency

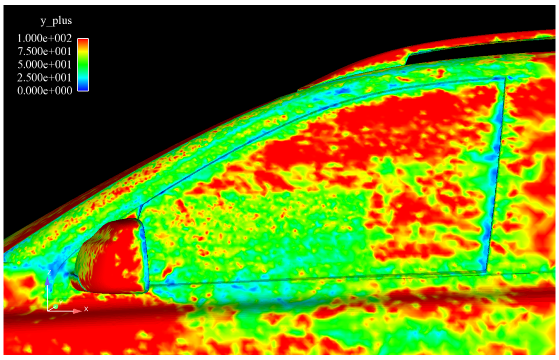
Different meshes with different resolutions in the region around the A-pillar and the side mirror were tested during the evaluation of this case. The meshes are described in section 3.3.1 and are found to yield quite different results. In Fig. 4.18 the RMS pressure and the corresponding y^+ values are given for OpenFOAM and Fluent. One can see that the levels of y^+ tend to be slightly lower for OpenFOAM than for Fluent. The range of y^+ in Figs. 4.18(a) and (c) is from 0 to 100, so that the green areas show levels of $y^+ \approx 30 - 70$. Even though OpenFOAM shows the lower levels in dimensionless wall distance, it is not able to predict the same levels of pressure fluctuations in the separated region downstream of the A-pillar. The excitation around the mirror foot is also missing almost entirely. However, examining the excitations for the mesh ICEM2, which are shown in Fig. 4.19, shows that Fluent also underpredicts the pressure fluctuations for ICEM1. This can be explained with insufficient mesh resolution.



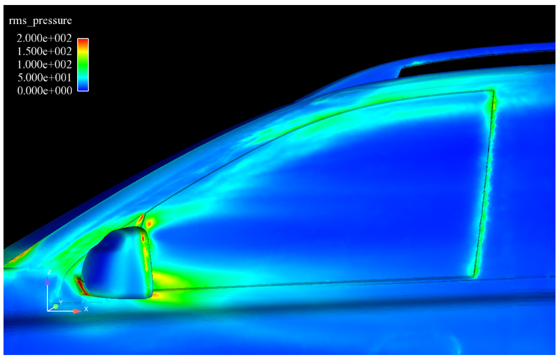
(a) y^+ OpenFOAM on ICEM1



(b) p_{rms} OpenFOAM on ICEM1



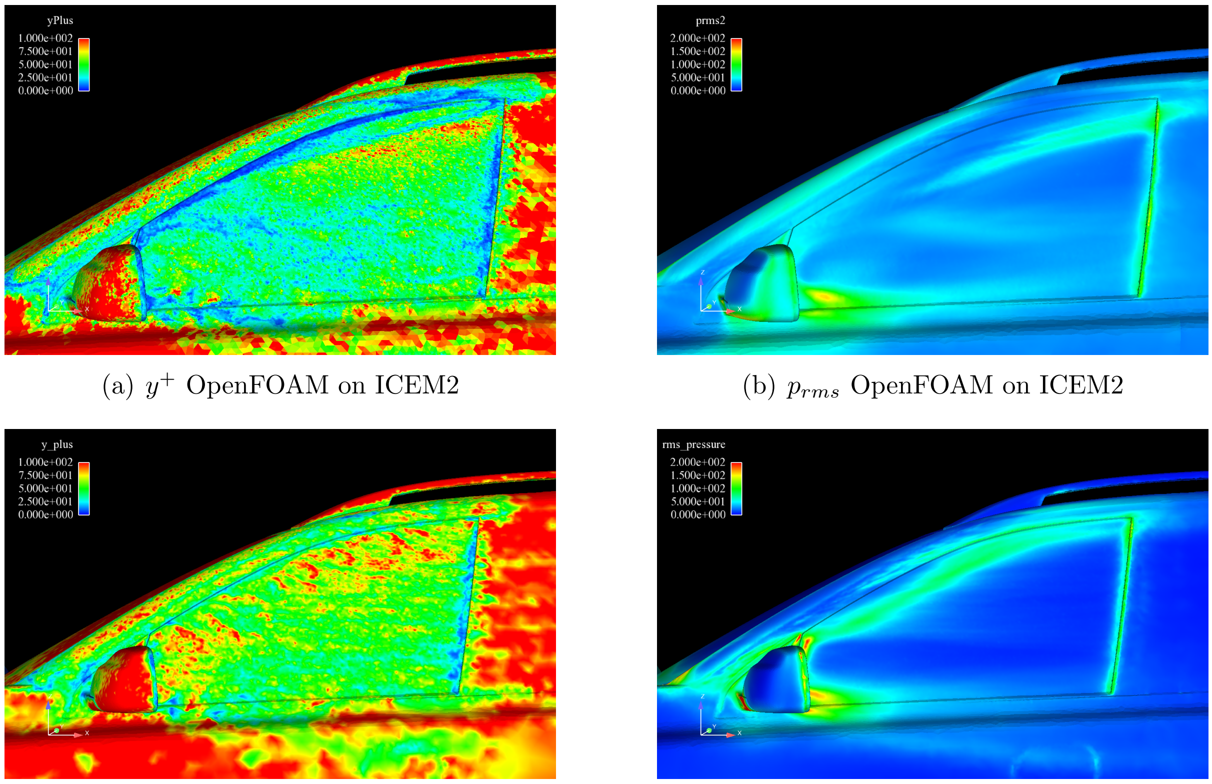
(c) y^+ Fluent on ICEM1



(d) p_{rms} Fluent on ICEM1

Figure 4.18: Comparison of OpenFOAM and Fluent for the mesh ICEM1, concerning y^+ and p_{rms} – y^+: red=100, blue=0; p_{rms}: red=100, blue=0

OpenFOAM shows higher levels of pressure fluctuations for the ICEM2 mesh. However, the areas under excitation are different from those in the Fluent results. A higher influence on the A-pillar itself and a misplaced trailing vortex are predicted by OpenFOAM, as shown in Fig. 4.19. Another explanation for the loss of information downstream the A-pillar can be that y^+ on the wind screen is too high and hence the flow does not carry the correct gradients towards the A-pillar and therefore no valid prediction can be done.



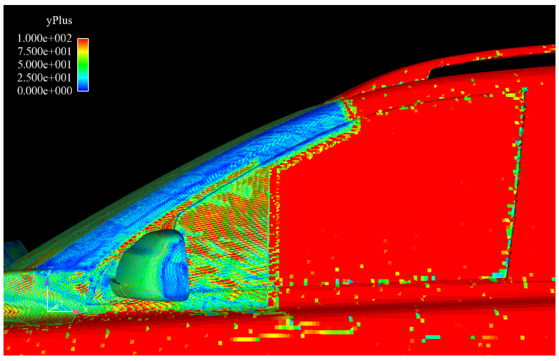
(a) y^+ OpenFOAM on ICEM2 (b) p_{rms} OpenFOAM on ICEM2

(c) y^+ Fluent on ICEM2 (d) p_{rms} Fluent on ICEM2

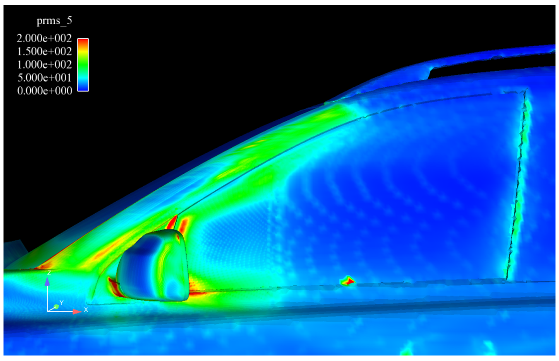
Figure 4.19: Comparison of OpenFOAM and Fluent for the mesh ICEM2, concerning y^+ and p_{rms} – y^+: red=100, blue=0; p_{rms}: red=100, blue=0

Figure 4.20 shows the result obtained with OpenFOAM for the mesh Harpoon1, which has a highly refined A-pillar. The effect of this can be seen in Fig. 4.20(a). The y^+ for the A-pillar is consistently low, so that wall resolved LES is possible. As a result of this, OpenFOAM is able to predict levels that are similar to those of Fluent on the two ICEM meshes. This leads to the assumption that the applied wall functions in OpenFOAM either work in a different way than in Fluent or do not work at all.

Further, investigating the results on Harpoon2, Fig. 4.21, it can be observed that OpenFOAM and Fluent yield very similar results for the same mesh, where OpenFOAM even predicts higher excitation levels. Again, the y^+ values in OpenFOAM are low and thus wall resolved LES is possible.

