
UNIVERSITÀ
DI TRENTO

Department of Information Engineering and Computer Science

Bachelor's Degree in
Computer Science

Final dissertation

# Modeling slipstreaming effects in vehicle platoons

**Supervisors**

Prof. Michele Segata

Prof. Renato Lo Cigno

**Student**

Andrea Stedile

Academic Year 2019/2020

# Acknowledgements

*I would like to express my sincere respect and gratitude to my supervisors, who have been caring and motivated my progress. I would like thank my whole family for their unconditional love and support. Also, I would like thank my friends Gianluca and Ramon, who are always there for me.*

≪ There will come a time to your life
When you ask yourself a series of questions
Am I happy with who I am?
Am I happy with the people around me?
Am I happy with what I am doing?
Am I happy with the way my life is going?
Do I have a life?
Or am I just living?
Do not let these questions restrain or trouble you
Just point yourself in the direction of your dreams
Find your strength from the sound
And make your transition ≫

# Contents

# Abstract

Platoons are convoys of vehicles that, with the use of communication and control algorithms, drive autonomously at very short distances. Most of the times, such vehicles enjoy reductions in air drag, which is a resistive force. Consequently, they require less tractive effort and consume less fuel. Further benefits of this application are optimized road utilization and traffic flow.

Research on platoons is ongoing and thriving. Traffic simulators represent highly valuable tools for the research community. However, they consider air drag to be constant. The consequence is twofold: first, they cannot be used to investigate how platooning reduces fuel consumption. Second, vehicle dynamic models are less precisely described.

The work of this thesis evolved through the synergy of two different fields of study. We used computational fluid dynamics (CFD) tools to simulate platoons with simplified vehicles, called Ahmed bodies, and investigate whether relationships exist between the number of vehicles, distances and magnitude of air drag. The data we gathered suggest that shorter distances lead to less air drag, although certain ones increase it. We created a GUI application to automate the setup of new simulations with desired vehicles and distances.

We made modifications to a traffic simulator. We added an elementary model that, under certain conditions, can recognize whether a vehicle is part of a platoon and, if so, update its drag coefficient with respect to the other vehicles and their distances. We enabled a vehicle consumption model and vehicle dynamics model to make use of realistic drag data. Finally, we performed a simulation to evaluate the impact of the modifications.

# 1 Introduction

## 1.1 Context

As road traffic increases and the burden on the environment grows, new mobility paradigms are starting to emerge. The idea of cooperative driving is to enable the cooperation among neighboring vehicles by means of communication, to safely reduce their mutual distance and optimize road utilization. Platooning [25] is a cooperative driving application. Platoons are convoys of vehicles in which only the leading one really drives, while the others follow autonomously at very short distances, without requiring the driver to steer, accelerate or brake. Each vehicle has the following components:

- A frontally mounted radar or sensor, which detects the vehicle ahead and measuring the headway;
- A device for wireless communication, which exchanges data about position, speed and acceleration;
- A control unit called CACC, standing for Cooperative Adaptive Cruise Control, which (with the use of the former items) computes the required acceleration and, accordingly, actuates the engine.

One fundamental benefit that comes from platooning is reduction of fuel consumption. In fact, platooning vehicles experience less air drag and, consequently, require less tractive effort. A study performed in the scope of the European SARTRE project has shown that all vehicles in the platoon save fuel by driving at a distance from 5 m to 8 m [16]: in particular, follower vehicles reduce their fuel consumption by up to 16%, while the leader by up to 8%.

## 1.2 Problem

The platooning application has an interdisciplinary nature: it requires control theory, vehicle dynamics, communication. Field operational tests of platooning applications are expensive in terms of time and cost. Research studies can resort to simulation analyses, whose trustworthiness depends on the realism of the models.

Platoon research has not yet addressed concerns such as: Given a certain set of vehicles, which arrangement saves the most fuel? Given different platoons, which is the most convenient for a vehicle to join?
In order to answer these questions, it is first and foremost necessary to model the air drag changes that platooning vehicles experience. However, as far as we are aware, no open source traffic simulator models such phenomenon and consequently none be used for such investigations.

## 1.3 Contributions

Chapters 2, 3 and 4 describe, respectively, computational fluid dynamics (CFD), the OpenFOAM CFD suite and the SUMO traffic simulator. The work of this thesis evolved through the synergy of two different fields of study and, accordingly, dissertation in chapter 5 is divided in two parts. In the first part we describe how we simulated a vehicle with a simplified geometry, called Ahmed body, both in isolation and in platoons with different distances. The goals are:

- Obtain the drag coefficient that the Ahmed body takes when driving in isolation and in platoons;
- Understand the implications of the number and distances of the Ahmed bodies on their drag coefficients.

The results suggest that shorter distances lead to reduced drag coefficients. Reductions are up to 28% for the leading Ahmed body and up to 15% for the followers. However, certain arrangements unexpectedly cause increased drag drag coefficients.

As additional contributions, we present two GUI applications:

- One application automates the setup of OpenFOAM simulations, which otherwise requires to manually define and edit a variety of files. It allows to choose the number and distances of vehicles, as well as another set of parameters.
- The other application can be used as a support for validating a simulation.

In the second part we describe how we modified the SUMO traffic simulator, which is used to test platooning applications but does not yet model the air drag changes that platooning vehicles experience. The work revolved around the implementation of a basic (although new) model for predicting such changes. We set the following goals:

- Make SUMO able to recognize whether a vehicle is part of a platoon and, if so, find the other vehicles and their distances;
- Make it able to keep track of and update a vehicle's drag coefficient;
- Expose an API to allow other other modules to read the vehicle's drag coefficient;
- Enable the user to specify data about air drag changes resulting from own CFD analyses;
- Make SUMO use such data when predicting air drag changes;
- Integrate the modifications in SUMO's energy model for electric vehicles .

As a conclusion, we perform a simple simulation about two platooning vehicles, the second of which is electric and enjoys air drag reduction, and observe that its battery discharges less rapidly.

# 2 Fundamentals

In this chapter we present some preliminary notions and terminologies that are necessary to understand the topics addressed by the thesis.

## 2.1 Aerodynamics and fluid dynamics

Aerodynamics [17] is the study of motion of air, particularly as interaction with a solid objects. It is a sub-field of fluid dynamics, and many aspects of its theory are common to these fields. We start by giving two definitions:

**Turbulence** When fluid motion has chaotic changes in pressure and trajectories, it is said to be turbulent or, simply, a turbulence. The flow around all kinds of vehicles, from trucks to cars, is turbulent.

**Drag** It is a force acting opposite to the relative motion of any object moving with respect to a surrounding fluid. It is generated by high-pressure zones in front of the vehicle and by the turbulent low-pressure zone, which forms at the tail, "pulling" the vehicle back (figure 2.1a) [25].

The following is the drag equation:

$$F_D = \frac{1}{2}\rho v^2 C_d A \tag{2.1}$$

where, for vehicles:

- $F_D$ is the drag force [N]
- $\rho$ is the density of air [kg/m$^3$]
- $v$ is the speed of the vehicle relative to the air [m/s]
- $A$ is the frontal area of the vehicle [m]
- $C_d$ is the drag coefficient of the vehicle (dimensionless)

The drag coefficient is an indicator of a vehicle's aerodynamic performance. Although it is not a constant, it can be considered as such when the vehicle reaches high speeds [18].

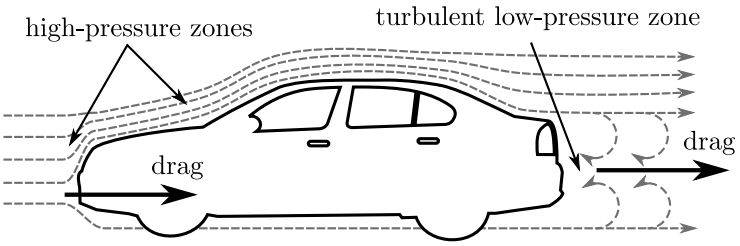
When two vehicles travel very close one another, the first is subject to a lower rear drag because the second disrupts the turbulent flow (2.1b). The second, instead, suffers a lower front drag because it travels in a lower density fluid [25]. This technique is called slipstreaming and allows platooning vehicles to reduce drag and, consequently, their drag coefficient.

## 2.2 Computational Fluid Dynamics

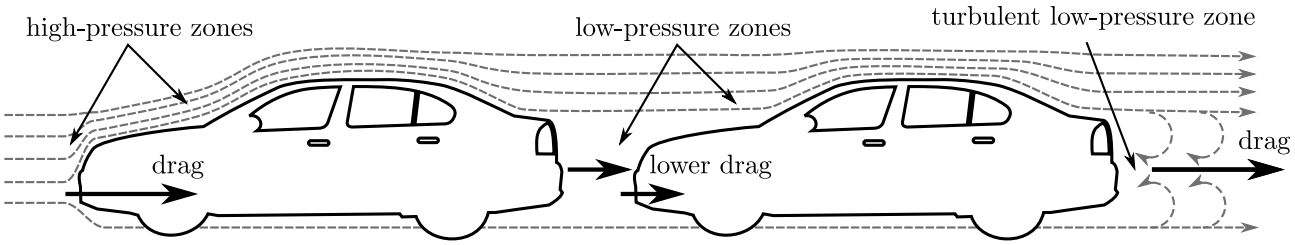
In this section we attempt to describe concepts that do not pertain to our field of study and provide not so rigorous explanations. We advise the reader to consult chapter 3 of [27] to gain a better understand of the concepts.

Computational Fluid Dynamics (CFD in short) is a branch of fluid mechanics that concerns computer simulations of flows. At the core of CFD are the Navier–Stokes equations, a set of partial differential equations which describe flows and how their physical properties change over time. The solution of the equations is a flow velocity, thus a vector field with flow direction and magnitude for every moment in time and every point in space.

The complexity of the Navier–Stokes equations is unpractical for solving turbulent flows. Therefore, efforts in research have been dedicated to the development of numerical methods to make them




(a) Single car




(b) Two close cars

Figure 2.1: Air drag for a single car and two cars traveling close to each other.

amenable to solution. A category of such numerical methods is represented by the Turbulence models for Reynolds-Averaged Navier–Stokes (RANS in short). They focus on the mean flow and the effects of turbulence on mean flow properties. RANS are time-averaged Navier–Stokes equations in which extra terms appear due to the interactions between various turbulent fluctuations. To model such terms, turbulence models are used. Turbulence models have different features and behaviors, which we omit to describe for sake of brevity. One of the most known and applied turbulence models is SST $k - \omega$.

