UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Sede Amministrativa: Università degli Studi di Padova

Dipartimento di Ingegneria Industriale

CORSO DI DOTTORATO DI RICERCA IN INGEGNERIA INDUSTRIALE
CURRICOLO: Ingegneria Meccanica
CICLO: XXIX

Numerical study of bicycle racing wheels aerodynamic performances

Tesi redatta con il contributo finanziario di Campagnolo Srl

Coordinatore: Ch.mo Prof. Giovanni Meneghetti
Supervisore: Ch.mo Prof. Nicola Petrone
Co-Supervisore: Ch.mo Prof. Marco Antonello

Dottorando: Pogni Matteo

Abstract

Improving the aerodynamic performance in one of the major challenges in the engineering research applied to racing bicycle. In fact, aerodynamic drag is the main source of losses in cycling and causes between 70% and 90% of total losses in flat road pace (i.e., when not climbing).

Moreover, also lateral forces imposed by crosswinds play an important role because they can destabilize the bike itself.

The body of the cyclist is actually the most important source of drag, because of its relevant frontal area. However, it is necessary to improve also the bike's components aerodynamics, which account for about the 33% of the total drag. This quite relevant percentage is mainly due to the wheels and the frame design.

According to *Greenwell,* wheel drag is responsible for 10% to 15% of total aerodynamic drag; therefore improving the design of this component can reduce the resistance of the bicycle by 2-3%. These numbers, in view of the high level required by either the today's competitions or the bicycle market justify the effort involved in cycling components aerodynamics.

The aim of this work isto assess the capability of CFD RANS simulations to predict the aerodynamic performance of modern racing bicycle wheels, and therefore build a numerical testing method to help the comparison between different solution and design.

The Thesis is subdivided in five chapters after an introduction to bicycle aerodynamics and a literature review of the previous literature regarding the wheel aerodynamics; we give a look at the theory regarding the computational fluid dynamics and the different model used in this work.

An initial method is build tested validated and refined showing the capabilities of the CFD to resolve the aerodynamic forces on a rotating wheel using a simpler steady-state analysis, applying the MRF method imposing a rotating frame to the region containing the wheel.

Different wheels were tested and the results compared with wind tunnel results obtained by Campagnolo; using this method we compared different design and introduced a performance index to characterize the wheel performances.

We performed brief analysis using and unsteady state model and a rigid body motion to compare the method with the steady model.

Good agreement with experimental wind tunnel studies suggests that the approach we outline holds considerable promise. Owing to the flexibility of this methodology, it is now possible to use CFD to provide more definitive answers on some of the open questions within the competitive cycling and triathlon communities.

Additional word regard the testing of an open-source code, because commercial codes are indeed expensive, we tried a simple three-spoke model with the OpenFOAM code, and highlight the pro and the cons of an open-source code.

Finally, there is a brief description of different force balance layouts.

Contents

List of Tables 9
List of Graph 9
List of Figures 10
List of Symbols 12
Introduction 15
1.1 Background 15
1.2 Thesis structure 16
Literature review 17
2.1 Introduction 17
2.2 Aerodynamics 17
2.3 Cycling 20
Theory overview 33
3.1 Introduction 33
3.2 Background 33
3.3 CFD Methodology 34
3.3.1 Pre-Processing 34
3.3.2 Solving 34
3.3.3 Post-Processing 35
3.4 Mesh Generation 35
3.5 The Mathematical Model 35
3.5.1 The governing equations 36
3.5.2 The different approaches of CFD 36
3.5.3 Time Averaging: Reynolds-Averaged Navier-Stokes 37
3.5.4 Direct numerical simulation 37
3.5.5 Large eddy simulation (LES) and Detached-Eddy Simulation (DES) 37
3.6 Moving Reference Frames 40
3.6.1 Governing Equations in Moving Reference Frames 40
3.7 Turbulence Modelling 42
3.7.1 Reynolds Stress Model 42
3.7.2 Boussinesq hypothesis 43
3.7.3 Spalart-Allmaras 43
3.7.4 Standard k-ε 44