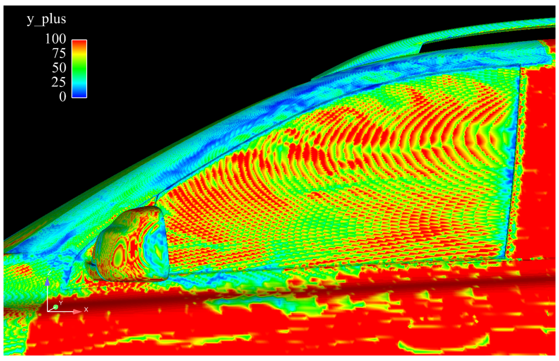


(a) y^+ OpenFOAM on Harpoon1

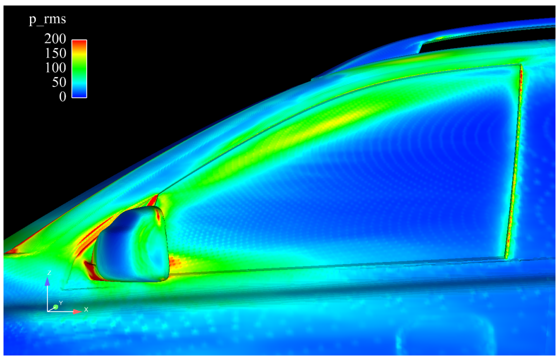


(b) p_{rms} OpenFOAM on Harpoon1

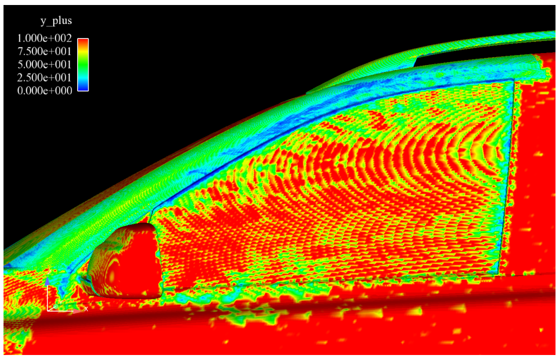
Figure 4.20: y^+ and p_{rms} for OpenFOAM on the mesh Harpoon1 – y^+: red=100, blue=0; p_{rms}: red=100, blue=0



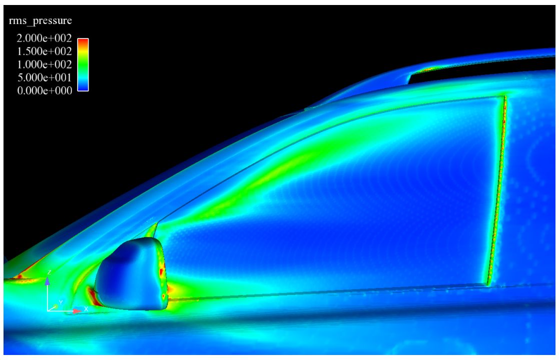
(a) y^+ OpenFOAM on Harpoon2



(b) p_{rms} OpenFOAM on Harpoon2



(c) y^+ Fluent on Harpoon2



(d) p_{rms} Fluent on Harpoon2

Figure 4.21: Comparison of OpenFOAM and Fluent for the mesh Harpoon2, concerning y^+ and p_{rms} – y^+: red=100, blue=0; p_{rms}: red=100, blue=0

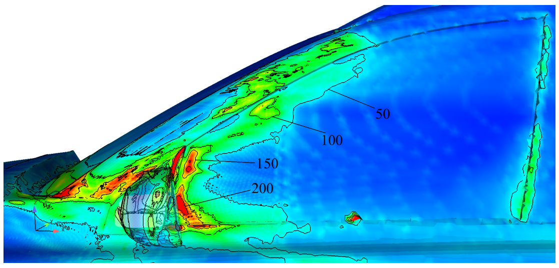
4.2.4 Convergence rate on the A-pillar

There are several possibilities for evaluating whether or not a simulation is converged. One can, for example investigate the development of the residuals and say that the simulation is converged once all the residuals fall below a defined threshold. Here, the development of the RMS pressure in the region of interest is investigated over time. Convergence is defined as the point in time at which no significant change in the solution is appearing anymore. First, the convergence on the mesh Harpoon1 is analyzed. Figure 4.22 shows the region around the side mirror and A-pillar at different timesteps.

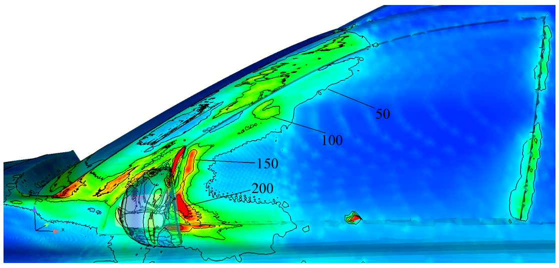
No significant changes can be seen in Figs. 4.22(a)-(h). However, the pressure fluctuations tend to decay slightly over time.

It is possible to determine the integral surface value of a surface patch in the model. In this case the integral over the A-pillar surface was calculated, since it is believed that it is most characteristic for the flow phenomena under investigation. Furthermore, the A-pillar region has the highest resolution.

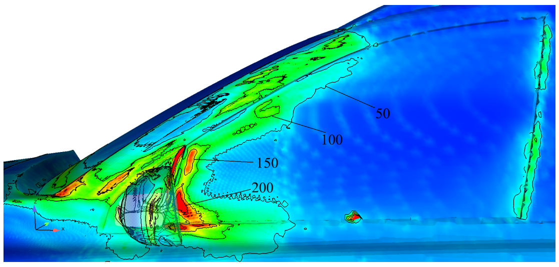
Figures 4.23 shows the value of the surface integral over time for Harpoon1 ($\diamond$) and Harpoon2 (x). One can see that the integral converges after about 0.35 seconds to a value of 6.12 $[Pa\ m^2]$ in the case of Harpoon1 and to a value of 10.18 $[Pa\ m^2]$ in the case of Harpoon2. This can be assessed as a sign that the chosen simulation length is sufficient in this case. Furthermore, it might be wise to begin extracting the statistics at a later point in time. For now, the extraction is initiated after 0.04 seconds (which are not shown in Fig. 4.23), but perhaps a time around 0.2 seconds could prove to be beneficial. The different integral value is caused by the fact that for Harpoon1 and Harpoon2 the A-pillar patch is defined differently and hence their surface area is different as well.



