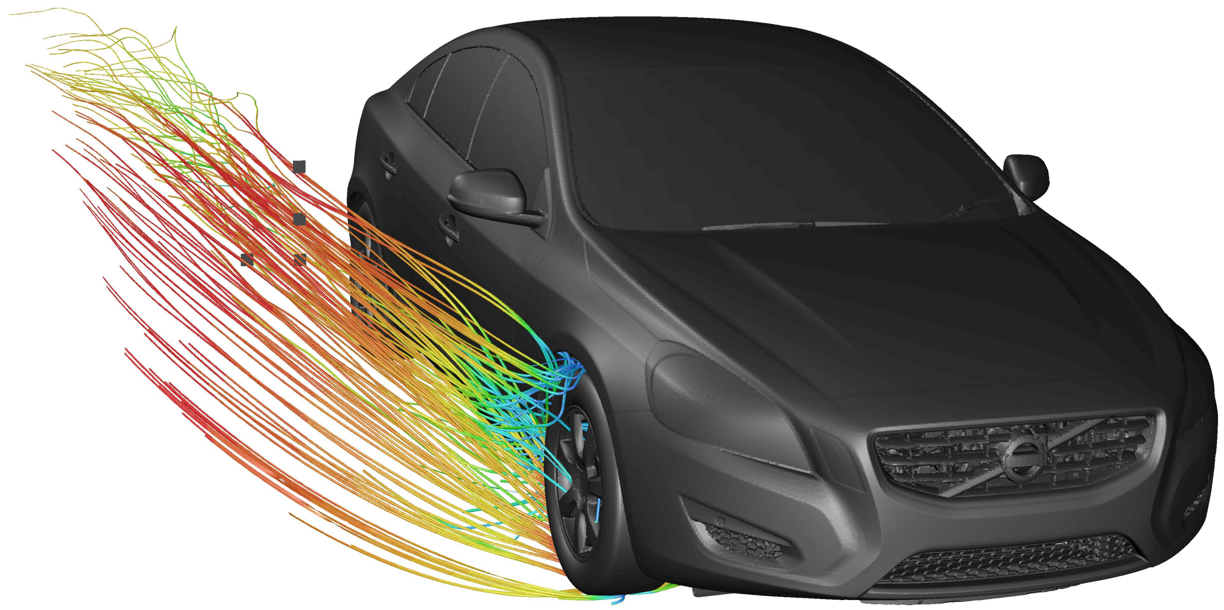
CHALMERS



Self Generated Snow Dust

Trajectory prediction of snow particles generated from the car

Master's thesis in Applied Mechanics

JIAN TAN

Department of Applied Mechanics
Division of Fluid Mechanics
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2014

Master's thesis 2014:81

MASTER'S THESIS IN APPLIED MECHANICS

Self Generated Snow Dust

Trajectory prediction of snow particles generated from the car

JIAN TAN

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

Göteborg, Sweden 2014

Self Generated Snow Dust
Trajectory prediction of snow particles generated from the car
JIAN TAN

Master's thesis 2014:81
ISSN 1652-8557
Department of Applied Mechanics
Division of Fluid Mechanics
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: +46 (0)31-772 1000

Cover:
Particle trajectories of generated snow dust from Volvo S60 under 50km/h driving speed. Velocity of discrete particles varies from approximately 0.05m/s to 14.7m/s.

Chalmers Reproservice
Göteborg, Sweden 2014

Self Generated Snow Dust
Trajectory prediction of snow particles generated from the car
Master's thesis in Applied Mechanics
JIAN TAN
Department of Applied Mechanics
Division of Fluid Mechanics
Chalmers University of Technology

Abstract

When a car is running on a snowy road, the snow on the ground can be lifted up as snow particles and these lifted particles can flow around the car. In this project, Computational Fluid Dynamics(CFD) method will be applied to study, simulate and replicate this specific phenomena. With the help of the test work, which has been done previously at Jokkmokk Proving Ground(JPG), there is a certain amount of experimental test data available, including precipitation of particles at different location, video logs and other environment information. The precipitation data will be the goal for the numerical simulation to reach, which is a validation of this method development. The numerical simulation to predict the particle trajectory and distribution of precipitation is done using CFD technique with multiphase flow approach. Both the experimental data and numerical predictions are presented, interpreted and discussed. It turns out that this method can partly replicate the experimental data, while there are certain specific phenomena that cannot be simulated or reflected from the numerical prediction.

This report begins with introduction of problem and aim. The implementation is described with motivations after the theory section. Results with discussions are presented after the implementation method. Possible improvements and future work are also recommended.

Keywords: CFD, Multihpase, Particle Trajectory, Contamination, Ground Vehicle, Snow Dust, Winter Driving

Acknowledgements

It is my honor and an unique privilege to have this opportunity to participate this very fascinating project in the Contamination group at Volvo Car Corporation, among a group of very talented, experienced and energetic engineers.

I want to express my greatest appreciation to my supervisors, **Carl Andersson** and **Dragos Moroianu**. Calle has been always very supported in many ways in my everyday life since the first day I came, both work and joy; while Dragos is the key to this project and makes sure I am always on the right track.

I am also very thankful both for my professor and examiner **Lars Davidson** at Chalmers Tekniska Högskola and for **Dr. Jonas Ask** at VCC, for helping me with the contact which leads to an opportunity to have this exciting experience.

I want to thank **Anders Björtin**, manager of Contamination Group, for his genuine concern for the project.

A special appreciation to **Joacim Larsson**, **Jonny Jonsson** and **Jörgen Frisfors**, for the experimental data, which plays a very important role in this project.

At last I want to thank my parents and relatives, for their years of unconditional understanding and support.

Nomenclature

Abbreviations

CAE	Computer Aided Engineering
CFD	Computational Fluid Dynamics
DRW	Discrete Random Walk
JPG	Jokkmokk Proving Ground
LES	Large Eddy Simulation
MRF	Multi Reference Frame
PID	Property ID
RANS	Reynolds-Averaged Navier-Stokes
SGS	Subgrid-Scale
SRS	Scale-Resolving Simulation
SST	Shear-Stress Transport
UDF	User Defined Function
VCC	Volvo Car Corporation

Constants

a	proportion factor for the friction velocity
ρ_a	density of air at normal condition
d	equivalent diameter
r	equivalent particle radius
α	shape parameter of Gamma distribution
β	scale parameter of Gamma distribution
a^*	damping coefficient for k-ω turbulence model
C_μ	constant for standard k-ε turbulence model
C_D	drag coefficient for particles
St	Stokes number
ρ_p	density for the discrete particle
d_p	diameter for the discrete particle
l	turbulence length scale at boundaries
L	characteristic length, diameter of the wind tunnel

Variables

v_i	velocity of continuous flow in i direction
x_i	coordinates
t	time variable
p	static pressure
f_i	external body force for discrete particles
T	temperature for continuous flow field
Φ	dissipation term for energy equation
c_p	heat capacity for energy equation
k	effective conductivity for energy equation
k	turbulent kinetic energy in k-ε and k-ω turbulence model
ε, ω	turbulent dissipation rate for k-ε and k-ω turbulence model
u_i	velocity for flow in i direction
μ_t	dynamic turbulence viscosity
S_{ij}	strain rate tensor
$\mathbf{u}$	velocity for the continuous flow field
$\mathbf{u}_p$	velocity for the discrete particle
$\dot{m}_p$	mass flow rate for discrete particles
u'_i	velocity fluctuations in i direction
u_{avg}	mean flow velocity

Contents

Abstract **i**

Acknowledgements **iii**

Nomenclature **v**

Contents **vii**

1 Introduction **3**
1.1 Background . . . 3
1.2 Problem formulation . . . 3
1.3 Aim . . . 3
1.4 Limitations . . . 3

2 Theory **5**
2.1 Creep, Saltation and Suspension . . . 5
2.2 Physical properties of snow particle . . . 6
2.2.1 Particle shape . . . 6
2.2.2 Particle size distribution . . . 7
2.2.3 Approximation of particle mean radius . . . 7
2.2.4 Signal processing . . . 9
2.3 Mesh quality: Skewness . . . 9
2.4 Governing equations for the flow . . . 9
2.4.1 The continuity equation . . . 9
2.4.2 The momentum equation . . . 9
2.4.3 The energy equation . . . 10
2.5 Turbulence model . . . 10
2.5.1 Reynolds Averaged Navier Stokes(RANS) model . . . 10
2.6 Multiphase model with Eulerian-Lagrangian approach . . . 11
2.6.1 Particle motion . . . 11
2.6.2 Mass, Heat and Momentum exchange . . . 12
2.6.3 Turbulent dispersion . . . 13
2.7 Boundary conditions . . . 13
2.7.1 Turbulence Properties for Velocity Inlet and Pressure Outlet . . . 13
2.7.2 Wall boundary condition . . . 14
2.7.3 Discrete particles . . . 14

3 Method **15**
3.1 Processed measurement data . . . 15
3.2 Properties of snow particles . . . 16
3.2.1 Density . . . 16
3.2.2 Particle size . . . 16
3.3 Mesh generation . . . 16
3.3.1 **ANSA** pre-processing: surface mesh . . . 17
3.3.2 **ANSA** pre-processing: surface smooth . . . 19
3.3.3 Volume mesh generation in **Harpoon** . . . 19
3.4 Various driving speed . . . 24
3.5 Boundary conditions . . . 24
3.5.1 Velocity Inlet for the wind tunnel . . . 24
3.5.2 Pressure Outlet for the wind tunnel . . . 24
3.5.3 Moving boundaries . . . 24
3.5.4 Inelastic reflection between the particles and the wall boundaries . . . 25
3.5.5 Precipitation sensor . . . 25
3.6 Implementation of particle injection . . . 25

3.6.1 Motivation of the implementation 26
3.6.2 Surface injection over the tyre 27
3.7 Single phase 28
3.7.1 Turbulence model 28
3.7.2 Convergence criteria 29
3.8 Multiphase 29
3.8.1 Stochastic Random Walk particle tracking 29
3.9 Verification 29
3.9.1 Varied implementation 29

4 Results **30**
4.1 Results from experimental data 30
4.1.1 50km/h 30
4.1.2 70km/h 30
4.1.3 90km/h 30
4.2 Results from numerical simulation 31
4.3 Differences between experimental and numerical results 34

5 Discussion **37**
5.1 The experimental data 37
5.2 Comparison between the experimental data and the simulated results 38
5.3 Tuning of the implementation 39
5.4 Simplified model 39
5.5 General observations of the steady state flow field 40
5.5.1 Quality of the converged solution 40
5.5.2 Velocity contour 40
5.6 Mesh sensitivity 40
5.6.1 Smooth mesh with target skew 40
5.6.2 Finding volumes on surfaces 41
5.6.3 Refinement boxes for particle trajectories 41
5.7 Particle trajectories 42
5.7.1 Particle force 42
5.7.2 Stochastic Random Walks model 42
5.7.3 Surface injection 42
5.8 Error source 42
5.8.1 Measurement data: huge fluctuations 42
5.8.2 Implementation based on visual observations from video logs: visual inaccuracy 43
5.8.3 Snow condition: various properties 43
5.8.4 Numerical calculation: various simplifications with errors 43

6 Conclusion **44**

7 Future work **45**

Appendices **47**

A PID list of the car model **48**

B Harpoon Configuration file **49**

C Fluent Journal file **50**
C.1 Steady state case 50
C.2 Injection and particle track 54

D User Defined Function **59**
D.1 First approach 59
D.2 Second approach 62

List of Figures

1.1 The mounted sensor on the car. The height of the sensor can be adjusted to reach different locations. The flux distribution with respect to different locations are obtained in such manner. The test was done under VCC at Jokkmokk Proving Ground(JPG) around February, 2014. . . 4
2.1 Three types of particle transportation from the ground. The particles are transported differently based on various drag forces and gravitational forces. 5
2.2 A demonstration for classification of the generated snow particles along the wheel based on a physical understanding and measurement films. 6
2.3 Snow density as a function of ambient temperature. 7
2.4 Irregular shape of snow particle with equivalent diameter $d = \sqrt{ab}$.[5] 8
2.5 An example of a 2-parameter Gamma distribution with $\bar{r} = 450\mu m$ and $\alpha = 2$. 8
2.6 Heat, Mass, and Momentum Transfer between the discrete and continuous phases[12]. 13
3.1 One of the tests data under 50km/h driving speed. The red dash line represents the mean value. 16
3.2 The surface mesh of the entire car in **ANSA**(Surface mesh of the wheels is not shown due to rendering reasons). 17
3.3 The surfaces representing the locations of the precipitation sensor domains. 18
3.4 The original surface representation is separated to two surface mesh representation. 18
3.5 The geometry of the created wind tunnel with the car inside. 20
3.6 A demonstration for the geometry of the refinement boxes besides the right two wheels. 21
3.7 The generated global mesh from **Harpoon** at $y = 0$. 21
3.8 The local mesh around car at $y = 0$. 21
3.9 Cut planes that are perpendicular to y axis to demonstrate and examine if the volume mesh is desirable. 22
3.10 Cut planes that are perpendicular to x axis to demonstrate and examine if the volume mesh is desirable. 23
3.11 Snapshot from films to help defining the injection points and directions. 26
3.12 Patched tyre surface defined in **ANSA**. 26
3.13 Velocity magnitude of v_y along $y-$direction. 27
3.14 Two approach for introducing the velocity component v_y along y-axis. 28
4.1 Distribution of converted mean mass flux with respect to different locations under 50km/h driving speed. 31
4.2 Flux distribution in percentage under 50km/h driving speed. 31
4.3 Distribution of converted mean mass flux with respect to different locations under 70km/h driving speed. 32
4.4 Flux distribution in percentage under 70km/h driving speed. 32
4.5 Distribution of converted mean mass flux with respect to different locations under 90km/h driving speed. 33
4.6 Flux distribution in percentage under 90km/h driving speed. 33
4.7 Comparison of distribution between experimental and numerical results. The driving speed is 50km/h. 34
4.8 Comparison of distribution between experimental and numerical results, bar chart. The driving speed is 50km/h. 35
4.9 Comparison of distribution between experimental and numerical results. The driving speed is 70km/h. 35
4.10 Comparison of distribution between experimental and numerical results, bar chart. The driving speed is 70km/h. 36
4.11 Comparison of distribution between experimental and numerical results. The driving speed is 90km/h. 36
4.12 Comparison of distribution between experimental and numerical results, bar chart. The driving speed is 90km/h. 37
5.1 All experimental data arranged with respect to sensor domains and driving speeds. 38
5.2 The steady state solution under 50km/h driving speed. The red dots represent the cells with a velocity higher than 30m/s. 40
5.3 The velocity contours at different planes and surfaces over the car. 41

5.4 A test case that the precipitation sensor reaches a maximum measuring range(5[V]) under most of the measuring time, because the sensor domain is blocked by the snow. 43

1 Introduction

1.1 Background

Modern personal cars are more and more advanced in different functions, hence can suffer various harsh environments while maintaining demanded performances. Driving on snow ground in winter is one of these extreme conditions that requires the car manufacture offering a complete solution. When a car is running over the snow ground, the snow will be first compressed by the tyre and a certain quantity of snow particles/crystals will be either lifted up from the ground, or attached on the tyre surface and detach later. During these processes, those snow particles can fly up in the air and part of those particles can keep suspended around the car because of the generated turbulent flow around the vehicle. This type of snow dust is referred to the *self generated snow dust.* Under a relatively long driving time period, these snow particles can attach and accumulate on different components of the car and influence their performances. In order to prevent or minimize the effect of this phenomena, various studies and tests have been done by the Contamination group at Volvo Car Corporation(VCC).

1.2 Problem formulation

A car running on the snow ground will generate snow dust particles and these particles are lifted in the air and remain suspended temporally. Those particles process specific trajectories. This can be reflected by measuring the volume flux at specific regions, see Figure 1.1.

1.3 Aim

The aim of this project is to develop a numerical method that can predict the snow particle trajectory with acceptable accuracy, compared with the experimental data. The accuracy is defined as the difference of flux distribution at different sensor domains between the experimental measurements and the numerical simulations.

1.4 Limitations

This project is a method development under certain ranges and scopes, which are listed in the following:

- The measurement test is not included in this project;
- There are a certain amount of factors that are not considered, more discussions will follow in Section 5.4;
- The aim defined above is considered as the first stage of the project, more work can be done if possible within the 20 weeks range.

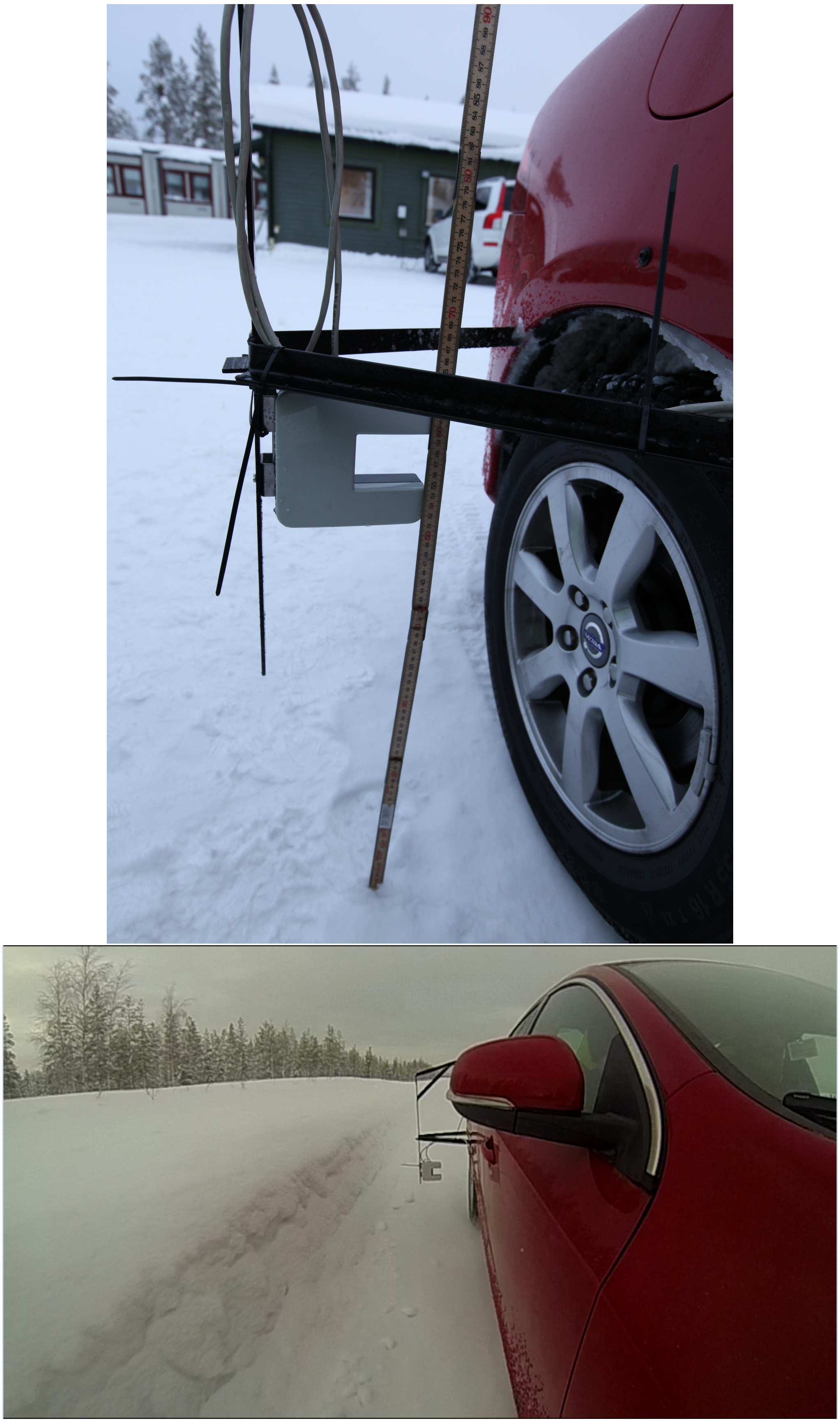


Figure 1.1: *The mounted sensor on the car. The height of the sensor can be adjusted to reach different locations. The flux distribution with respect to different locations are obtained in such manner. The test was done under VCC at Jokkmokk Proving Ground(JPG) around February, 2014.*

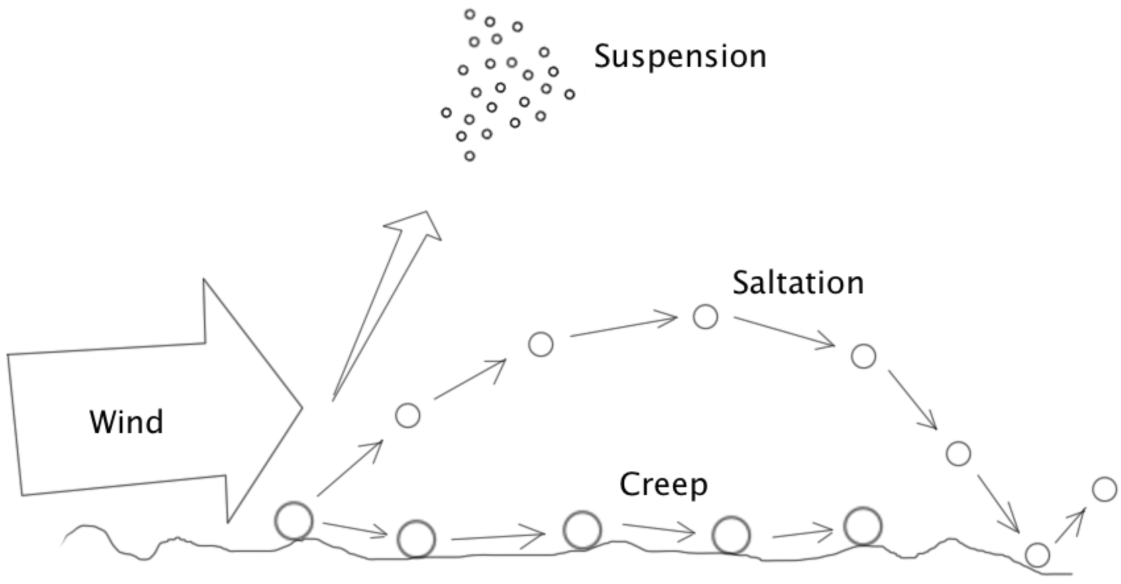


Figure 2.1: *Three types of particle transportation from the ground. The particles are transported differently based on various drag forces and gravitational forces.*

2 Theory

In this chapter, the fundamental principles of both physical properties of the snow (particles) and how the computational simulation is undertaken are introduced. The physical interpretation will be a reliable foundation for the computational implementation.

2.1 Creep, Saltation and Suspension

In order to obtain a better understanding of the movement of snow particles, it is helpful to study how they are transported, which is also relevant to this specific project. The snow particles on the ground can be transported by three types of fundamental movements: creep, saltation and suspension[1]. These three types of movements are geological terms and are shortly described below:

- the creep refers to snow particles rolling on the ground;
- the saltation refers to snow particles that are drifted away from the ground, but settle back after flying over a certain distance over the ground;
- the suspension refers to the particles that are lifted above the ground and suspended in the air for a relatively longer time period;

see Figure 2.1 for illustration.

In this project, snow dust generated from the wheels will have more contribution to the measurements and therefore will be taken into account. The creep process for this situation will not give much contribution, since the particles do not leave the ground in principle. The saltation particles will give very much contribution to the measured data as well since particles can reach the sensor domain no matter if they fall back to the ground or not. Therefore saltation will be taken into account. The suspended particles are also relatively interesting. A sketch based on this physical understanding can be seen in Figure 2.2 for demonstration of the wheel generated snow dust for the situation in this specific project.