3.7.5 RNG k-ε 44
3.7.6 Realizable k-ε 44
3.7.7 Standard k-ω 45
3.7.8 k-ω SST 46
3.8 Near wall modelling 46
3.8.1 Using Wall Treatment Models 48
3.8.2 Wall laws 49
3.9 Boundary Conditions 50
3.9.1 Inlet 50
3.9.2 Outlet condition 50
3.9.3 Wall 51
3.9.4 Symmetry 51
3.10 Numerical method 51
3.10.1 The finite volume method 51
3.10.2 Discretization 52
3.11 The Solver 54
3.11.1 Segregated solver 54
3.11.2 Pressure-velocity coupling 54
3.11.4 Coupled solver 55
3.12 Convergence 55
Development of simulation methods 56
4.1 Introduction 56
4.2 Early modelling 56
4.2.1 Model geometric definition 56
4.1.2 Method Testing 58
4.2 Model 2 60
4.2.1 Model modification 60
4.2.2 Surface Roughness 63
4.2.3 Spoke effects 63
4.2.4 Test methods 64
Results 65
5.1 Introduction 65
5.2 Small fluid domain 67
5.3 Wide fluid domain – Model 2 69

5.3.1 Simulation Results 70
5.4 Wheel design comparison 73
5.4.1 Introduction 73
5.4.2 Rim inner radius 74
5.4.3 Rim width R_w 75
5.5 Performance index 77
5.5.1 Introduction 77
5.5.2 Mean performance index 77
5.5.3 Rotating wind performance index 78
5.6 Unsteady state model 80
5.6.1 Introduction 80
5.6.2 The Model 80
5.6.3 Results 81
Evaluation of the Method on OpenFOAM code 84
6.1 Introduction 84
6.2 About OpenFOAM 84
6.3 OpenFOAM Data Structure 85
6.4 OpenFOAM model structure 86
6.4.1 System Directory 86
6.4.2 Constant Directory 87
6.4.3 Time Directories 87
6.5 OpenFOAM Utilities 88
6.5.1 Pre-processing 88
6.5.2 Meshing 88
6.5.3 Domain Decomposition 89
6.5.4 Post-processing 89
6.5.5 Run-time Processing 90
6.5.6 Solution Monitoring 90
6.6 Hypotheses of the OpenFoam TriSpoke model 90
6.7 Meshing 91
6.8 Initial conditions 92
6.9 Controls 93
6.10 Results 93
Conclusions 98

7.1 Conclusions 98
7.2 Raccomandations for future works 98
OpenFoam Files 100
A1 blockMeshDict 100
A2 snappyHexMesh 101
A3 createPatchDict 108
A4.1 initial conditions P 109
A4.2 initial conditions U 110
A4.3 initial conditions k 111
A4.4 initial conditions nut 113
A4.5 initial conditions Epsilon 114
A5 dynamicMeshDict 115
A6 transportProprierties 116
A7 turbolenceProprierties 117
A8 controlDict 118
A9 fvSolution 119
Aerodynamic Balance Force preliminary study 122
A2.1 Introduction 122
A2.2 General Layout 122

List of Tables

Page
Table 1. Summary of Kyle (1991a) study on wheels 26
Table 2. Greenwell C_D×A comparison data 28
Table 3. Tew and Sayers C_D×A 29
Table 4. Literature comparison 30
Table 5. Surface roughness 63
Table 6. Wheel details. 66
Table 7. Resume of the wheel tested and relative domain/method 67
Table 8. Comparison of Drag and side forces for the disc wheels, calculated by using the standard and the realizable k-ε models. 67
Table 9. Drag and side forces results for the spoke wheel 68
Table 10. Calculated normalized drag 74
Table 11. Calculated normalized drag 76