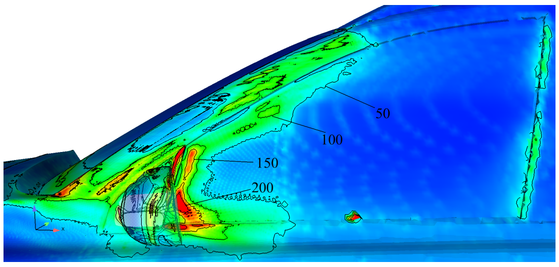
(a) p_{rms} after 0.05 sec



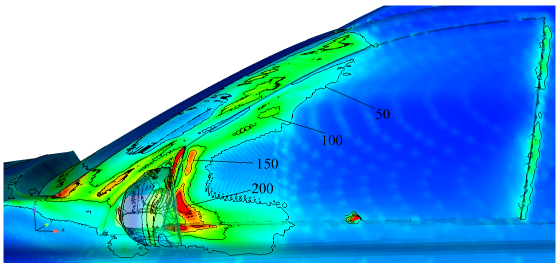
(b) p_{rms} after 0.1 sec



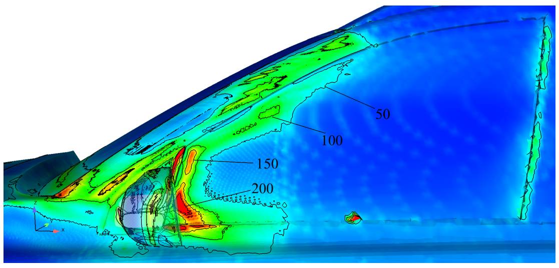
(c) p_{rms} after 0.15 sec



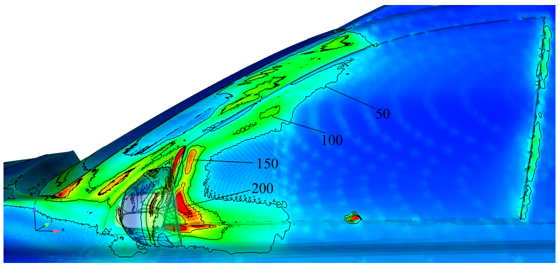
(d) p_{rms} after 0.2 sec



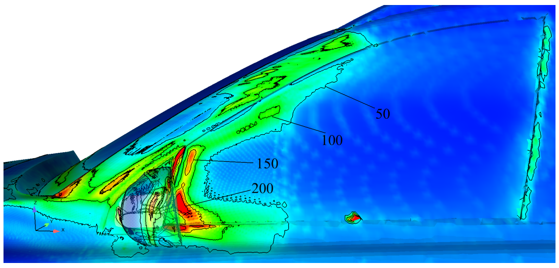
(e) p_{rms} after 0.25 sec



(f) p_{rms} after 0.3 sec



(g) p_{rms} after 0.35 sec



(h) p_{rms} after 0.4 sec

Figure 4.22: Development of the RMS value of pressure over time

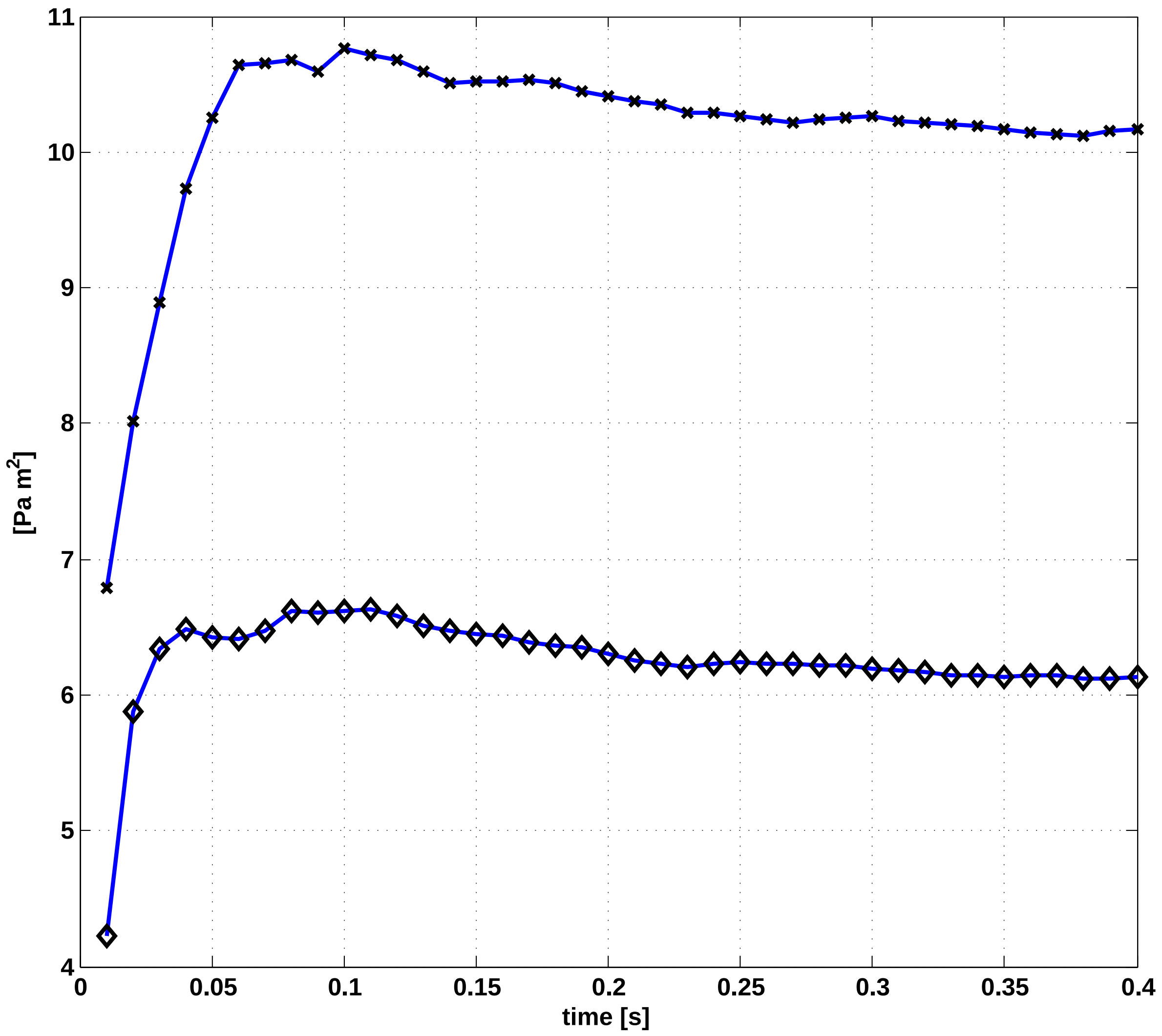
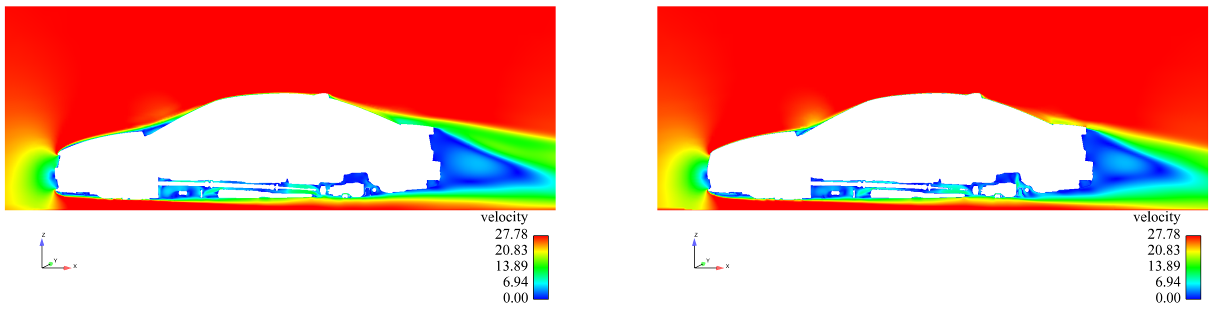


Figure 4.23: Integration of the RMS value of pressure over the A-pillar surface, Harpoon1 ($\diamond$) and Harpoon2 (x)

4.3 S80

4.3.1 Flow results

This section shows the flow results for the simulations of the S80 and a comparison between the OpenFOAM and the Fluent solution is made. First, the overall flow field is plotted on the symmetry plane ($y = 0$) of the vehicle, as shown in Fig. 4.24. In order to get a better view, the vehicle itself is blanked out. One can see that the velocity behaves very similarly in both cases. The wake, especially, is predicted almost identically in both cases, and the flow along the underbody is practically the same. Differences can easily be noted on the upper side of the vehicle, especially for the flow along the hood, at the wind screen and in the region above the trunk; they are more strongly separated in OpenFOAM than in Fluent. This will be explained later in further detail.

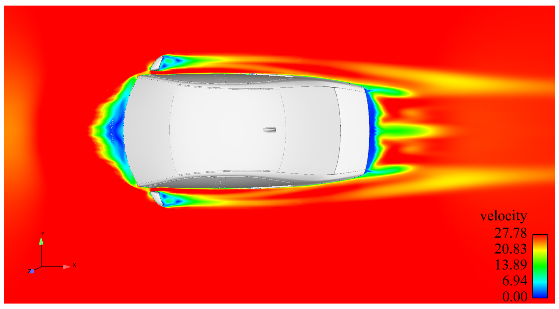


(a) OpenFOAM (b) Fluent

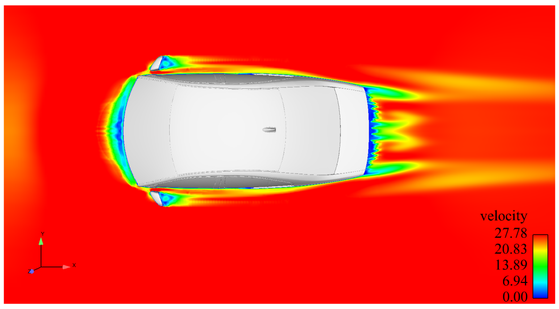
Figure 4.24: Velocity field on the symmetry plane of the vehicle for OpenFOAM (a) and Fluent (b)

Figure 4.25 shows the flow field at the belt line. One can see the influence of the side mirrors and the separation from the trunk. OpenFOAM predicts a similar but slightly different behavior at the rear end of the trunk as compared to Fluent. The Fluent solution shows a lot of small structures, which make the simulation seem almost transient and which are not found in the OpenFOAM solution. The reason for this can be found in the difference in separation from the rear screen of the vehicle. Since the flow separates quite early for OpenFOAM, the structures at the rear end cannot be found. Furthermore, the mirror wake reaches further downstream for OpenFOAM than for Fluent, but still shows a comparable behavior.

Investigating the flow at a plane parallel to the ground that cuts through the center of the wheels yields the results shown in Fig. 4.26. Obviously Fluent provides a different wake behavior in comparison to OpenFOAM. Close behind the vehicle, there is a contraction of



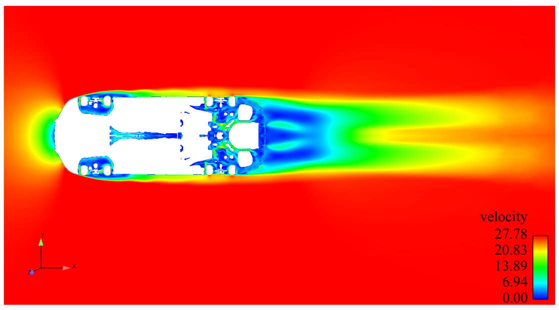
(a) OpenFOAM



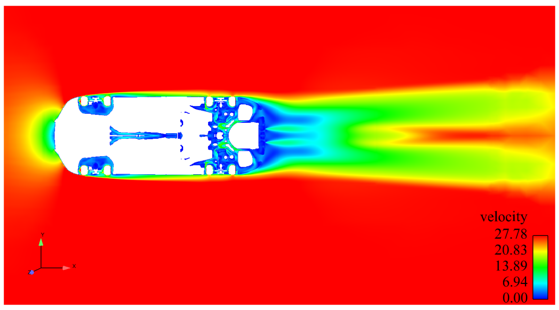
(b) Fluent

Figure 4.25: Velocity field seen from above in a plane at the belt line, OpenFOAM (a) and Fluent (b)

the wake. After this, the wake starts expanding and actually splits up into two separate streaks. OpenFOAM also predicts that the wake splits into two separate streaks; however, the contraction and the expansion are not as strong as in the Fluent result. Looking at the cut through the underbody in the same figure reveals that both solvers provide almost the same result. There is a difference concerning the wheels, which will be explained later in detail.