For most purposes of aerodynamic interest, it is unnecessary to resolve the details of the turbulent fluctuations, which are mostly random and change case by case. Information about time-averaged properties of the flow (such as mean velocities, mean pressures etc.) are considered satisfying. Therefore, the vast majority of turbulent flow simulations is performed with RANS-based turbulence models.

CFD is firmly established in the automotive sector to analyze vehicle aerodynamics. We now describe, without demand of rigor, the sub-processes involved in such analyses. We advise the reader to consult chapters 1 and 7 of [23], as well as [13].

### Pre-processing

The process begins by modeling two elements: first, the vehicle geometry; second, the volume in which flow is to be simulated. The resulting volume is discretized in a mesh of cells, which should be of high quality, as defined by measures of orthogonality and skewness.

When meshing, considerations about $y^+$ should be made. To explain what the $y^+$ is, we must first introduce the concept of boundary layer. The boundary layer is the region in close proximity to the surface of the vehicle, which is more properly called wall. Here, the flow transitions from a laminar (regular) regime to turbulent. This plays a key role in determining the aerodynamic forces.

Turbulence models use two different approaches to resolve this flow feature:

- The first approach is called near-wall model and consists in resolving the flow all the way down to the wall. It requires meshing the boundary layer in such a way that a series of many, thin layers of prism cells expand from the wall. This requirement is difficult to meet, because small cells easily compromise the mesh quality.

- The second approach is to model the flow with so-called wall functions. Use of wall functions requires layers of prism cells too, but in this case far less layers are required. This requirement is easier to meet, because mesh quality can be easier preserved.

$y^+$ is a dimensionless value that is used to verify if the flow near wall is resolved sufficiently and if the mesh resolution is in correct range for the turbulence model in use. Usually, near-wall models

require $y^+ < 1$ and wall functions $30 < y^+ < 200$. As the $y^+$ can be verified only upon completion of the simulation, finding the correct settings for the prism layers is often a trial and error process.

The process is continued by setting initial conditions, such as the flow speed, and boundary conditions. The last step is choosing the discretization schemes for terms, such as derivatives in equations, that are calculated during a simulation. Moreover, control parameters for the linear solvers are adjusted. In the OpenFOAM suite, of which we later given an overview, this involves (for example) selecting first order methods, which are bounded and stable, but diffusive; or second order ones, which are more accurate but might become oscillatory [3]. Such settings can greatly impact the simulation and results.

### Processing

Computers calculate the solution of the mesh in an iterative fashion, usually until a convergence criterion is met or certain iterations have passed. Computing time depends on various factors, the mesh fineness above all, as well as the settings of numerical schemes and linear solvers. Usually, CFD software can decompose the mesh in multiple parts and use parallelism to accelerate the calculation.

### Post-processing

The results are validated through a series of steps:

- Verify that the $y^+$ falls within the targeted range. If not, the boundary layer was not correctly modeled and the results are likely to be incorrect.
- Plot the equation residuals to verify that they converge. Residuals are the differences between the equation results of two consecutive iterations, and thus are error magnitudes. As they keep decreasing, the equation results reach values that change less and less. This is known as convergence.
- Plot the quantities of interest, such as aerodynamic forces and coefficients, to verify they are bounded and steady.
- If possible, compare the quantities of interest with experimental results, and verify they are reasonably close.
- Finally, plot the velocity and pressure with a dedicated application, such as [8].

## 2.3 Road traffic simulations

Road traffic simulations can have different goals. They can be used to evaluate the goodness of a road network's design, making use of different metrics such as the road's response to a certain traffic demand and the mean fuel consumption[15]. They represent a highly valuable tool for the research community: for example, they allow testing of new cooperative driving applications.

Depending on the focus of the research, the details of traffic can be resolved to different extents, ranging from the dynamics of average quantities down to individual vehicle motion. Macroscopic simulations concern traffic flows, whereas microscopic ones model vehicles explicitly. The idea of microscopic modeling is to describe the dynamics of each individual vehicle as a function of the positions and velocities of the neighboring vehicles [21]. An example of microscopic model is the car-following, which is based on the assumption that a vehicle can change its velocity only if that does not violate any safety constraint (for example, crashing into others).

# 3 The OpenFOAM CFD suite

In this chapter we give an overview of OpenFOAM, the CFD suite we used to investigate the aerodynamics of single vehicles and platoons. OpenFOAM [7], standing for Open-source Field Operation And Manipulation, allows to simulate several physics phenomena, including turbulence. It enjoys a widespread use in the industry, academia and research community, and is under active development.

## 3.1 OpenFOAM essentials

OpenFOAM does not come with a native graphical user interface. In facts, its paradigm is centered around the operation of the command line and the definition of text files called dictionaries, which are the most common means of specifying data. Therefore, its learning curve can be a bit steep, but it comes with some tutorial cases for learning purposes: for example, the Motorbike tutorial implements a setup for automotive simulations using the $k-\omega$ SST turbulence model.

### Case structure

The following is the typical structure of an OpenFOAM case implementing an aerodynamic simulation:

`0` This directory contains dictionaries specifying initial values, such as air velocity, pressure and turbulence, as well as boundary conditions, such as wall functions.

`constant` Contains files encoding the geometry and a dictionary specifying the type of turbulence modeling to be used, such RAS (Reynolds-Averaged Stress), as well as the model itself, such as $k-\omega$ SST.

`system` Contains dictionaries defining the domain dimensions, the mesh to be produced and, importantly, dictionaries for setting parameters associated with the solution procedure itself: the numerical schemes and the equation solvers, tolerances and algorithms.

### Applications

OpenFOAM comes with a large set of applications to be executed from a command line. They are categorized as follows:

**Solvers** They are each designed to solve a specific problem in CFD. An example is `simpleFoam`, a steady-state solver for incompressible, turbulent flows. It can also be used to compute the $y^+$ upon completion of a simulation.

**Utilities** They perform pre- and post-processing tasks. Examples are `blockMesh`, that can create flow domains, `snappyHexMesh`, that generates meshes and `checkMesh`, that checks mesh quality.

# 4 The SUMO traffic simulator

In this chapter we give an overview of SUMO, the traffic simulation we modified in accordance with the goals set in chapter 1. The overview is meant to enable the reader to understand the work done and put the future user at ease with necessary tools.

SUMO [20] [9], standing for Simulation of Urban MObility, is a road traffic simulator, whose development is mainly lead by the Institute of Transportation Systems at the German Aerospace Center. It allows to address a large set of traffic management topics. We chose SUMO for the following reasons:

- It allows for the simulation of platooning systems.
- It is open source, thus its source code can be studied and modified. It allows to implement and evaluate new algorithms.
- It models traffic with the microscopic approach, thus the events concerning each vehicle can be analyzed.

SUMO's development is being lead with the design goal that the software shall be fast and modular: the core of the simulator is written in the C++ programming language, and the various utilities require to operate the command line. However, it also comes with a graphical user interface for visualizing a running simulation.

## 4.1 SUMO essentials

The simplest SUMO simulation requires a road network and vehicles to be specified. A road network is, at a coarse scale, a directed graph of nodes, usually called junctions, and edges. Edges are single-directed streets between two nodes, and contain at least a lane. An edge can be part of one or more routes, which are network paths on which vehicles can drive.

A vehicle has a specific vehicle type, which describes its physical properties (such as its length, maximum speed and acceleration) and employs a specific car-following model. Only a default vehicle type is provided, which is automatically assigned to a vehicle unless the user create and assign a custom one. Unless otherwise specified, vehicles taking part in a simulation use a collision-free car-following model.

The time step is the temporal resolution into which the position of each vehicle is updated. Intuitively, a large time step produces a fast and rough simulation, whereas a small value produces a slower but more accurate one. SUMO provides a useful mechanism, called `TraCI` (Traffic Control Interface), that allows at each time step to suspend the simulation and access its objects, such as vehicles and road network elements. It consists in a set of APIs that are accessible by C++, Python and Java programs. It allows, for example, to inject vehicles and generate their trajectories, to retrieve their information and change their behavior.

### 4.1.1 Emission and consumption models

Traffic puts a burden on the environment in terms of consumed fuel and emitted pollutants, and vehicle emission models can be helpful in predicting vehicle emission levels. SUMO implements various emission models for gasoline vehicles [19] and an energy consumption model for electric ones [22]. The energy consumption model is the only one defining and using the vehicle's drag coefficient. Therefore, it is also the only we can possibly update. The problem is that the model is insensitive to the vehicle's surrounding, because it considers the drag coefficient to be constant. This implies that the model can overestimate the vehicle's energy consumption.

## 4.2 PLEXE

PLEXE, short term for PLatooning EXtension for vEins, is an extension to Veins for the simulation of platooning systems, which has been recently integrated in SUMO. It was developed by M.Segata et al. [25] [26].

At the core of PLEXE is a car-following model that implements a CACC: it is responsible for the longitudinal control of the vehicle. PLEXE features a realistic vehicle dynamics model, meaning that it models dynamics about engine, brakes, gears ratios and, notably, external forces such as air and rolling resistances. In PLEXE, a vehicle's maximum acceleration is a function of such parameters, and the vehicles are allowed to crash into each other. This is in contrast with SUMO's default car-following model, which is collision free, meaning that a vehicle never engages in a potentially dangerous situation.

PLEXE's models are insensitive to the vehicle's surrounding and considers the vehicle's drag coefficient to be constant. When the vehicle's maximum acceleration is calculated, it might be improperly capped, because platooning often reduces drag and thus alleviates engine load.

# 5 Contributions

The work of this thesis evolved through the synergy of two different fields of study and, accordingly, this dissertation is divided in two parts.

## 5.1 OpenFOAM CFD simulations of platoons

We should reiterate from the outset that the CFD discipline is vast and requires technical competence in its wholeness, thus the use of CFD software must be meticulous. The contents of this section are the result of skills we acquired without formal guidance and by our own efforts. Although most of the work has a rationale, the explanations represent our simplified perspective, which may help or hinder and not satisfy an informed reader. We will state any assumption made in the analyses so that the reader are able to evaluate the results and draw their conclusions.