List of Graph

Page
Graph 1 Comparison between Godo CFD results and Zipp wind tunnel results 31
Graph 2. Grid sensitivity study 58
Graph 3. Small Fluid Domain Comparison, standard k-ε 59
Graph 4. Comparison between the calculated C_D and the experimental C_D of the S2 wheel 60
Graph 5. Zipp sub9 drag coefficient, experimental data from literature 62
Graph 6. CFD early results compared to the given data 62
Graph 7. Comparison between different spokes shape and no spokes normalized on the “aero” spoke results 63
Graph 8. Drag and Side forces realizable and standard k-ε 67
Graph 9. CFD Drag and side forces graphs for Spoke wheels 68
Graph 10. Drag experimental measurements for S1, S2, S5 and S6 wheel 70
Graph 11. . Comparison between CFD results and S1 wheel (Bora 50) 70
Graph 12. Comparison between cfd results and S2 wheel (Zipp 404) 71
Graph 13. S5 high profile wheel drag 72
Graph 14. S6 Prototype drag 73
Graph 15. Calculated normalized drag R8,R6, R10 74
Graph 16. Comparison between R8, R6 and R10 75
Graph 17. Calculated normalized drag "normal" rim +2mm and -2mm 76
Graph 18. Calculated normalized drag "normal" rim +2mm and -2mm 76
Graph 19. Mean of the drag based on the angle tested 78
Graph 20. Mean of the drag based on the angle tested 79
Graph 21. Performance index, calculated between 0° and 180° 79
Graph 22. Unsteady model, drag results Cd x A [m^2] compared to the wind tunnel 82

Graph 23. Unsteady model, side results Cs x A [m^2] compared to the wind tunnel.....82
Graph 24. Unsteady model, side results Cm x A [m^3] compared to the wind tunnel.....82

List of Figures

Page

Figure 1. Consequence of an adverse pressure gradient when fluid flows Lukes (2006) 18
Figure 2. The normalized boundary layer of a laminar and turbulent flow (Massey, 1998).....19
Figure 3. Methods of reducing drag at Red=200,000: (a) smooth infinite-length cylinder, (b) rough infinite-length cylinder and (c) streamlined infinite-length form (from Achenbach (1968) and White (2003)).....19
Figure 4. A prestigious race; (a) Laurent Fignon and (b) Greg Lemond at the Tour de France 1989. 22
Figure 5. Wind and ground speed relative to the biker's direction.....23
Figure 6. Wheel basic dimensions.....23
Figure 7. The six general wheel forms: (a) traditional, (b) deep-section rim, (c) five-spoke, (d) four-spoke, (e) tri-spoke and (f) lenticular (Lukes 2006) 25
Figure 8. Zdravkovich (1992) experimental set-upl 27
Figure 9. Greenwell (1995) experimental set-up.....28
Figure 10. Schematic foundations of CFD experiments.....36
Figure 11. The three divisions of the near-wall region (Fluent manual) 47
Figure 12. A representation of the computational mesh for the two near-wall modelling approaches (Fluent Manual).....48
Figure 13. A basic two-dimensional mesh used to demonstrate the finite volume technique (Fluent manual) 52
Figure 14. A schematic of how the discretization schemes calculate φ at cell faces (φ_w and φ_e) using: (a) first-order upwind, (b) second-order upwind and QUICK scheme (c) 52
Figure 15. The segregated solver 54
Figure 16. Wheel profiles from Godo (2009) 56
Figure 17. Wheel profile reconstructed using Dassault Systemes CATIA Software 57
Figure 18. Domain and region definition 57
Figure 19. Wheel design used for the simulation, based on Zipp Sub9 design taken from the literature 59
Figure 20. y+ value for a spoked wheel 59
Figure 21. Modification to the original model 61
Figure 22. Aero Spokes.....63
Figure 23. Forces direction.....64
Figure 24. Disc wheels a) D1 b) D2.....65
Figure 25. S1 (a), S2 (b), S3 (c), S4 (d), S5 (e) and S6 (f) 66
Figure 26. GST Windkanal wind tunnel.....69
Figure 27. Zipp 404 wheel surface from zipp.com.....71
Figure 28. Rim used to test the dimpled surface (not the actually zipp surface).....72
Figure 29. Wheel inner radius the original radius R8 compared to R6 and R10 74
Figure 30. Wheel width comparison the original wheel width 28 compared to 26 and 30mm 75