(a) OpenFOAM



(b) Fluent

Figure 4.26: Velocity field seen from above in a plane cutting through the wheel houses and wheels, OpenFOAM (a) and Fluent (b)

Figure 4.27 gives a detailed view of the side mirror region. One can see that the area of the wake with low flow velocities is larger in the OpenFOAM result and the flow separates from the inside of the mirror head further upstream in the OpenFOAM simulation. Also, on the outside of the mirror head, a stronger separation occurs, which makes the OpenFOAM solution predict a broader wake. In the Fluent solution, there is some interaction between the side mirror wake and the flow stays attached to the side window. This phenomenon cannot be found in OpenFOAM.

At the front end of the vehicle, a different flow behavior can be observed. As shown in

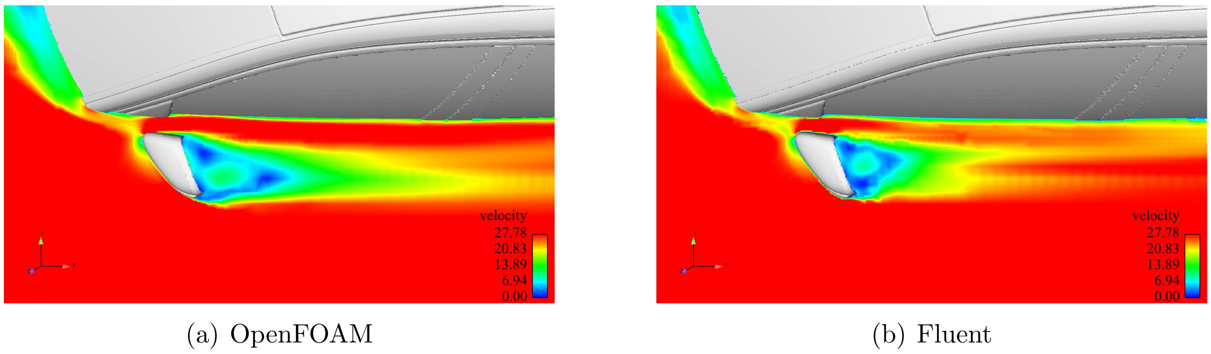


Figure 4.27: Detailed view of the velocity field around the side mirror, OpenFOAM (a) and Fluent (b)

Fig. 4.28(a), the flow separates at the front end of the hood and does not really reattach. Therefore also the flow on the wind screen cannot attach as early as in Fluent. This leaves a rather large region with low flow velocities in the plenum. It is believed that this behavior is caused by the small difference the models show at the air intake. Due to the fact that OpenFOAM had problems converging the original Fluent mesh, it had to be rebuilt. Unfortunately, the mesh used in OpenFOAM is built from an open front model, which explains the different shape at the air intake. Since the air intake for the OpenFOAM mesh shows a sharp edge in comparison to the Fluent mesh, higher separation can be expected. However, the influence of this mismatch is believed to have a rather small influence on the drag value. Furthermore, the shape of the intake yields a larger region with high pressure (low velocity) in front of the vehicle. This causes naturally a higher drag value for the OpenFOAM solution.

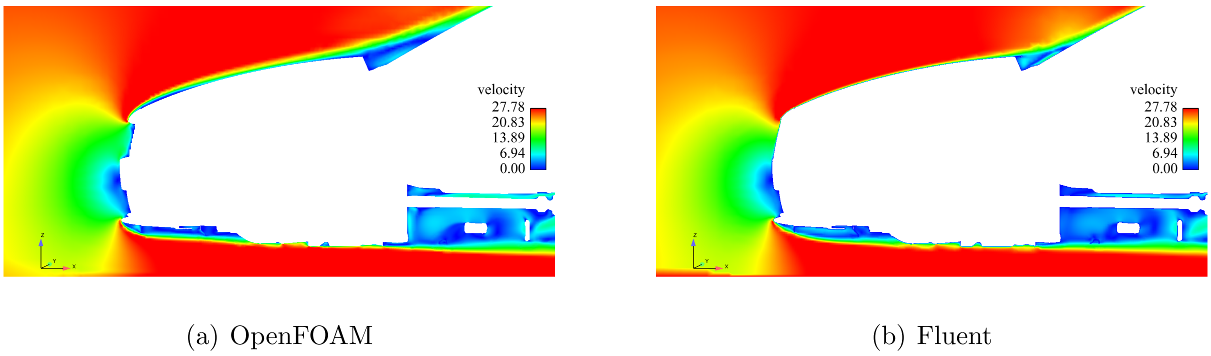
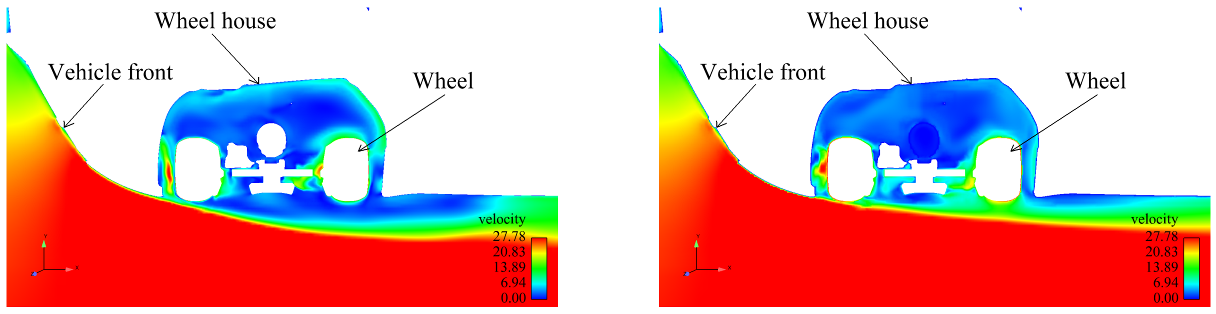


Figure 4.28: Detailed view of the velocity field at the front end of the vehicle, OpenFOAM (a) and Fluent (b)

In Fig. 4.29, the front wheel region is shown. As both simulations (OpenFOAM, Fluent) have the same boundary conditions for the wheels, the difference in result is quite large.

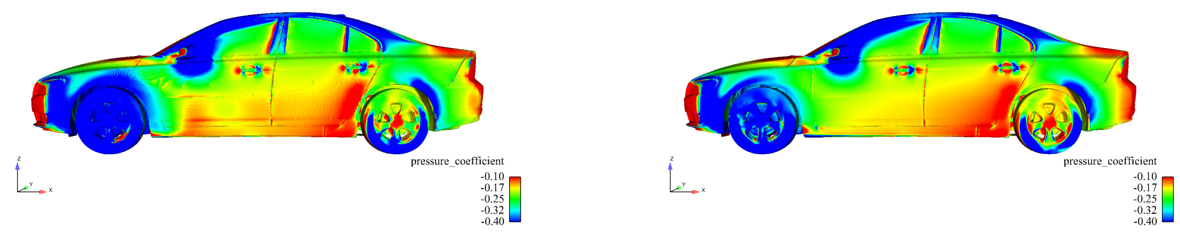
In the OpenFOAM simulation, the flow separates from the upstream end of the wheel and stays separated for a long distance. It only reattaches some distance downstream of the wheel house. The Fluent simulation, in contrast, shows a reattachment of the flow already on the downstream part of the wheel. This behavior of Fluent was also noted in the VRAK case.



(a) OpenFOAM (b) Fluent

Figure 4.29: Detailed view of the velocity field in the region of the left front wheel and wheel house, OpenFOAM (a) and Fluent (b)

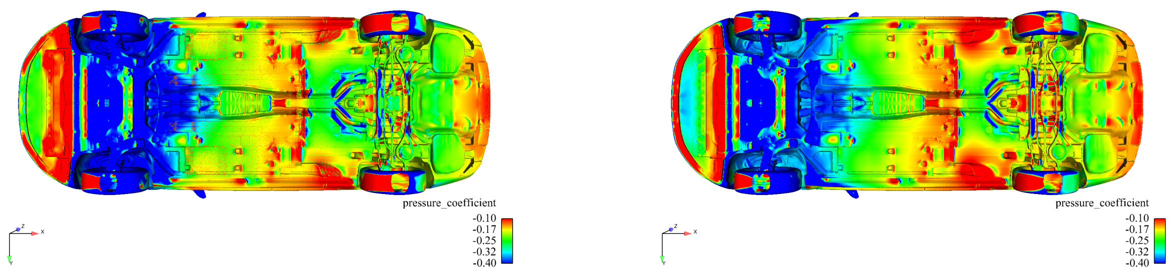
Aside from the general flow field, the pressure distribution can also be analyzed. The latter can be investigated in a convenient way by looking at the pressure coefficient. Figure 4.30 displays the pressure coefficient on the S80 in a side view. Both codes are in good agreement, even though the front wheel shows differences. As already described in the VRAK case, OpenFOAM predicts a stronger separation than Fluent, which can be seen as well in Fig. 4.29.