### 5.1.1 Limitations and scope of the research

OpenFOAM cases can be configured to run in parallel. The computing resource at our disposal was a server with abundant memory (about 100 Gb) but with a scarce number of CPU cores (only 8). Consequently, we used a simplified vehicle called Ahmed body [14], which can be meshed quickly.

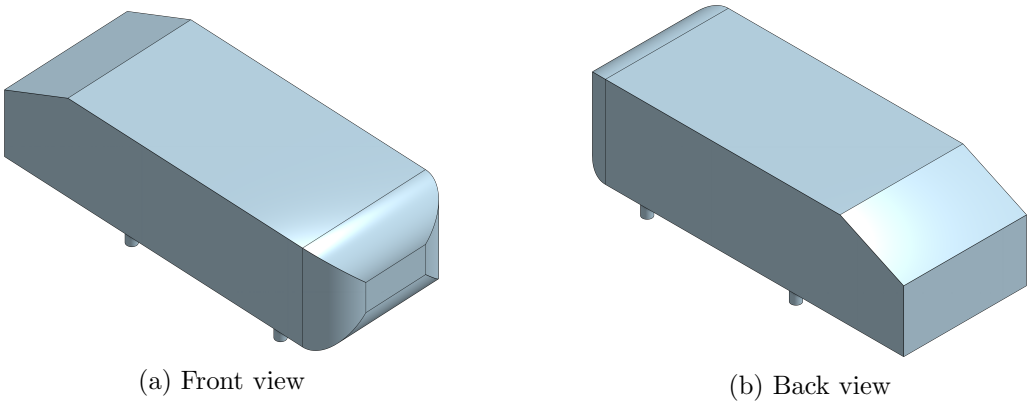
Our goal was to investigate whether relationships exist between the number of vehicles, their distances and their drag coefficients. To achieve this goal, we first simulated an Ahmed body in isolation and obtained its drag coefficient. We then simulated platoons of two to five Ahmed bodies, repeating each simulation for five different distances. Because of time constraint, we were not able to perform simulations to investigate relationships between the size, ordering and positioning of the vehicles (such as their offset on the lane and their angle on a curvilinear stretch of road).

Moreover, we omitted to fully validate the simulations, as it was beyond our abilities. Nevertheless, we followed guidelines for CFD processes and used tools for assessing the goodness of the results.

### 5.1.2 Ahmed body in isolation

The Ahmed body, originally described in [14] and depicted in figure 5.1, has a simplified geometry that retains some salient features of the flow around real cars. It has has a length of 1.044 m, a width of 0.389 m and, including the legs, a height of 0.388 m. It has undergone numerous studies and, eventually, has become a standard used to validate simulation tools.

The drag on the Ahmed body is mostly generated at the rear end and depends on the rear slant angle. We now describe our simulation of the Ahmed body with a 25° rear slant angle, which has an experimental value of of 0.298 [14] [1].



(a) Front view

(b) Back view

Figure 5.1: The Ahmed body.

**Pre-processing** It is suggested that beginners in OpenFOAM start their journey with the tutorials. Therefore we copied and modified the Motorbike tutorial, that can be used for aerodynamic simulations and uses the $k - \omega$ SST turbulence model.

We changed the initial values and turbulence parameters in accordance with the validation case at [1]. Table 5.1 reports them.

We used the Onshape website [6] to model the Ahmed body geometry. We exported it as `STL` (a standard CAD format) with a high quality preset, and substituted the Motorbike geometry. We adjusted the wind tunnel dimensions similarly to [1] (length of 12 m, width of 3 m and height of 5 m).

To reduce meshing and processing time, we applied a boundary condition called symmetry that consists in splitting the domain into two symmetrical halves and simulate only one. We adjusted the meshing parameters to achieve a $y^+ > 30$. Table 5.2 reports the relevant ones. We adjusted some other parameters to increase the refinement of the wake region.

Meshing took approximately 4 m. Upon completion, we inspected the meshing log to ensure that the prism layers were correctly inserted (table 5.3). We also verified that the mesh had a good quality in terms of orthogonality (max: 61.2, average: 4.1) and skewness (max: 2.5).

**Processing** In chapter 3 we mentioned the importance of the numerical schemes and solvers tolerances. For sake of learning and curiosity, we performed the simulation with two different setups, whose differences are reported in table 5.4:

- The tutorial's original setup, which uses first order methods;
- A setup with second methods, tighter tolerances and smaller relaxation factors, which we determined consulting [3].

The simulations take little more than 3 h to complete.

**Post-processing** Figure 5.3 and 5.4 show the residuals and aerodynamic coefficients of the two simulations. We focus on the second one:

- Its coefficients are less oscillatory and its residuals converge to lower values. The $\omega$ residuals have some spikes: most likely, it is because divergence schemes are unbounded. The coefficients, however, have no artifacts.
- Its $y^+$ is reported in table 5.5 and plotted in figure 5.2. We observe that it is sometimes outside of required range. However, the value looks fine in the back region, where the most important interaction (flow separation) occurs.
- To avoid selecting local minima or maxima, we define the drag coefficient as the mean value taken in the last 100 iterations. Therefore, the drag coefficient is 0.303, and is within a 2% error margin of 0.298 (obtained in [14]) and within a 1% of 0.30 (obtained in [24]).

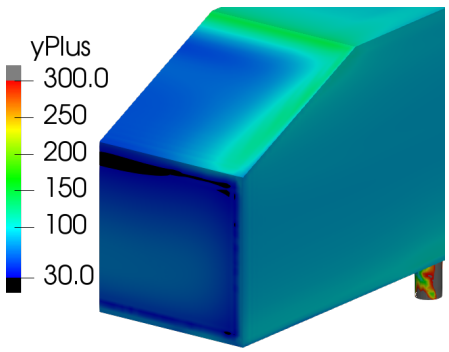



Figure 5.2: $y^+$ plot of the second simulation. Black color indicates $y^+ < 30$, grey color indicates $y^+ > 300$.

| | |
|---|---|
| Free stream velocity [m/s] | 63.7 |
| Turbulence kinetic energy | 21.9 |
| Turbulence specific dissipation rate | 29215 |
| Air kinematic viscosity [kg/ms] | $1.5295 \cdot 10^{-5}$ |
| Air density [kg/m$^3$] | 1.196 |

Table 5.1: Initial values and turbulence parameters.

| | Ahmed body | Floor |
|---|---|---|
| Layers | 4 | 2 |
| First layer thickness [mm] | 1.044 | 3.3 |
| Expansion ratio | 1.2 | 1.2 |

Table 5.2: Boundary layer parameters.

| | Ahmed body | Floor |
|---|---|---|
| Layers | 3.98 | 2 |
| Overall thickness [mm] | 5.59 | 7.26 |
| Layer coverage [%] | 99.7 | 100 |

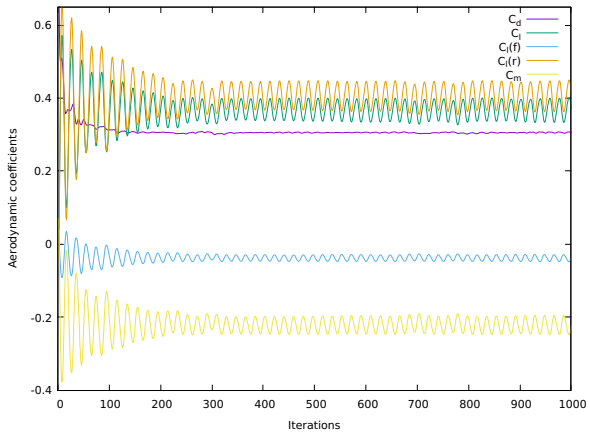
Table 5.3: Resulting boundary layer.

| | First setup | Second setup |
|---|---|---|
| Gradient term | | |
| `U` | cellLimited Gauss linear 1 | Gauss linear |
| Divergence term | | |
| `k` | bounded Gauss upwind | Gauss linear |
| `omega` | bounded Gauss upwind | Gauss linear |
| `U` | bounded Gauss linearUpwindV grad(U) | Gauss linearUpwindV Gauss linear |
| Solver tolerance | | |
| `p` | 0.01 | 0.0001 |
| `U` | 0.1 | 0.001 |
| `k` | 0.1 | 0.001 |
| `omega` | 0.1 | 0.001 |
| Relaxation factor | | |
| `p` | - | 0.3 |
| `U` | 0.9 | 0.7 |

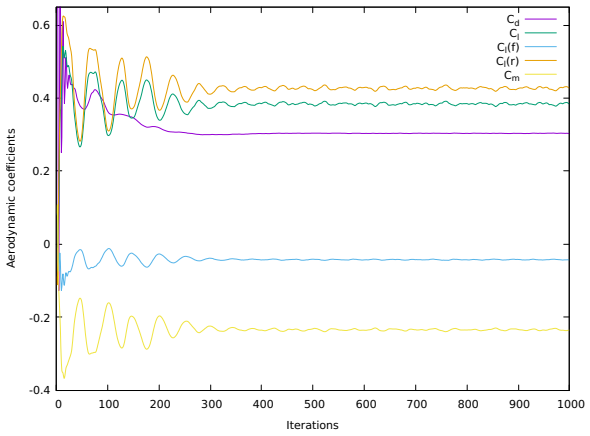
Table 5.4: Differences between the two simulation setups.

| | Body | Legs | Floor |
|---|---|---|---|
| Min | 2 | 14 | 2 |
| Max | 1372 | 615 | 428 |
| Average | 89 | 224 | 66 |

Table 5.5: $y^+$ of the second simulation.