Figure 31. Relative speed and angles....78
Figure 32. Testing procedure for a S1 wheel....79
Figure 33. Speed field at 12m/s at 10° for a Bora 80 the second picture is taken 20ms after the first, the third after 25ms from the first and the last at 30ms from the first....81
Figure 34. Overview of OpenFOAM structure, extracted from official guide....84
Figure 35. Two cell in polyhedral mesh, J. Casacuberta Puig (2014)....85
Figure 36. OpenFOAM linked list mesh representation of two cell mesh....86
Figure 37. OpenFOAM case directory structure....86
Figure 38. Wheel used....90
Figure 39. Example of the stl file converted using the open-source software Salome....91
Figure 40. Floor and wheel model shown in ParaVIEW....91
Figure 41. Wheel meshed, in white the solidWall patch in red the movingWall patch shown in ParaVIEW....92
Figure 42. snappyHexMesh result in ParaVIEW....92
Figure 43. Velocity magnitude at 0,1 seconds....93
Figure 44. Velocity magnitude at 0,2 seconds....94
Figure 45. Velocity magnitude at 0,4 seconds....94
Figure 46. Velocity magnitude at 0,65 seconds....94
Figure 47. Velocity magnitude at 0,85 seconds....95
Figure 48. Velocity magnitude on a similar application on Star-CCM+ v11.04.12; t=0.2s....95
Figure 49. Velocity magnitude on a similar application on Star-CCM+ v11.04.12; t=0,4s....96
Figure 50. Lever design....123
Figure 51. Later design....123
Figure 52. Force balance particular....124
Figure 53. Force balance....125
Figure 54. Proposed bike support....125

List of Symbols

a	Acceleration	[m/s^2]
a_1	Model constant for the SST k-ω turbulence model	
A	Reference Area; in the wheel study it is the area calculated using the external diameter of the wheel	[m^2]
A_{cv}	Cross sectional area of the control volume face	[m^2]
C	Courant Number	
$C_{1\varepsilon}$	Model constant for the k-ε turbulence model	
$C_{2\varepsilon}$	Model constant for the k-ε turbulence model	
C_{b2}	Model constant for the Spalart-Allmaras turbulence model	
C_D	Drag coefficient	
$C_D \times A$	Drag coefficient normalized to the Frontal area	
C_μ	Model constant for the k-ε turbulence model	
d	Characteristic length	[m]
D_ω	Cross-diffusion term	
E	An empirical constant in the law-of-the-wall	
f	Frequency of vortex shedding	[Hz]
F	Force	[N]
F_2	Blending function in SST k-ω turbulence model	
g	acceleration due to gravity	[m/s^2]
G_k	Generation of turbulent kinetic energy due to mean velocity gradients	
G_b	Generation of turbulent kinetic energy due to buoyancy	
G_v	Production of turbulent viscosity	
G_ω	Generation of ω	
k	Turbulence kinetic energy	[kgm^{2}s^{-2}]
l_s	Length scale	[m]
m	Mass	[kg]
P	Time-averaged pressure	[Pa]
p_s	Static pressure	[Pa]
$p_{s\infty}$	Static pressure of the infinite flow	[Pa]
q	Dynamic pressure	[Pa]
Re	Reynolds number	
Re_y	Turbulent Reynolds number	
S	Strain rate magnitude	
S_k	User defined source term	

S_ω	User defined source term	
S_ϕ	Source of ϕ per unit volume	
t	Time	[s]
T	Temperature	[K]
U	Time-averaged velocity in the x-direction	[ms^{-1}]
U_f	Frictional velocity	[ms^{-1}]
U^+	U plus	
u'	Instantaneous velocity in the x-direction	[ms^{-1}]
V	Time-averaged velocity in the y-direction	[ms^{-1}]
v	Velocity	[ms^{-1}]
$v\tilde{}$	Transported variable in the Spalart-Allmaras turbulence model	
v'	Instantaneous velocity in the y-direction	[ms^{-1}]
V_s	Velocity scale	[ms^{-1}]
W	Time-averaged velocity in the z-direction	[ms^{-1}]
w'	Instantaneous velocity in the z-direction	[ms^{-1}]
x	Displacement in the x-direction	[m]
y	Displacement in the y-direction	[m]
Y_M	contribution of the fluctuating dilation in compressible turbulence to the overall dissipation rate	
Y_k	Dissipation of k due to turbulence	
Y_ω	Dissipation of ω due to turbulence	
Y_v	Destruction of turbulent viscosity	
y^+	Y plus	
$y*$	Y star	
z	Displacement in the z-direction	[m]