(a) OpenFOAM (b) Fluent

Figure 4.30: Comparison of the pressure coefficient C_p on the vehicle surface in side view for OpenFOAM (a) and Fluent (b)

The pressure coefficient can also be studied in Fig. 4.31, where the underbody of the vehicle is shown. The vehicle front with the spoiler lip is pointing to the left in the figures. OpenFOAM and Fluent predict different flow behaviors for this region. In the Fluent result, the flow actually hits the spoiler lip, causing a high pressure in front of it. Then the flow separates from the spoiler lip and excites a vortex like structure in a low pressure region behind the lip. OpenFOAM does not show this behavior. Again, the flow separates more noticeably from the front end of the vehicle and does not even hit the spoiler lip anymore. This entails a lower velocity upstream as well as downstream of the spoiler lip and therefore also a higher pressure. It is remarkable that this significantly varying flow behavior at the front end does not have a great effect on the rest of the underbody flow.



(a) OpenFOAM (b) Fluent

Figure 4.31: Comparison of the pressure coefficient C_p on the underbody for OpenFOAM (a) and Fluent (b)

In Fig. 4.32, the pressure coefficient in the rear of the vehicle is shown in a side-by-side comparison, with OpenFOAM on the left hand side and Fluent on the right hand side. It can be observed that OpenFOAM shows some small patches with lower pressure on the lower end of the vehicle. Furthermore, there are some small structures at the top end of the trunk in the Fluent solution, which cannot be found in OpenFOAM. These structures are missing since the flow has previously separated from the rear screen, which yields a rather smooth separated flow at the rear end of the trunk.

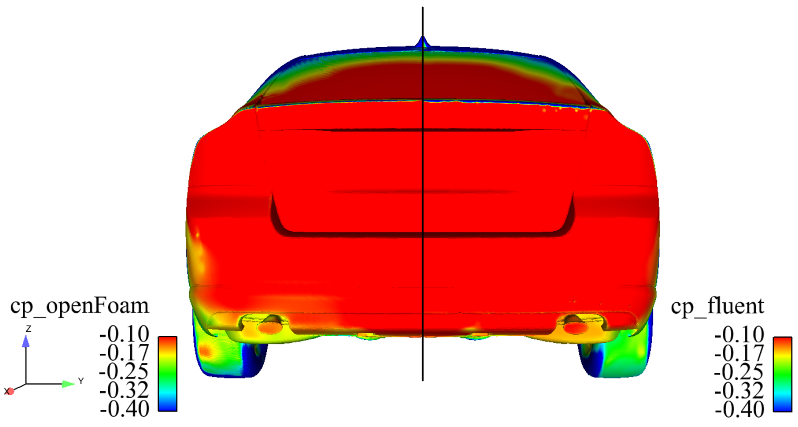


Figure 4.32: Pressure coefficient C_p in the base of the S80, left half: OpenFOAM, right half: Fluent

4.3.2 Drag and lift evaluation

As this is an aerodynamic test case, the drag and lift coefficients must be examined. Figures 4.33 and 4.34 illustrate the development of the drag and lift coefficient, respectively. The reference value from a Fluent simulation is given by the turquoise line. In Fig. 4.33, the red line represents the drag coefficient (multiplied with the frontal projection area), which is obtained by OpenFOAM. The different curves in Fig. 4.34 stand for the total lift (red, fluctuating curve), the rear lift (blue, fluctuating curve) and the front lift (green, fluctuating curve) obtained by OpenFOAM, whereas the solid lines correlate to the corresponding Fluent values.

One can see that the drag value is overpredicted by OpenFOAM. It is about 9% higher than the corresponding Fluent value. This is partly due to the incorrect separation at the front end of the hood and the larger zone of high pressure in front of the vehicle, as shown in Fig. 4.28. Further problems are caused by the separation from the rear screen. This can be seen in Fig. 4.36, where the drag value is better using the *linearUpwindV* scheme. Having a look at Figs. 4.35(a) and (b) reveals that the early separation cannot be found if the enhanced divergence scheme is used. Furthermore, one can see that the drag coefficient converges fairly quickly. Already after about 1000 iterations no major changes in value appear.

If one investigates the lift coefficient, one notices especially that the total lift correlates very well with the given Fluent results. The front and rear lift are instead slightly under- and overpredicted, respectively. This means that the Δ value between the two lift components

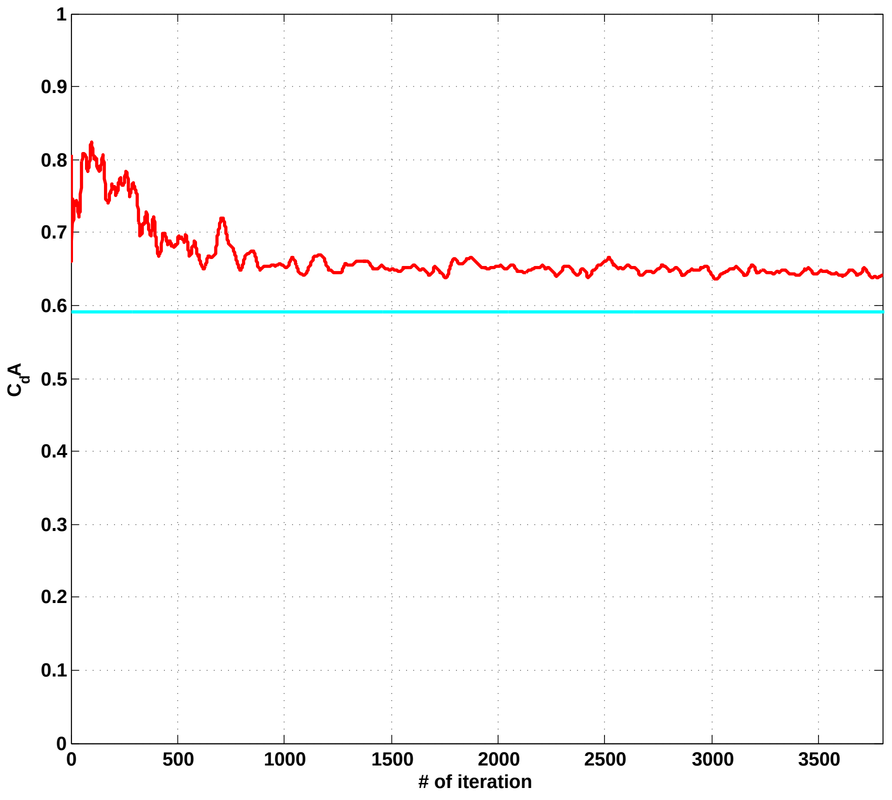


Figure 4.33: C_dA of the S80, obtained with OpenFOAM (red line). The turquoise line represents the Fluent reference value.

is greater for the OpenFOAM solution than it is for the Fluent solution. Δ is calculated in the following way Eq. (4.1).

$$\Delta = |C_{l,rear} - C_{l,front}| \tag{4.1}$$

where the two vertical lines denote the absolute value.

This yields $\Delta_{Fluent} = 0.159$ and $\Delta_{OpenFOAM} = 0.1901$. However, the lift coefficient converges rather late, in comparison to the drag coefficient.

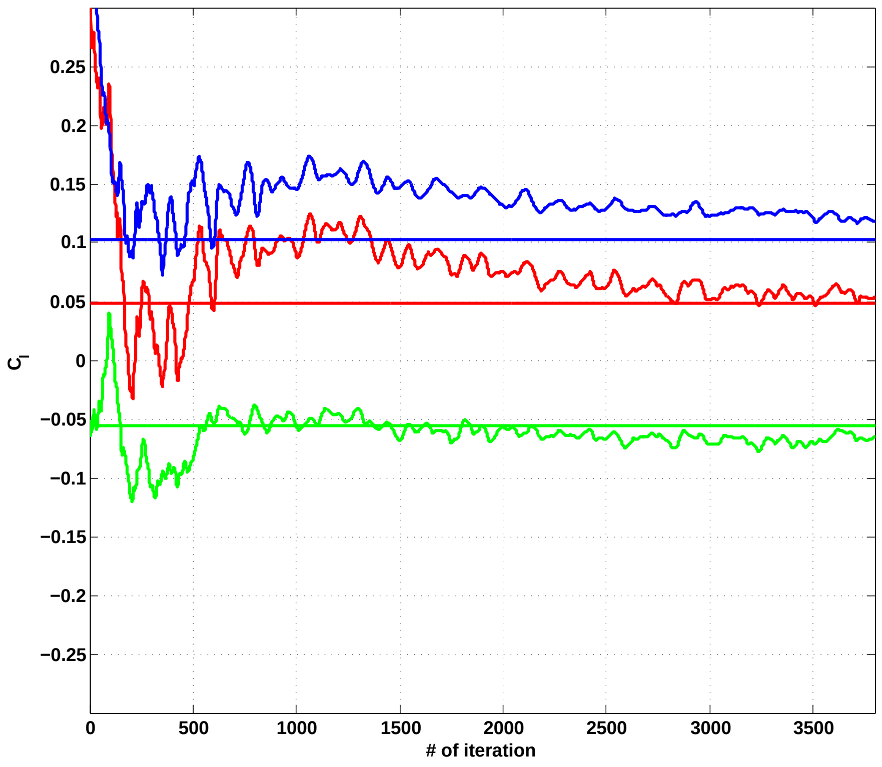


Figure 4.34: C_l of the S80, obtained with OpenFOAM (red, blue and green line). The turquoise line represents the Fluent reference value. red: total lift; blue: rear lift; green: front lift

Regarding the distribution of the front and rear lift, it is particularly remarkable that one will often find a down force (negative lift) on the front axle and lift force on the rear axle when investigating car models in CFD. In wind tunnel tests, however, this behavior is not shown. Usually the front and rear lift are about the same value.