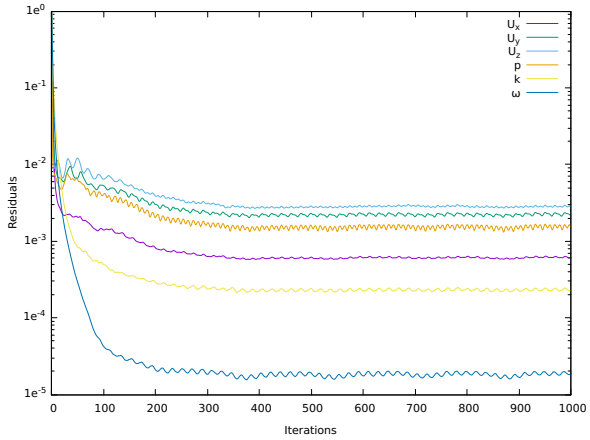



(a) First setup

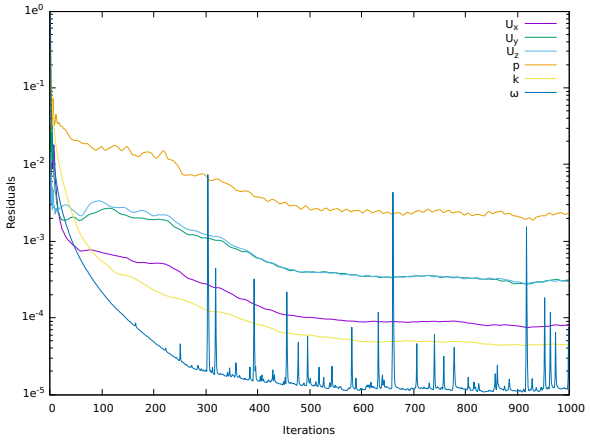



(b) Second setup

Figure 5.3: Aerodynamic coefficients of the second simulation about the Ahmed body in isolation. $C_d$, $C_l$, $C_l$(f), $C_l$(r), $C_m$ are, respectively, drag, lift, front lift, rear lift and pitching moment coefficients [17].

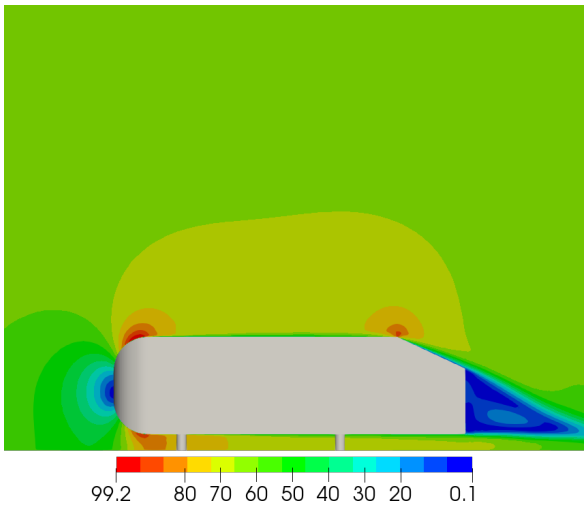



(a) First setup

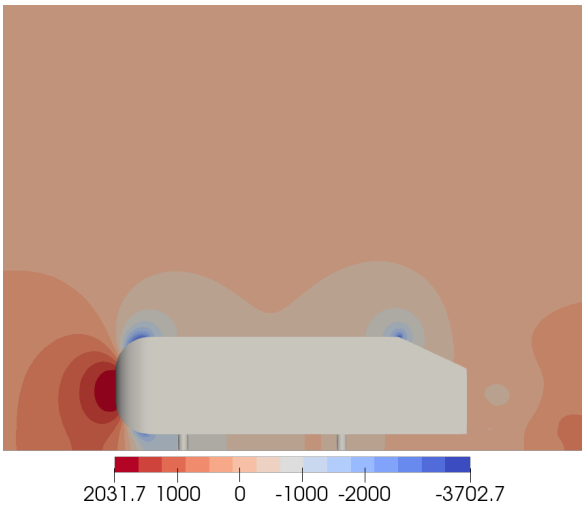



(b) Second setup

Figure 5.4: Convergence of the second simulation about the Ahmed body in isolation. $U_x$, $U_y$ and $U_z$ are the velocity components residuals; $p$ are pressure residuals; $k$, $\omega$ are turbulence variables residuals.




(a) Velocity plot [m/s])




(b) Pressure plot [Pa]

Figure 5.5: Velocity and pressure plots of the second simulation about the Ahmed body in isolation.

### 5.1.3 Ahmed bodies in platoons

We simulated two to five Ahmed bodies in platoons, using the setup of the second simulation about the Ahmed body in isolation, which seemed to yield the best results. We repeated each simulation with seven different distances, for a total of 28 simulations. As we added more Ahmed bodies, we took care of enlarging the wind tunnel.

In real life, vehicles in platoons have distances of 5 to 20 m. We needed to consider that the Ahmed bodies are much smaller than cars. We considered the length of the third generation Fiat Panda [2], which is 3.653 m. We obtained the Ahmed bodies distances with the following proportion:

$$\frac{\text{Ahmed Body's length}}{\text{Fiat Panda's length}} = \frac{\text{Distance } D}{\text{Scaled distance } D_s} \tag{5.1}$$

The following table reports the distances with which we repeated the simulations, together with the scaled up ones:

| Distance $D$ [m] | Scaled distance $D_s$ [m] |
|---|---|
| 0.714 | 2.5 |
| 1.000 | 3.5 |
| 1.429 | 5 |
| 1.715 | 6 |
| 2.143 | 7.5 |
| 2.858 | 10 |
| 4.287 | 15 |

Table 5.6: Distances with which the simulations are repeated.

Figure 5.5 depicts pressure and flow speed around the Ahmed body. Tables A.1 to A.4 in the attachments contain complete drag coefficients measurements. Figures 5.6 to 5.9 summarize the outcomes of our simulations:

- In most scenarios, the Ahmed bodies have reduced drag coefficients. There is a downward trend in drag coefficients as the distances are shortened.
- At very small distances, the leading Ahmed body has a reduced drag coefficient;
- The net change in drag coefficients is a always a reduction.

An extensive study [24] about two Ahmed bodies in platoons, conducted by means of wind tunnel tests, observes the same behavior.

### 5.1.4 GUI applications

**$y^+$ histogram** In chapter 3 we mentioned that, upon completion of a simulation, it is necessary to check whether the $y^+$ has fallen within the target range. In reality, it necessary that the majority of the surface's $y^+$ falls within. One of our contributions is a GUI for plotting the $y^+$ distribution as a histogram. The application allows to adjust five different points of interest and reports the total surface area and average $y^+$. Its source code and documentation can be found at [5].

Figure B.1 in the attachments depicts the $y^+$ distribution of the second simulation about the Ahmed body in isolation.

**OpenFOAM case builder for platoons** In chapter 3 we mentioned that the use of OpenFOAM requires to manually define and edit a variety of files. One of our contributions is a GUI that automates this process, avoiding possible sources of error. The GUI comprises a main window in which the vehicles and their distances are specified, and a wizard that walks the user through a series of choices, such as the initial values, turbulence variables and meshing parameters. When the user launch the creation, the program arranges the required OpenFOAM directories and files. Its source code and documentation can be found at [4].

Figure B.2 in the attachments contains screenshots of the application.

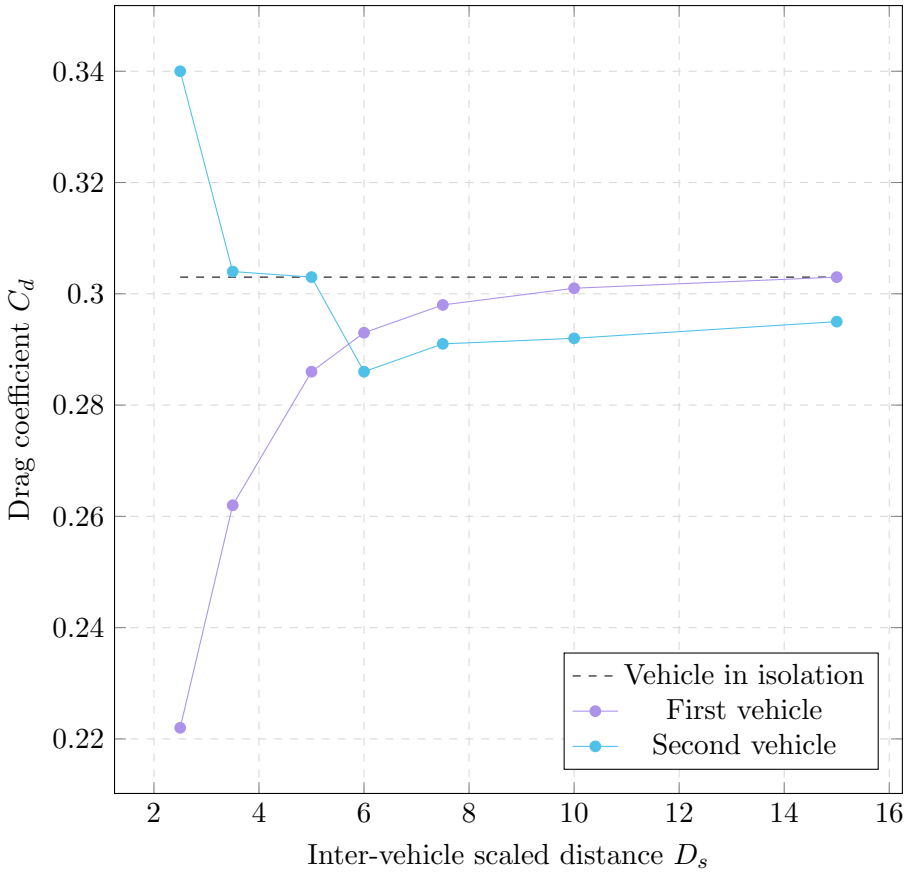



Figure 5.6: Two Ahmed bodies in platoon.