There is another type of snow dust that is not generated by the wheel but generated by the turbulence flow from the moving car. When wind blows over the snow(in this case, the turbulence and vortex generated

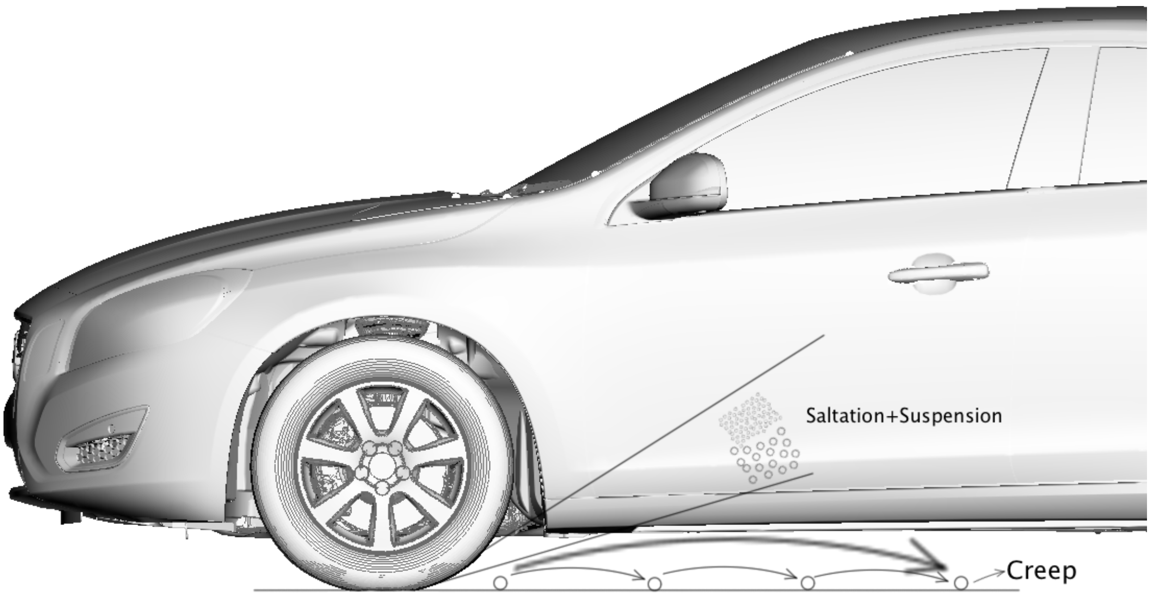


Figure 2.2: *A demonstration for classification of the generated snow particles along the wheel based on a physical understanding and measurement films.*

by the moving car), a certain amount of snow particles will be lifted and transported. This is another source of snow dust generation.

Particle ejection threshold In order to determine the condition when the particle is lifted from the ground due to vortex, a certain criteria has to be defined. In this case, the friction velocity, as a particle injection criteria, is defined in Equation 2.1 below[2]:

$$u_* \propto \sqrt{\frac{\tau_0}{\rho_\mathrm{a}}}, \tag{2.1}$$

with a proportion factor $a \approx 4.16$ suggested by M.Gordon (or without factor a). Here the τ_0 is the fluid shear stress and ρ_a is the density of air. A typical value of this friction velocity when the snow particle is ejected is $\approx 0.25m/s$ for fresh snow and $1m/s$ for wind hardened snow, suggested by T.K. Thiis[3]. However this type of source of snow particles will not be considered or implemented in this project.

2.2 Physical properties of snow particle

In order to implement the most accurate reality into the numerical model, basic physical properties of snow or snow particle have to be understood and defined. Physical parameters such as density, particle size and its distribution are discussed in the following paragraphs.

Density Density of fresh snow bulk can vary from below to above 100[kg/m^3] [1]. Density ρ_s[kg/m^3] of the snow bulk as a function of ambient temperature T_a[°C] can be estimated with an equation developed by J.W.Pomeroy and E.Brun[4] as the following:

$$\rho = 67.9 + 51.25\exp(\frac{T_\mathrm{a}}{2.59}), \quad -10°\mathrm{C} < T_\mathrm{a} < 1°\mathrm{C}. \tag{2.2}$$

Equation 2.2 is plotted in Figure 2.3 and indicates that under $0°C$ the density of snow varies approximately between $70kg/m^3$ and $120kg/m^3$.

2.2.1 Particle shape

In nature the snow particle has irregular shape, with a maximum length a and a maximum width b perpendicular to the length, see Figure 2.4. The equivalent diameter of the snow particle is defined as $d = \sqrt{ab}$[5]. All

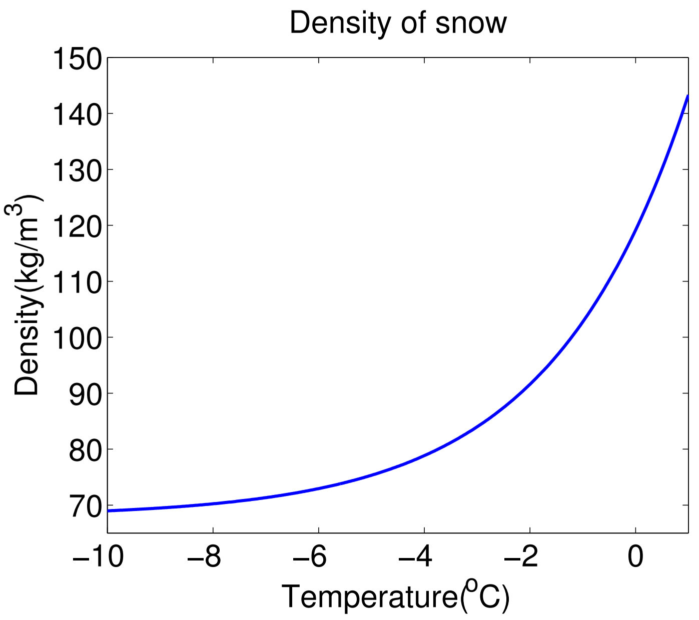


Figure 2.3: *Snow density as a function of ambient temperature.*

diameters or radius regarding to the size of the particles are based on this equivalent diameter in the following discussion.

2.2.2 Particle size distribution

The size distribution of snow particles is fitted by a 2-parameter Gamma distribution, motivated by e.g. M.Gordon[2], R.A. Schmidt[7], etc. M.Gordon formulates the following Gamma distribution[2]:

$$f(r) = \frac{N r^{\alpha-1} \exp(-r/\beta)}{\beta^{\alpha}\Gamma(\alpha)}, \tag{2.3}$$

where N is the total number density, r is the equivalent particle radius ($r = d/2$, where d is the equivalent diameter introduced above), Γ is the Gamma function, α and β are the shape and scale parameters, which can be expressed by mean radius $\bar{r}$ and variance σ: $\alpha = \bar{r}^2/\sigma^2$ and $\beta = \sigma^2/\bar{r}$. The experimentally determined shape parameter α varies from 0.6 to 16.1, motivated by M.Gordon, who also suggests $\alpha = 2$. This leaves to determine the distribution by defining the mean radius $\bar{r}$. M.Gordon suggests that the range of the mean radius $\bar{r}$ is between $36\mu m$ and $144\mu m$[6], while R.A. Schmidt demonstrate a range up to $0.8mm$[7]. There are further discussions regarding how to define the mean radius of particles in the following section.

2.2.3 Approximation of particle mean radius

Based on the measurement given for this project regarding to determine the particle size, only videos are available. Therefore it is rather difficult to give a precis information about snow particle radius or radius range. However, certain approximations of radius (or radius range) based on observations of videos and information of filming devices can be obtained.

Resolution of naked eye It is important to study the resolution of naked eye in order to give a plausible approximation of the radius, since visual observation from videos is the first approach. The maximum angular resolution of the human eye is about 1/60 degree[8]. The Go Pro camera is installed approximately $1m$ away from filming region. Therefore the minimum visible particle has a radius approximately $0.02/360 \cdot 2\pi/2 \approx 0.2mm$. It is assumed that most of the generated snow particles have radius above this value. According to the size distribution introduced above, a reasonable assumption of mean radius $\bar{r}$ can be $\approx 0.45mm$. Such distribution is shown in Figure 2.5. This mean radius will be implemented and adjusted during the simulation process.

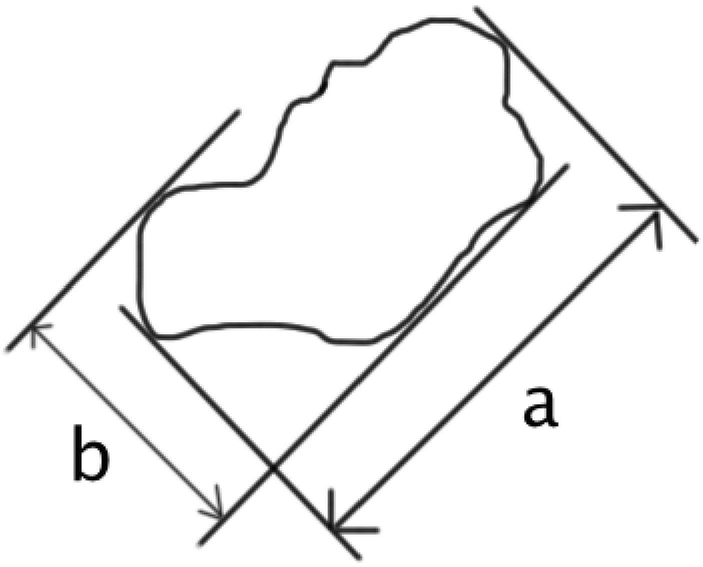


Figure 2.4: *Irregular shape of snow particle with equivalent diameter* $d = \sqrt{ab}$*.[5]*

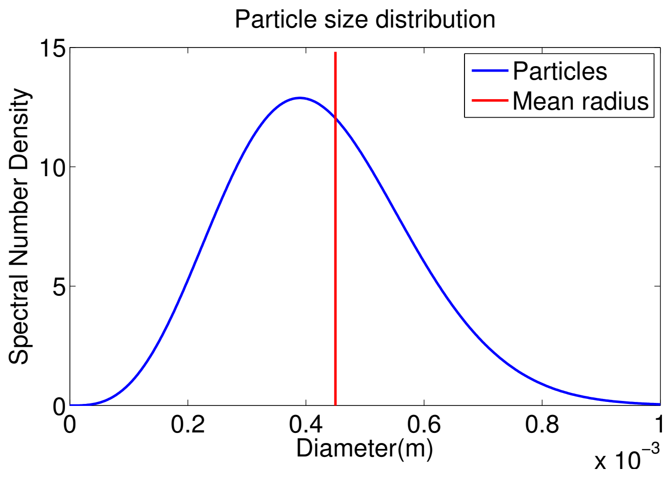


Figure 2.5: *An example of a 2-parameter Gamma distribution with* $\bar{r} = 450\mu m$ *and* $\alpha = 2$.

2.2.4 Signal processing

Precipitation sensor The measurement data is obtained from a *Thies Clima* precipitation sensor[9], with an active sensor surface of $5cm \times 5cm$ and a minimum particle size of $0.2mm$, which is generally valid for the assumption of snow particle size in this project. The output measurement unit is mm/min, which refers to the precipitation of water droplets. Here it is assumed that the snow particles can be detected and measured in the same way the water droplets are measured.

Conversion of measuring unit The output unit is mm/min and it is converted to mas flux of the sensor domain in order to compare with the simulation result and/or adjust the implementation. With the measured output $p[mm/min] = p/60[mm/s]$ and the active sensor surface $25cm^2 = 2500mm^2$, the measured output can be converted to $p \cdot 2500/60[mm^3/s]$. The total mass flux per second is $\dot{m} = p \cdot 2500/60 \cdot \rho_{\text{snow}}[kg \cdot s^{-1}]$. The measurement data can be converted in this way to compare with the simulation data or vice versa. The conversion sets up a relation so that the simulation results and measurements can be compared with.

2.3 Mesh quality: Skewness

Skewness is a term that represents the quality of one individual cell in the mesh. It is the ratio between the shape of the practical cell and the shape of an equilateral cell. For example, skewness for a quadrilateral cell can be defined as the following[10]:

$$\text{Skewness}_{\text{quad}} = \max\left[\frac{\theta_{\max} - 90}{90}, \frac{90 - \theta_{\min}}{90}\right]. \tag{2.4}$$

Therefore, for a hex cell type, all angles should be as closed to 90° as possible. Hence the skewness is near to 0 for a cell with high quality, while a low quality cell with high skewness closed to 1 can cause numerical instabilities.

2.4 Governing equations for the flow

For any flow, the conservation of mass and momentum are solved in order to simulate the flow. For flow with heat transfer, the equation of conservation of energy is solved additionally. There can be other additional governing equations, however these three equations represent the fundamental physical law for any Newtonian flow motion. A basic introduction is given in the following sections in order to demonstrate the mathematical understanding of the flow, where the numerical simulation is based on as well.

2.4.1 The continuity equation

The continuity equation is also called the equation of mass conservation since it is derived based on the fact that the mass of the continuous flow is unchanged within a certain control volume. The continuity equation for any incompressible flow is formulated as the following[11]:

$$\frac{\partial v_i}{\partial x_i} = 0, \tag{2.5}$$

where v_i is the flow velocity, x_i is the location and $i = 1, 2, 3$ for three dimensional space.

2.4.2 The momentum equation

As it is for mass conservation, the momentum is also a conserved physical parameter within the control volume. For any incompressible flow with constant viscosity, the transport equation for momentum, which is derived with the help of the continuity equation, reads[11]:

$$\rho\frac{Dv_i}{Dt} = -\frac{\partial p}{\partial x_i} + \mu\frac{\partial^2 v_i}{\partial x_j \partial x_j} + \rho f_i, \tag{2.6}$$

where t is the time, p is the static pressure, μ is the dynamic viscosity and ρ is the density of the fluid(is a constant for incompressible flow). f_i is the external body force and gravitational force, it can also be defined as forces from other sources. The transport equation for momentum is also called for *Naiver Stokes* equation.

2.4.3 The energy equation

The equation of energy conservation, as the total energy is balanced within the control volume, reads as the following:

$$\rho c_p \frac{DT}{Dt} = \Phi + \frac{\partial}{\partial x_i}\left(k\frac{\partial T}{\partial x_i}\right), \quad \Phi = 2\mu S_{ij}S_{ij}, \tag{2.7}$$

where T is the temperature, Φ is the dissipation term, S_{ij} is the strain-rate tensor, c_p is the heat capacivity and k is the effective conductivity. If the heat conductivity coefficient is constant then term $\frac{\partial T}{\partial x_i}$ can be rewritten to $\alpha\frac{\partial^2 T}{\partial x_i \partial x_i}$, where $\alpha = k/(\rho c_p)$ is the thermal diffusion[11].

2.5 Turbulence model

Turbulence is a key feature for the flow. It is usually transited from laminar flow when the Reynolds number is approximately around 10^3 to 10^4. Turbulence is a common phenomena for flow behaviors, therefore it is not a specific property for different types of fluids as long as the fluid is Newtonian. Turbulence may exist in many flow situations, for example flow moving towards any blunt bodies, boundary layers, etc. There is no specific definition of the turbulence. One common characteristic feature of the turbulence is the irregularity, that the turbulence is chaotic. Turbulence is also always three dimensional in space and unsteady. It is assumed that there is a certain amount of eddies with different length scales and life time spans. The eddies with relatively larger sizes will give energy to smaller eddies, while the smallest eddies are transformed to thermal energy. Thus the turbulence is also dissipative and this energy transform is called as Cascade process. Since the sizes of the eddies are much larger than the molecular scale, the turbulent flow is considered continuous as well.[11]

In computational fluid dynamics, the velocity of turbulent flow is usually treated as a time-averaged part $\bar{v}_i$ and a fluctuating part $v'_i(v_i = \bar{v}_i + v'_i)$. This is called the Reynolds averaging technique, first proposed by Osborne Reynolds. Turbulence models that use the Naiver Stokes equations that are treated by this averaging techniques are called the *Reynolds Averaged Navier Stokes* model.

2.5.1 Reynolds Averaged Navier Stokes(RANS) model

There is a certain amount of RANS turbulence models that are based on turbulence kinetic energy k and certain specific dissipation rate ε or ω. Different models have their strong and weak points.

k-ε model The standard k-ε model is a 2-equations eddy-viscosity model based on transport of turbulent kinetic energy k and dissipation rate ε, assuming that the flow is fully turbulent. It has been widely used in both academic and practical engineering branches. In **Fluent** it is formulated as Equation 2.8 and 2.9 in the following[14]:

$$\frac{\partial}{\partial t}(\rho k) + \frac{\partial}{\partial x_i}(\rho k u_i) = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] + G_k + G_b - \rho\varepsilon - Y_M + S_k, \tag{2.8}$$

and

$$\frac{\partial}{\partial t}(\rho \varepsilon) + \frac{\partial}{\partial x_i}(\rho \varepsilon u_i) = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_\varepsilon}\right)\frac{\partial \varepsilon}{\partial x_j}\right] + C_{1\varepsilon}\frac{\varepsilon}{k}(G_k + C_{3\varepsilon} + G_b) - C_{2\varepsilon}\rho\frac{\varepsilon^2}{k} + S_\varepsilon, \tag{2.9}$$

with constants $C_{1\varepsilon}$, $C_{2\varepsilon}$, $C_\mu(\mu_t = \rho C_\mu k^2/\varepsilon)$, σ_k and σ_ε.
The k-ε model has a relatively poor prediction at the near-wall region. Therefore the k-ω model is developed to have a better performance near the wall.

k-ω model The standard k-ω is a 2-equation model based on the transport of turbulent kinetic energy k and the specific dissipation rate ω. This model considers in low-Reynolds number effects compared with k-ε. In

Fluent it is formulated in Equation 2.10 and 2.11 in the following[14]:

$$\frac{\partial}{\partial t}(\rho k) + \frac{\partial}{\partial x_i}(\rho k u_i) = \frac{\partial}{\partial x_j}\left(\Gamma_k \frac{\partial k}{\partial x_j}\right) + G_k - Y_k + S_k, \tag{2.10}$$

and

$$\frac{\partial}{\partial t}(\rho \omega) + \frac{\partial}{\partial x_i}(\rho \omega u_i) = \frac{\partial}{\partial x_j}\left(\Gamma_\omega \frac{\partial \omega}{\partial x_j}\right) + G_\omega - Y_\omega + S_\omega, \tag{2.11}$$

where $\Gamma_k = \mu + \mu_t/\sigma_k$ and $\Gamma_\omega = \mu + \mu_t/\sigma_\omega$. $\mu_t = a^*\rho k/\omega$ and a^* is the damping coefficient. G is the production term and Y is the dissipation term. A weak point of this model is the sensitivity outside the shear layer. Therefore a turbulence model that beholds the strong point of both k-ε and k-ω model is needed.

Shear-Stress Transport(SST) model The Shear-Stress Transport(SST) k-ω model is a combination between k-ε model and k-ω model so that the model has high accuracy at the near-wall region as the k-ω contributes, meanwhile it also has a free-stream independence in the far field as the k-ε contributes. SST model has also a limitation of the shear stress in the adverse pressure gradient regions[11].

Realizable k-ε model The realizable k-ε model is also based on the transport equation of turbulent kinetic energy k and turbulent dissipation rate ε as the standard k-ε model. However the realizable k-ε model processes a different formulation of turbulent viscosity. The ε equation is derived from an exact equation for the transport of the mean-square vorticity fluctuation[12], which is also different from the standard k-ε equation. These differences improve that the realizable k-ε model is consistent with the physical description of turbulent flows. Neither standard k-ε nor k-ω is consistent with the physics[12]. The k and ε equations in the realizable k-ε model in **Fluent** are formulated as the following[12]:

$$\frac{\partial}{\partial t}(\rho k) + \frac{\partial}{\partial x_j}(\rho k u_j) = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] + G_k + G_b - \rho\varepsilon - Y_M + S_k, \tag{2.12}$$

and

$$\frac{\partial}{\partial t}(\rho \varepsilon) + \frac{\partial}{\partial x_j}(\rho \varepsilon u_j) = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_\varepsilon}\right)\frac{\partial \varepsilon}{\partial x_j}\right] + \rho C_1 S\varepsilon - \rho C_2 \frac{\varepsilon^2}{k + \sqrt{\nu\varepsilon}} + C_{1\varepsilon}\frac{\varepsilon}{k}C_{3\varepsilon}G_b + S_\varepsilon, \tag{2.13}$$

where

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

The modeled constants are: $C_{1\varepsilon} = 1.44$, $C_2 = 1.9$, $\sigma_k = 1.0$ and $\sigma_\varepsilon = 1.2$. Compared with standard k-ε model, the C_μ is no longer a constant. Other constants are different from other turbulence models as well.

2.6 Multiphase model with Eulerian-Lagrangian approach

The Eulerian-Lagrangian approach is usually applied when the particles are relatively discrete and they own a relatively lower volume fraction with respect to the continuous phase. Particles trajectories in the carrier phase can be calculated by following the particles moving path, while the carrier phase is calculated in a usual RANS calculations.

In one way coupling, which is applied in this project, the particles are affected only by the continuous phase. The discrete particles will not affect the continuous flow field. Therefore there is, only mass, heat and momentum exchange from the carrier phase to the discrete particles. There are no interactions between particles either.

2.6.1 Particle motion

In Eularian-Lagrangian framework, injected computational parcels are treated and tracked individually. The motion of these parcels is determined by the applied forces.

Computational parcel When running numerical simulations using a Eulerian-Lagrangian fluid-particle approach, a group/bunch of realistic particles is considered as one computational parcel, which has the density as the mean density and mass as the total mass of these particles. Such a computational parcel is considered as one discrete particle in numerical calculations and is stated as particles in the following text.

The discrete particles trajectory is predicted by calculating their moving paths. The particles are moved mainly by the drag force from the carrier phase. Based on the Newton second law of motion, the movement of a discrete particle has the following formulation[13]:

$$\frac{d\mathbf{u}_p}{dt} = F_D(\mathbf{u} - \mathbf{u}_p) + \frac{\mathbf{g}(\rho_p - \rho)}{\rho_p} + \mathbf{F}_{\text{other}}, \tag{2.14}$$

where F_D is the drag force:

$$F_D = \frac{18\mu}{\rho_p d_p^2}\frac{C_D \text{Re}}{24}.$$

Here the $\mathbf{u}$ is the velocity of the continuous phase, $\mathbf{u}_p$ is the velocity of the discrete particle, μ is the dynamic viscosity of the carrier phase, ρ is the density of the continuous phase, ρ_p is the density of the discrete particle, d_p is the diameter of the particle and Re is the relative Reynolds number between the continuous and discrete phase: $\text{Re} = \rho d_p|\mathbf{u}_p - \mathbf{u}|/\mu$. C_D is the drag coefficients of the discrete particle. $\mathbf{F}_{\text{other}}$ in Equation 2.14 refers to other forces, one example is the virtual mass[13]:

$$\mathbf{F}_{\text{other}} = \frac{1}{2}\frac{\rho}{\rho_p}\frac{d}{dt}(\mathbf{u} - \mathbf{u}_p), \tag{2.15}$$

which is significant when continuous phase and discrete phase have similar densities. Another example is the pressure gradient force[12]:

$$\mathbf{F} = \left(\frac{\rho}{\rho_p}\right)\mathbf{u}_p \nabla \mathbf{u}. \tag{2.16}$$

The drag coefficients, C_D, can be formulated with different particle shapes: In **Fluent**, the drag coefficient for smooth, spherical particles can be defined as the following:

$$C_D = a_1 + \frac{a_2}{\text{Re}} + \frac{a_3}{\text{Re}^2}, \tag{2.17}$$

where a_1, a_2 and a_3 are constants. For non-spherical particles, drag coefficients in **Fluent** is defined as the following:

$$C_D = \frac{24}{\text{Re}_{sph}}\left(1 + b_1 \text{Re}_{sph}^{b_2}\right) + \frac{b_3 \text{Re}_{sph}}{b_4 + \text{Re}_{sph}}, \tag{2.18}$$

Stokes number The Stokes number is an important dimensionless number for a particle suspended in a continuous flow field. It represents how fast the particle will follow along the continuous flow field. It is defined as the following[13]:

$$\text{St} = \frac{\tau_V}{\tau_F}, \tag{2.19}$$

where τ_F is the characteristic time of the continuous flow field and τ_V is the momentum response time:

$$\tau_V = \frac{\rho_p d_p^2}{18\mu}. \tag{2.20}$$

If the Stokes number St $\ll 1$, this means the response time of particles is far less than carrier flow, which indicates that the particle will basically follow the movements of the continuous phase as its motion; while if St $\gg 1$, the response time of particles will be much longer than the carrier flow, which indicates that the particle will basically not follow the continuous phase.

2.6.2 Mass, Heat and Momentum exchange

In one way coupling, the discrete particles are affected by the continuous phase surrounding them(the particles can however not affect the continuous flow field). The mass, heat and momentum of each discrete particle can be affected by the continuous phase(all three physical parameters or some of them, based on the problem definition or particle type). In **Fluent** these three exchanges are treated as the discrete particles move over each control volume, see Figure 2.6 as an illustration[12].