Table 4.3 summarizes the aerodynamic results.

Unfortunately, there are no experimental references available, since no comparable closed front test were carried out.

Table 4.3: Summary of drag and lift results for the VRAK case

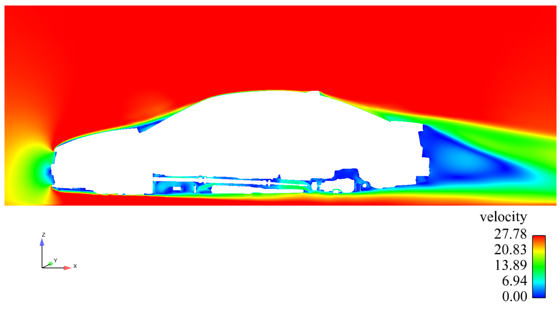
Property	*OpenFOAM*	*Fluent*
C_d	0.278	0.255
$C_d A$	0.6446	0.592
$SD(C_d A)$	0.0033	—
$C_{l,total}$	0.0559	0.048
$SD(C_{l,total})$	0.0042	—
$C_{l,front}$	-0.0671	-0.056
$C_{l,rear}$	0.123	0.103
Δ	0.1901	0.159

4.3.3 Discretization schemes

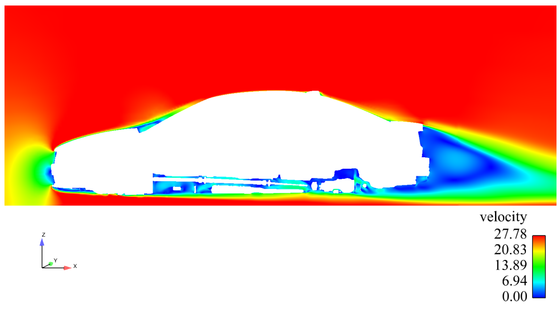
While evaluating the S80 case, different discretization schemes for the momentum divergence were tested. In order to keep comparability with Fluent as close as possible, only different versions of the *linearUpwind* scheme were used. OpenFOAM offers special schemes for vector fields, which are selected by adding a "V" to the name of the normal scheme. These schemes are supposed to be superior to the normal schemes because they take into account the directions of the field. In the following, a small comparison between the *linearUpwind* and the *linearUpwindV* scheme is carried out.

Figure 4.35 shows that the *linearUpwindV* scheme performs better for all the locations shown. It produces very similar results to Fluent as shown above. Investigating the drag and lift values (see Figs. 4.36 and 4.37) reveals that a significantly better lift value is obtained with the normal scheme. The improved scheme shows a slightly better behavior for the drag value.

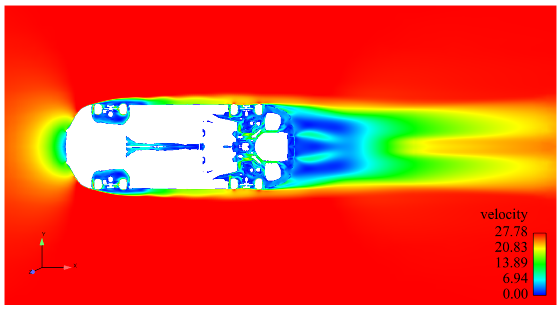
Taking into account that the drag and lift prediction is most important in an aerodynamic simulation and in order to preserve consistency with the VRAK case, the *linearUpwind* scheme is used here. The VRAK case showed significantly better results using the normal scheme in comparison to the improved scheme.



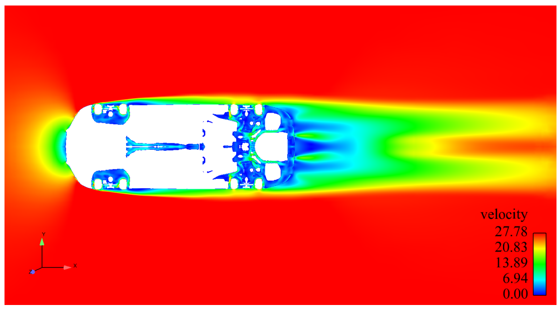
(a) linearUpwind



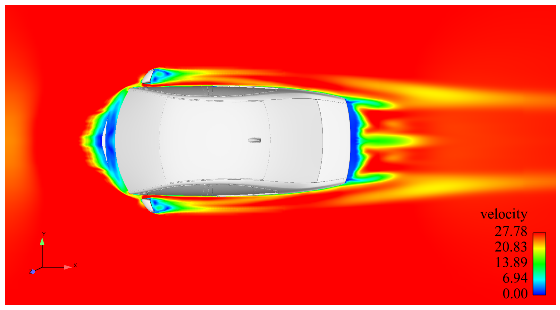
(b) linearUpwindV



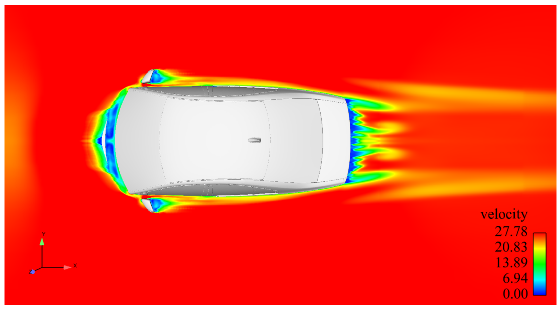
(c) linearUpwind



(d) linearUpwindV



(e) linearUpwind



(f) linearUpwindV

Figure 4.35: Comparison of the flow field between *linearUpwind* (left column) and *linearUpwindV* (right column)

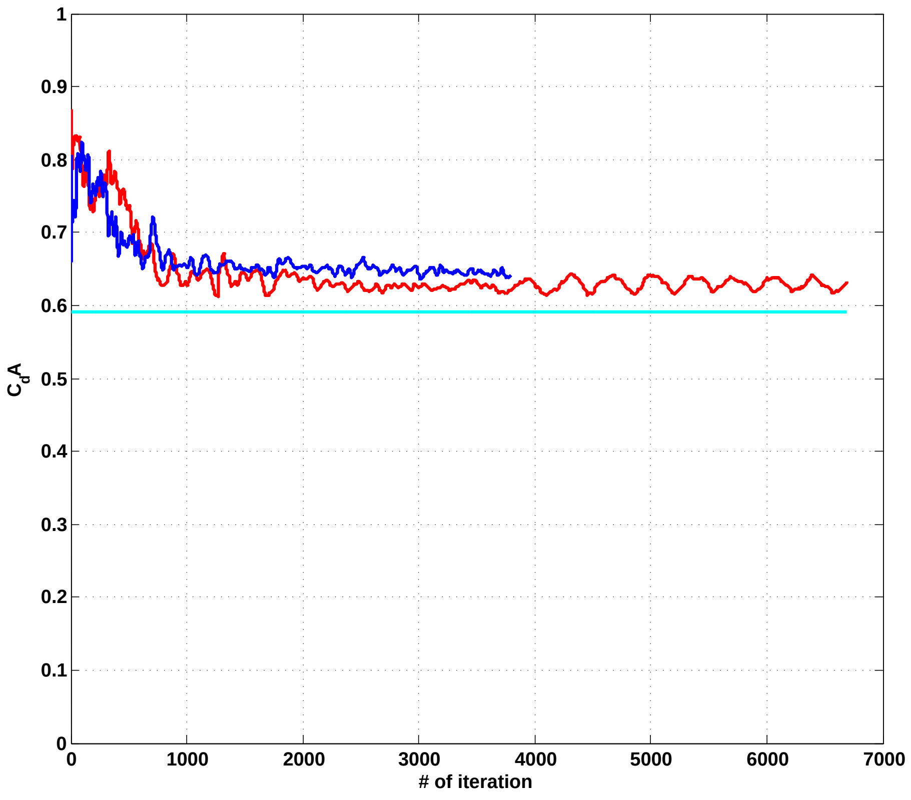


Figure 4.36: C_dA of the S80, obtained with *linearUpwind* (blue line) and *linearUpwindV* (red line). The turquoise line represents the Fluent reference value.