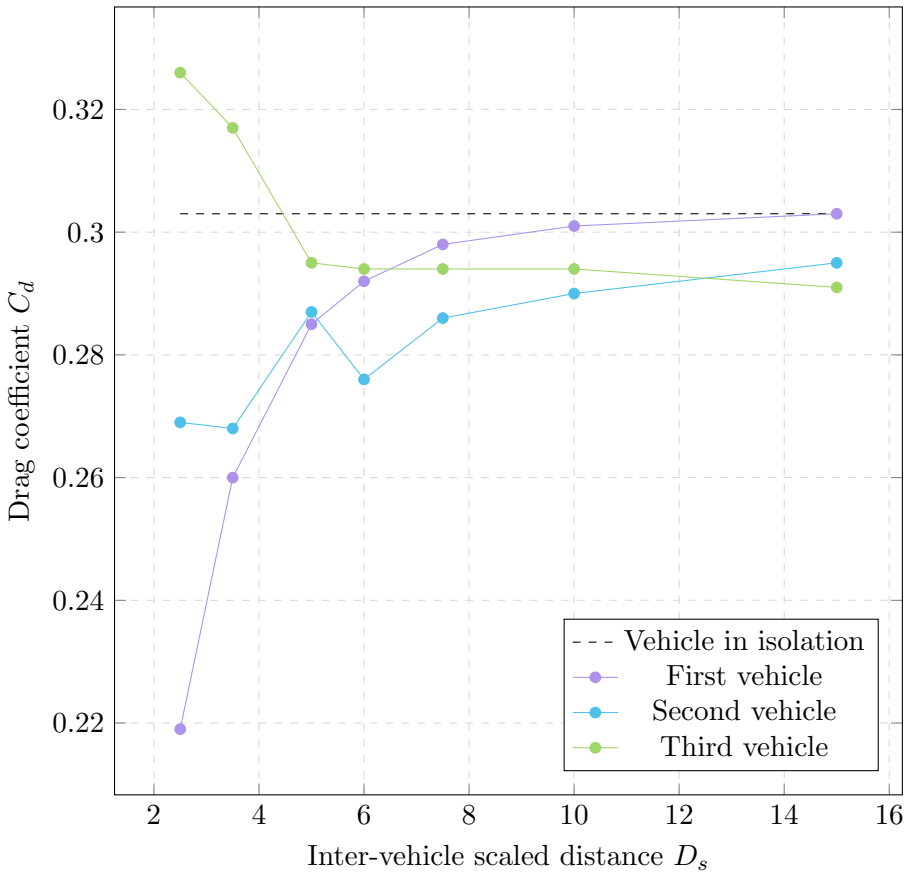



Figure 5.7: Three Ahmed bodies in platoon.

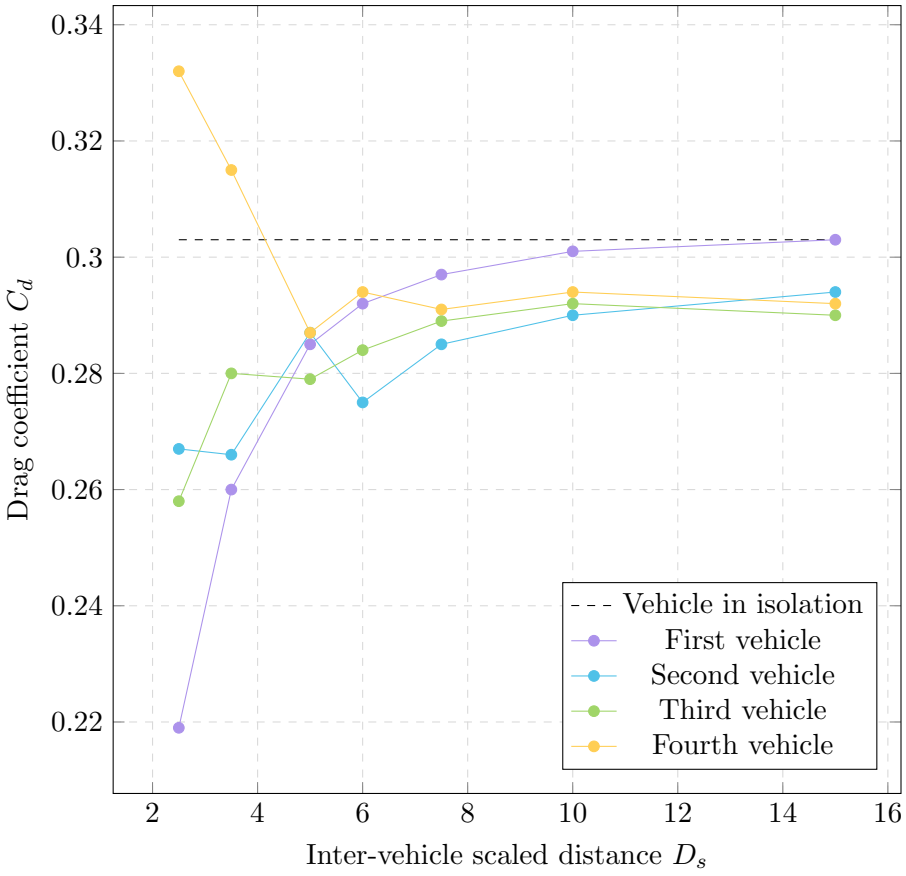



Figure 5.8: Four Ahmed bodies in platoon.

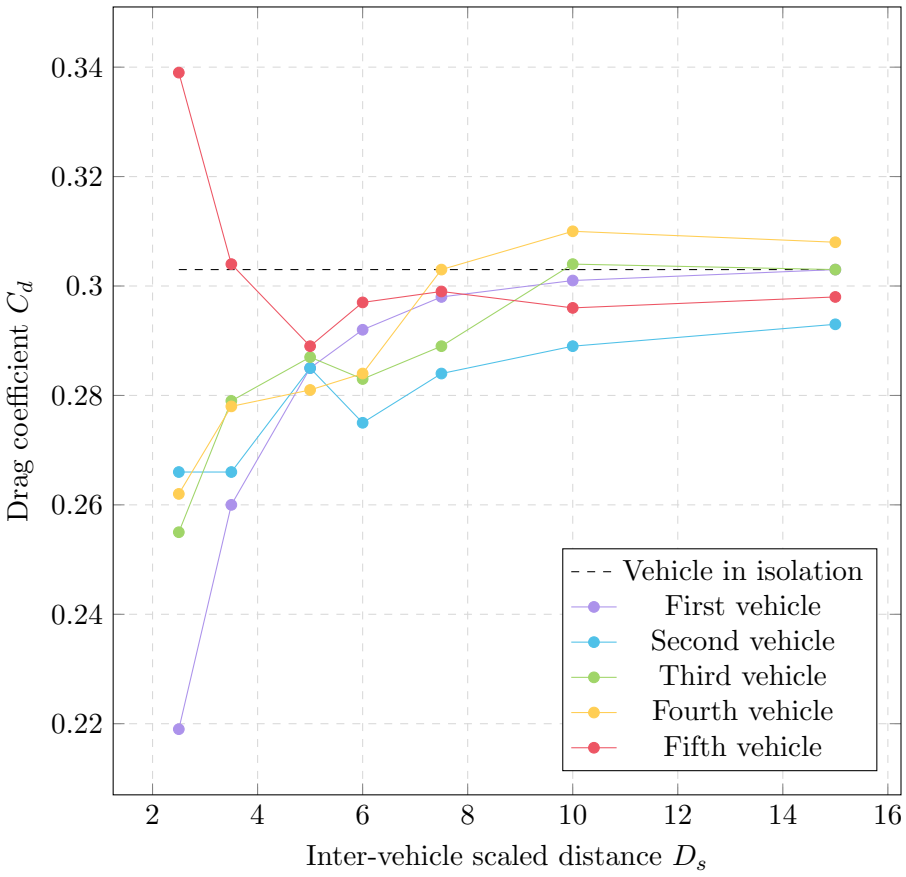



Figure 5.9: Five Ahmed bodies in platoon.

## 5.2 Modeling air drag changes in SUMO

The ultimate goal of this thesis was to modify the SUMO traffic simulator allowing the analysis of the air drag changes that platooning vehicles experience and the investigation of the correspondence between an electric vehicle's air drag and energy consumption.

### 5.2.1 Limitations and scope of the development work

In section 5.1 we performed simulations whose outcomes suggest that, depending on the number and distances of the vehicles, platooning vehicles take different drag coefficients. Consequently, our work must at least take these two variables into account. However, there is a catch: although SUMO provides functions for obtaining a vehicle's predecessor on the lane, no equivalent ones exist for obtaining its successor.

Consider now, for sake of discussion, a platoon with three vehicles, and suppose to know the drag coefficients they take when driving at distances of 5, 10 and 15 m:

- What drag coefficients should they take when driving at 7.5 m? And at 20 m? In general, what happens when their distances change due to acceleration or braking?
- What coefficients should they take when another vehicle joins at the tail?
- What happens if they enter a curvilinear strech of road?

It is difficult to answer such questions, because turbulence has unpredictable outcomes. We agree on the following simplifications:

- We narrow down the problem to the case of platoons driving on freeways and not entering curves. Moreover, vehicles are equally distanced.
- If the platooning vehicles need new drag coefficients, we can try to interpolate them, provided that we know the coefficients they take when driving at (at least) one shorter and one longer distance. Consequently, in the preceding example, we would have been able to obtain new coefficients only for the 7.5 m case.
- If such interpolation is not possible, a vehicle takes its normal drag coefficient.

### 5.2.2 Encoding platoons and CFD data

The goals of this section are to enable the user to specify data about drag coefficient changes resulting from own CFD analyses, program SUMO to load such data at runtime and save it. The user must provide the data by means of a text file formatted as follows:

- It must be subdivided in blocks, each block encoding a different platoon.
- The first line of a block is a list of vehicles. More specifically, the list contains vehicle types, which we discussed in chapter 4, and is ordered from the leader to the last follower.
- The second line of a block is a list that contains the distance of each vehicle from its predecessor. As the leader is not preceded by any other vehicle, the first value must be 0. To enforce the simplification that the vehicles are equally distanced, any subsequent value must be the same.
- The third line of a block is a list that contains the drag coefficient changes of the vehicles in the form of percentages. A negative value represents a reduction; a positive value represents an increase, which we observed to be possible.
- Any line that starts with the `#` character is a comment which is not read.

We then programmed the following classes:

`CfdVehicle` It holds data about a vehicle, such as its type, its distance from the predecessor (and successor), its percentage of drag coefficient change. It holds a reference to the predecessor (and successor), if any.

`CfdPlatoon` It is essentially a vector of `CfdVehicle`.

`Cfd` It is a singleton class that loads the data file and, with the use of the two previous classes, saves it. This class also constitutes an API which is described in the next section.