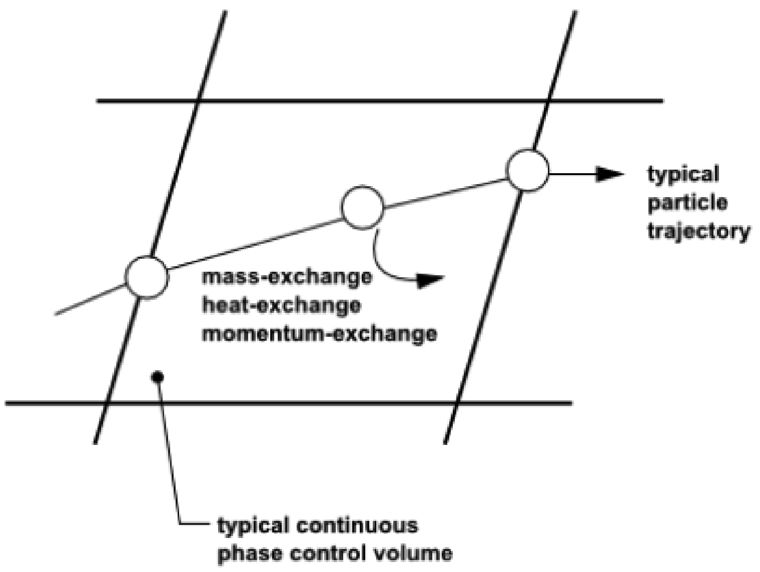


Figure 2.6: *Heat, Mass, and Momentum Transfer between the discrete and continuous phases[12].*

Momentum Exchange The momentum exchange between discrete particles and continuous phase is formulated in **Fluent** according to the following equation[12]:

$$\mathbf{F} = \Sigma\left(\frac{18\mu C_D \mathrm{Re}}{\rho_p d_p^2 24}(\mathbf{u}_p - \mathbf{u}) + \mathbf{F}_{\mathrm{other}}\right)\dot{m}_p \Delta t, \tag{2.21}$$

where all the physical parameters are introduced in the previous sections.

The mass exchange is formulated using continuity equation while the heat exchange is formulated by changing the thermal energy of the particles.

2.6.3 Turbulent dispersion

The turbulence generated in a flow can influence the motion of discrete particles by dispersion. However, as it has been mentioned before in section 2.5.1, time-averaged turbulence models do not possess either turbulence or velocity fluctuation u'. In this case the turbulence that causes the dispersion will need to be modeled. In **Fluent** the Stochastic Tracking can be used to predict the particle trajectory. By using Discrete Random Walk model(DRW), the velocity fluctuation can be modeled, hence also the instantaneous velocity. The Stochastic Tracking method will perform a sufficient number of calculations of trajectory to represent the random effects.

Step Length Factor In **Fluent**, the step length factor influences the time step size for the integration of trajectories. The integration time step is defined as the following[12]:

$$\Delta t = \frac{\Delta t^*}{\lambda}, \tag{2.22}$$

where Δt^* is the estimated transit time and λ is the step length factor. This implies that integration time is inversely proportional to step length factor.

2.7 Boundary conditions

The usual types of boundary conditions are velocity inlet, pressure outlet, symmetry, wall boundary conditions regarding to the continuous flow. For discrete particles, there can be escape, trap, reflect boundary conditions.

2.7.1 Turbulence Properties for Velocity Inlet and Pressure Outlet

For an applied turbulence viscous model, different turbulence parameters for different viscous turbulence models will be defined for specific boundaries.

Turbulent intensity The turbulent intensity is defined as the fraction between root-mean-square of the velocity fluctuations(u') and the mean flow velocity(u_{avg}), $\frac{u'}{u_{avg}}$. In **Fluent**, for a fully developed pipe flow, the turbulent intensity is estimated as follows[12]

$$I \equiv \frac{u'}{u_{avg}} = 0.16\big(Re_{D_H}\big)^{-1/8}, \text{ where } Re_{D_H} = \frac{\rho u L}{\mu}. \tag{2.23}$$

Here L is the characteristic length, μ is the dynamic viscosity of the air and $\mu_{air} = 1.983 \cdot 10^{-5}$. The turbulent intensity at boundaries can be approximated using this equation.

Turbulence length scale Turbulence length scale refers to the side of the large eddy of the turbulence flow. An approximation of the turbulence length scale of a pipe flow is that the turbulence length scale cannot be larger than a fraction of the pipe size[12]:

$$l = 0.07L, \tag{2.24}$$

where L is the diameter of the pipe.

2.7.2 Wall boundary condition

The wall boundary condition is usually used to bound/block the interior fluid. With a viscous turbulence model, the no-slip condition is always defined for the wall.

Wall motion The translational or rotational motion of a moving wall can be defined, by specifying the translational speed component, or rotational speed with its rotational axis and direction.

2.7.3 Discrete particles

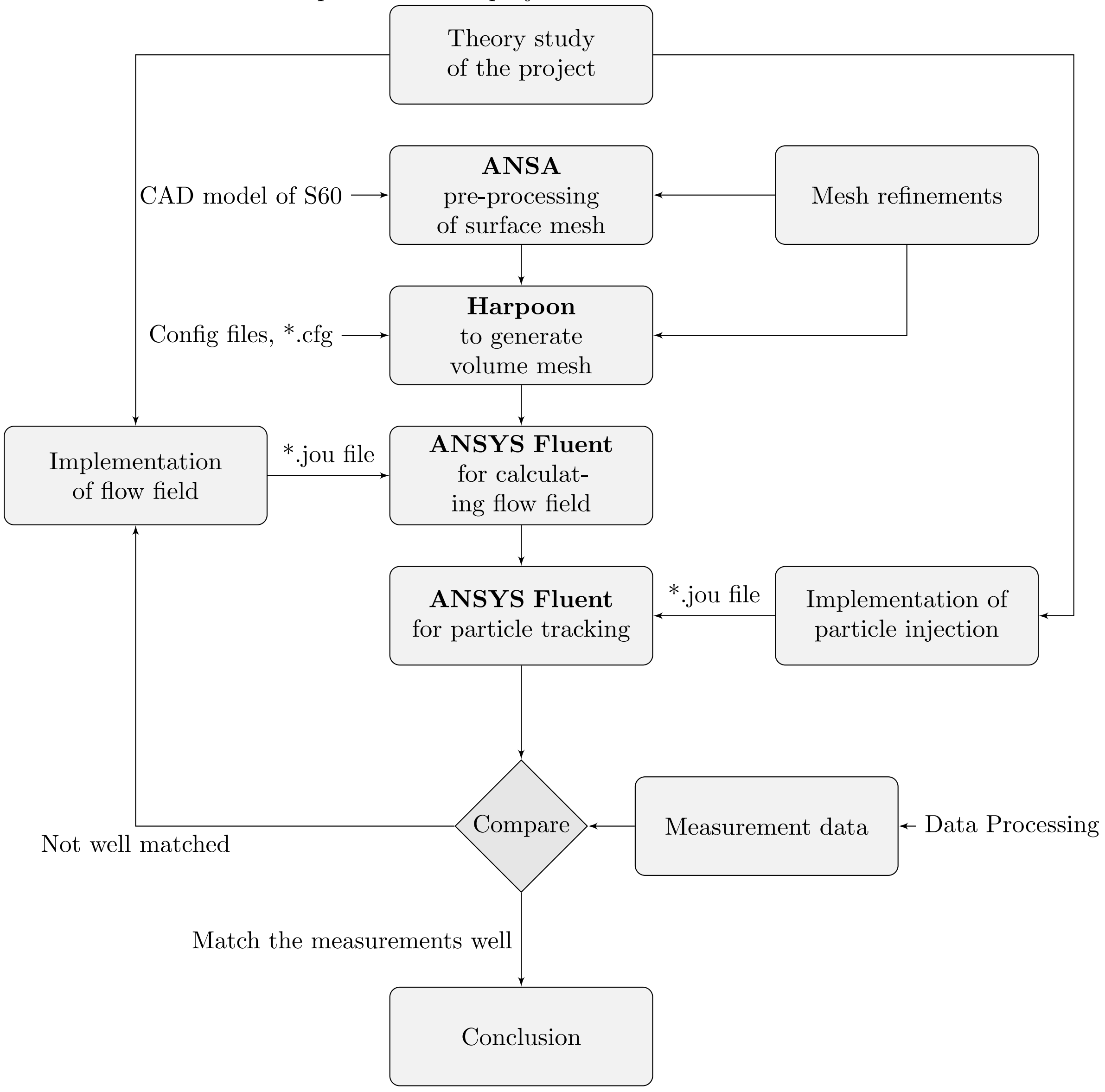
The discrete particles can hit the boundary and bounce back, escape through the surface or get trapped on the surface.

Reflect boundary condition In **Fluent**, the velocity with its components after reflection can also be manually defined as a proportion of the velocity before the particle hits the boundary surface.

3 Method

This chapter describes every implementation of the simulation with its motivation.

The project is undertaken in such procedure: first the geometry model(here Volvo S60) will be pre-processed in **ANSA** and a steady-state flow field will be calculated under different driving speeds; the discrete particles will be injected into this steady flow field and particle trajectories will be calculated and simulated; with the help of the prediction a particle distribution can be obtained. This distribution will be compared with the experimental data obtained from the test at Jokkmokk Proving Ground(JPG) in February, 2014. If the numerical prediction of distribution is far from the experimental data, certain implementations will be adjusted and tuned and the comparison will be done after tuning. This loop repeats until the numerical results are satisfying. A flow chart is presented to demonstrate the process for this project.


Theory study of the project
CAD model of S60
ANSA pre-processing of surface mesh
Mesh refinements
Config files, *.cfg
Harpoon to generate volume mesh
Implementation of flow field
*.jou file
ANSYS Fluent for calculating flow field
ANSYS Fluent for particle tracking
*.jou file
Implementation of particle injection
Not well matched
Compare
Measurement data
Data Processing
Match the measurements well
Conclusion

3.1 Processed measurement data

The experimental measurement data from the sensor is processed to provide a general understanding of the flux distribution. In those measurement data, each test was undertaken within one or two minutes, with different velocities and with sensor located in different places. The mean value of precipitation (converted from volume

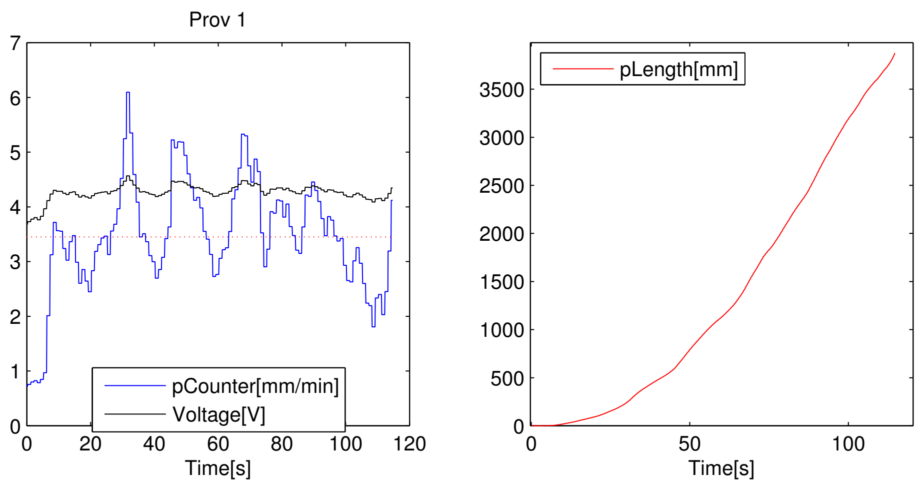


Figure 3.1: *One of the tests data under* $50km/h$ *driving speed. The red dash line represents the mean value.*

flux to mass flux, described in section 2.2.4) can be extracted from each run, see Figure 3.1 as an example. Distributions with respect to different locations under different specific speed can be calculated in this way.

3.2 Properties of snow particles

In order to simulate the situation as closed to measurements as possible, the physics behind the situation has to be sufficiently well understood. Based on the theories that have been introduced in the previous section 2.2, the physical information can be converted for implementation.

3.2.1 Density

The snow bulk density varies approximately from 70kg/m^3 to 120kg/m^3 below $0°C$ according to Figure 2.3 in section 2.2 previously. The density will be considered as a constant here. Other heat sources or heat exchange will not be taken into account in this project. Therefore the temperature is not implemented for simulation, but only to determine the density of the snow particle. However, the ground snow is compressed by the tyre and the particles are generated afterwards, the density of the particle can be much higher than its original density. The snow bulk density may also have a different value compared with snow particle density mentioned above(70 $\sim$ 120kg/m^3). In this project, snow particle density is approximated as 120kg/m^3 as a initial guess and will be modified later if needed.

3.2.2 Particle size

The snow particles have a specific average size and a size distribution as it has been introduced previously. However, uniform sized snow particles will be implemented as a first approach to simplify the problem. The trajectories of the snow particles from simulation can also be adjusted by changing the (average) size of the particles in order to meet the measurement data, since the size can change the particle mass. Specific distributions can be implemented later as an option for improving the simulation result.

3.3 Mesh generation

The mesh has to be generated for the numerical calculation later on, with the help of the CAD model of the car that has been created in advance.

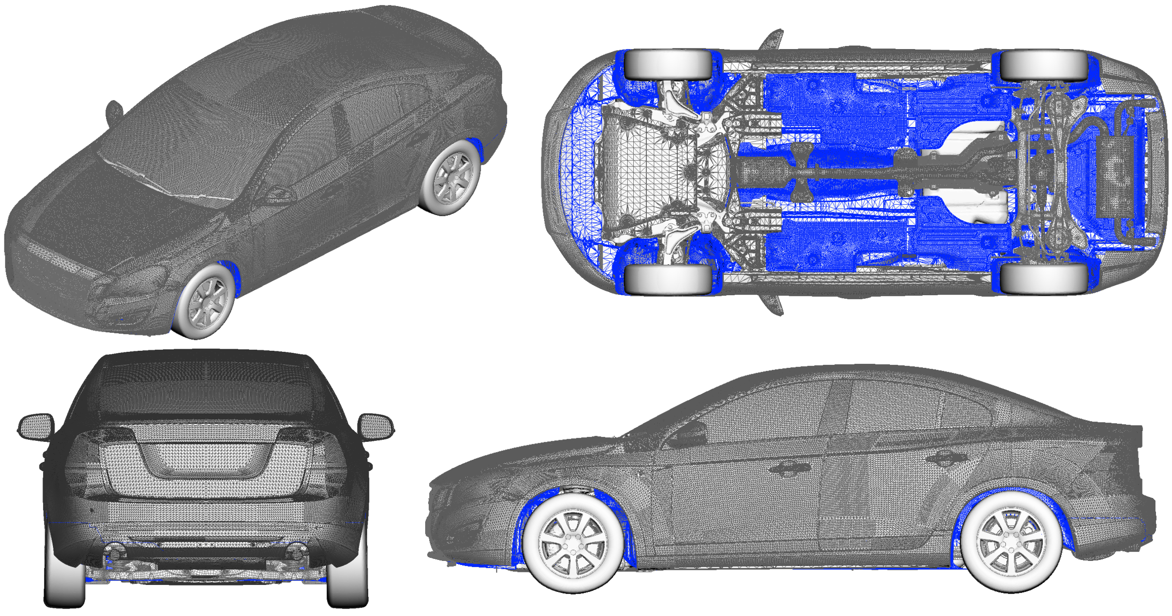


Figure 3.2: *The surface mesh of the entire car in* ***ANSA****(Surface mesh of the wheels is not shown due to rendering reasons).*

3.3.1 ANSA pre-processing: surface mesh

Based on previous experience, the CAD model of this Volvo S60 is imported first into **ANSA** for surface mesh generation. Here the surface mesh is approved as long as the geometric properties of the car are represented, since it will be imported further to **Harpoon** for volume mesh generation. Therefore the mesh quality is not consulted here. The key is merely that the mesh can represent the shape of the car with well enough details and high enough quality over the surface. A finished surface mesh of the car model is shown in Figure 3.2.

Closed space Another important key for the surface mesh of the geometry is to make sure that the geometry is physically plausible. By saying physically plausible it means that there are no cracks over the surface, nor skewed, unrealistic surfaces. One critical requirement is that the coupé and other solid component have to be a closed space. Otherwise the coupé space inside, for example, will be meshed (which it is both unnecessary and can cause numerical instability in this project) while generating the volume mesh in **Harpoon** later on. This can be prevented in **ANSA** by checking if there are small holes or cracks on the surface. Whether the coupé or other volumes are closed can be examined in **Harpoon**. If there is volume mesh generated at regions where they should not be, then the geometry or surface mesh has to be corrected.

Specify the mesh size with different PIDs The mesh qualities are different between different surfaces in order to maximize the accuracy meanwhile minimizing the computational load. This is achieved by naming the PIDs(Property IDs) with mesh size and later in **Harpoon**, specify the surfaces with specific mesh size/level by checking names of the PIDs. For example, the PID name *wall-suspension-rear-08mm* implies that the first layer of mesh over the rear suspension part will have 8mm as its mesh size. **Harpoon** will find out all surfaces with names that contain *08mm* and define the mesh size over these surfaces as 8mm. This first layer thicknesses of the mesh over the surfaces can be calibrated to optimize the result or the computational load as well. A list consisting of all PIDs for this Volvo S60 model is attached in Appendix A.

Sensor domains In order to capture the information of discrete particles flying through the precipitation sensors, several surfaces representing the location of the sensor domain are generated in **ANSA**, see Figure 3.3. From the figure it can be seen that sensor 3 and 4, sensor 5 and 6 cover each other. Thus the particle flux information for each individual sensor cannot be obtained simultaneously. In order to obtain the information for particle flux for each sensor domain, it must be guaranteed that no domain is covered. Consequently, two sets of surface mesh will be exported which both contain several but not all sensor domains, see Figure 3.4 for illustration.

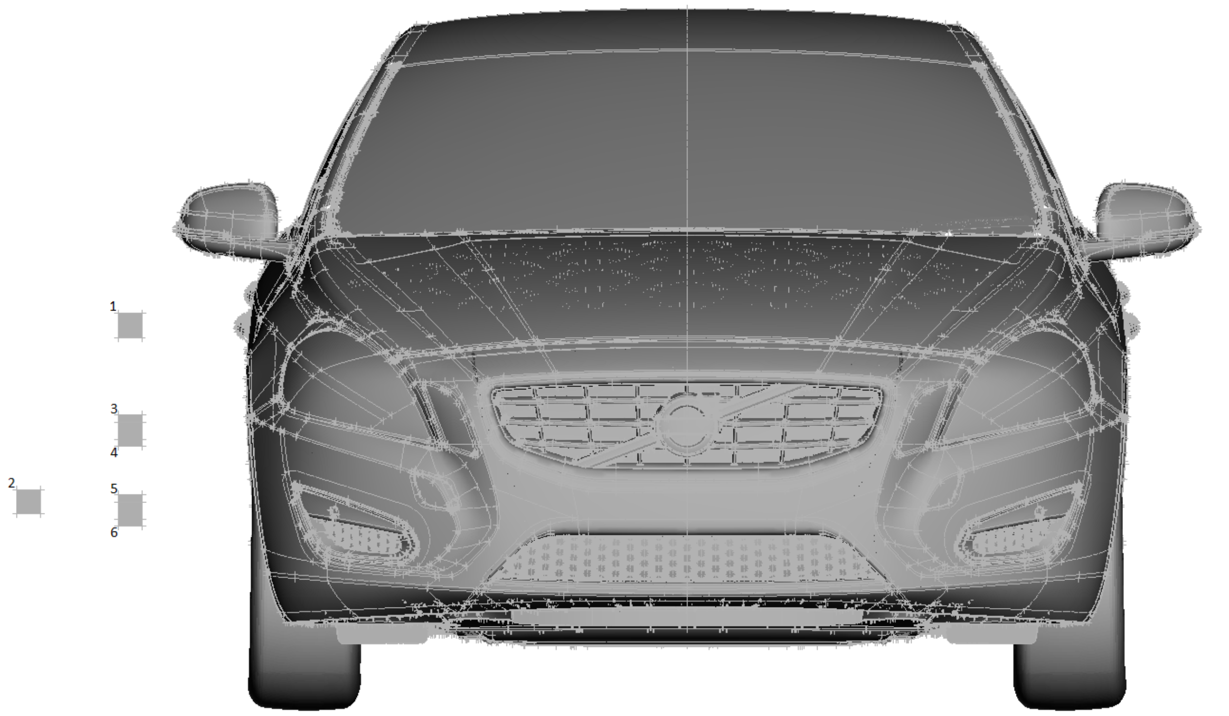


Figure 3.3: *The surfaces representing the locations of the precipitation sensor domains.*

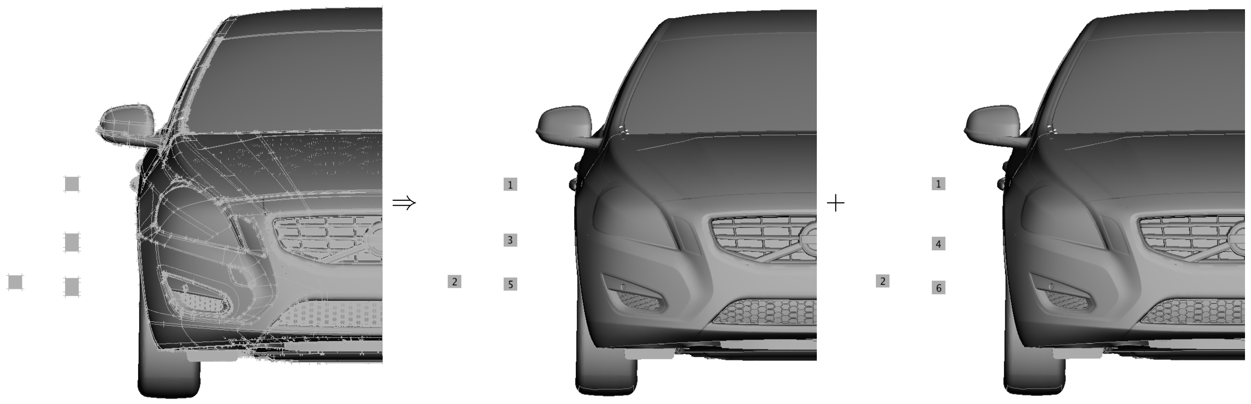


Figure 3.4: *The original surface representation is separated to two surface mesh representation.*

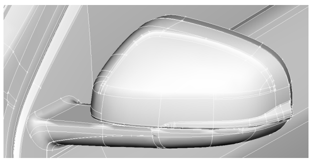
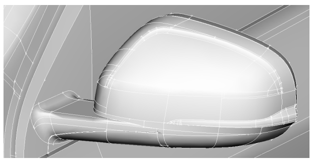
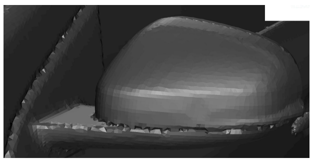
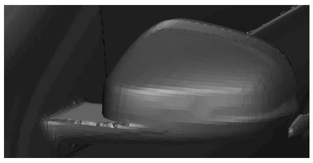
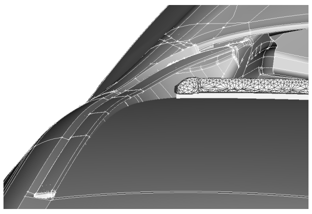
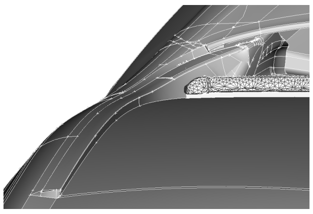
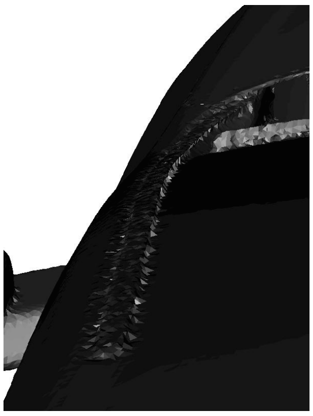
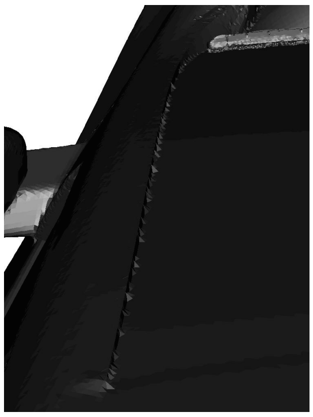
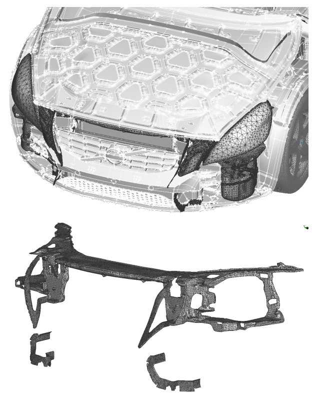
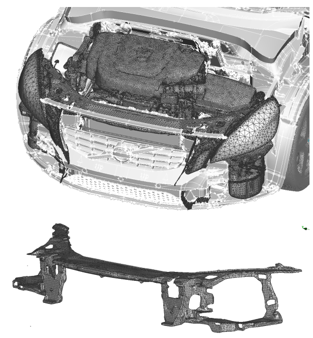
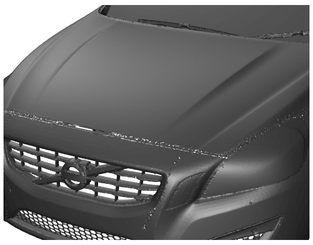
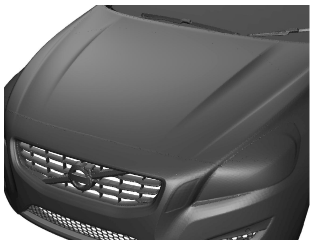
ANSA:before	**ANSA**:after	Mesh:before	Mesh:after
			
			
			

Table 3.1: Changes in **ANSA** and the consequences for the volume mesh surface.