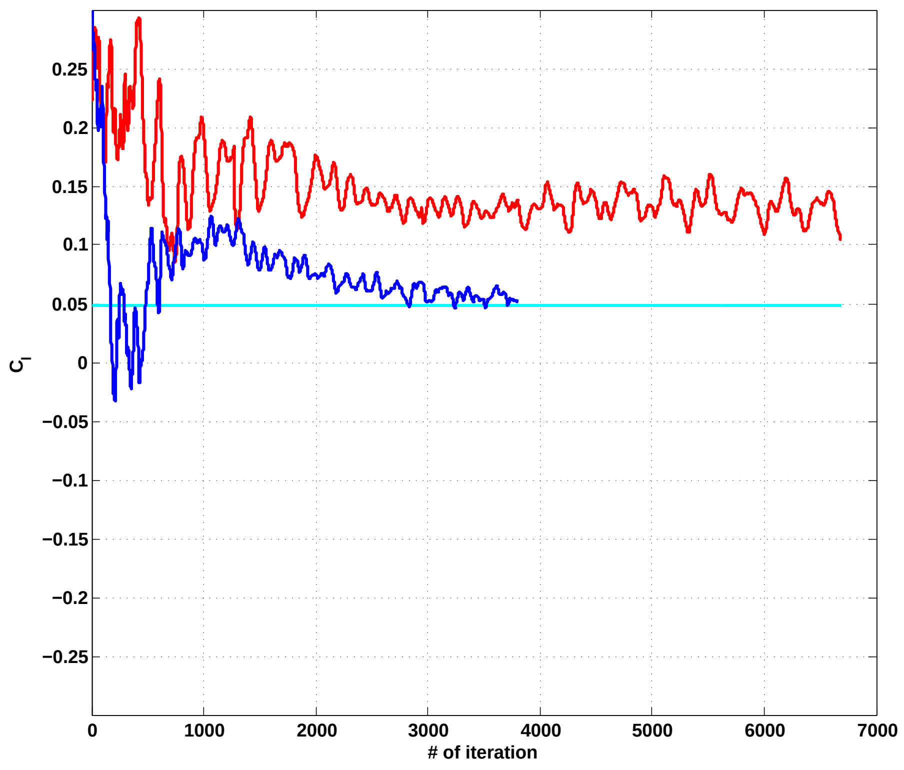


Figure 4.37: C_l of the S80, obtained with *linearUpwind* (blue line) and *linearUpwindV* (red line). The turquoise line represents the Fluent reference value.

4.3.4 Mesh quality issues

In order to make the simulation converge a more complex approach has to be used, as compared to the other steady-state case (VRAK). Using the second order upwind scheme from the beginning, yields divergence, which is probably due to the strong fluctuations at the beginning of the simulation. The turbulence quantities especially are likely to diverge. The reason for this can be identified by investigating the results of a crashed simulation, where only a few highly skewed cells of the mesh (usually on the underbody of the model) contained very high values that made the entire simulation collapse. This led to the conclusion that a more refined underbody would help to circumvent the divergence problem. Eventually it turned out to be most effective to refine the underbody of the model and concurrently correct for the most skewed cells in Harpoon. A maximum skewness of 0.92 turned out to be sufficient.

4.3.5 Treatment of baffles

One of the objectives of this investigation was to find out how OpenFOAM was able to handle baffles. There are routines in OpenFOAM that enable a treatment of baffles similarly to the way they are treated in Fluent. As described in section 2.1.5, the very thin walls are split into two separate walls in order to obtain two walls with clearly identifiable normal vectors. However, the use of the available utilities is very abstract and not well documented. Hence, for now, it is easier to make a detour and exploit Fluent for this purpose instead. It is advantageous to export the mesh from Harpoon in Fluent format. Then it is possible to load the mesh into Fluent and save it again in ASCII format, which can then be processed in OpenFOAM. The baffles are treated automatically while the mesh is loaded into Fluent. Also the list of surfaces is updated automatically.

Chapter 5

Conclusion

This work is a comparative study between two CFD codes, Fluent and OpenFOAM, where the latter is open source. Three different test cases are investigated; two of which focus on aerodynamics and one which focuses on aeroacoustics. The two aerodynamics cases address different levels of abstraction. The first case is a generic vehicle model (called VRAK) without any details, whereas the second one is a fully detailed production vehicle (Volvo S80). Both cases are solved with steady state RANS simulations. The aeroacoustics case is solved with LES in order to capture the pressure fluctuations in the region downstream of the side mirror and the A-pillar for a generic SUV (Volvo XC90).

The VRAK case shows good agreement between Fluent and OpenFOAM, both in terms of accuracy and simulation speed. Discretization schemes were shown to have a large influence on the behavior of OpenFOAM.

The XC90 showed that OpenFOAM delivers results, which are in good accordance with both, measurements and Fluent simulations. It is shown that the results are strongly dependent on the mesh resolution. Also the meshing technique is found to have a strong influence on the behavior. Harpoon meshes show much faster simulation times and are advantageous for the quality of the results in comparison to ICEM meshes.

Finally, the S80 shows that the mesh quality is essential for a stable simulation. Fluent proved to be capable of handling grids, even though simulations on the same grid were diverging in OpenFOAM. These issues can be partially resolved by refining the mesh in the problematic regions, but this results in a longer simulation time. The flow pattern in general has a similar appearance in both Fluent and OpenFOAM, where OpenFOAM shows more significant separations, especially in the vehicle front and at the front wheel. In the S80 case, the treatment of baffles and the implementation of rotating wheels in

OpenFOAM was investigated. Baffles cannot be handled in an intuitive way and hence the automatic baffle treatment of Fluent was exploited. However, the implementation of rotating boundaries at the wheels is rather simple.

Since there is no GUI in OpenFOAM, it is less intuitive to use as compared to Fluent. An advantage that OpenFOAM has over Fluent is the user's ability to change almost all parameters of a simulation during run time, which makes it easy to study the impact that different settings have on the behavior of the simulation. The objective of this work was to assess, if OpenFOAM can replace Fluent at VCC. Therefore the final remark is the following:
OpenFOAM is capable of delivering as good results as Fluent, even if in some cases a more refined mesh might be necessary. This makes it in general possible to change from Fluent to OpenFOAM. However, this change can only be done gradually and further research is inevitable.

Chapter 6

Recommendations for further work

- Investigate the differences in the available discretization schemes and their behavior. Knowing the behavior in detail enables a better understanding of the numerics and thus makes it possible to achieve more accurate results.
- Mesh dependency seems to be crucial in order to obtain stability and to get reasonable results (at least in part). It could be helpful to investigate this issue.
- Since there is no GUI in OpenFOAM and one has to handle a considerable amount of text files, automation would be helpful, especially, if more complex cases (e.g. the S80 or the XC90) are run, since they require many small commands by the user. Therefore working with OpenFOAM is quite time consuming right now and would certainly be improved by having an automated course of events.
- Investigate the usage of Multiple Reference Frames (MRF) in OpenFOAM. At the moment, only a simple rotating boundary is set for the wheels, so the accuracy of the results can probably be improved by applying MRF.
- For the aeroacoustics case, it might be helpful to initiate the extraction of statistics at a later point in time.
- Investigate the application of wall functions for LES in OpenFOAM. For now, a wall function for ν_{SGS} is applied, but there is no certainty that it actually works. Only for wall resolved LES was an acceptable result found.
- Try out the use of the meshing tool, *snappyHexMesh*. It is considered to create really good meshes, which show fast convergence, not only for OpenFOAM, but also for different codes. Thus, even if OpenFOAM does not become the tool of choice at VCC, Fluent simulations could possibly be sped up.

Bibliography

[1] O. Reynolds. *An experimental investigation of the circumstances which determine whether the motion of water shall be direct or sinuous, and of the law of resistance in parallel channels.* None, 1883.

[2] J. Boussinesq. Théorie de l'écoulement tourbillant. *Mem. Présentés par Divers Savants Acad. Sci. Inst. Fr.*, 23:46–50, 1877.

[3] D.C. Wilcox. *Turbulence Modeling for CFD, second edition.* DCW Industries, Inc., 2000.

[4] P.R. Spalart and S.R. Allmaras. *A One-Equation Turbulence Model for Aerodynamic Flows.* AIAA paper, 1992.

[5] A.N. Kolmogorov. Equations of turbulent motion of an incompressible fluid. *Physics*, 6:56–58, 1942.

[6] W.P. Jones and B.E. Launder. The prediction of laminarization with a two-equation model of turbulence. *International Journal of Heat and Mass Transfer*, 15:301–314, 1972.

[7] J.W. Deardorff. A numerical study of three-dimensional turbulent channel flow at large reynolds numbers. *Journal of Fluid Mechanics*, 41:453–480, 1970.

[8] A.N. Kolmogorov. The local structure of turbulence in incompressible viscous fluids for very large reynolds numbers. *Proceedings of the USSR Academy of Sciences*, 30:299–303, 1941.

[9] U. Piomelli and J.R. Chasnov. *Large-Eddy Simulations: Theory and Applications.* Kluwer Academic Publishers, 1996.

[10] J.H. Ferziger. *Large Eddy Numerical Simulations of Turbulent Flows.* AIAA paper, 1976.

[11] J. Smagorinsky. General circulation experiments with the primitive equations. i. the basic experiment. *Monthly Weather Review*, 91:99–164, 1963.

[12] R.H. Barnard. *Road Vehicle Aerodynamic Design.* MechAero Publishing, 2001.

[13] R. Cordes. online - http://rc.opelgt.org/indexcw.php, 2010-05-06.

[14] Openfoam user guide, version 1.6, 2009.

[15] A. Shabbir Z. Tang T.-H. Shih, W.W. Liou and J. Zhu. A new k-epsilon eddy viscosity model for high reynolds number turbulent flows. *Computers and Fluids*, 24:227–238, 1995.

[16] O.A. Marzouk and E.D. Huckaby. *Simulation of a Swirling Gas-Particle Flow Using Different k-epsilon Models and Particle-Parcel Relationships.* Engineering letters advanced online, 2010.

[17] J. Ask and L. Davidson. Flow and dipole source evaluation of a generic suv. *Journal of Fluids Engineering*, 132:1–12, 2010.