### 5.2.3 Modeling air drag changes

The goal of the second task was to program an API that, on the basis of the data that the user provides, estimates (and returns) the percentage by which the drag coefficient of a platooning vehicle should change. The API consists of a function (`getDragCoefficientReduction`) that can be called by passing the considered vehicle's type, a vector containing its predecessors' types (ordered from the closest to the farthest) and the inter-vehicle distance.

Consider a situation in which a vehicle type $t$ and a vector of length $n$ are passed as parameters. The function reverses the vector, so that it becomes ordered from the farthest to the closest. A `CfdPlatoon` of length $m$ and the vector and are considered compatible if the following conditions are met:

- $m \geq n + 1$;
- The first $n$ types in the `CfdPlatoon` match, one by one, the types in the vector;
- Type at position $n + 1$ in the `CfdPlatoon` is $t$.

We previously mentioned that SUMO does not currently provide functions for obtaining a vehicle's successors. The above conditions mean that, when we consider a vehicle:

- In the best case it is at the platoon's tail, thus we can obtain all the preceding vehicles and effectively model its drag coefficient;
- In the worst case it is the platoon's leader, thus we have no information about the succeeding vehicles and cannot model its drag coefficient.
- In the average case it is in the middle, thus we can only use partial information to reconstruct the platoon it is part of.

For example, suppose that the user provide data about two platoons with three cars, the first having, in order, a truck and two cars, and the second having a truck, a car and a truck. Suppose that the user simulate the second platoon. Consider the car in the middle: it cannot be known whether it is succeeded by a car or a truck, thus it is impossible to match a platoon with certainty. To solve the issue, we decided to disallow data about platoons whose vehicles differ in number and in types, meaning that we programmed SUMO to abort if it encounters such lists. In the future, if a function for obtaining a vehicle's successor becomes available, we can update our model and allow such specifications.

The function works as follows. It separates compatible `CfdPlatoons` in two different groups: those with, respectively, a shorter and a longer inter-vehicle distance. If one of them is empty, it returns 0% change. Otherwise, it sorts the two groups by distance. Let `S` and `L` be, respectively, the `CfdPlatoons` of the first and second group with the shortest inter-vehicle distance. Let $d_{\texttt{S}}$ and $d_{\texttt{L}}$ be, respectively, `S`'s and `L`'s inter-vehicle distances. Let $c_{\texttt{S}}$ and $c_{\texttt{L}}$ be, respectively, `S`'s and `L`'s $n$+1-th vehicles' percentages of air drag change. Let $d$ be the provided inter-vehicle distance. The function estimates (and returns) the percentage of reduction in air drag with the following linear interpolation:

$$\frac{c_{\texttt{S}}(d_{\texttt{L}} - d) + c_{\texttt{L}}(d - d_{\texttt{S}})}{d_{\texttt{L}} - d_{\texttt{S}}} \tag{5.2}$$

### 5.2.4 A new SUMO device

SUMO devices [10] are containers for functionality and data that can be added to a vehicle. When a traffic simulation is running and a vehicle with a device moves, the device gets notified and carries out some planned activity. There exist devices for the measurement of the average speed, emissions and

more. Devices can generate output and are accessible with the `TraCI` interface. From a programmer's point of view, they are C++ classes.

We have created a new device, called `Slipstream`, that keeps track of a vehicle's drag coefficient. It contains the following variables:

- `myRefDragCoefficient`: It is the drag coefficient that the vehicle takes when driving in isolation: in other words, it is the reference value and does not change over the course of the simulation. The user must specify it when defining the vehicle's type (see section 4.1).
- `myDragCoefficient`: It is the drag coefficient that the vehicle takes when driving inside a platoon.
- `precedingVehicles`: It is a vector that contains references to the vehicle's predecessors on the lane, and is ordered from the closest to the furthest one.
- `distances`: It is a vector that contains the distance that each of the `precedingVehicles` has from its own predecessor.

When notified upon vehicle movement, the device performs the following actions:

1. It updates the `precedingVehicles` vector, making use of existing SUMO's functions. After the update, the vector is such that each vehicle is not further than 20 m from its own predecessor, and the last vehicle is not further than 120 m from the vehicle holding the device.
2. It verifies that each value in the `distances` vector has the same rounded value. This enforces the simplification that the vehicles are equally distanced. If this is not the case, it sets `myDragCoefficient` to `myRefDragCoefficient` and returns.
3. It invokes the function for the estimation of drag coefficient change. It does so by passing the vehicle's type, a vector containing the types of the vehicles in the `precedingVehicles` vector and the average value of the `distances` vector.
4. When the percentage is returned, it sets `myDragCoefficient` to `myRefDragCoefficient` minus the percentage of that value.

### 5.2.5 Updates to existing SUMO code

In section 4.1.1 we mentioned that SUMO's energy consumption model for electric vehicles and PLEXE's vehicle dynamics model consider a vehicle's drag coefficient to be constant. We updated them so that, when they are about to calculate the quantities of interest, they first check whether the vehicle is equipped with a Slipstream device and, if so, they use its drag coefficient. This required to apply the following logic:

```
1 MSDevice_Slipstream* slipstream =
2   dynamic_cast<MSDevice_Slipstream*>(veh.getDevice(typeid(MSDevice_Slipstream)));
3 if (slipstream != nullptr) {
4     /* Use the device's drag coefficient */
5 }
```

## 5.3 Proof of concept

We performed a simulation that showcases the work of this thesis. The simulation is set in a stretch of freeway and concerns two vehicles having realistic dimensions: the leading one is a truck, the second one is a Tesla Model S P85, an electric car whose properties can be found at [11] and [12]. The vehicles, initially, drive at 90 km/h and are 50 m apart. The Tesla starts to accelerate and joins the truck at 5 meters of distance, forming a minimal platoon.

We repeated the simulation twice: in the second one the Tesla is equipped with Slipstream device. The goal was to compare how the its energy consumption differed.
We started by defining the following vehicle types:

```
1  <vType id="Truck"
2    vClass="trailer"
3    guiShape="truck/semitrailer"
4
5    minGap="0"
6    lanesCount="4"
7    carFollowModel="CC">
8  </vType>
```

Listing 5.1: Vehicle type for the truck. Lines 5 to 7 enable PLEXE's platooning models.

```
1  <vType id="TeslaSlip"
2    length="4.971"
3    width="1.963"
4    height="1.445"
5    accel="4.4"
6    decel="5"
7    maxSpeed="58.33"
8
9    minGap="0"
10   lanesCount="4"
11   carFollowModel="CC">
12
13   <param key="has.battery.device" value="true"/>
14   <param key="maximumBatteryCapacity" value="85000"/>
15   <param key="maximumPower" value="350000"/>
16   <param key="vehicleMass" value="2112"/>
17   <param key="frontSurfaceArea" value="2.34"/>
18   <param key="airDragCoefficient" value="0.24"/>
19   <param key="internalMomentOfInertia" value="0"/>
20   <param key="radialDragCoefficient" value="0"/>
21   <param key="rollDragCoefficient" value="0"/>
22   <param key="constantPowerIntake" value="0"/>
23   <param key="propulsionEfficiency" value="0.9"/>
24   <param key="recuperationEfficiency" value="0"/>
25   <param key="stoppingTreshold" value="0.1"/>
26
27   <param key="has.slipstream.device" value="true"/>
28   <param key="dragCoefficient" value="0.24"/>
29 </vType>
```

Listing 5.2: Vehicle type for the Tesla. Lines 2 to 7 define custom properties. Lines 9 to 11 enable PLEXE's platooning models. Lines 13 to 25 define the Battery device (parameters set to 0 are not used in the calculations). Lines 29 and 30 define the Slipstream device, which can be enabled or disabled with attribute `value`.

We defined the data file, which is required for the second simulation:

```
1  # Platoon Nr. 1
2  # vehicle types
3  Truck TeslaSlip
4  # distances
5  0 2.5
6  # air drag change
7  -4 -50
8  # Platoon Nr. 2
9  Truck TeslaSlip
10 0 5
11 -2 -45
12 # Platoon Nr. 3
13 Truck TeslaSlip
14 0 7.5
15 -1 -36
16 # Platoon Nr. 4
17 Truck TeslaSlip
18 0 10.
19 0 -23
```

```
20  # Platoon Nr. 5
21  Truck TeslaSlip
22  0 15.
23  0 -12
```

Listing 5.3: Data file containing the drag coefficient changes of the vehicles, to be used by the API.

We compiled SUMO with all debug directives defined so that, when the second simulation is run, it produces some output explaining the `Slipstream` device and API logic:

```
1  device 'slipstream_v1' notifyMove
2  Preceding vehicles:
3  [6.13 m] v0
4  getDragCoefficient
5  Vehicle's type: Tesla
6  Preceding vehicles' types: Truck
7  Inter-vehicle distance: 6.13
8  Compatible platoons with longer inter-vehicle distances:
9  Truck, -1.00% [ 7.50 m ] Tesla, -36.00%
10 Truck, 0.00% [ 10.00 m ] Tesla, -23.00%
11 Truck, 0.00% [ 15.00 m ] Tesla, -12.00%
12 Compatible platoons with shorter inter-vehicle distances:
13 Truck, -4.00% [ 2.50 m ] Tesla, -50.00%
14 Truck, -2.00% [ 5.00 m ] Tesla, -45.00%
15 [Interpolated values]:
16 7.50m [6.13m] 5.00m
17 -36.00% [-40.92%] -45.00%
18 Drag coefficient: 0.24 -> 0.14
```

Listing 5.4: An excerpt of the output produced by the second simulation. Lines 1 to 3 and 18 are produced by the `Slipstream` device. Lines 4 to 17 are produced by the API's `getDragCoefficientReduction` function.

Figure 5.10 depicts the Tesla's energy consumption. We can observe that, when Slipstream device is used, battery discharges less rapidly.