3.3.2 ANSA pre-processing: surface smooth

The car model contains all details from small scale to large scale. Some of these scales are important as they will influence the flow properties; some of these details are small and irrelevant for fluid field but will add difficulty for volume mesh generation, because the scale of the details are smaller than the mesh scale. This leads to bad quality mesh and hence convergence difficulty later on for the RANS calculation. With respect to these issues, some of the details have to be removed so the surface can be smooth, which provides a high quality mesh and better convergence for RANS model. Some of the surfaces representing different small details are deleted and some gaps are filled. Some examples are demonstrated in Table 3.1 to show the consequences. The exterior wall of the car is more sensitive than other parts due to the high velocity flow and thus requires smoother surface in volume mesh.

3.3.3 Volume mesh generation in Harpoon

A surface mesh file is exported from the pre-processed ANSA-file with the surface mesh and imported to **Harpoon** for volume mesh generation. Based on previously developed method, a configuration file will be created first. This file contains all configurations needed to create the wind tunnel geometry respective volume mesh with refinements. A demonstration in **ANSA** shows the appearance of the wind tunnel in Figure 3.5. The wind tunnel can be defined using command *farfield xmin* + coordinates.

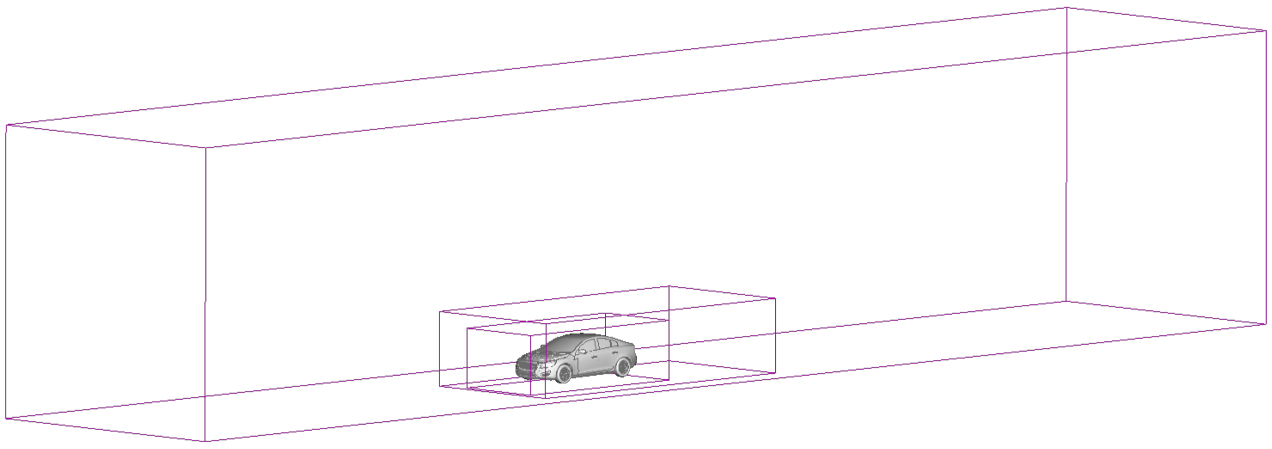


Figure 3.5: *The geometry of the created wind tunnel with the car inside.*

Refinement box There are several refinement boxes defined within the wind tunnel. Two boxes with different sizes are enclosing the entire car to refine the mesh around it, which are obviously demonstrated in Figure 3.5. The refinement boxes can be defined using the following command[15] as an example:

```
refine **AROUND CAR
0 0
−500 −1500 0
8000 1500 2000
```

where *refine* is the command for creating a refinement box; the first 0 at the next line refers to cartesian box as box type, the second 0 refers to a resolution of $32mm$ cell size; the next two lines refers to the location for the diagonal line section of the cartesian box[15].

There are another two refinement boxes located along the wheels on the right side(see Figure 3.6) where large amount of snow particles are expected. The situation at the left side of the car is not taken into account, since the measurements are done at the right side and the car can be considered as approximately symmetric. It also reduces the number of cells and increases the computational performance of the simulation. These three refinement boxes can ensure a better prediction of the snow particles trajectories. One of these refinement boxes have cylinder shape and is defined in the similar way[15] as an example in the following:

```
refine **SNOWY REGION
3 3
1740 900 195
4800 1150 510
1740 880 220
4800 880 800
```

where *3 3* below *refine* refers to the cylinder as box type and mesh quality(4mm here); the two coordinates after that define the two center points of the top and bottom of the cylinder, while the last two coordinates define points on edges of the top and bottom circle. There is another rectangular refinement box located between the two cylinders.

The configuration file also takes care of defining the mesh type (command *type hex* for hex dominant), voiding the mesh in coupé(command *vptkeep+coordinate*), smoothing the mesh and eliminating the volume mesh on the surface since these cells usually have bad qualities(command *smooth* and *vfind*), setting boundary conditions(command *setbc*), etc.[15]

Figure 3.7 presents a cut plane at symmetry plan of the generated mesh from **Harpoon**, with a detailed observation around car in Figure 3.8. From Figure 3.8 it can be seen that the coupé, the engine bay and other solid geometries are empty without any volume mesh, which is the correct situation. Figure 3.9, 3.10 for cut planes at different locations are presented to verify that the volume mesh is desirable globally.

Number of cells The number of cells determines the computational load. For a steady-state simulation the number of cells can be as large as $\approx$ 100 million cells for a 2000 iterations calculation. For unsteady simulations,

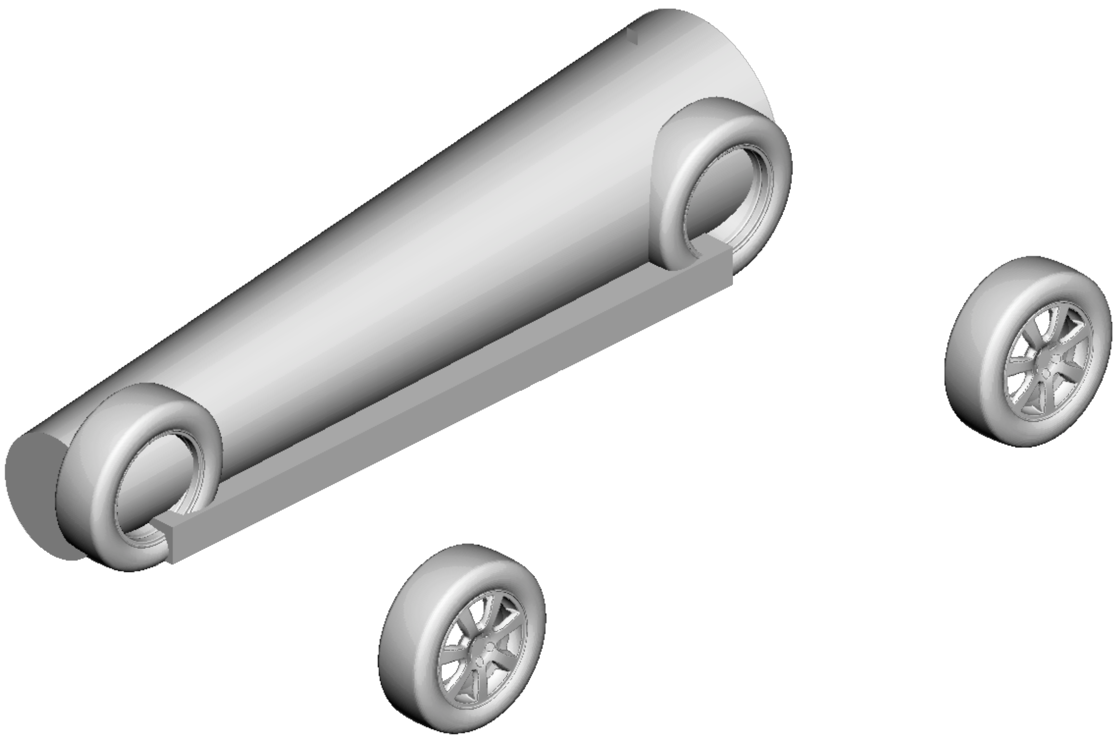


Figure 3.6: *A demonstration for the geometry of the refinement boxes besides the right two wheels.*

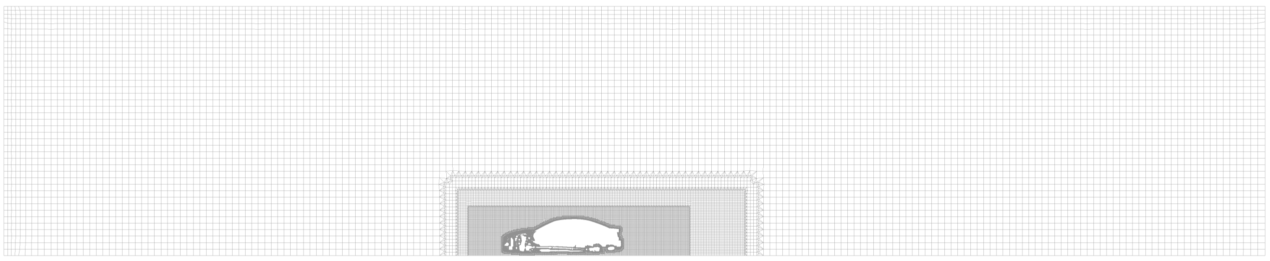


Figure 3.7: *The generated global mesh from **Harpoon** at $y = 0$.*

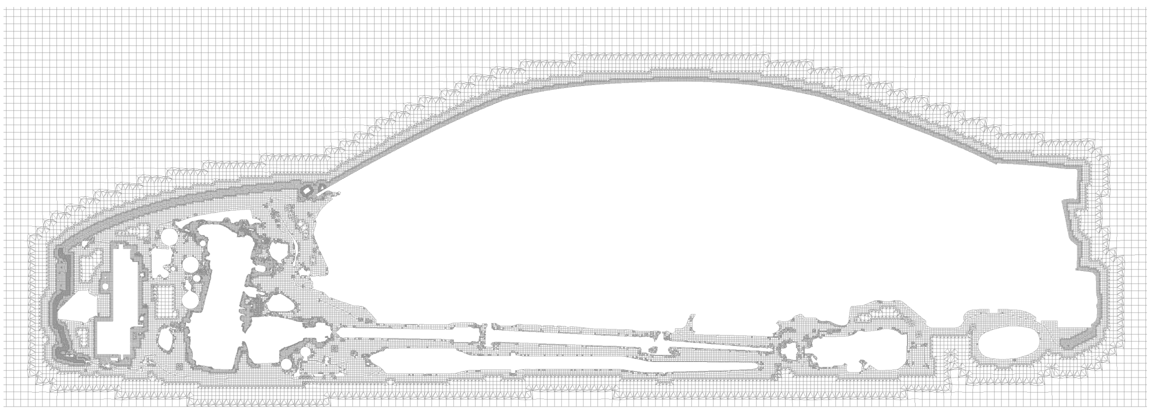
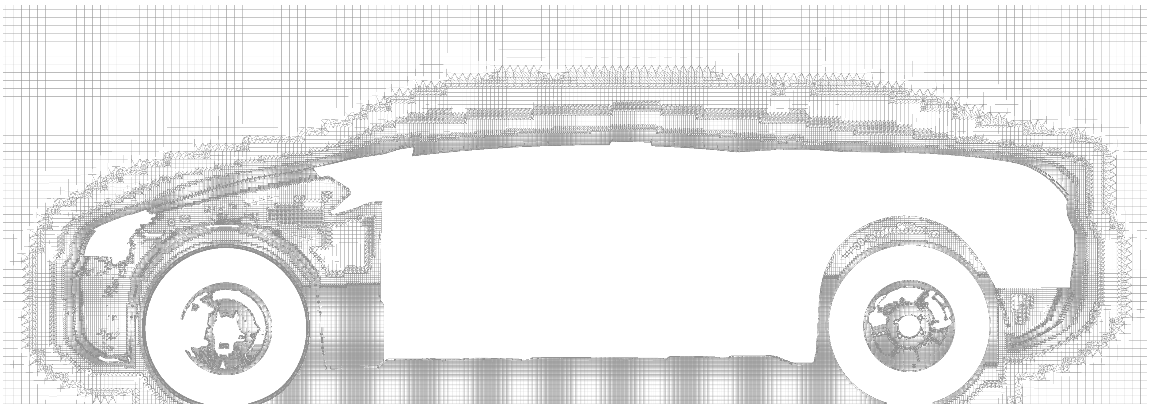
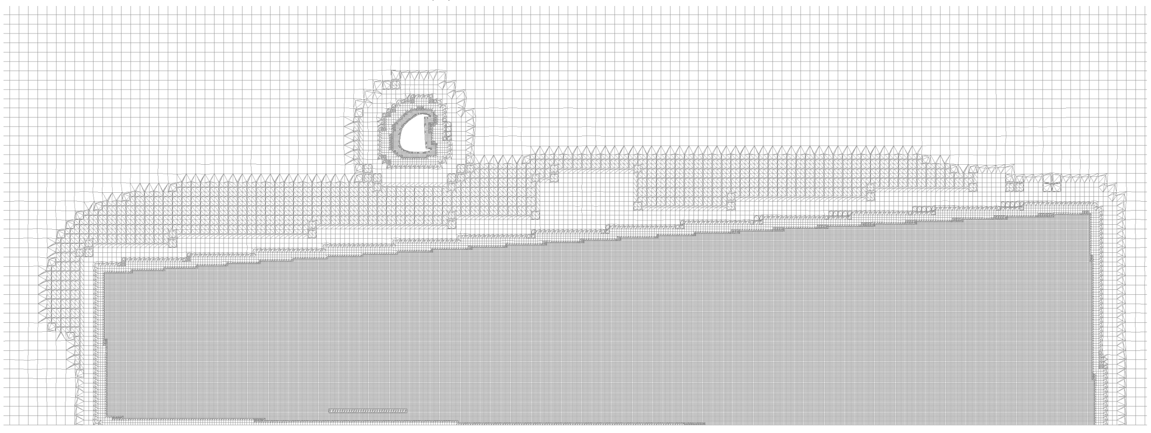


Figure 3.8: *The local mesh around car at $y = 0$.*

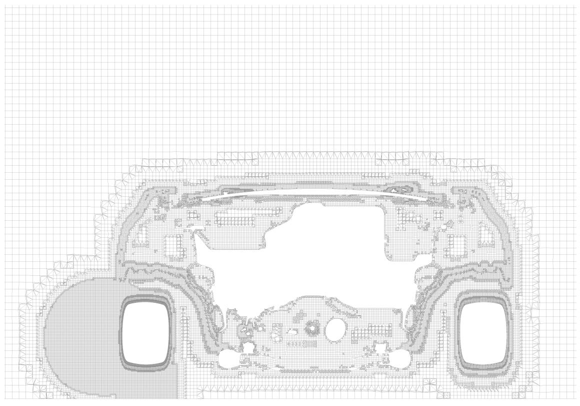


(a) *Plane cut at $y = 0.78m$.*

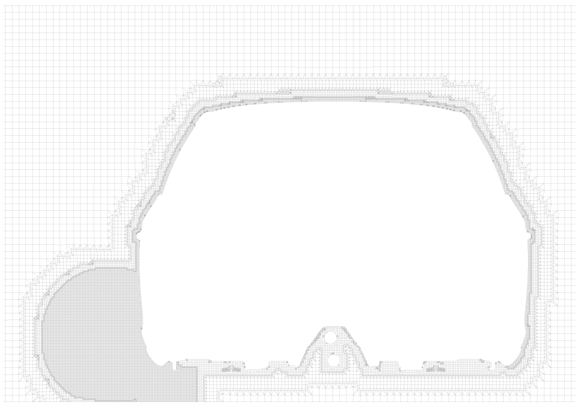


(b) *Plane cut at $y = 1m$.*

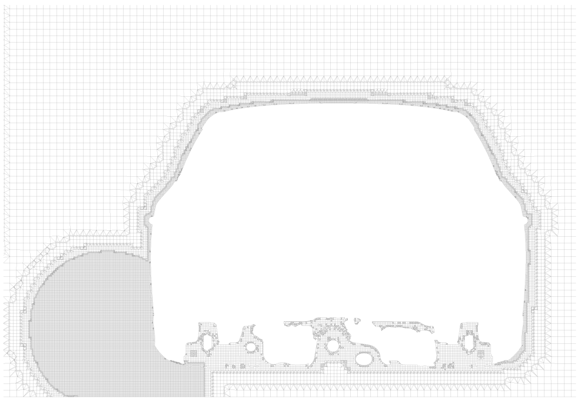
Figure 3.9: *Cut planes that are perpendicular to y axis to demonstrate and examine if the volume mesh is desirable.*



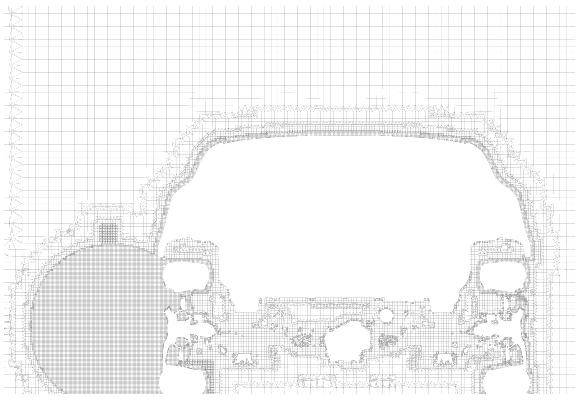
(a) *Plane cut at $x = 2m$.*



(b) *Plane cut at $x = 3m$.*



(c) *Plane cut at $x = 4m$.*



(d) *Plane cut at $x = 4.5m$.*

Figure 3.10: *Cut planes that are perpendicular to x axis to demonstrate and examine if the volume mesh is desirable.*

Driving speed(km/h)	Velocity magnitude(m/s)	turbulence intensity(%)	turbulence length scale
50	13.9	2.16	0.1
70	19.4	2.16	0.1
90	25	2.16	0.1

Table 3.2: Defined values for velocity inlet boundary condition under different driving speeds.

Gauge pressure(Pa)	turbulence intensity(%)	turbulence length scale(m)
0	3	0.1

Table 3.3: Defined values for pressure outlet boundary condition under all driving speeds.

the number of cells will have to be reduced to around $50 \sim 60$ million cells at most. An appropriate number of cells will demand the least computational resource while giving a relatively more focus on interesting regions and hence a more precise prediction of the flow and particles. The number of cells in this project is fixed around 62 million.

The configuration file of the volume mesh generation for **Harpoon** is attached in Appendix B.

3.4 Various driving speed

In this project, the experimental measurements were undertaken under different driving speeds: 50km/h, 70km/h and 90km/h, with driving direction towards negative x direction. The steady flow field are therefore simulated under these different velocities with different boundary conditions. The particle injection is implemented differently correspondingly as well.

3.5 Boundary conditions

In this section the boundary conditions for both continuous flow field and the discrete particles are described.

3.5.1 Velocity Inlet for the wind tunnel

Velocity inlet defines the flow velocity coming into the wind tunnel. The flow velocity here is the same as the car velocity. The turbulence intensity and turbulence length scale can be approximated using Equation (2.23) and (2.24). Here the turbulence intensity and length scale are same under all circumstances. The defined values for velocity inlet under different driving speeds are listed in Table 3.2.

3.5.2 Pressure Outlet for the wind tunnel

The pressure at outlet boundary is defined as 1 bar as normal condition, therefore the gauge pressure is 0. Turbulence intensity and length scale are defined the same way as in Velocity Inlet. Values are the same under all different driving speeds and are listed in Table 3.3.

3.5.3 Moving boundaries

Translation movement of the ground Since the car is treated as the reference of the system, the ground will be defined to have a translation movement in positive x direction and having the same velocity magnitude as the car. Values are listed in Table 3.4.

Driving speed(km/h)	Velocity magnitude(m/s)	Unit vector of wall translation(x, y, z)
50	13.9	$(1, 0, 0)$
70	19.4	$(1, 0, 0)$
90	25	$(1, 0, 0)$

Table 3.4: Defined values for translation movement of the ground

Driving speed(km/h)	Rotational speed, front wheel(deg/s)	Rotational speed, rear wheel(deg/s)
50	43.64	42.29
70	61.08	59.21
90	78.55	76.13

Table 3.5: Defined values for rotational speeds for both the front wheel and the rear wheel.

Location of rotation-axis, front wheel(x, y, z)(m)
$(1.7071, 0, 0.4785)$
Location of rotation-axis, rear wheel(x, y, z)(m)
$(4.4820, 0, 0.4919)$
Rotation-axis direction(x, y, z)
$(0, -1, 0)$

Table 3.6: Defined values for rotation-axis location respective direction.

Rotating wheel In order to replicate the physical situation as much as possible, the tyre surface will be set as rotating surface as well. Based on the observation of the wheel from **ANSA**, the coordinates for the center of the wheel can be approximately determined. The rotation-axis is parallel with the y-axis. The rotational speed of the wheel can be obtained from the car velocity: $\omega = v/r$. All surfaces of one indivitual wheel(tyre surfaces, rim structures, etc.) are defined with the same rotational speed and the same axis. Since the wheel is rotational, it could be ideal to simulate the rotation of the entire rim structure, e.g. using Multi-Reference Frame (MRF) method. However this requires generating a new mesh for each time step for the transient simulation and is practically computational prohibited since it takes more than 20 minutes to just generate a volume mesh with ~ 60 millions cells. Therefore the wheel structure is frozen and the surface (rotational) velocity is applied, no matter using steady or unsteady RANS. The information of coordinates and direction of the rotational axis can be obtained from the car model in **ANSA**. The defined values are listed in Table 3.5 and Table 3.6.

3.5.4 Inelastic reflection between the particles and the wall boundaries

The car model consists of a large number of different parts defined as reflective wall boundary condition for discrete particles. Several of these walls are defined as inelastic collision when the particles hit on them. This is achieved by manually defining the normal and tangential velocity component after the particles are reflected:

$$v_t' = \text{coeff1} \cdot v_t,$$
$$v_n' = \text{coeff2} \cdot v_n,$$

where coeff1 and coeff2 are the coefficients that determine how much the velocity is damped after the reflection.

3.5.5 Precipitation sensor

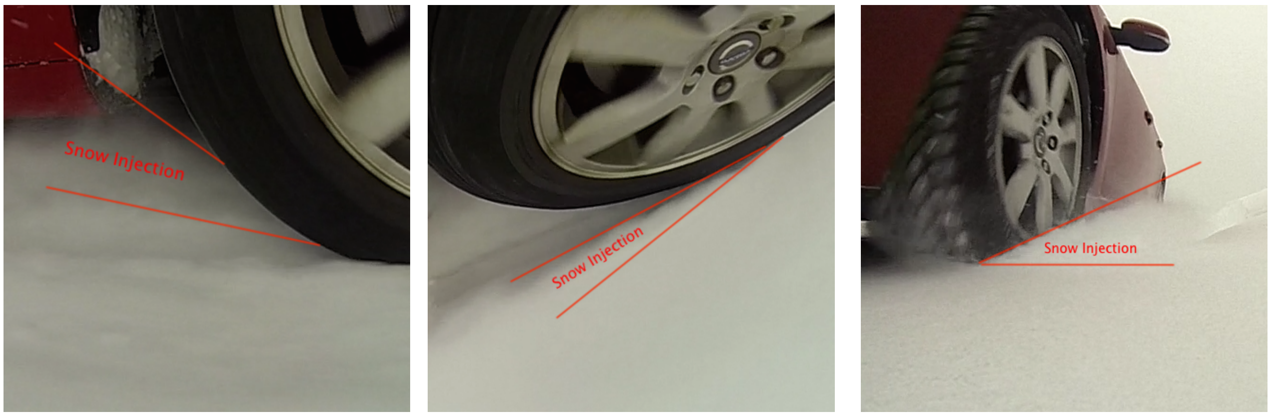
With the steady flow field, the particles will be released after the flow simulation as post-processing. The boundary condition of continuous phase for these surfaces are defined as *fan* boundary condition so the flow can pass through these surfaces, while discrete particles can either flow through them as usual, either get trapped if needed. Therefore boundary conditions of discrete particles at those surfaces can be defined as two types:

- *interior*, which means the particles will travel through as usual as traveling through interior;
- *trap*, which means the particles will be captured when they travel through the surfaces. Number of trapped particles will be reported and hence the information of number flux can be obtained.