Appendix A

Files used in OpenFOAM

In the following the files of the VRAK case are presented in accordance to sections 2.2.1 and 3.1. This section is supposed to support the reader in understanding the file structure and the possibilities of individual adjustment of the solver. The files are sorted according to their appearance in the directories as shown in Fig. 2.6.
First, the files of the *0* folder are presented. One can see that these files are giving the boundary and initial conditions for a RANS simulation with the $k - \varepsilon$ model (since there are files for ε,k ,p and U).

A.1 *0* folder

epsilon The file contains in the beginning a header with information about the software and its version. Then, file information are given in the curly braces, following to the header. After that the actual specification of the initial and boundary field has to be done. For the sake of clarity and to save space, the header and the file information part, is not presented again i quoting the rest of the files.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.6.x                                 |
|   \\  /    A nd           | Web:      http://www.OpenFOAM.org               |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      epsilon;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
dimensions      [0 2 -3 0 0 0 0];

internalField   uniform 3.3398e-5;

boundaryField
{
    body
    {
        type            zeroGradient;
    }
    rear_body
    {
        type            zeroGradient;
    }
    rear_body_lamproof
    {
        type            zeroGradient;
    }
    wheels
    {
        type            zeroGradient;
    }
    outlet
    {
        type            zeroGradient;
    }
    inlet
    {
        type            fixedValue;
        value           uniform 3.3398e-5;
    }
    farfield
    {
        type            symmetryPlane;
    }
    tunnelfloor
    {
        type            zeroGradient;
    }
}
// ************************************************************************* //
```

k

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0.001157;

boundaryField
{
    body
    {
        type            zeroGradient;
    }
    wheels
    {
        type            zeroGradient;
    }
    rear_body
    {
        type            zeroGradient;
    }
    rear_body_lamproof
    {
        type            zeroGradient;
    }
    outlet
    {
        type            zeroGradient;
    }
    inlet
    {
        type            fixedValue;
        value           uniform 0.001157;
    }
    farfield
    {
        type            symmetryPlane;
    }
    tunnelfloor
    {
        type            zeroGradient;
    }
}
// ************************************************************************* //
```

p

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    body
    {
        type            zeroGradient;
    }
    wheels
    {
        type            zeroGradient;
    }
    rear_body
    {
        type            zeroGradient;
    }
    rear_body_lamproof
    {
        type            zeroGradient;
    }
    outlet
    {
        type            fixedValue;
        value           uniform 0;
    }
    inlet
    {
        type            zeroGradient;
    }
    farfield
    {
        type            symmetryPlane;
    }
    tunnelfloor
    {
        type            zeroGradient;
    }
}
// ************************************************************************* //
```

U

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

internalField   uniform (27.778 0 0);

boundaryField
{
    body
    {
        type            fixedValue;
      value           uniform (0 0 0);
    }
    wheels
    {
        type            fixedValue;
      value           uniform (0 0 0);
    }
    rear_body
    {
        type            fixedValue;
      value           uniform (0 0 0);
    }
    rear_body_lamproof
    {
        type            fixedValue;
      value           uniform (0 0 0);
    }
    outlet
    {
        type            zeroGradient;
    }
    inlet
    {
        type            fixedValue;
        value           uniform (27.778 0 0);
    }
    farfield
    {
        type            symmetryPlane;
    }
    tunnelfloor
    {
        type            fixedValue;
        value           uniform (27.778 0 0);
    }
}
// ************************************************************************* //
```

A.2 *constant* folder

RASproperties

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
RASModel            realizableKE;

turbulence          on;

printCoeffs         on;
// ************************************************************************* //
```

transportProperties

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
transportModel  Newtonian;

nu              nu [0 2 -1 0 0 0 0] 1.48498e-05 ;
// ************************************************************************* //
```

turbulenceProperties

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
turbulenceModel realizableKE;

turbulence      on;

laminarCoeffs{}

kEpsilonCoeffs
    {
        Cmu             Cmu [ 0 0 0 0 0 0 0 ] 0.09;
        C1              C1 [ 0 0 0 0 0 0 0 ] 1.44;
        C2              C2 [ 0 0 0 0 0 0 0 ] 1.92;
        alphaEps        alphaEps [ 0 0 0 0 0 0 0 ] 0.76923;
    }
realizableKeCoeffs
    {
        Cmu             Cmu [0 0 0 0 0 0 0] 0.09;
        A0              A0 [0 0 0 0 0 0 0] 4.04;
     C2             C2 [0 0 0 0 0 0 0] 1.92;
        alphak          alphak  [0 0 0 0 0 0 0] 1.0;
        alphaEps        alphaEps  [0 0 0 0 0 0 0] 0.76923;
        alphah          alphah  [0 0 0 0 0 0 0] 1.0;
    }
SpalartAllmarasCoeffs
    {
        alphaNut        alphaNut [ 0 0 0 0 0 0 0 ] 1.5;
        Cb1             Cb1 [ 0 0 0 0 0 0 0 ] 0.1355;
        Cb2             Cb2 [ 0 0 0 0 0 0 0 ] 0.622;
        Cw2             Cw2 [ 0 0 0 0 0 0 0 ] 0.3;
        Cw3             Cw3 [ 0 0 0 0 0 0 0 ] 2;
        Cv1             Cv1 [ 0 0 0 0 0 0 0 ] 7.1;
        Cv2             Cv2 [ 0 0 0 0 0 0 0 ] 5;
    }
```

```
wallFunctionCoeffs
    {
        kappa           kappa [ 0 0 0 0 0 0 0 ] 0.4187;
        E               E [ 0 0 0 0 0 0 0 ] 9;
    }
// ************************************************************************* //
```

A.2.1 *constant/polymesh* folder

boundary

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
8
(
    body
    {
        type            wall;
        nFaces          364055;
        startFace       33359387;
    }
    rear_body
    {
        type            wall;
        nFaces          125781;
        startFace       33723442;
    }
    rear_body_lamproof
    {
        type            wall;
        nFaces          195958;
        startFace       33849223;
    }
    wheels
    {
        type            wall;
        nFaces          81698;
        startFace       34045181;
    }
    outlet
    {
        type            patch;
        nFaces          465;
        startFace       34126879;
    }
    inlet
    {
        type            patch;
        nFaces          465;
        startFace       34127344;
    }
    farfield
    {
        type            symmetryPlane;
        nFaces          53060;
        startFace       34127809;
    }
    tunnelfloor
    {
        type            wall;
        nFaces          68522;
        startFace       34188091;
    }
)
// ************************************************************************* //
```

A.3 *system* folder

controlDict

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
application     simpleFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;
```

```
endTime         3000;

deltaT          1;

writeControl    timeStep;

writeInterval   3000;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression compressed;

timeFormat      general;

timePrecision   6;

runTimeModifiable yes;

functions
    (
    forces
        {
            type forces;
            functionObjectLibs ("libforces.so");
            patches (body wheels rear_body rear_body_lamproof);
            rhoName rhoInf;
            pName p;
            Uname U;
            log true;
            rhoInf 1.205;
            CofR (-0.049 0 -0.23);
            outputControl timeStep;
            outputInterval 1;
        }
    );
// ************************************************************************* //
```

decomposeParDict

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
numberOfSubdomains 40;

method          metis;

simpleCoeffs
    {
        n               (2 1 1);
        delta           0.001;
    }

hierarchicalCoeffs
    {
        n               (2 2 1);
        delta           0.001;
        order           xyz;
    }

metisCoeffs
    {}

manualCoeffs
    {
        dataFile        "";
    }

distributed     no;

roots
    ();
// ************************************************************************* //
```

fvSchemes

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
ddtSchemes
    {
    default steadyState;
    }
gradSchemes
    {
    default         Gauss linear;
    }
divSchemes
    {
    default         none;
    div(phi,U)      Gauss upwind;
    div(phi,k)      Gauss upwind;
    div(phi,epsilon) Gauss upwind;
    div((nuEff*dev(grad(U).T()))) Gauss linear;
    }
laplacianSchemes
    {
    default         Gauss linear;
    }
interpolationSchemes
    {
    default         linear;
    interpolate(U)  linear;
    }
snGradSchemes
    {
    default         limited 0.5;
    }
fluxRequired
    {
    default         no;
    p;
    }
// ************************************************************************* //
```

fvSolution

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
solvers
    {
        p
        {
            solver           GAMG;
            tolerance        1e-6;
            relTol           0.1;
            smoother         GaussSeidel;
            nPreSweeps       0;
            nPostSweeps      2;
            cacheAgglomeration on;
            agglomerator     faceAreaPair;
            nCellsInCoarsestLevel 6;
            mergeLevels      1;
        };

        U
        {
            solver           smoothSolver;
            smoother         GaussSeidel;
            tolerance        1e-6;
            relTol           0.1;
            nSweeps          1;
        };

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

        k
        {
            solver           smoothSolver;
            smoother         GaussSeidel;
            tolerance        1e-14;
            relTol           0.1;
            nSweeps          1;
        };
    }

SIMPLE
    {
        nNonOrthogonalCorrectors 0;
    }

relaxationFactors
    {
        p               0.3;
        U               0.8;
        k               0.7;
        epsilon         0.7;
    }
// ************************************************************************* //
```