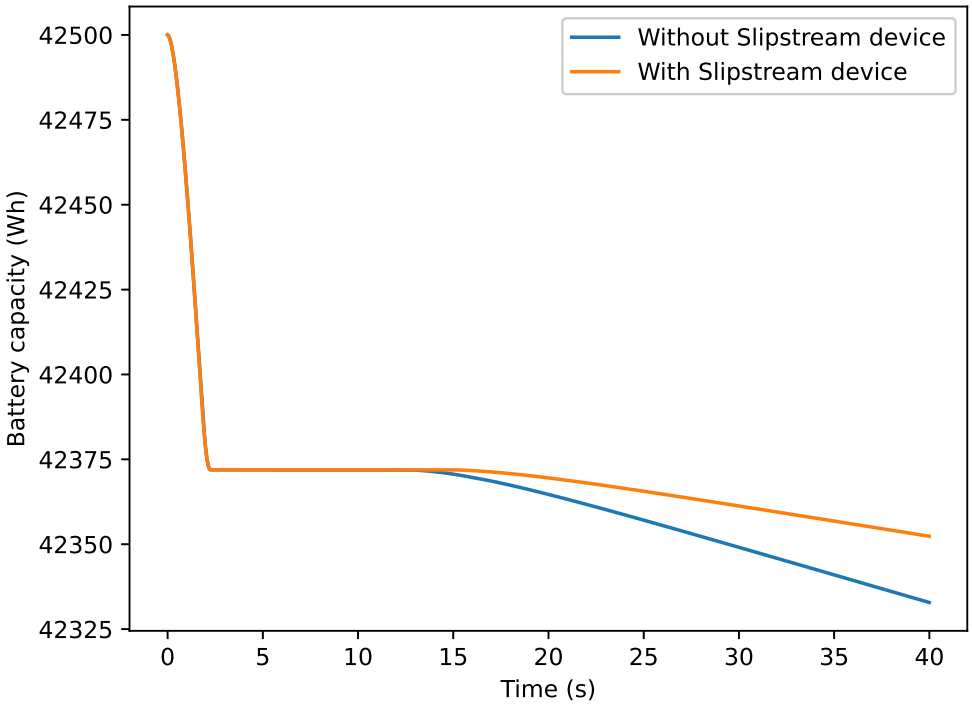



Figure 5.10: Battery consumption of an electric vehicle joining its predecessor on the lane.

# 6 Conclusions

As far as we are aware, no open source traffic simulator models the air drag changes that platooning vehicles are subject to. We focused on the SUMO traffic simulator, which considers a vehicle's drag coefficient to be constant. We added a model that, under certain conditions, is capable of updating a platooning the vehicle's drag coefficient. The model is basic but new and, as such, may represent a first, small step towards sophisticated ones.

When we started working on this thesis we were not familiar with CFD concepts, terminology and tools, which we had to learn without someone's formal guidance. The process involved interacting with the online CFD community and discussion forums, on which simplified clarifications were provided.

We performed OpenFOAM simulations about a vehicle with simplified geometry, called Ahmed body, both in isolation and in platoons with different distances. We observed that there is a downward trend in drag as the distances are shortened. However, some anomalies are present, as certain arrangements cause increase in drag. On one side, this proves that platooning is a legitimate strategy for reducing net drag; on the other, it suggests that certain vehicles have increased drag, thus consume more fuel.

We created a GUI application to automate the setup of new OpenFOAM simulations about Ahmed bodies in platoons, which should accelerate the CFD process for obtaining new data.

## 6.1 Future work

There are two prerequisites to further research. First, we need to add a new function in SUMO for obtaining a vehicle's successor. This lack of functionality currently poses some limits on our model, which are described in section 5.2.3. Second, we need to answer the following question: Do Ahmed bodies in platoons retain sufficient features of flow around real cars to allow proper aerodynamic analyses? If not, we must simulate real vehicles, with higher computational costs associated.

Also pertaining SUMO, we can observe that the use of `Slipstream` devices causes redundant data. In fact, each `Slipstream` device maintains own vectors for saving the preceding vehicles and their distances. Shared data structures could perhaps solve this problem.

When considering future work, some scenarios come to our mind. We could partner with a competent CFD group and organize simulation campaigns. This would allow us to simulate more vehicle types together, such as trucks and cars, much in the same fashion as what done in [16]. Moreover, this would allow us to fully validate the simulations.

If such a partnership produces abundant and reliable data, we could start designing a sophisticated model, perhaps with the use of Machine Learning techniques.

# Bibliography


[1] Ahmed body's validation case. https://www.simscale.com/docs/validation-cases/aerodynamics-flow-around-the-ahmed-body/. Last access 11/07/2020.

[2] Fiat Panda. https://www.fiat.ch/content/dam/fiat/ch/pricelist/Panda-IT.pdf. Last access 11/07/2020.

[3] Finite Volume Method: A Crash introduction. http://www.wolfdynamics.com/wiki/fvm_crash_intro.pdf. Last access 11/07/2020.

[4] GUI: OpenFOAM case builder for platoons. https://github.com/andreastedile/openfoam-platoons.

[5] GUI: $y^+$ Histogram. https://github.com/andreastedile/yplushistogram.

[6] OnShape. https://www.onshape.com/. Last access 11/07/2020.

[7] OpenFOAM. https://openfoam.com/. Last access 11/07/2020.

[8] Paraview. https://www.paraview.org/. Last access 11/07/2020.

[9] SUMO. https://openfoam.com/. Last access 11/07/2020.

[10] SUMO devices. https://sumo.dlr.de/docs/Developer/How_To/Device.html. Last access 11/07/2020.

[11] Tesla Model S P85. https://evcompare.io/cars/tesla/tesla_model_s_p85/. Last access 11/07/2020.

[12] Tesla Model S P85: drag coefficient and frontal area. https://www.tesla.com/sites/default/files/blog_attachments/the-slipperiest-car-on-the-road.pdf. Last access 11/07/2020.

[13] Wall functions and $y^+$. https://www.simscale.com/forum/t/what-is-y-yplus/. Last access 11/07/2020.

[14] S.R. Ahmed, G. Ramm, and G. Faltin. Some salient features of the time-averaged ground vehicle wake. In *SAE Technical Paper*. SAE International, 02 1984.

[15] R. J. Blokpoel, D. Krajzewicz, and R. Nippold. Unambiguous metrics for evaluation of traffic networks. In *13th International IEEE Conference on Intelligent Transportation Systems*, pages 1277–1282, 2010.

[16] Arturo Dávila and Mario Nombela. Sartre - safe road trains for the environment reducing fuel consumption through lower aerodynamic drag coefficient. In *SAE Technical Paper*. SAE International, 10 2011.

[17] Wolf-Heinrich Hucho. Aerodynamics of road vehicles. *SAE International*, 1986.

[18] Christiana L Katsoulos. An experimental study on drag reduction of aftermarket additions on an suv.

[19] Daniel Krajzewicz, Michael Behrisch, Peter Wagner, Raphael Luz, and Mario Krumnow. Second generation of pollutant emission models for sumo. In Michael Behrisch and Melanie Weber, editors, *Modeling Mobility with Open Data*, pages 203–221, Cham, 2015. Springer International Publishing.

[20] Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. Recent Development and Applications of SUMO - Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements*, 5(3&4):128–138, December 2012.

[21] Stefan Krauß. *Microscopic modeling of traffic flow: Investigation of collision free vehicle dynamics*. PhD thesis, 1998.

[22] Tamás Kurczveil, Pablo Álvarez López, and Eckehard Schnieder. Implementation of an energy model and a charging infrastructure in sumo, 2014.

[23] Tomislav Maric, Jens Hopken, and Kyle Mooney. *The OpenFOAM technology primer*. Sourceflux, 2014.

[24] Riccardo Pagliarella. *On the aerodynamic performance of automotive vehicle platoons featuring pre and post-critical leading forms*. PhD thesis, RMIT University, 2009.

[25] Michele Segata. *Safe and efficient communication protocols for platooning control*. PhD thesis, University of Trento, 2016.

[26] Michele Segata. Platooning in SUMO: an open source implementation. In *SUMO User Conference*, pages 51–62, 2017.

[27] Henk Kaarle Versteeg and Weeratunge Malalasekera. *An introduction to computational fluid dynamics: the finite volume method*. Pearson education, 2007.

# Attachment A Drag coefficients data

| Distance [m] | v1 | v2 |
|---|---|---|
| 2.5 | 0.222 | 0.34 |
| 3.5 | 0.262 | 0.304 |
| 5 | 0.286 | 0.303 |
| 6 | 0.293 | 0.286 |
| 7.5 | 0.298 | 0.291 |
| 10 | 0.301 | 0.292 |
| 15 | 0.303 | 0.295 |

(a) Drag coefficients

| Distance [m] | v1% | v2% | Net% |
|---|---|---|---|
| 2.5 | -26.7 | +12.2 | -14.5 |
| 2.5 | -13.5 | +0.3 | -13.2 |
| 5 | -5.6 | - | -5.6 |
| 6 | -3.3 | -5.6 | -8.9 |
| 7.5 | -1.7 | -4 | -5.7 |
| 10 | -0.7 | -3.6 | -4.3 |
| 15 | - | -2.6 | -2.6 |

(b) % of change

Table A.1: Two Ahmed bodies in platoon.

| Distance [m] | v1 | v2 | v3 |
|---|---|---|---|
| 2.5 | 0.219 | 0.269 | 0.326 |
| 3.5 | 0.26 | 0.268 | 0.317 |
| 5 | 0.285 | 0.287 | 0.295 |
| 6 | 0.292 | 0.276 | 0.294 |
| 7.5 | 0.298 | 0.286 | 0.294 |
| 10 | 0.301 | 0.29 | 0.294 |
| 15 | 0.303 | 0.295 | 0.291 |

(a) Drag coefficients

| Distance [m] | v1% | v2% | v3% | Net% |
|---|---|---|---|---|
| 2.5 | -27.7 | -11.2 | +7.6 | -31.3 |
| 3.5 | -14.2 | -11.6 | +4.6 | -21.2 |
| 5 | -5.9 | -5.3 | -2.6 | -13.8 |
| 6 | -3.6 | -8.9 | -3 | -15.5 |
| 7.5 | -1.7 | -5.6 | -3 | -10.3 |
| 10 | -0.7 | -4.3 | -3 | -8 |
| 15 | - | -2.6 | -4 | -6.6 |

(b) % of change

Table A.2: Three Ahmed bodies in platoon.