Greek Letters

α^*	Damping coefficient for turbulent viscosity	
α_k	Inverse effective Prandtl numbers for k	
α_ε	Inverse effective Prandtl numbers for ε	
ε	Dissipation of turbulent kinetic energy	
ϕ	Scalar quantity	
Γ_k	Effective diffusivity of k	
Γ_ω	Effective diffusivity of ω	
Γ_ϕ	Diffusion coefficient of ϕ	
κ	von Karman's constant	
μ	Dynamic Viscosity	[kgm^{-1}s^{-1}]

μ_t	Eddy (turbulent) viscosity	[$kgm^{-1}s^{-1}$]
ρ	Fluid density	[kgm^{-3}]
σ_k	Turbulent Prandtl number for k	
σ_ε	Turbulent Prandtl number for ε	
σ_ω	Turbulent Prandtl number for ω	
σ_v	Model constant for the Spalart-Allmaras turbulence model	
τ	Reynolds stress	[$kgm^{-1}s^{-2}$]
τ_w	Wall shear stress	[$kgm^{-1}s^{-2}$]
ω	Specific dissipation rate	

Chapter 1

Introduction

1.1 Background

Aerodynamic drag is highly influential to cycling performance. In elite road and track cycling, aerodynamics is by far the most dominant loss, it causes between 70% and 90% of total losses in flat road pace (i.e., when not climbing) *Chowdhury (2011).* Throughout the sport's history numerous examples have occurred that demonstrate the large influence that controlling and reducing aerodynamics can have. An excellent example is the progression of the world hour record; which is simply how far a cyclist can ride in one hour. Between 1993 and 1996 the record advanced by 5.7 km, and previously it had taken 51 years for the same progression. This dramatic increase can be largely attributed to the bikes becoming more aerodynamically efficient and, more importantly, allowing the riders to adopt more aerodynamic positions. In 2000 the governing body for cycling, the Union Cycliste Internationale, tightened their grip on aerodynamic advancements. A result was that the current holder of the world record, Chris Boardman, attempted the record on a bike similar to a 1970s track bike. Chris Boardman covered 49.4 km on his conventional track bike, whereas his previous record on his unconstrained bike was, and still is, 56.4 km. This example demonstrates the importance of aerodynamics in cycling, and most importantly that there is scope for valued research into cycling aerodynamics so that the findings may be exploited to give a performance advantage.

The body of the cyclist is actually the most important source of drag, because of its relevant frontal area *Greenwell (2009)*. However, it is necessary to improve also the bike's components aerodynamics, which account for about the 33% of the total drag. This quite relevant percentage is mainly due to the wheels and the frame design.

The examination and optimization of cycling aerodynamics is not a new field. In fact, ever since the invention of the safety bicycle in the late 19th century the reduction of aerodynamic resistance seems to have been a priority for manufacturers and riders, but the use of Computational Fluid dynamics (CFD) is quite an un explored field in cycling: technical difficulties, initial investment and knowledge were barriers to the developing of numerical fluid dynamics so the application of CFD to the sport is not so well established. The studies that have used CFD to analysing cycling aerodynamics are fewer, and only a small part regard the aerodynamics of racing bike wheels *(i.e: Godo 2009).* This is where the current study seeks to make advancements in the knowledge, because, if applied correctly CFD could be an extremely powerful analysis tool for the study of cycling aerodynamics. CFD has the highly advantageous ability of being able to scrutinize flow structures and resulting forces in fine levels of detail, which would otherwise be extremely difficult and very expensive by other means, as wind tunnel test using sophisticated sensors or expensive particle image systems, on another hand, CFD needs good experimental results for validation purposes.

From an industrial point of view, the integration of the CFD in the CAD design can speed up the development of new products, which requires wind tunnel experimental analysis only for a small selection of models.

According to *Greenwell et al (1995)* wheel drag is responsible for 10% to 15% of total bicycle aerodynamic drag; therefore improving the design of this component can reduce the resistance of the bicycle by 2-3%. These numbers, in view of the high level required by either the today's competitions or the bicycle market justify the effort involved in cycling components aerodynamics.