The generalization of snow particles is introduced in section 3.6.

3.6 Implementation of particle injection

The presence of the self generated snow dust is implemented by defining particle injections, which is also the critical part of the project.



(a) *Snow particle injection, type 1.* (b) *Snow particle injection, type 2.* (c) *Snow particle injection, type 2.*

Figure 3.11: *Snapshot from films to help defining the injection points and directions.*

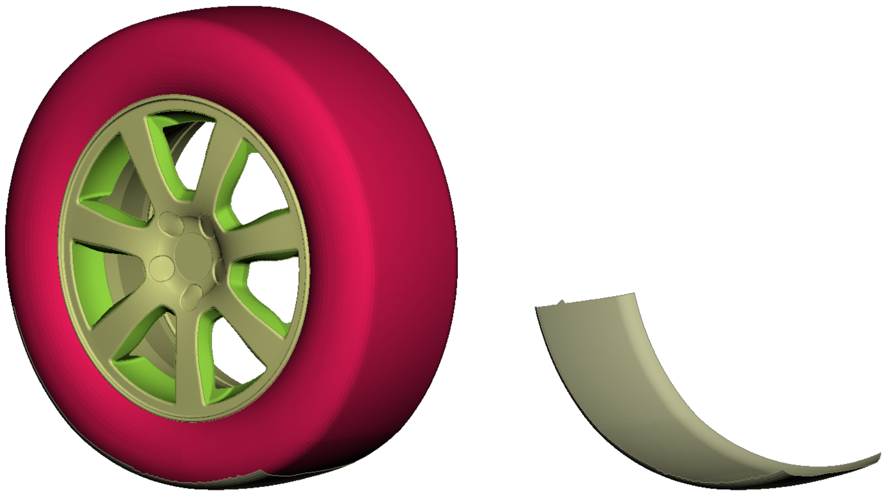


Figure 3.12: *Patched tyre surface defined in* ***ANSA****.*

3.6.1 Motivation of the implementation

The locations of injections is decided by observing the videos. There are two types of injection points, see Figure 3.11a and 3.11b. Both of them locate on the tyre and are listed below:

- One type of injection points are located on (part of) the tyre surface, see Figure 3.11a. The injection points will be defined as uniformly distributed, close to the tyre. (In practical numerical simulation the injection points cannot be defined exactly at the tyre surface since they must be located inside a computational cell). The velocity magnitudes of the injected particles are the same as the rotating wheel surface velocity magnitudes and the directions are same as the rotating wheel.

- The other type of points are located on the edge of the tyre(Figure 3.11b) as single point, solid cone injection, causing a splash of the snow, generating snow particles on the side. Figure 3.11c also gives contribution to define the angle of this type injection. The velocity magnitudes in this case is more or less unknown, with a span-wise direction approximately 26° from the ground.

The type 2 injection caused by splash is unstable and unreliable compared with type 1, since it depends on the depth of the snow and snow ground condition(whether it is packed or not). Furthermore this splash phenomena cannot be observed clearly from the videos most of the time. Therefore type 2 **will not be considered** as a first approach.

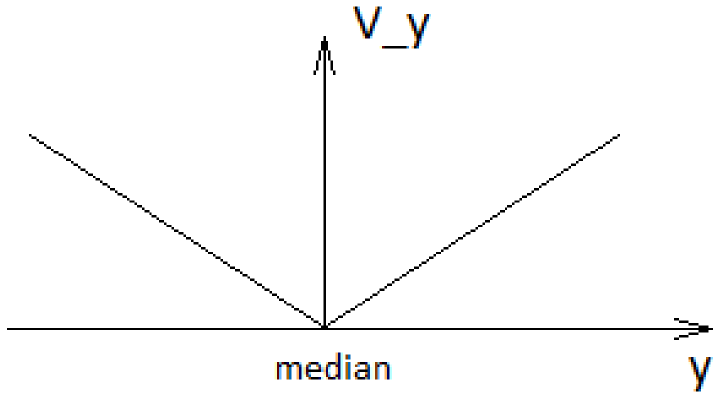


Figure 3.13: *Velocity magnitude of v_y along $y-$direction.*

3.6.2 Surface injection over the tyre

Since the tyre surface (or part of the tyre surface) is considered to inject particles, a "surface-type" injection is needed to be implemented(yet not exactly *surface* type injection in **Fluent**). This is fulfilled by using two UDFs (User Defined Function). The two UDFs are *wallVelocity.c* and *cellCentroids.c*[1]. UDF *wallVelocity.c* generates a velocity field for the first layer of cells over the surface of the tyre with the specified velocities, while UDF *cellCentroids.c* loops over all cells that are located over the surface, extracts the centroid of these cells, defines them as the injection point and single particle injection as injection type and finally generates an injection file. Therefore the snow particles generated from the tyre are implemented in this manner. These two UDFs are combined and modified to a *injection.c* UDF later and compiled in **Fluent**.

Patched tyre surface for surface injection Based on observing the videos, for example Figure 3.11a, it is assumed that only a part of the entire tyre is considered as the surface that injects particles. This patched surface, which is presented in Figure 3.12, will be first defined in **ANSA** and later utilized by the UDF as the injecting surface.

The 'manually' added velocity component at y direction Based on previous projects and experiences[2], adding a span-wise velocity component for the injection will not only contribute but also be necessary. The magnitude of this span-wise v_y contribution has a linear distribution based on the y coordinate and can be formulated as the following:

$$v_y = C \cdot (y - y_{\text{median}}), \quad \text{where } C = \frac{C_y}{y_{\max} - y_{\text{median}}}. \tag{3.1}$$

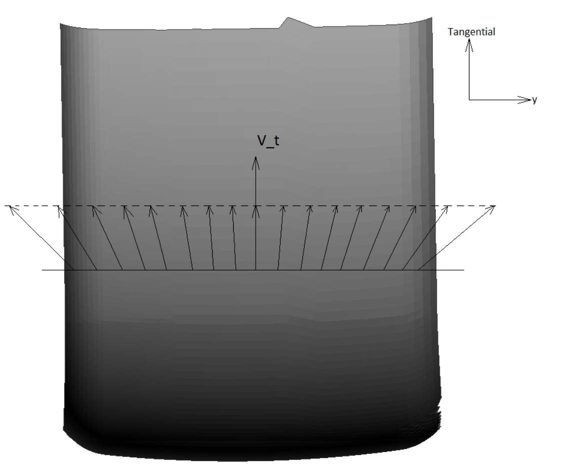
Here $y_{\max}$ refers to the coordinate of the edge of the tyre, y_{median} is the middle of the tyre and y is any y coordinate within the tyre. Figure 3.13 demonstrates approximately how v_y is distributed along the $y-$axis: where V_y refers to the velocity magnitude. This indicates that the particle will be injected not only tangential to the surface, but also will be injected 'outside' the wheels. How large this v_y will be is unknown and can be adjusted (by changing constant C_y) if needed.

The consequence of introducing this v_y is that the total velocity magnitude can exceed the driving speed. This can be either realistic or inappropriate. In this project two approaches are applied as the following:

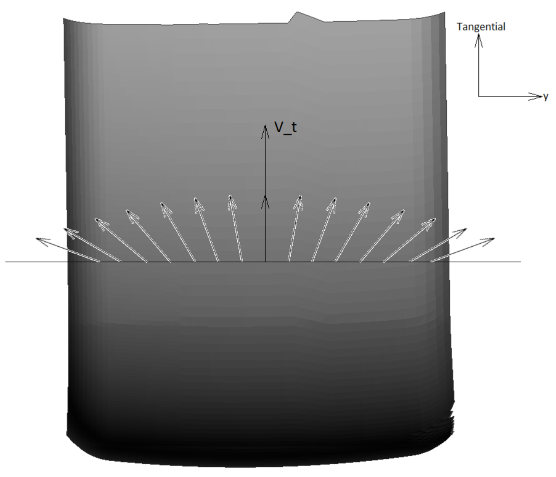
- The first approach is that only v_y is applied with a reduced velocity v_t. Besides there is no further implementation. This leads to a relatively high velocity magnitude of particle injection near the edge of the tyre, see Figure 3.14a for illustration;

- The second approach is that a reduction of tangential velocity magnitude v_t is applied after introducing v_y. In order to make sure the total velocity magnitude is the same as the original velocity magnitude,

[1]UDFs are written by Dragos Moroianu and are slightly modified by author.
[2]The water splash simulation of a car using LES and Volume of Fluid(VOF) technique at VCC, done by Carl Anderson and Dragos Moroianu.



(a) *Velocity for the first approach, tangential velocity component v_t is unvaried along y direction.*



(b) *Velocity for the second approach, tangential velocity component v_t is decreased proportionally.*

Figure 3.14: *Two approach for introducing the velocity component v_y along y-axis.*

Driving speed	Volume mesh A(for sensor 1,2,3,5)	Volume mesh B(for sensor 1,2,4,6)
50km/h	50a.cas.gz & 50a.dat.gz	50b.cas.gz & 50b.dat.gz
70km/h	70a.cas.gz & 70a.dat.gz	70b.cas.gz & 70b.dat.gz
90km/h	90a.cas.gz & 90a.dat.gz	90b.cas.gz & 90b.dat.gz

Table 3.7: Total six steady-state simulation cases.

which is the same as the driving speed, the tangential velocity over the surface is modified as the following:

$$v_t' = \sqrt{v_t^2 - v_y^2},$$
$$v_x' = v_x \cdot \frac{v_t'}{v_t},$$
$$v_z' = v_z \cdot \frac{v_t'}{v_t},$$

where v_t, v_x, v_z are original velocity components, while v_t', v_x', v_z' are the modified velocity components after introducing v_y. Figure 3.14b illustrate the velocity profile of this approach.

The UDFs for generating injection file are attached in Appendix D. The generated injection file contains coordinates of these single injection points, the velocity component of the injections, the diameter and the temperature of these particles.

3.7 Single phase

The simulation will be held under single phase condition as a first stage to reach a steady state flow field. The discrete particle trajectories will be simulated and tracked using this stable flow field after as post-processing. The boundary conditions for steady states simulations are described in section 3.5. There are totally 6 steady state simulations to calculate: 2 cases for 2 volume mesh under 3 different driving speeds, which are listed in Table 3.7.

3.7.1 Turbulence model

In the steady-state simulation, standard k-ε is applied first to get converged results for evaluation. Later, realizable k-ε is applied as a more reliable turbulence model for more precise results.

3.7.2 Convergence criteria

There is no convergence criteria to reach(since it can take far more computational resource than necessary due to large amount of data) for the steady-state simulation but a fixed number of iterations for the simulation to reach and stop. Here the number of iterations is set 2500, based on previous experiences.

A journal file that contains all settings of the steady-state simulation is imported in **Fluent** for calculation. This journal file is attached to the Appendix C.1.

3.8 Multiphase

Predictions of particle trajectories can be done using different methods. In this project the particle trajectories are predicted and numerically simulated using a steady-state flow field with Stochastic Random Walk model. There is also a certain amount of surfaces that are defined using an inelastic reflection approach as introduced previously in Section 3.5.4.

3.8.1 Stochastic Random Walk particle tracking

According to Equation 2.22 the step length factor is set as 8 and total number of steps is set as 15000 to make sure most of the particles are either trapped or escaped. Number of tries is set as 5 to give enough statistical meaning while keep the computational load small.

A journal file that contains all configurations for the particle tracking is attached in Appendix C.2.

3.9 Verification

The simulated flux distribution from the simulation will be compared with precipitation data from the measurement to verify if the simulation is appropriate with its implementation. Certain configurations will be adjusted in order to satisfy the measurement data.

3.9.1 Varied implementation

There are several variables for tuning so that the simulated flux distribution meets the experimental data as much as possible. Different variables with motivations are listed:

- Constant C_y in Equation 3.1 to adjust the velocity profile for surface injection;
- Diameter of particles, this will affect the mass of the particles;
- Density of material, this can also change the mass of the particles;
- Shape coefficient, this will change the drag force when the particle drag law is applied.

4 Results

This chapter contains all relevant, explained results from simulations with different configurations representing different realistic situations. The simulation results are demonstrated and compared with experimental data for all cases in the following sections.

4.1 Results from experimental data

The experimental data are converted as mass flux and mean values are obtained.[1] Converted data under different driving speeds are presented in the following.

4.1.1 50km/h

The mean value for converted mass flux at all locations under 50km/h is shown in Figure 4.1. A more straight forward figure demonstrating the flux distribution in percentage values from the experimental result (Figure 4.1) is presented in Figure 4.2.

4.1.2 70km/h

The mean value for converted mass flux at all locations under 70km/h is shown in Figure 4.3. A more straight forward figure demonstrating the distribution from experimental result is presented in Figure 4.4.

4.1.3 90km/h

The mean value for converted mass flux at all locations under 90km/h is shown in Figure 4.5. A more straight forward figure demonstrating the distribution from experimental result is presented in Figure 4.6.

In Figure 4.1, 4.3 and 4.5, there are data marked with green round circle, which indicate the sensor domain located at sensor 2. The data marked with red color indicate that the snow ground is pressed and packed hence generated less amount of snow particles. The data marked with red color are not utilized.

[1]The conversion is same as mentioned previously in Section 3.1, here the density ρ is still assumed 120kg/m^3 as first guess. However it doesn't matter since only the distribution is consulted.

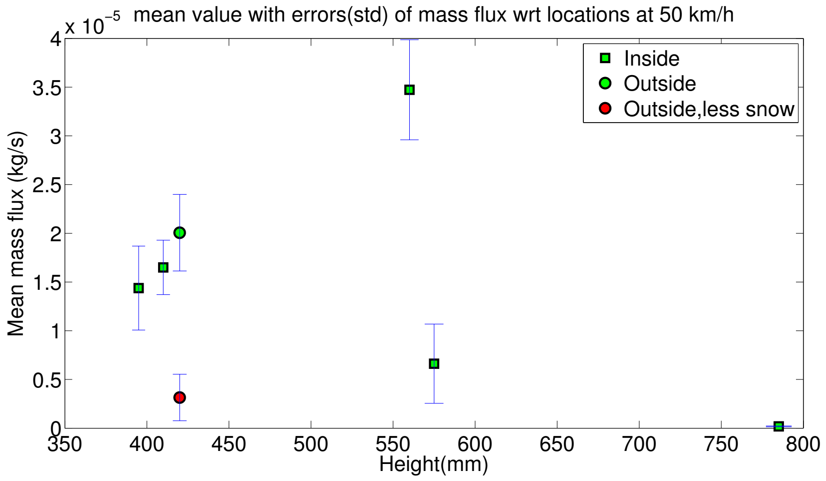


Figure 4.1: *Distribution of converted mean mass flux with respect to different locations under* 50*km/h driving speed.*

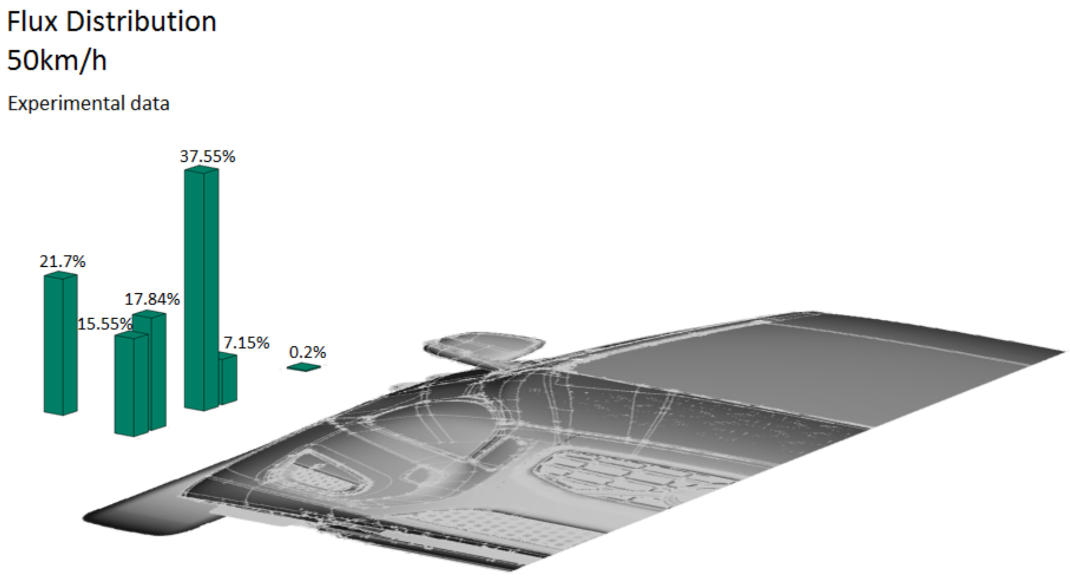


Figure 4.2: *Flux distribution in percentage under* 50*km/h driving speed.*

4.2 Results from numerical simulation

The particle injection is based on the *first approach*, which has been formulated previously in Section 3.6.2.The numbers of trapped particles with respect to each individual sensor domain under different driving speeds are listed in Table 4.1[2]; the flux distribution is presented in Table 4.2. The implementations for particle injection under different circumstances are listed in Table 4.3.

[2]Sensor 1 and sensor 2 process the average value from two sets of data since they exist in both volume mesh.

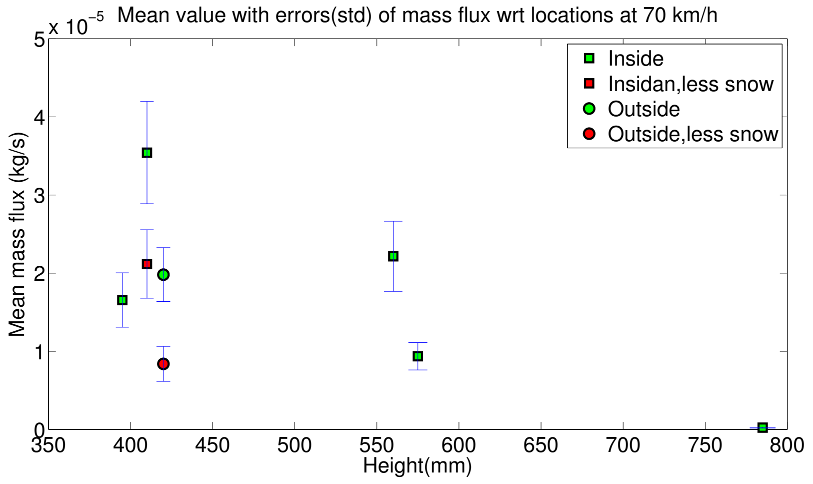


Figure 4.3: *Distribution of converted mean mass flux with respect to different locations under* 70*km/h driving speed.*

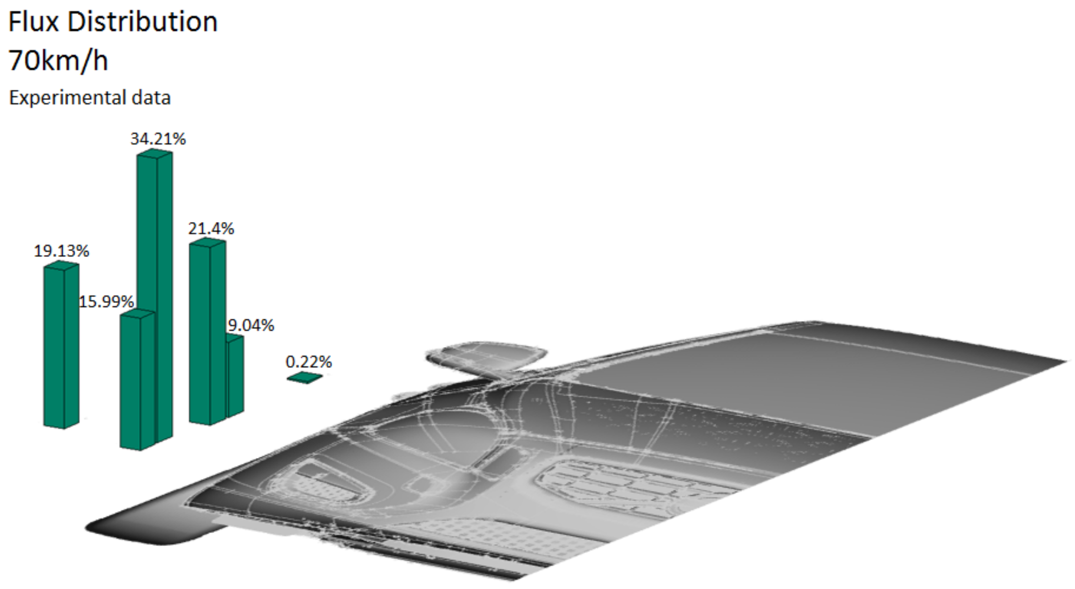


Figure 4.4: *Flux distribution in percentage under* 70*km/h driving speed.*

Speed	Sensor 1	Sensor 2	Sensor 3	Sensor 4	Sensor 5	Sensor 6
50km/h	0	2437	1362	1586	2349	2748
70km/h	39	2043	1672	1698	4736	4554
90km/h	3	2494	1725	1860	4468	4821

Table 4.1: Number of trapped particles at different sensor domains under different driving speeds.

Speed	Sensor 1	Sensor 2	Sensor 3	Sensor 4	Sensor 5	Sensor 6
50km/h	0%	23.25%	12.99%	15.13%	22.41%	26.22%
70km/h	0.26%	13.86%	11.34%	11.52%	32.13%	30.89%
90km/h	0.02%	16.23%	11.22%	12.11%	29.07%	31.36%

Table 4.2: Percentage/Distribution of trapped particles at different sensor domains under different driving speeds.

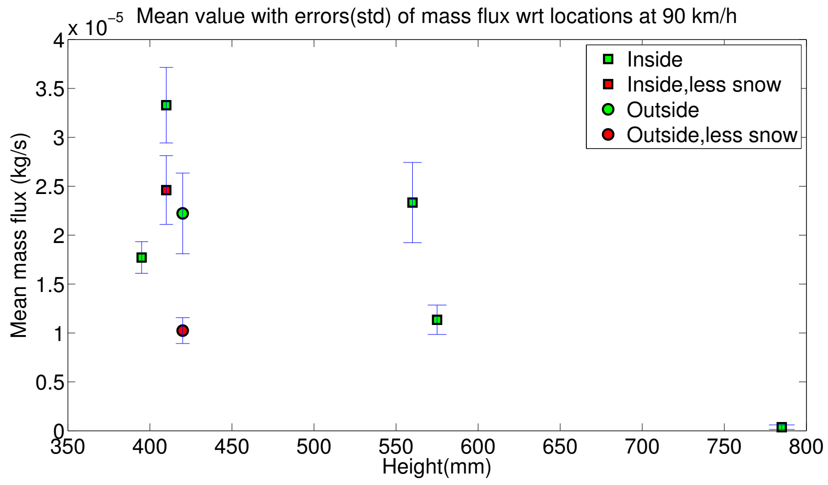


Figure 4.5: *Distribution of converted mean mass flux with respect to different locations under 90km/h driving speed.*

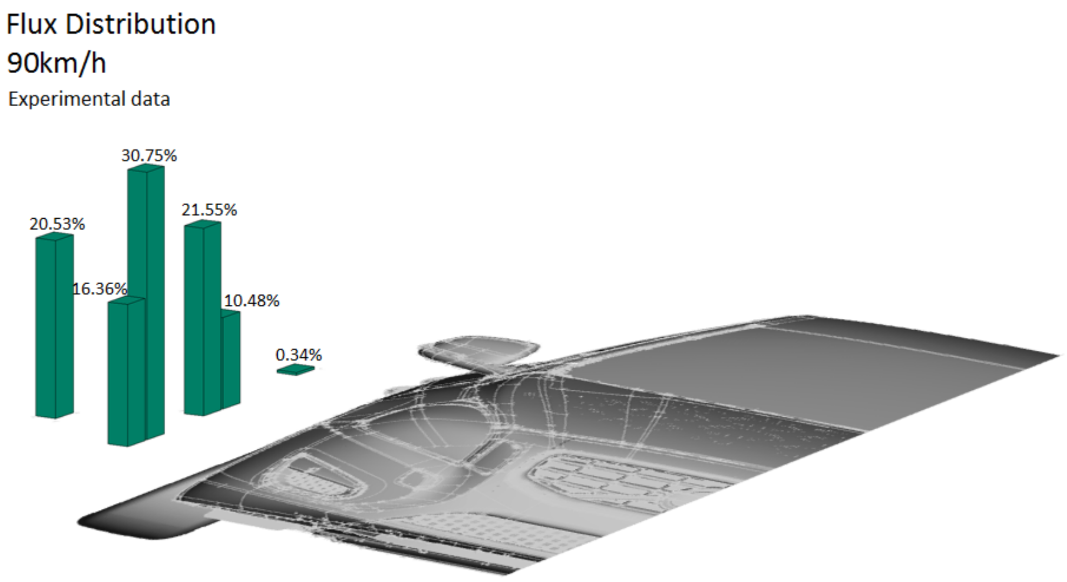


Figure 4.6: *Flux distribution in percentage under 90km/h driving speed.*

Implementation	50km/h	70km/h	90km/h
Particle density:	350kg/m^3	350kg/m^3	350kg/m^3
Reduced v_t**:**	9.8m/s	13.7m/s	13m/s
Coefficient C_y**:**	9.8	13	16
Drag law: spherical	spherical	spherical	spherical
Particle diameter:	0.6mm	0.6mm	0.6mm

Table 4.3: Implementation for particle injection under 50km/h, 70km/h and 90km/h driving speed.