| Distance [m] | v1 | v2 | v3 | v4 |
|---|---|---|---|---|
| 2.5 | 0.219 | 0.267 | 0.258 | 0.332 |
| 3.5 | 0.26 | 0.266 | 0.28 | 0.315 |
| 5 | 0.285 | 0.287 | 0.279 | 0.287 |
| 6 | 0.292 | 0.275 | 0.284 | 0.294 |
| 7.5 | 0.297 | 0.285 | 0.289 | 0.291 |
| 10 | 0.301 | 0.29 | 0.292 | 0.294 |
| 15 | 0.303 | 0.294 | 0.29 | 0.292 |

(a) Drag coefficients

| Distance [m] | v1% | v2% | v3% | v4% | Net% |
|---|---|---|---|---|---|
| 2.5 | -27.7 | -11.9 | -14.9 | +9.6 | -44.9 |
| 3.5 | -14.2 | -12.2 | -7.6 | +4 | -30 |
| 5 | -5.9 | -5.3 | -7.9 | -5.3 | -24.4 |
| 6 | -3.6 | -9.2 | -6.3 | -3 | -22.1 |
| 7.5 | -2 | -5.9 | -4.6 | -4 | -16.5 |
| 10 | -0.7 | -4.3 | -3.6 | -3 | -11.6 |
| 15 | - | -3 | -4.3 | -3.6 | -10.9 |

(b) % of change

Table A.3: Four Ahmed bodies in platoon.

| Distance [m] | v1 | v2 | v3 | v4 | v5 |
|---|---|---|---|---|---|
| 2.5 | 0.219 | 0.266 | 0.255 | 0.262 | 0.339 |
| 3.5 | 0.26 | 0.266 | 0.279 | 0.278 | 0.305 |
| 5 | 0.285 | 0.285 | 0.287 | 0.281 | 0.289 |
| 6 | 0.292 | 0.275 | 0.283 | 0.284 | 0.297 |
| 7.5 | 0.298 | 0.284 | 0.289 | 0.303 | 0.299 |
| 10 | 0.301 | 0.289 | 0.304 | 0.31 | 0.296 |
| 15 | 0.303 | 0.293 | 0.303 | 0.308 | 0.298 |

(a) Drag coefficients

| Distance [m] | v1% | v2% | v3% | v4% | v5% | Net% |
|---|---|---|---|---|---|---|
| 2.5 | -27.7 | -12.2 | -15.8 | -13.5 | +11.9 | -57.3 |
| 3.5 | -14.2 | -12.5 | -7.9 | -8.3 | +0.7 | -42.2 |
| 5 | -5.9 | -5.9 | -5.3 | -7.3 | -4.6 | -29 |
| 6 | -3.6 | -9.2 | -6.6 | -6.3 | -2 | -27.7 |
| 7.5 | -1.7 | -6.3 | -4.6 | - | -1.3 | -14 |
| 10 | -0.7 | -4.6 | +0.3 | +2.3 | -2.3 | -5 |
| 15 | - | -3.3 | - | +1.7 | -1.7 | -3.3 |

(b) % of change

Table A.4: Five Ahmed bodies in platoon.

# Attachment B GUI applications

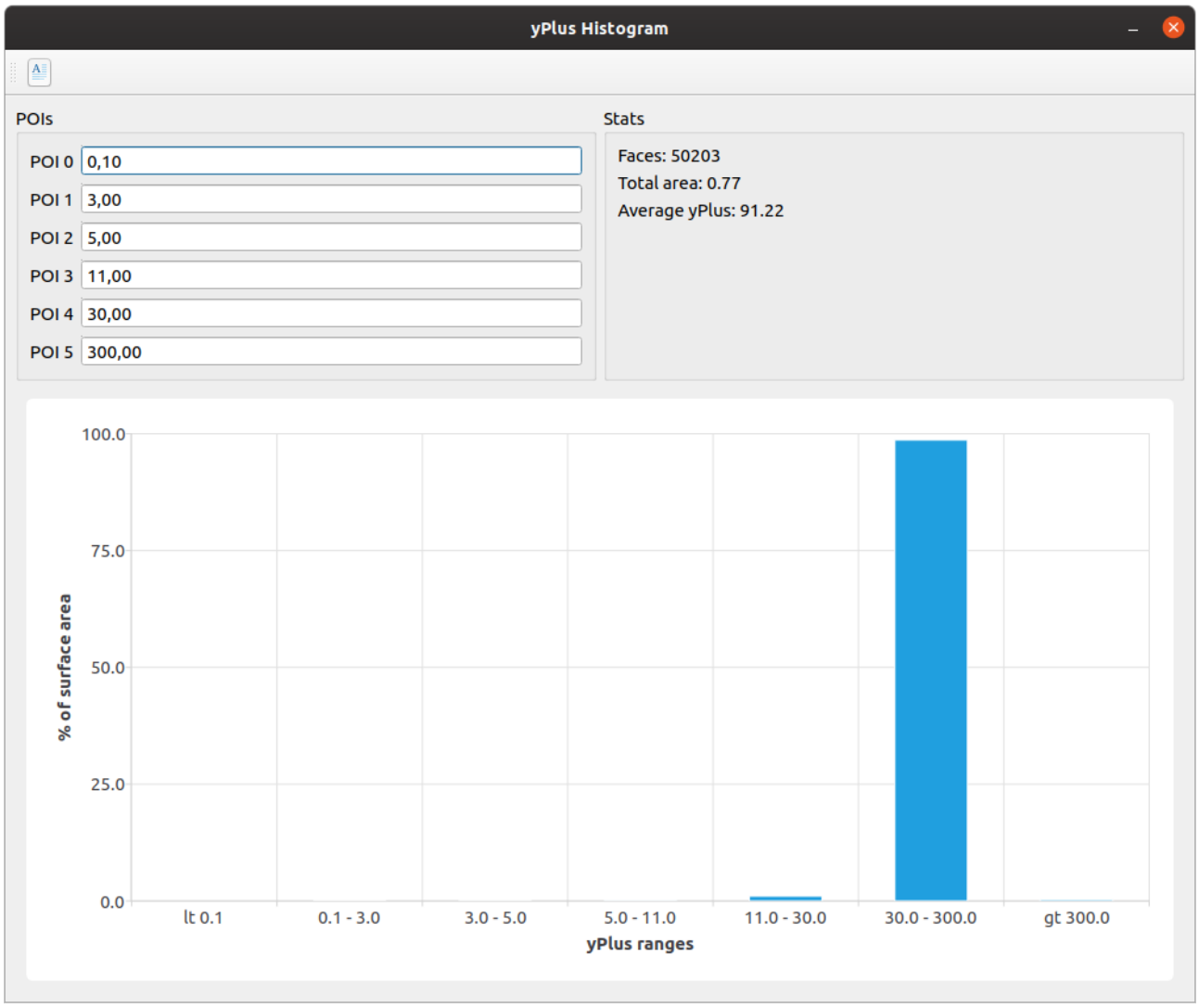
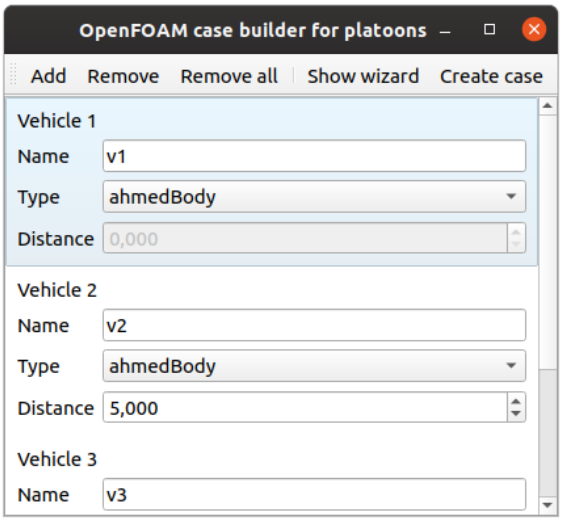
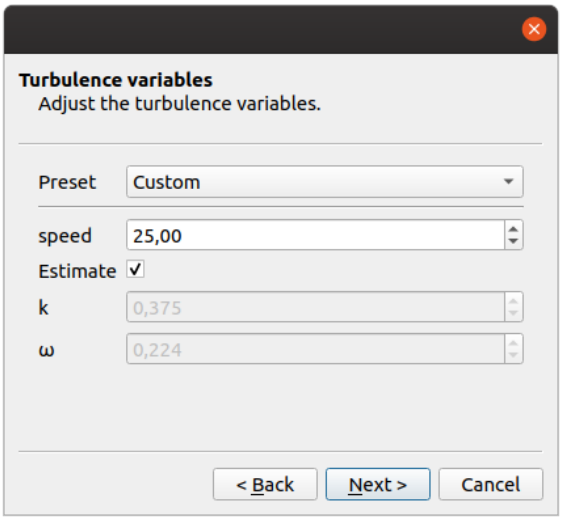


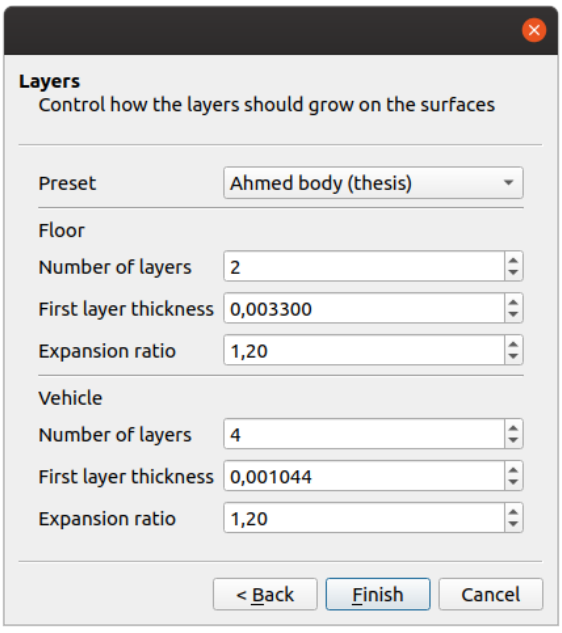

Figure B.1: $y^+$ histogram.




(a) Main window for selecting vehicles and distances




(b) Wizard page for selecting turbulence parameters




(c) Wizard page for selecting turbulence parameters

Figure B.2: Some screenshots of the OpenFOAM case builder for platoons.