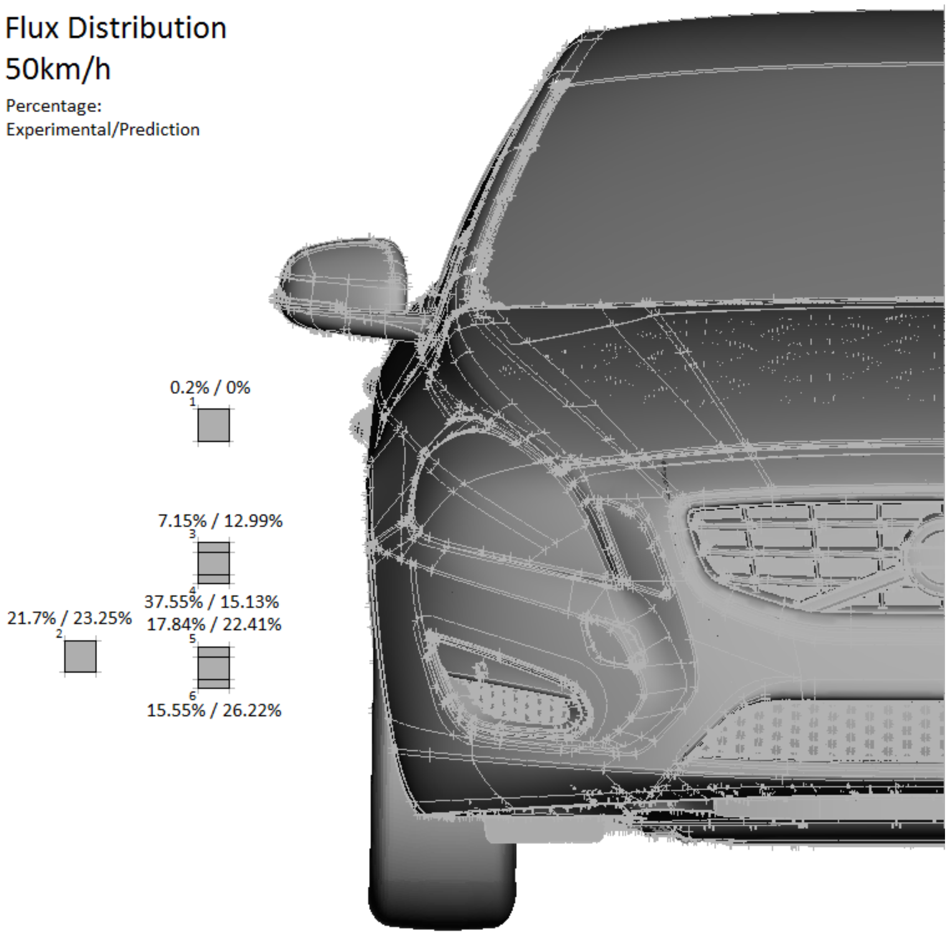


Figure 4.7: *Comparison of distribution between experimental and numerical results. The driving speed is* $50 km/h$.

4.3 Differences between experimental and numerical results

The differences between the experimental data and numerical prediction under different driving speeds are represented in Figure 4.7 - 4.12.

50km/h The difference between experimental and numerical results is presented in Figure 4.7, while the bar chart that represents the difference is presented in Figure 4.8.

70km/h The difference between experimental and numerical results is presented in Figure 4.9, while the bar chart that represents the difference is presented in Figure 4.10.

90km/h The difference between experimental and numerical results is presented in Figure 4.11, while the bar chart that represents the difference is presented in Figure 4.12.

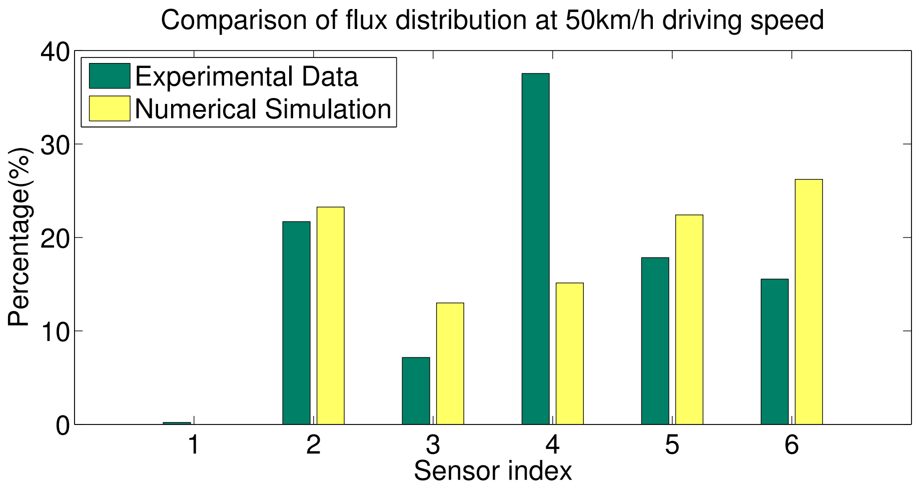


Figure 4.8: *Comparison of distribution between experimental and numerical results, bar chart. The driving speed is* 50km/h.

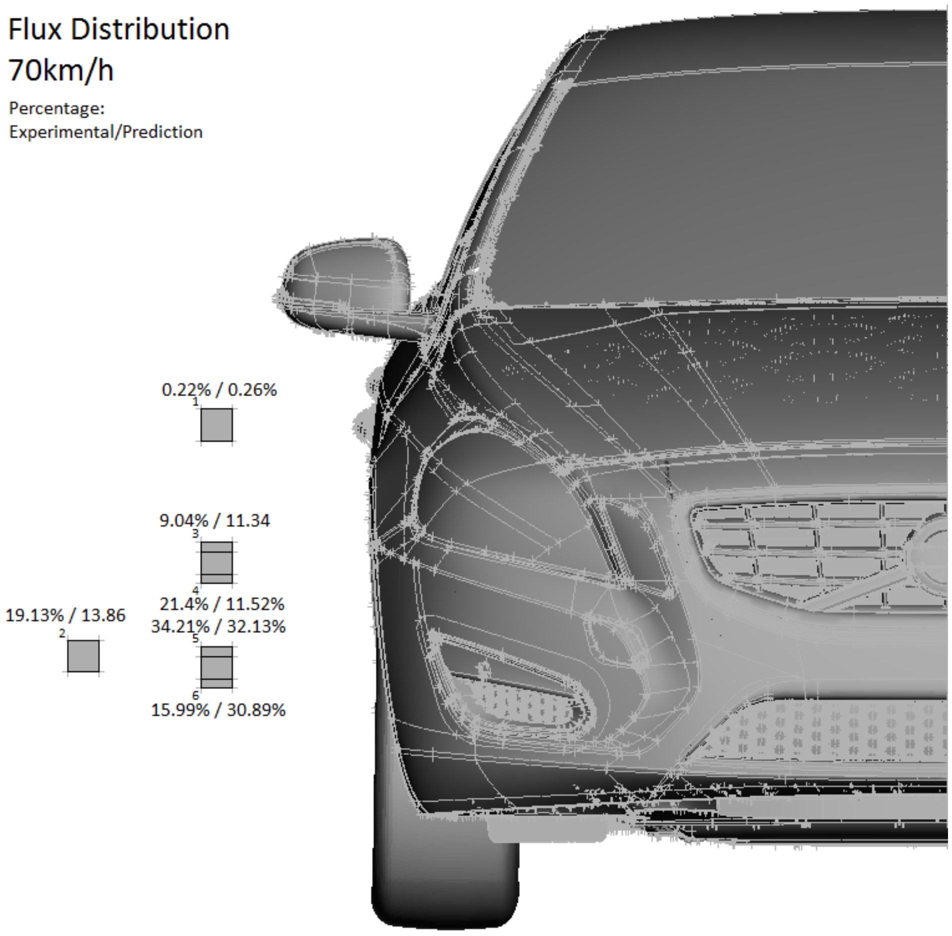


Figure 4.9: *Comparison of distribution between experimental and numerical results. The driving speed is* 70km/h.

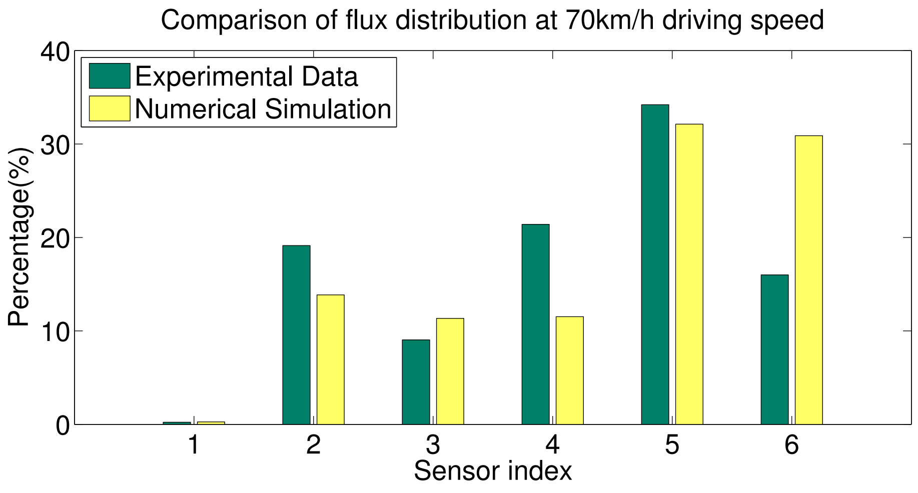


Figure 4.10: *Comparison of distribution between experimental and numerical results, bar chart. The driving speed is* 70*km/h.*

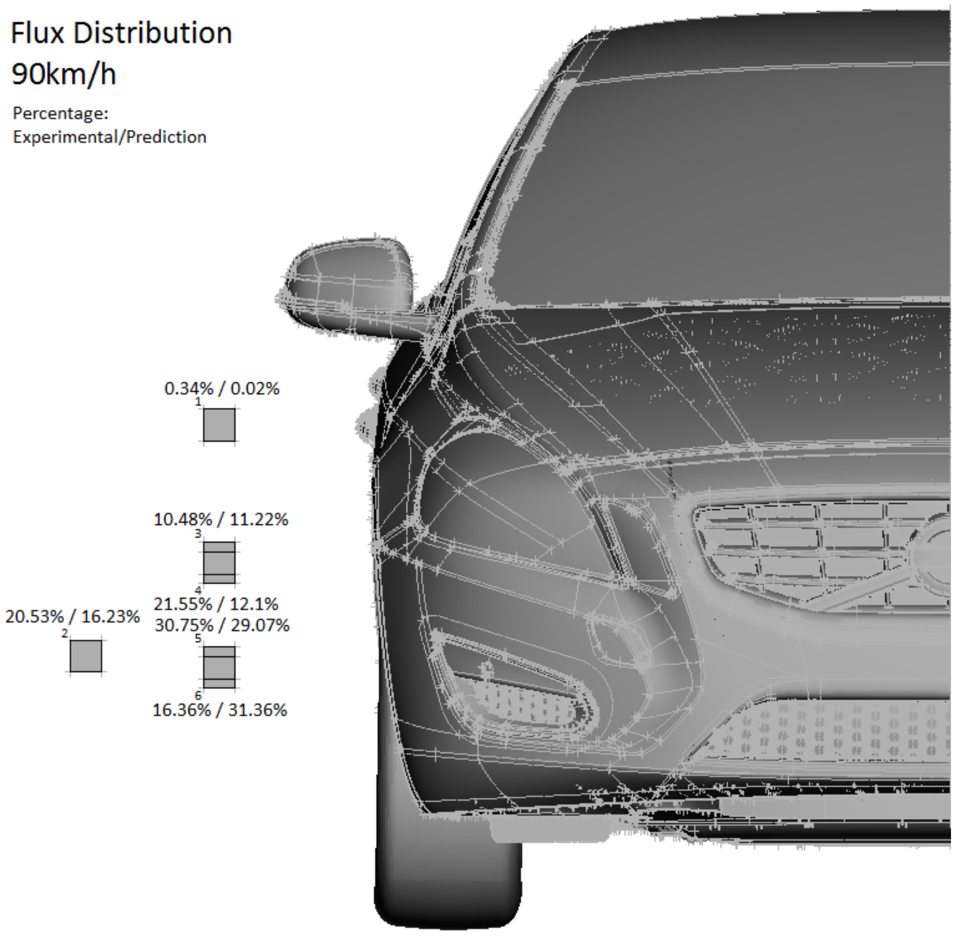


Figure 4.11: *Comparison of distribution between experimental and numerical results. The driving speed is* 90*km/h.*

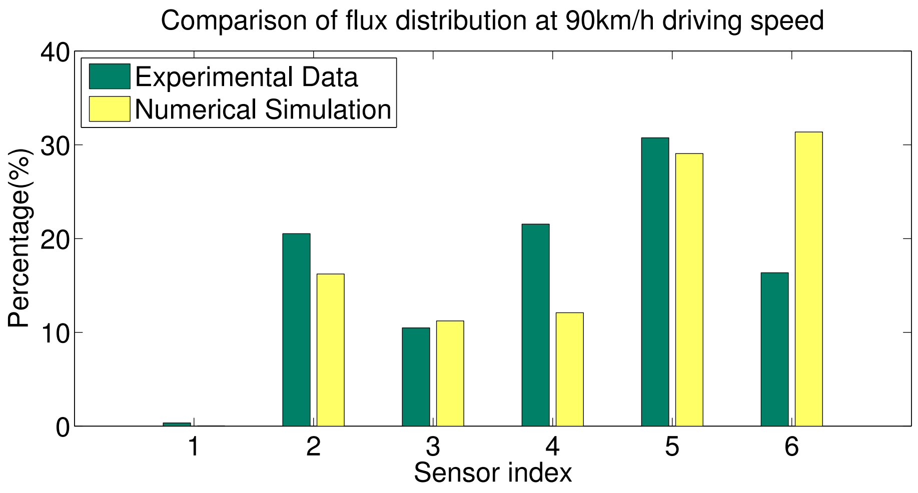


Figure 4.12: *Comparison of distribution between experimental and numerical results, bar chart. The driving speed is* $90km/h$.

5 Discussion

In this chapter, some note worthy problems during the process of this project are introduced and discussed with regard to previous chapters and sections.

5.1 The experimental data

The experimental data, obtained at Jokkmokk Proving Ground(JPG), is considered as the goal for the simulated results to reach. The data under different driving speeds is presented and compared between each other in Figure 5.1. There are several conclusions and regularities that can be pointed out:

- Precipitation at sensor 1 (which has the highest location) always possesses a very low distribution. This can be rather easily understood that most of the snow particles cannot fly high enough to reach that location;
- Precipitation at sensor 2 (which has the most outside location from the car) always has a flux distribution around 20% under all driving speeds. The flux distribution is therefore speed invariant;
- Precipitation at sensor 6 (which possesses the lowest location, nearest to the ground) remains a flux distribution around 15%;
- Precipitation at sensor 3 varies relatively larger, but not much, approximately between 7% and 10%.

However, there is one specific phenomenon from the experimental data that remains unclear, which can be found in all cases:

- At 50km/h driving speed, (see Figure 4.2), it can be pointed out that the flux distribution at sensor 4 has more than 5 times higher value than distribution at sensor 3;
- At 70km/h driving speed, (see Figure 4.4), it can be pointed out that the flux distribution at sensor 4 is twice as distribution at sensor 3; while distribution at sensor 6 is only a bit less than half of distribution at sensor 5;
- At 90km/h driving speed, (see Figure 4.6), it can be pointed out that the flux distribution at sensor 4 is a bit more than twice as distribution at sensor 3; while distribution at sensor 6 is a bit more than half of distribution at sensor 5.

There are several possible reasons to this behavior:

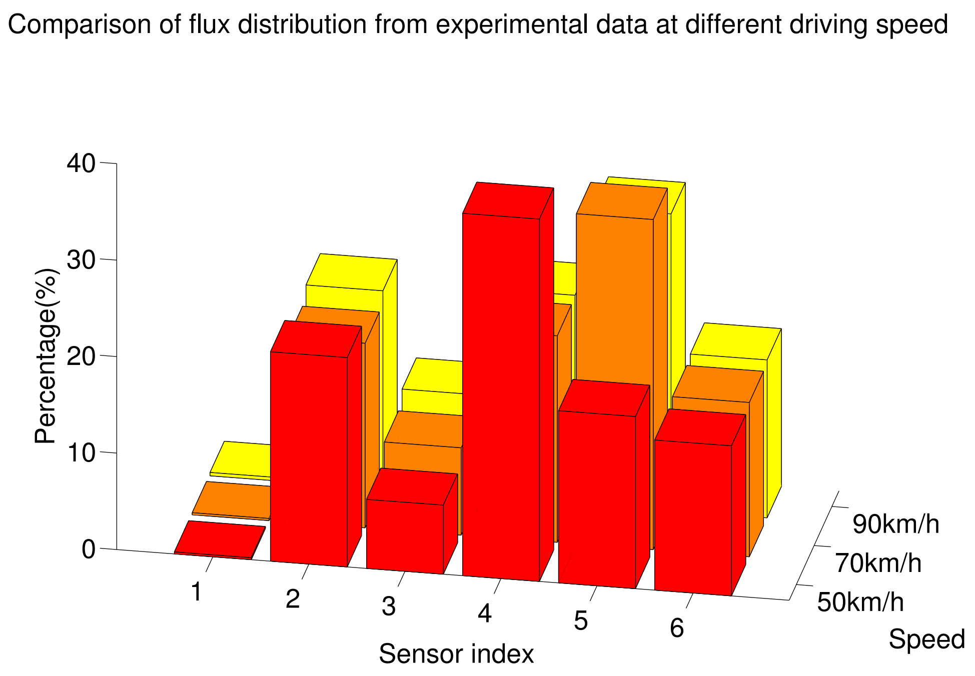


Figure 5.1: *All experimental data arranged with respect to sensor domains and driving speeds.*

- there is only one sensor, and these measurements regarding to different locations were undertaken in different trials(not measured at same time), therefore the snow conditions on the ground can be different and hence caused different scenarios of snow dust generation for each individual measurement;
- the sensor's body itself with the installation structure could influence the flow field, since the sensor body is a relatively large block of geometry, the flow was blocked and forced to change direction, not right in front of the sensor domain, but would still change the flow field through it;
- the distribution of snow dust at these sensor domains caused by the flow field behaves exactly as measurement data represented. This is the minimum possibility as considered (because the locations are extremely close while the data differ much more than thought) but such possibility cannot be eliminated.

Generally speaking, the flux difference between sensor 3 and sensor 4, difference between sensor 5 and sensor 6 are relatively big in most cases, while sensor 3 and 4, sensor 5 and 6 share 70% of their sensor domain areas. This big difference cannot be replicated from this particle trajectory prediction method development.

5.2 Comparison between the experimental data and the simulated results

The differences between the experimental data and the simulated distribution can be seen above in Section 4.3. From the figures it can be stated that there are specific sensor domains that give relatively dramatic differences between the experimental and numerical results: sensor 4 and sensor 6. At sensor 4, the numerical simulated data is always smaller than the experimental value; while at sensor 6, the numerical result gives higher distribution than the experimental data. This indicates that most of the snow particles flow at a relatively low height. However flux distribution at sensor 3, which locates only 15mm higher than sensor 4, possesses a very low flux distribution than sensor 4 does. An opposite situation can be found at sensor 6, that the simulated flux distribution is much higher than the experimental flux distribution. These differences are a part of the problem mentioned above in section 5.1, that the big differences between sensor 3 and 4, between sensor 5 and 6 cannot be replicated from this particle trajectory prediction method.

5.3 Tuning of the implementation

The implementation of the snow particle generation has to be tuned to meet the measurement data. During the tuning process, there are several variables (see Section 3.9.1) that are tuned to influence the flux distribution from the numerical calculation. These variables with their influences are listed below:

- Constant C_y in Equation 3.1: this constant gives an effective contribution to increase the flux distribution at sensor 2, which indicates that this constant C_y will make the particle spread more outwards from the car exterior body.
- Diameter of particles: the change of diameter of particles leads to the change of mass of particles. Within a certain range, the increasing diameter gives a higher flux distribution at sensor 1, 3 and 4, which are located relatively higher above the ground, since a heavier particle with the same speed can travel a relatively longer range. However when the diameter exceeds a specific value, the result becomes the opposite.
- Density of material: this variable has a similar effect compared with the diameter of particles. It has to be pointed out that the final value, 350kg/m^3, is very close to the value given from a previous study, which provides a density range from 390kg/m^3 to 420kg/m^3 at Volvo Climate Wind Tunnel[17].
- Shape coefficient: this affects the path of the particles. With lower coefficient (which indicates a more skewed particle shape), the particle will follow the flow more, which can also be treated as higher drag force.

In the practical tuning process, the drag law is kept as *spherical* and diameter is kept as 0.6mm, under most circumstances to simplify the tuning process.

5.4 Simplified model

In practical experimental conditions, there are many factors that can influence the behavior of the measurement data. However the implemented model for numerical simulation is, to large extent, simplified. Factors that can influence the measurement yet not included in the implementation are listed below:

- The snow depth on the ground can vary at different locations and the amount of snow particles generated by splashing can be highly unstable while the car is running. The various snow depth is not considered in this project;
- The temperature model (energy equation) is not included in the implementation, there are no temperature dependence for densities(for both snow particle and air) and no diffusion;
- The vortex shedding and other unsteady effects for a running car are not implemented or cannot be represented in the steady-state simulation;
- The tyre pattern is not contained and the tyre surface is considered as a smooth surface. The packed snow pattern on the ground after the wheel rolled over is not implemented in the model either;
- The snow particles generated by the flow over the snow ground(or particles dragged by the wake of the car) are not studied or implemented in this project;
- Particles generated from only part of the right, front wheel is studied, since the particle generation at the rear wheel is not of interest and the measurement instrument is only located at the right side of the car;
- How and where the snow particles are packed or accumulated on the surfaces of different components of the car is beyond scope of this project;
- The discrete snow particles have no interaction with the continuous flow field, nor with other snow particles(particle-particle interaction). The particle will not coalesce or brake up;
- The size distribution of snow particles is not implemented, a uniformly distributed particle size is considered;

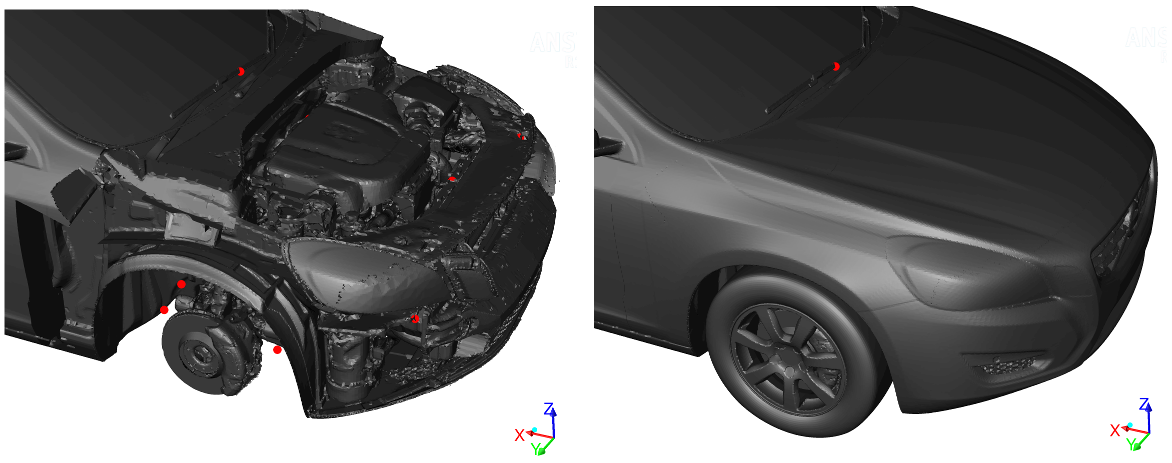


Figure 5.2: *The steady state solution under 50km/h driving speed. The red dots represent the cells with a velocity higher than 30m/s.*

- Other possible yet unknown scenarios.

These listed factors have various contribution to the experimental data. The influence from these factors are excluded from the numerical implementation is unknown.

5.5 General observations of the steady state flow field

5.5.1 Quality of the converged solution

The steady state simulation converged relatively stable after 2500 iterations, however there is a certain amount of cells that contain higher velocities, which is because the simulation is not completely well converged. An example can be shown in Figure 5.2 under a 50km/h case, where there are around 25 cells that possess velocities higher than 30m/s. Those cells locate mostly inside the engine bay, which will not give a big influence of the exterior flow field. Therefore the result is acceptable.

5.5.2 Velocity contour

The velocity contour at symmetry plane, a cut plane close to the ground and the right front tyre is shown in Figure 5.3. In the figure it can be seen that a higher velocity field is formed at the lower front tyre region, generating a relatively complex vortex at the rear of the front wheel. Another strong vortex is formed at a region that is both close to the ground and the tyre. These vortexes can influence the injected snow particles motion in a relatively more complex process. At the top of the car the flow is also accelerated, due to the flow expansion.

5.6 Mesh sensitivity

Mesh is a critical issue for numerical simulation. A high quality mesh can not only increase the numerical stability but also offer better convergence with shorter computational time. In this project, the mesh is generated in **Harpoon** and the number of cells is around 62 million and a large quantity of the cells are $4mm$ or below as size. There are several methods to improve the quality of mesh and points during the project that need to be pointed out and discussed.

5.6.1 Smooth mesh with target skew

The mesh quality with respect to skewness is controlled in **Harpoon** using *smooth* command as the following:

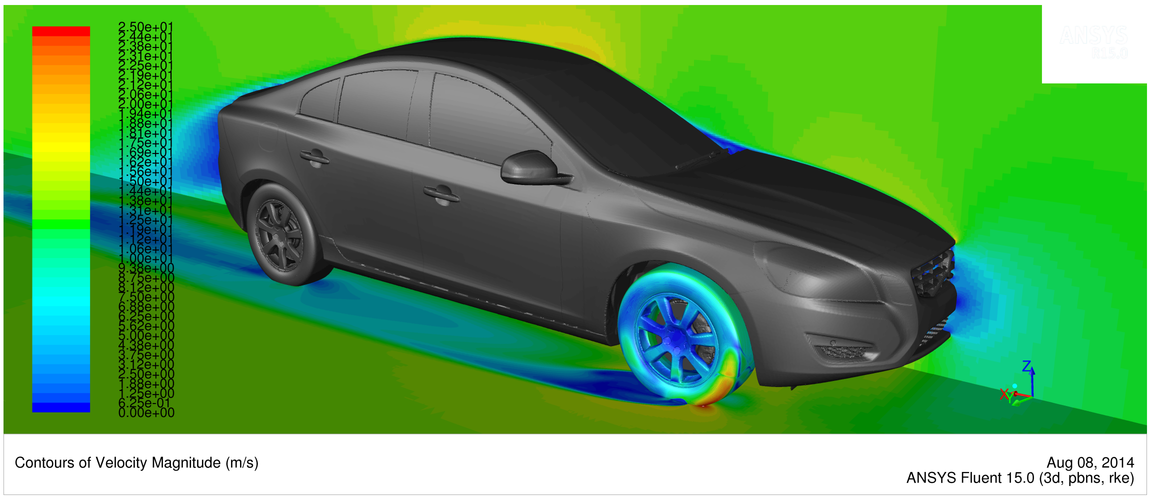


Figure 5.3: *The velocity contours at different planes and surfaces over the car.*

	Sensor 1	Sensor 2	Sensor 3	Sensor 5
With refinement boxes	0	2437	1362	2349
Without refinement boxes	0	2781	1623	2012

Table 5.1: Difference of flux distribution between a case *with* refinement boxes and a case *without* refinement boxes. The driving speed is 50km/h.

smooth 2 0.98
smooth 2 all
smooth 2 0.98

Harpoon will first smooth all cells with target skew of 0.98 twice, and then smooth all cells twice and finally smooth all cells with target skew of 0.98 twice again. In this way the mesh quality can be controlled or improved, if the numerical simulation is not well converged or results are not desirable. It must be pointed out that even a target skew of 0.98 smooth is set, not all cells with skewness higher than 0.9 will be refined. There will be still a number of mesh with skewness higher than 0.98. The functionality of this *smooth* command is therefore questionable.

5.6.2 Finding volumes on surfaces

Command *vfind* will check if there are volume mesh or cells that penetrate specific surfaces. It is important to run *vfind* over all surface mesh to avoid volume mesh inside the boundary, which will lead to skewed cells. During the project these cells cause serious convergence problems, even though the bad cells were located and refinement boxes with much higher quality cells were applied. In this case **Fluent** reports with message "*turbulent viscosity limited to viscosity ratio of 1.00000+e05 in 355 cells*", this message will not disappear as expected and has the possibility to finally cause numerical divergence, which is usually caused by cells with low quality or high skewness.

5.6.3 Refinement boxes for particle trajectories

The refinement boxes are important in order to predict the trajectories, since a bigger cell generates larger misalignment with respect to trajectory prediction. And with a group of many big cells, the misalignment can be accumulated and the trajectories will be very poorly predicted. In the project, the refinement boxes have a $4mm$ size as cell size. Smaller cell size will cause high number of cells (with $2mm$ size on refinement boxes, the number of cells will exceed 100 million) that can be applied if there are enough computational resources. A steady state flow field without the refinement boxes for particle trajectories is applied with stochastic random walk particle tracking and the difference is presented in Table 5.1. From the table it can be seen that without the refinement boxes the prediction provides a flux distribution with higher value at sensor 2, sensor 3 and lower value at sensor 5. This indicates a decreased accuracy at sensor 2, sensor 3, and a increased accuracy at sensor 5. However, there is no clear proof whether the refinement box is a preliminary requirement or not, since a tuning process may make the result more ideal.

Different shapes of refinement boxes There are several different types of shape for refinement box in **Harpoon**. The distribution of snow trajectories is approximated based on observations at the beginning of the project and the refinement boxes are defined(here is cylinder as box type) based on this approximation. When the trajectories are calculated and the understandings of trajectories are more complete, the refinement boxes can be improved so that trajectories will be contained inside the refinement boxes as much as possible, while the refinement box generate less cells. In this way the particle trajectories can be simulated with relatively high accuracy with a minimum number of cells to offer a better computational efficiency.

5.7 Particle trajectories

Besides the mesh size discussed above in Section 5.6.3, there are other factors that can influence the particle trajectory.

5.7.1 Particle force

The particle trajectory is based on the forces that are applied on the particles. Therefore it is critical to account for all possible forces while keeping the calculation effective, that means forces that have little impact on particle motion can be neglected to some extent. In this project only drag force is applied, other forces either have very little impact, or are not suitable for snow particles.

5.7.2 Stochastic Random Walks model

Step length factor The step length factor is set as 8. In this project, a higher step length factor makes the particles flow a longer range while keeping a larger height. A small step length factor (take 1 as an example), makes the particles fall downwards more quickly.

Number of tries There are around 90000 single injection points over the injection surface, therefore the number of tries should be limited within a reasonable value. Here the values is 5, which results totally around 450000 injections.

Eddy life time Because of the lack of understanding of eddy life time, the *random eddy life time* is applied with a constant of 0.15, which is recommended in the **Fluent User Guide**[14].

5.7.3 Surface injection

The snow dust particles detached from the tyre surfaces when a car is moving is implemented as injected particles in **Fluent** as it has been discussed in previous sections. However, whether the particles are generated from the *entire* tyre surface or *part* of the tyre surface cannot be decided completely. Either how much of the part of the entire tyre is uncertain, because of the uncertainty from a visual observation. This will furthermore decide the shape of the refinement boxes as well.

5.8 Error source

5.8.1 Measurement data: huge fluctuations

The measurement data is taken from the precipitation sensor and stored in the data files. Mean values are extracted from the data, see Figure 3.1 as an example. From this example it can be seen that the data is highly fluctuated and irregular with respect to precipitation (mm/min). The mean value can be considered as usable but a high error margin exists and it cannot be estimated unless more measurements can be preformed.
The precipitation sensor can also operate without a proper working condition, for example the sensor domain can be blocked and covered by the snow. This will give a maximum reading from the data for the entire time, which has been proved in several measurements, see Figure 5.4. The sensor also has a effective detecting range

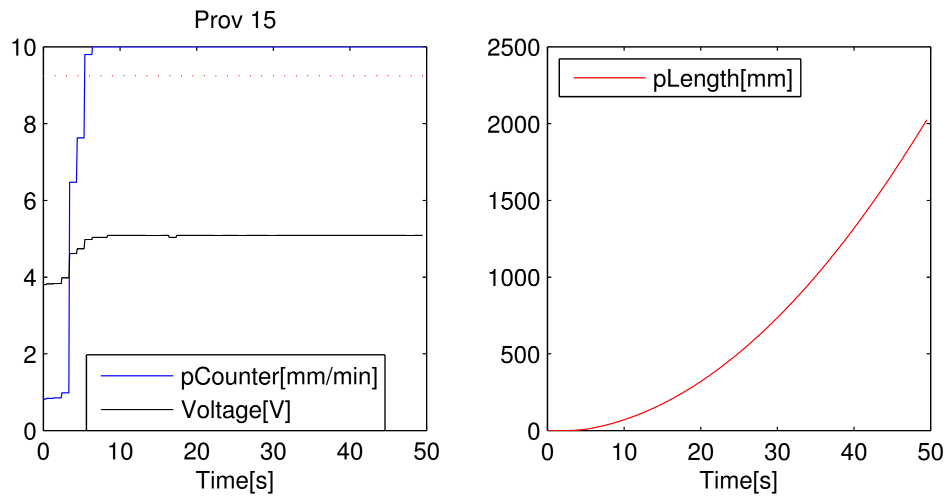


Figure 5.4: *A test case that the precipitation sensor reaches a maximum measuring range(5[V]) under most of the measuring time, because the sensor domain is blocked by the snow.*

of droplet size above 0.2mm. This can give uncertainties when particles below this size is passing through the sensor domain as well.

5.8.2 Implementation based on visual observations from video logs: visual inaccuracy

Parts of implementation of the numerical simulation are based on the visual observations from the videos(defined injection angles and patched area), which were filmed while measurements were undertaken. As it has been introduced in previous sections, implementation of particle injection (injection angles, directions and velocities) is based on a visual observation from the videos. To decide the angles in this way is no doubt not a reliable method and large inaccuracy exists. Even though this is used as an initial assumption and the implementation is tuned thereafter, this will contribute inaccuracies and cannot be neglected. Also there is lens distortion and resolution issues that can influence the visual observation.

5.8.3 Snow condition: various properties

As mentioned above in section 5.4 and 2.2, in realistic situations, the snow particle has a varying density with varying environment temperature. The particles can also coalesce and collide with other particles. These factors will cause properties change and affect the trajectories and precipitation in reality. While in the project these factors are neglected and not contained in implementation. Thus only part of the generated snow particles are simulated, this will not represent everything happening and tuning in the post-process lacks accurate motivations.

5.8.4 Numerical calculation: various simplifications with errors

Since the numerical method is to discretize the partial differential equations to linear equations with different techniques, this method itself will give numerical errors more or less.

Turbulence model In this project, the standard k-ε turbulence model is applied. Compared with other turbulence models in **Fluent**(e.g. k-ε RNG, k-ε realizable, k-ω SST), the standard k-ε has a relatively poor prediction at the free stream region, while other models have better prediction, as it has been discussed in section 2.5. In this project, a well-converged steady-state flow field is achieved with the realizable k-ε turbulence model. Therefore all particle trajectory predictions are based on this k-ε model. Higher precision of the steady flow field or better predictions of the trajectories can be expected with other turbulence models.

6 Conclusion

According to the results and discussions above, some conclusions can be stated:

- The flux distribution from the experimental data can partly be replicated from a numerical simulation. The numerical simulated flux distribution at sensor 1, 2, 3, 5 can be reproduced with relatively higher accuracy;
- Some characteristics are consistent between realistic tests and numerical prediction. The general trend can be formulated as the following: distribution that the sensors located higher above the ground possess smaller flux distribution, while the sensors located lower near the ground possess bigger flux distribution. This trend is represented in both experimental data and numerical simulation;
- There is one specific characteristic that cannot be replicated in this project: the big difference between 2 sensors that are located very close to each other (sensor 3 and 4; sensor 5 and 6). The numerical prediction gives a more even distribution.

Those conclusions above indicate that the performance of this developed method for particle trajectory prediction is acceptable.

7 Future work

The future work can be an extension of this method development. For further consideration, several possible plans and scenarios are listed below:

- More work can be put to refine the accuracy of the prediction of this flux distribution. Certain optimization procedure can be implemented. A possible formulation can be: the difference between the simulated flux distribution and the measured flux distribution can be the objective to minimize or optimize; certain factors (such as constant C_y, diameter of particles, and other tuning variables discussed in Section 5.3) can be defined as the variables that define the function domain. This optimization process can provide the opportunity to greatly speed up the tuning process and obtain more ideally simulated results.
- The quality of the steady-state simulation can be refined. This can be done either by improving the geometry model in **ANSA**, by improving the volume mesh quality in **Harpoon**, or by choosing better simulation settings(more accurate turbulence models, higher order schemes, etc.). The residuals can be lowered down by running more iterations.
- If the computational resource is available, more complex models can be implemented, such as a transient simulation can be performed to provide a more detailed flow scenario. The transient turbulence model can either be URANS, DES or LES. The flowing particles affected by vortex shedding generated from the car can provide more detailed observations and better understandings of this *self generated snow dust* phenomenon. Two way coupling of particles and other possible body force that are applied on the particle can also be implemented. Any other possible physical models that can affect the results are also welcome to be implemented in.
- More variables that are not included in this project (such as various diameter of particles, which are mentioned in Section 5.4) can be defined.
- The experimental method can also be improved. Possible improvements can be such as: more evenly distributed locations of the sensor, more circumstances with different driving speeds, more advanced measuring techniques, etc.
- This project, that simulating the snow particles has a certain amount of similarities compared with the dust generation from the car, which means that when the car is running on the dusty road, the dust particles will also be lifted up, just as the way the snow particles are generated. This leads to a possibility that this specific method for particle trajectory prediction of snow dust can also be used to predict the dust particles. Under the tuning process(mentioned in Section 5.3), similarities are observed with respect to dust flux distribution under with certain specific cases, which provides a strong support that this method is capable for both dust and snow dust simulations.
- The snow accumulation and its positive/negative affections to the performance of different components of the car can be a fairly long scope of this project. However there are studies and simulations are done, that a governing equation of snow particle accumulation is implemented with a Eulerian-Eulerian approach[18]. In this way it is also possible to predict the accumulated snow particles at different locations, numerically. There are other studies about snow drifting which also make it possible.[19]

Bibliography

[1] J.A.Pomeroy, D.H.Male, Steady-state suspension of snow. *Journal of Hydrology*, 135, p.275-301, 1972.

[2] M.Gordon, S.Savelyev, P.A.Taylor. Measurements of blowing snow, part II: Mass and number density profiles and saltation height at Fraklin Bay NWT Canada. *Cold Regions Science and Technology*, 55, p.75-85. 2009.

[3] T.K.Thiss. Large scale studies of development of snowdrifts around buildings. *Journal of Wind Engineering and Industrial Aerodynamics*, 91, p.829-839. 2003.

[4] J.W Pomeroy, E.Brun. Physical Properties of snow. *Snow Ecology: an Interdisciplinary Examination of Snow-covered Ecosystems*, p.45-118. 2001.

[5] R.A.Schmidt. Estimates of threshold windspeed from particle sizes in blowing snow. *Cold Region Science and Technology*, 4, p.187-193. 1981.

[6] M.Gordon, P.A.Taylor. Measurements of blowing snow Part I Particle shape size distribution velocity and number flux at Churchill Manitoba Canada. *Cold Regions Science and Technology*, 55, p.63-74. 2009.

[7] R.A.Schmidt. Measuring particle size and snowfall intensity in drifting snow. *Cold Region Science and Technology*, 9, p.121-129. 1984.

[8] Visual Acuity of the Human Eye. `https://www.nde-ed.org/EducationResources/CommunityCollege/PenetrantTest/Introduction/visualacuity.htm`

[9] `http://www.thiesclima.com/precipitation-sensor.html`

[10] B.Marshall, P.Paul, *Handbook of Computational Geometry.* Elsevier Science. 2000.

[11] L.Davidson, *Fluid mechanics, turbulent flow and turbulent modelling*, Division of Fluid Dynamics. Gothernburg. 2012.

[12] ANSYS Fluent Theory Guide. Release 15.0. 2013.

[13] C.T.Crowe, J.D.Schwarzkopf, M.Sommerfeld and Y.Tsuji, *Multiphase Flows with droplets and particles,* second edition, CRC Press. 2012.

[14] ANSYS Fluent User Guide. Release 15.0. 2013.

[15] Harpoon. User Guide for Version 5.2.

[16] ANSYS Fluent Text Command List. 2013

[17] Engine Air Intake Optimization for Winter Driving. D.Engebretsen. Master's Thesis. 2006:28.

[18] Y.Tominaga, T.Okaze, A.Mochida. CFD modeling of snowdrift around a building An overview of models and evaluation of a new approach. *Building and Environment*, 46, p.899-910. 2011.

[19] M.Beyers, B.Waechter. Modeling transient snowdrift development around complex 3D structures. *Journal of Wind Engineering and Industrial Aerodynamics*, 96, p.1603-1615. 2008.

Appendices

A PID list of the car model

```
fan-coolpack-cac-in-04mm
fan-coolpack-cac-out-04mm
fan-coolpack-cond-in-04mm
fan-coolpack-cond-out-04mm
fan-coolpack-fan-large-in-04mm
fan-coolpack-fan-large-out-04mm
fan-coolpack-fan-shroud-gap-04mm
fan-coolpack-fan-small-in-04mm
fan-coolpack-fan-small-out-04mm
fan-coolpack-flaps-04mm
fan-coolpack-rad-in-04mm
fan-coolpack-rad-out-04mm
fan-eng-bay-AIS-intake-04mm
fan-eng-bay-AIS-suction-inlet-04mm
fan-exterior-close-grille-02mm
fan-exterior-close-spoiler-02mm
if-cac-ds-fan-04mm
if-cac-ds-sides-04mm
if-cac-us-fan-04mm
if-cac-us-sides-04mm
if-cond-ds-fan-04mm
if-cond-ds-sides-04mm
if-cond-us-fan-04mm
if-cond-us-sides-04mm
if-rad-ds-fan-04mm
if-rad-ds-sides-04mm
if-rad-us-fan-04mm
if-rad-us-sides-04mm
wall-coolpack-accessories-04mm
wall-coolpack-cac-sides-04mm
wall-coolpack-cac-tanks-04mm
wall-coolpack-cond-sides-04mm
wall-coolpack-cond-tanks-04mm
wall-coolpack-fan-blade-large-04mm
wall-coolpack-fan-blade-large-rot-04mm
wall-coolpack-fan-blade-small-04mm
wall-coolpack-fan-blade-small-rot-04mm
wall-coolpack-fan-shroud-04mm
wall-coolpack-fan-shroud-large-04mm
wall-coolpack-fan-shroud-small-04mm
wall-coolpack-flaps-04mm
wall-coolpack-rad-sides-04mm
wall-coolpack-rad-tanks-04mm
wall-eng-bay-corse-04mm
wall-eng-bay-finest-04mm
wall-eng-bay-medium-04mm
wall-eng-bay-fine-04mm
wall-exterior-body-04mm
wall-exterior-front-04mm
wall-exterior-grille-mesh-04mm
wall-exterior-spoiler-mesh-04mm
wall-exterior-wipers-04mm
wall-floor-front-08mm
wall-floor-front-panels-08mm
wall-floor-front-spoiler-08mm
wall-floor-front-undershield-08mm
wall-floor-fuel-system-08mm
wall-floor-fwd-08mm
wall-floor-heat-shields-08mm
wall-floor-heatshield-tunnel-08mm
wall-floor-pannels-08mm
wall-floor-rear-2WD-08mm
wall-floor-wheelhouse-front-08mm
wall-floor-wheelhouse-rear-08mm
wall-powertrain-04mm
wall-powertrain-04mm
```

```
wall—powertrain—cd—plate—I5D—std—08mm
wall—powertrain—drive—shaft—08mm
wall—powertrain—exhaust—08mm
wall—suspension—front—brake—disc—04mm
wall—suspension—front—brake—disc—omega—04mm
wall—suspension—front—coarse—08mm
wall—suspension—front—medium—08mm
wall—suspension—rear—08mm
wall—suspension—rear—brake—disc—08mm
wall—suspension—rear—brake—disc—omega—08mm
wall—wheel—rear—rim—stationary—04mm
wall—floor—underbody—heatshield—08mm
wall—wheel—rear—tyre—omega—02mm
wall—wheel—rear—rim—omega—04mm
wall—wheel—front—tyre—omega—01mm
wall—wheel—front—rim—stationary—04mm
wall—wheel—front—rim—omega—04mm
wall—powertrain—coarse—08mm
fan—preceptioncensor1—04mm
fan—preceptioncensor2—04mm
fan—preceptioncensor3—04mm
fan—preceptioncensor4—04mm
fan—preceptioncensor5—04mm
fan—preceptioncensor6—04mm
```

B Harpoon Configuration file

Lines start with "**" character are commented.

```
import tgrid ../HARPOON/surf0717a.msh
baselev 32
farfield global
farfield xmin —19000
farfield ymin —4700
farfield zmin 172
farfield xmax 30850
farfield ymax 4700
farfield zmax 10000
**refine    Start kommando
**0 3       Position 1: Box typ, 0=rectangular, Position 2: Size of  mesh 3 = 4 mm (med ...
    baselevel 32), 1—16mm
** 700 —450 350 Koordinat: xmin, ymin, zmin
** 1250 450 950 Koordinat: xmax, ymax, zmax
**COOLPACK
**refine
**0 3
**900 —475 350
**1250 475 950
**SNOWY
** 1 CYLINDERS
refine
3 3
1450 950 445
4800 1150 510
1450 880 220
4800 880 800
**refine
**3 3
**1740 675 200
**4800 570 225
**1740 700 220
**4800 686 350
** 1 RECTANGULAR
refine
0 3
1740 909 172
```

```
4481 659 320
**AROUND CAR
refine
0 0
-500 -1500 0
8000 1500 2000
**MELLANLIGGANDE
refine
0 -1
-800 -2500 0
10000 2500 2500
type hex
expand slow
mesh external
volume -3
remove
level 1
gminlev 1
gmaxlev 5
plevel *01mm 6 6 0
plevel *02mm 5 5 0
plevel *04mm 4 4 0
plevel *08mm 3 3 0
plevel *16mm 2 2 0
vfind all
** DELETE COUPE VOLUME
vptkeep 0 0 5000 ** YTTRE FLUID
**vptkeep 1113 89 777 ** KYLPAKETET
vptrename 0 0 5000 fluid-main
**vptrename 1113 89 777 fluid-1-mrf
smooth 2 0.98
smooth 2 all
smooth 2 0.98
** SET BOUNDARY CONDITIONS ALL FAN TO radiator FOR MEASUREMENT
**setbc fan-eus-* radiator
setbc fan-coolpack-* radiator
setbc fan-exterior-* radiator
setbc fan-precipitationcensor* radiator
**setbc fan-wheelhouse-* radiator
setbc farfield_maxx pressure-outlet
setbc farfield_minx velocity-inlet
setbc farfield_maxy symmetry
setbc farfield_miny symmetry
setbc farfield_maxz symmetry
**save harpoon ../HARPOON/harpoon_volmesh.hrp
export fluent vol ../HARPOON/0717a.msh
**harpoon -Ver 5.2a7 -fillhole 10 -UTM -oldutm -batch harpoon.cfg -omp 4
```

C Fluent Journal file

Lines start with ";;" character are commented. For more details, see Fluent Text Command List.[16]

C.1 Steady state case

```
/file/set-batch-options yes yes no
rc ../../HARPOON/0717a.msh          ;Edit filename (%%.gz?%%)
/grid/scale/ 0.001 0.001 0.001
/solve/set/gradient-scheme/
yes ; Options:Green-Gauss Cell-Based
/define/models/viscous/ke-realizable?
yes
;/define/models/viscous/turbulent-expert/rke-cmu-rotation-term?
;yes
/define/models/viscous/near-wall-treatment/enhanced-wall-treatment?
```

```
no
/define/models/solver pressure-based
yes
/define/boundary-conditions/modify-zone

zone-type farfield_minx velocity-inlet
zone-type farfield_maxx pressure-outlet
zone-type farfield_miny symmetry
zone-type farfield_maxy symmetry
zone-type farfield_maxz symmetry

/define/boundary-conditions/velocity-inlet
farfield_minx ;zone id/name
no            ;velocity specification method: magnitude and direction
no            ;                               components
yes           ;                               magnitude, normal to boundary
yes           ;reference frame: absolute
no            ;use profile for velocity magnitude?
13.8888888888 ;velocity magnitude(m/s)
no            ;use profile for supersonic/initial gauge pressure?
0             ;supersonic/initial gauge pressure (pascal) (%%ignored for subsonic%%)
no            ;turbulent specification method: k and epsilon %%
yes           ;                                 intensity and length scale
2.16          ;turbulent intensity(%)
0.1           ;turbulent length scale(m)
;no            ;use profile for turbulent kinetic energy?
;0.1           ;turbulent kinetic energy (m2/s2)
;no            ;use profile for turbulent dissipation rate?
;200           ;turbulent dissipation rate (m2/s3)

/define/boundary-conditions/pressure-outlet
farfield_maxx ;zone id/name
no            ;use profile for gauge pressure?
0             ;gauge pressure
no            ;backflow direction specification method: direction vector
yes           ;                                          normal to boundary
no            ;turbulent specification method: k and epsilon
yes           ;                                 intensity and length scale %%
;yes           ;                                 intensity and viscosity ratio
3
0.1
;0.1           ;backflow turbulent intensity (%)
;5             ;backflow turbulent viscosity ratio
no            ;radial equilibirium pressure distribution
no            ;average pressure specification?
no            ;specify targeted mass flow rate
;DEFINE THE MOVING GROUND
/define/boundary-conditions/wall
farfield_minz    ;zone-id/name
yes              ;wall motion[motion-bc-stationary]: change current value?
motion-bc-moving ;
no               ;shear boundary condition[shear-bc-noslip]
yes              ;define wall motion relative to adjacent cell zone?
no               ;apply a rotational velocity to this wall?
13.8888888888    ;velocity magnitude (m/s) [0]  ;19.4444444444, 25
1                ;x-component of wall translation
0                ;y-component of wall translation
0                ;z-component of wall translation
no               ;define wall velocity components?
no               ;use profile for wall roughness height?(without wall-enhance treatment..)
0
no
0.5
;DEFINE THE ROTATING WHEELS: FRONT
/define/boundary-conditions/wall
wall-wheel-front-rim-omega-04mm    ;zone-id/name
yes              ;wall motion[motion-bc-stationary]: change current value?
motion-bc-moving ;
no               ;shear boundary condition[shear-bc-noslip]
yes              ;define wall motion relative to adjacent cell zone?
yes               ;apply a rotational velocity to this wall?
```

```
no                ;define wall velocity components?
no                ;use profile for wall roughness height?
0                 ;wall roughness height(m)
no                ;use profile for wall roughness constant?
0.5               ;wall roughness constant
43.6308992074     ;rotational speed(rad/s) ;60.3430901214, 77.5839730132
1.70714           ;x-position of rotation-axis origian(m)
0                 ;y-position of rotation-axis origian(m)
0.4784505         ;z-position of rotation-axis origian(m)
0                 ;x-component of rotation-axis direction
-1                 ;y-component of rotation-axis direction
0                 ;z-component of rotation-axis direction
/define/boundary-conditions/wall
wall-wheel-front-rim-stationary-04mm    ;zone-id/name
yes               ;wall motion[motion-bc-stationary]: change current value?
motion-bc-moving ;
no                ;shear boundary condition[shear-bc-noslip]
yes               ;define wall motion relative to adjacent cell zone?
yes                ;apply a rotational velocity to this wall?
no                ;define wall velocity components?
no                ;use profile for wall roughness height?
0                 ;wall roughness height(m)
no                ;use profile for wall roughness constant?
0.5               ;wall roughness constant
43.6308992074     ;rotational speed(rad/s)
1.70714           ;x-position of rotation-axis origian(m)
0                 ;y-position of rotation-axis origian(m)
0.4784505         ;z-position of rotation-axis origian(m)
0                 ;x-component of rotation-axis direction
-1                 ;y-component of rotation-axis direction
0                 ;z-component of rotation-axis direction
/define/boundary-conditions/wall
wall-wheel-front-tyre-omega-02mm    ;zone-id/name
yes               ;wall motion[motion-bc-stationary]: change current value?
motion-bc-moving ;
no                ;shear boundary condition[shear-bc-noslip]
yes               ;define wall motion relative to adjacent cell zone?
yes                ;apply a rotational velocity to this wall?
no                ;define wall velocity components?
no                ;use profile for wall roughness height?
0                 ;wall roughness height(m)
no                ;use profile for wall roughness constant?
0.5               ;wall roughness constant
43.6308992074     ;rotational speed(rad/s)
1.7071            ;x-position of rotation-axis origian(m)
0                 ;y-position of rotation-axis origian(m)
0.4784505         ;z-position of rotation-axis origian(m)
0                 ;x-component of rotation-axis direction
-1                 ;y-component of rotation-axis direction
0                 ;z-component of rotation-axis direction
/define/boundary-conditions/wall
wall-wheel-front-tyre-omega-right-01mm    ;zone-id/name
yes               ;wall motion[motion-bc-stationary]: change current value?
motion-bc-moving ;
no                ;shear boundary condition[shear-bc-noslip]
yes               ;define wall motion relative to adjacent cell zone?
yes                ;apply a rotational velocity to this wall?
no                ;define wall velocity components?
no                ;use profile for wall roughness height?
0                 ;wall roughness height(m)
no                ;use profile for wall roughness constant?
0.5               ;wall roughness constant
43.6308992074     ;rotational speed(rad/s)
1.7071            ;x-position of rotation-axis origian(m)
0                 ;y-position of rotation-axis origian(m)
0.4784505         ;z-position of rotation-axis origian(m)
0                 ;x-component of rotation-axis direction
-1                 ;y-component of rotation-axis direction
0                 ;z-component of rotation-axis direction
/define/boundary-conditions/wall
wall-wheel-front-tyre-omega-right-injec-01mm    ;zone-id/name
```

```
yes                ;wall motion[motion—bc—stationary]: change current value?
motion—bc—moving ;
no                 ;shear boundary condition[shear—bc—noslip]
yes                ;define wall motion relative to adjacent cell zone?
yes                 ;apply a rotational velocity to this wall?
no                 ;define wall velocity components?
no                 ;use profile for wall roughness height?
0                  ;wall roughness height(m)
no                 ;use profile for wall roughness constant?
0.5                ;wall roughness constant
43.6308992074      ;rotational speed(rad/s)
1.7071             ;x—position of rotation—axis origian(m)
0                  ;y—position of rotation—axis origian(m)
0.4784505          ;z—position of rotation—axis origian(m)
0                  ;x—component of rotation—axis direction
—1                  ;y—component of rotation—axis direction
0                  ;z—component of rotation—axis direction
;DEFINE THE ROTATING WHEELS: REAR
/define/boundary—conditions/wall
wall—wheel—rear—rim—omega—04mm    ;zone—id/name
yes                ;wall motion[motion—bc—stationary]: change current value?
motion—bc—moving ;
no                 ;shear boundary condition[shear—bc—noslip]
yes                ;define wall motion relative to adjacent cell zone?
yes                ;apply a rotational velocity to this wall?
no                 ;define wall velocity components?
no                 ;use profile for wall roughness height?
0                  ;wall roughness height(m)
no                 ;use profile for wall roughness constant?
0.5                ;wall roughness constant
42.2950441374      ;rotational speed(rad/s)  ;59.4177064765, 76.3941940413
4.482              ;x—position of rotation—axis origian(m)
0                  ;y—position of rotation—axis origian(m)
0.4919345          ;z—position of rotation—axis origian(m)
0                  ;x—component of rotation—axis direction
—1                  ;y—component of rotation—axis direction
0                  ;z—component of rotation—axis direction
/define/boundary—conditions/wall
wall—wheel—rear—rim—stationary—04mm    ;zone—id/name
yes                ;wall motion[motion—bc—stationary]: change current value?
motion—bc—moving ;
no                 ;shear boundary condition[shear—bc—noslip]
yes                ;define wall motion relative to adjacent cell zone?
yes                ;apply a rotational velocity to this wall?
no                 ;define wall velocity components?
no                 ;use profile for wall roughness height?
0                  ;wall roughness height(m)
no                 ;use profile for wall roughness constant?
0.5                ;wall roughness constant
42.2950441374      ;rotational speed(rad/s)
4.482              ;x—position of rotation—axis origian(m)
0                  ;y—position of rotation—axis origian(m)
0.4919345          ;z—position of rotation—axis origian(m)
0                  ;x—component of rotation—axis direction
—1                  ;y—component of rotation—axis direction
0                  ;z—component of rotation—axis direction
/define/boundary—conditions/wall
wall—wheel—rear—tyre—omega—04mm    ;zone—id/name
yes                ;wall motion[motion—bc—stationary]: change current value?
motion—bc—moving ;
no                 ;shear boundary condition[shear—bc—noslip]
yes                ;define wall motion relative to adjacent cell zone?
yes                 ;apply a rotational velocity to this wall?
no                 ;define wall velocity components?
no                 ;use profile for wall roughness height?
0                  ;wall roughness height(m)
no                 ;use profile for wall roughness constant?
0.5                ;wall roughness constant
42.2950441374      ;rotational speed(rad/s)
4.482              ;x—position of rotation—axis origian(m)
0                  ;y—position of rotation—axis origian(m)
```

```
0.4919345        ;z—position of rotation—axis origian(m)
0                ;x—component of rotation—axis direction
—1                ;y—component of rotation—axis direction
0                ;z—component of rotation—axis direction

/define/boundary—conditions/symmetry
farfield_miny
/define/boundary—conditions/symmetry
farfield_maxy
/define/boundary—conditions/symmetry
farfield_maxz
/define/operating—conditions/gravity
yes
0
0
—9.82

/solve/set p—v—coupling 24 ;pressure velocity coupling scheme
/solve/set/discretization—scheme/mom 0 ;convective discretization scheme for momentum
/solve/set/p—v—controls 10 0.2 0.2
/solve/initialize/compute—defaults velocity—inlet
farfield_minx
/solve/initialize initialize—flow
/solve/initialize fmg—init ;enable FMG intialization?
yes
/solve/monitors/residual/check—convergence?
no ;check convergence of continutiy residuals?
no ;check convergence of x—velocity residuals?
no ;check convergence of y—velocity residuals?
no ;check convergence of z—velocity residuals?
no ;check convergence of k residuals?
no ;epsilon residuals?

;/solve/iterate
;500
/solve/set/discretization—scheme/mom 1
/solve/iterate
2500

wcd 0717a_50.cas.gz                  ; Edit file name
/exit ;
yes
;fluent.run —Ver 15.0.7 —Job test_24h —Host cs2cfdcar —Ncpu 408 —Que cfdcar_16 —Case ...
    steady_50kph.jou
```

C.2 Injection and particle track

```
/file/set—batch—options yes yes no
rcd 0717a_50.cas.gz             ;Edit filename (%%.gz?%%)

;INJECTION FILE SHOULD HAVE THE NAME: surf_injecpatched50.inj
/define/injections/create—injection
surf_tyre   ;injection name
no          ;Particle type[inert]: change current value?
yes         ;Injection type[single]: change current value?
file
yes         ;Injection material [anthracite]: change current value?
helium—liquid ;TEST CASE materials
surf_injecpatched50.inj_cy8 ;File name
yes         ;Stochastic tracking?
yes         ;Random eddy lifetime?
5           ;Number of tries
0.15            ;Time scale constant
no          ;Modify laws?
no          ;Set user defined initialization function?
no          ;Cloud tracking?
```

```


/define/models/dpm/numerics/tracking 15000 no 10
/define/models/dpm/options/pressure-gradient-force no

;CHANGE MATERIAL DENSITY
/define/materials/change-create helium-liquid
helium-liquid
yes             ;change density?
constant        ;method
400             ;value
no              ;change Cp?



/define/models/dpm/injections/injection-properties/set/pick-injections-to-set
no          ;list all available injections before picking one or more of them?
surf_tyre



/define/models/dpm/injections/injection-properties/set/physical-models/drag spherical

;DEFINE REFLECT BOUNDARY CONDITIONS FOR DISCRETE PARTICLES
/define/boundary-conditions/wall
wall-wheel-front-tyre-omega-right-injec-01mm
no          ;Wall Motion: change current value?
no          ;Shear Boundary Condition: Change current value?
yes         ;Define wall motion relative to adjacent cell zone?
yes         ;apply a rotational velocity to this wall?
no
no

no

no

1
0.3











/define/boundary-conditions/wall
wall-wheel-front-tyre-omega-right-01mm
no          ;Wall Motion: change current value?
no          ;Shear Boundary Condition: Change current value?
yes         ;Define wall motion relative to adjacent cell zone?
yes         ;apply a rotational velocity to this wall?
no
no

no

no

1
0.3






```

```





/define/boundary-conditions/wall
wall-floor-wheelhouse-front-04mm
no
no
no
0
no
0.5
no
polynomial
1
0.3
polynomial
1
0.8

/define/boundary-conditions/wall
wall-floor-wheelhouse-front-04mm-shadow
no
no
no
0
no
0.5
no
polynomial
1
0.3
polynomial
1
0.8

/define/boundary-conditions/wall
wall-floor-wheelhouse-front-04mm.1
no
no
no
0
no
0.5
no
polynomial
1
0.3
polynomial
1
0.8

/define/boundary-conditions/wall
wall-exterior-body-04mm
no
no
no
0
no
0.5
no
polynomial
1
0.3
polynomial
1
0.8


/define/boundary-conditions/wall
```

```
wall—exterior—body—04mm.1
no
no
no
0
no
0.5
no
polynomial
1
0.3
polynomial
1
0.8


/define/boundary—conditions/wall
wall—exterior—body—04mm.1—shadow
no
no
no
0
no
0.5
no
polynomial
1
0.3
polynomial
1
0.8


/define/boundary—conditions/wall
farfield_minz
no
no
yes
no

1
0
0
no
no
0
no
0.5
yes              ;Discrete Phase BC type[reflect] Change?
escape


/define/boundary—conditions/wall
wall—floor—rear—2WD—08mm
no
no
no
0
no
0.5
no
polynomial
1
0.3




;CHECK TRAPED PARTICLES FOR CENSOR 3
/define/boundary—conditions/radiator
fan—precipitationcensor3—04mm   ;zone—id
```

```
constant             ;loss coeff. method:
0                    ;value
constant             ;heat—transfer—coeff. method:
0                    ;value
yes                  ;discrete phase BC type: change current value?[no]
trap                 ;

/solve/dpm—update
;CHECK TRAPED PARTICLES FOR CENSOR 5
/define/boundary—conditions/radiator
fan—precipitationcensor3—04mm   ;zone—id
constant             ;loss coeff. method:
0                    ;value
constant             ;heat—transfer—coeff. method:
0                    ;value
yes                  ;discrete phase BC type: change current value?[no]
interior             ;

/define/boundary—conditions/radiator
fan—precipitationcensor5—04mm   ;zone—id
constant             ;loss coeff. method:
0                    ;value
constant             ;heat—transfer—coeff. method:
0                    ;value
yes                  ;discrete phase BC type: change current value?[no]
trap                 ;

/solve/dpm—update
;CHECK TRAPED PARTICLES FOR CENSOR 2
/define/boundary—conditions/radiator
fan—precipitationcensor5—04mm   ;zone—id
constant             ;loss coeff. method:
0                    ;value
constant             ;heat—transfer—coeff. method:
0                    ;value
yes                  ;discrete phase BC type: change current value?[no]
interior             ;

/define/boundary—conditions/radiator
fan—precipitationcensor2—04mm   ;zone—id
constant             ;loss coeff. method:
0                    ;value
constant             ;heat—transfer—coeff. method:
0                    ;value
yes                  ;discrete phase BC type: change current value?[no]
trap                 ;

/solve/dpm—update
;CHECK TRAPED PARTICLES FOR CENSOR 1
/define/boundary—conditions/radiator
fan—precipitationcensor2—04mm   ;zone—id
constant             ;loss coeff. method:
0                    ;value
constant             ;heat—transfer—coeff. method:
0                    ;value
yes                  ;discrete phase BC type: change current value?[no]
interior             ;

/define/boundary—conditions/radiator
fan—precipitationcensor1—04mm   ;zone—id
constant             ;loss coeff. method:
0                    ;value
constant             ;heat—transfer—coeff. method:
0                    ;value
yes                  ;discrete phase BC type: change current value?[no]
trap                 ;

/solve/dpm—update
;wcd 0719ap_50.cas.gz                   ; Edit file name
/exit ;
yes
```

```
;fluent.run -Ver 15.0.7 -Job test_24h -Host cs2cfdcar -Ncpu 408 -Que cfdcar_16 -Case ...
    steady_50kph.jou
```

D User Defined Function

D.1 First approach

The UDF using the approach that a *uniformly* reduced v_t and v_y are applied.

```
#include "udf.h"

/************ wallVelocity.c modifiable area ************************************/
/*origin [m]*/
#define OX 1.70714
#define OY 0.868
#define OZ 0.478451

/*rotation axis []*/
#define xAxis 0.0
#define yAxis -1.0
#define zAxis 0.0

/*rotation velocity [rad/s]*/
#define omega (9.8/0.3183269)

/*number of surfaces, and the array of surfaces where the udf is applied*/
#define NR_SURF 1
int surfaces[NR_SURF] = {165}; /*For case without cooling package*/

/*Vy component*/
real const ymax = 0.896;
    real const ymedian = 0.793;
real const Cy = 9.8;

/************ cellCentroids.c modifiable area ************************************/
/*number of surfaces, and the array of surfaces where the udf is applied*/
/*#define NR_SURF 1*/
/*int surfaces[NR_SURF] = {171};*/

char *filename="surf_injecpatched50tiny.inj";

/************ end user modifiable area ************************************/

/************ wallVelocity.c ************************************/
int setVelocity(Thread *tf, int i)
{
    face_t f = 0;
    cell_t c = 0;
    Thread *t = NULL;
    real NV_VEC(x);
    real NV_VEC(r);
    real NV_VEC(axis);
    real NV_VEC(Omega);
    real mag = 0;
    real NV_VEC(orig);
    real NV_VEC(vel);
    real C;

        C = Cy/(ymax-ymedian);

    NV_D(orig,=,OX,OY,OZ);
    NV_D(axis,=,xAxis,yAxis,zAxis);
    if(1e-9 < (mag = NV_MAG(axis)))
        NV_S(axis,/=,mag);
    NV_VS(Omega,=,axis,*,omega);

    if(! BOUNDARY_FACE_THREAD_P(tf))
        Error("Thread h%x is not a boundary face thread!\n",tf);

    if(NULL == tf)
        Error("Null pointer face thread!\n");

    begin_f_loop(f,tf) /* loops over faces in a face thread */
    {
        if(NULL == (t = THREAD_T0(tf)))
            if(NULL == (t = THREAD_T1(tf)))
                Error("Null pointer cell thread!\n");
        if(0 > (c = F_C0(f,tf)))
            if(0 > (c = F_C1(f,tf)))
                Error("Negative cell index!\n");
        C_CENTROID(x,c,t);
        NV_VV(r,=,x,-,orig);
        if(3 == ND_ND)
            NV_CROSS(vel,Omega,r);
        else
        {
            vel[0] = -omega*r[1];
            vel[1] = omega*r[0];
        }

        switch (i)
        {
            case 0: if(x[0]<OX)
                        C_U(c,t) = -vel[0];
                    else
                    C_U(c,t) = vel[0];
                break;
            case 1: C_V(c,t) = vel[1]+C*(x[1]-ymedian);
                break;
#if RP_3D
            case 2: if(x[0]<OX)
                        C_W(c,t) = -vel[2];
                else
                        C_W(c,t) = vel[2];
                break;
#endif
            default: Error("Unknown index: %d\n",i);
                break;
        }
    }
    end_f_loop(f,tf)
    return 1;
}

int correctCells(Domain *d)
{
    /*set 0 velocity and 0 flux in cells with an initial velocity magnitude larger than 240 ...
        m/s*/
    Thread *t, *tf;
    cell_t c;
    face_t f;
    int n;
    const real velMax = 240*240;
    real NV_VEC(vel);
    int count = 0;

    thread_loop_c(t,d)
    {
        begin_c_loop(c,t)
        {
            ND_SET(vel[0],vel[1],vel[2],C_U(c,t),C_V(c,t),C_W(c,t));
            if(velMax < NV_MAG2(vel))
            {
                C_U(c,t) = 0;
                C_V(c,t) = 0;
                C_W(c,t) = 0;
                count++;
                c_face_loop(c,t,n)
                {
                    f = C_FACE(c,t,n);
                    tf = C_FACE_THREAD(c,t,n);
                    if(NULL != tf)
                        F_FLUX(f,tf) = 0;
                }
            }
        }
        end_c_loop(c,t)
    }

    count = PRF_GISUM1(count);
    if(0 < count)
        Message0("Number of cells corrected: %d\n",count);

    return 1;
}

/************ cellCentroids.c ************************************/
int saveCentroids(Thread *tf)
{
    face_t f = 0;
    cell_t c = 0;
    Thread *t = NULL;
    real NV_VEC(x);
    real NV_VEC(v);
    real diameter = 6e-4;/* 0.6mm */
    real mass_flow = 5e-2;/* 50g/s */
    real temperature = 273; /* 273 K */
    FILE *pf;

    if(! BOUNDARY_FACE_THREAD_P(tf))
        Error("Thread h%x is not a boundary face thread!\n",tf);

    if(NULL == tf)
        Error("Null pointer face thread!\n");

    if(NULL != (pf = fopen(filename,"r")))
    {
        Message0("The file '%s' already exists. New data will be appended to it!\n",filename);
        fclose(pf);
    }

    if(NULL == (pf = fopen(filename,"a")))
        Error("Could not open the file '%s'!\n",filename);


    begin_f_loop(f,tf) /* loops over faces in a face thread */
    {
        if(NULL == (t = THREAD_T0(tf)))
            if(NULL == (t = THREAD_T1(tf)))
                Error("Null pointer cell thread!\n");
        if(0 > (c = F_C0(f,tf)))
            if(0 > (c = F_C1(f,tf)))
                Error("Negative cell index!\n");
        C_CENTROID(x,c,t);
        v[0] = C_U(c,t);
        v[1] = C_V(c,t);
        v[2] = C_W(c,t);
        fprintf(pf,"((%e %e %e %e %e %e %e %e %e) ...
            %d)\n",x[0],x[1],x[2],v[0],v[1],v[2],diameter,temperature,mass_flow,f);
    }
    end_f_loop(f,tf)

    fclose(pf);

    return 1;
}
/*Generate injection points based on cell centroids*/
```

```
DEFINE_ON_DEMAND(on_demand)
{
    Domain *d = Get_Domain(1);
    Thread *tf;
    int i;

#if PARALLEL
    Error("For the moment, this udf works only in serial!\n");
#endif
    if(NULL == d)
        Error("No case loaded!\n");

    for(i = 0; i < NR_SURF; i++)
    {
        if(NULL == (tf = Lookup_Thread(d,surfaces[i])))
            Error("No such zone: %d!\n",surfaces[i]);
        Message0("Adding zone %d to '%s'!\n",surfaces[i],filename);

        Message0("Done!\n",surfaces[i]);
        setVelocity(tf,0);
        setVelocity(tf,1);
#if RP_3D
        setVelocity(tf,2);
#endif
        saveCentroids(tf);
    }
    correctCells(d);
}
```

D.2 Second approach

The UDF using the approach that a *constant* v_t and v_y are applied.

```
        switch (i)
        {
            case 0: if(x[0]<OX)
                        C_U(c,t) = -0.2*vel[0]*sqrt(13.888*13.888-(vel[1]+C*(x[1]-ymedian))
                               *(vel[1]+C*(x[1]-ymedian)))/13.888;
                    else
                        C_U(c,t) = vel[0]*sqrt(13.888*13.888-(vel[1]+C*(x[1]-ymedian))
                               *(vel[1]+C*(x[1]-ymedian)))/13.888;
                break;
            case 1: C_V(c,t) = vel[1]+C*(x[1]-ymedian);
                break;
#if RP_3D
            case 2: if(x[0]<OX)
                        C_W(c,t) = -0.2*vel[2]*sqrt(13.888*13.888-(vel[1]+C*(x[1]-ymedian))
                               *(vel[1]+C*(x[1]-ymedian)))/13.888;
                    else
                        C_W(c,t) = vel[2]*sqrt(13.888*13.888-(vel[1]+C*(x[1]-ymedian))
                               *(vel[1]+C*(x[1]-ymedian)))/13.888;
                break;
```