The influence that aerodynamics has on cycling and the advantages of CFD for aerodynamic analyses for racing wheel provide the motivation for this study.

This PhD research activities have been supported by the Campagnolo srl company which is one of the world main producers of high end components for racing bicycles, enduring a previous work based on the development of methods for the engineering characterization of cycling components following an integrated approach which relates the technical analysis of the engineering characteristics to the user quality requirements.

The previous work *Giubilato (2010)*, was based primarily on the structural side of racing wheels, mocking the aerodynamics of the wheel using a simple parameter based on the wheel depth; therefore, a more complete analysis was needed.

1.2 Thesis structure

The thesis is subdivided in 7 parts, the first part of the thesis describe the state of the art in this field, the main part describes the model construction, then the model use with stead and unsteady methods:

1. INTRODUCTION

A brief introduction to the thesis.

2. LITERATURE REVIEW

A literature review on the subject.

3. THEORY OVERVIEW

Brief introduction to Computational Fluid Dynamics: basic theory, RANS equations, turbulence models, wall treatments and solvers.

4. MODEL DEVELOPMENT

The first adopt model, meshing and model first test. Then model testing and refinement.

5. RESULTS DISCUSSION

A wheel aerodynamic classification method based on the previously obtained result

6. OPENFOAM MODEL
7. CONCLUSION
8. ATTACHMENT: OPENFOAM FILES AND INITIAL FORCE BALANCE PROJECT

Chapter 2

Literature review

2.1 Introduction

In this chapter, a background to the subjects explored in this thesis will be presented. The aim is to provide a starting point of the thesis by showing the current understanding of the subject and exposing the gaps in knowledge that this research aims to fill.

The chapter begins with a brief exploration of aerodynamics by presenting the fundamental concepts that should be understood when examining this work. The application of aerodynamics to racing bycicle is the main topic of the thesis, and the wheel will be explored next. After a brief explanation of some of the important aspects of the sport the discussion focuses on previous research in the field of wheels aerodynamics, making up the main body of the chapter.

2.2 Aerodynamics

This section introduces some fundamental aerodynamic principles, the understanding of which will be important when reading the thesis. Alongside the descriptions will be some historical references to the scientist who made relevant discoveries.

Any projectile travelling through a fluid will experience a resistive force. Recordings of this fact stretch back as far as Ancient Greece, to Aristotle, who astutely observed that a projectile travelling through a vacuum would continue indefinitely unless acted on by another object. This resistive force is due to the pressure distribution and skin friction distribution around the object and can therefore be divided into two, pressure drag and skin friction drag, respectively. If the projectile is asymmetric or rotating, common examples being an aerofoil or spinning sphere, then the force experienced will not act directly against the motion of the object. This out-of-plane force can be divided into x, y and z components, called drag, side and lift force, respectively. Drag force is the most significant of these, as far as the thesis is concerned. Drag force (F) is normally represented by:

$$F = C_D A \frac{1}{2} \rho v^2$$

where C_D is drag coefficient, ρ is fluid density, *v* is fluid velocity relative to the projectile and *A* is the frontal area of the projectile. Many great scientists had a hand in deriving and proving this relationship, including Sir Isaac Newton, John Smeaton, Sir George Cayley and Lord Rayleigh. What emerges immediately from this equation is that reducing any term on the left hand side of the equation will reduce the drag force.

Density and velocity in the great majority of cases are beyond one's control. C_D is a very important term for an aerodynamicist, as is a dimensionless coefficient that defines the efficiency of a projectile and in most cases will change with Reynolds number (*Re*). Frontal area, *A*, is equally important and much simpler to control, therefore there is no mystery behind its reduction.

Fluid flows have a number of different classifications depending on their speed and idealization. In this thesis the flow examined will be viscous, incompressible and subsonic flow.

Prandtl (1904) found that the flow separates from the surface when subject to an adverse pressure gradient. An adverse pressure gradient will be experienced when a fluid flows over a convex object, for example the flow over the top of a cyclist's helmet. Figure 1 demonstrates this concept.

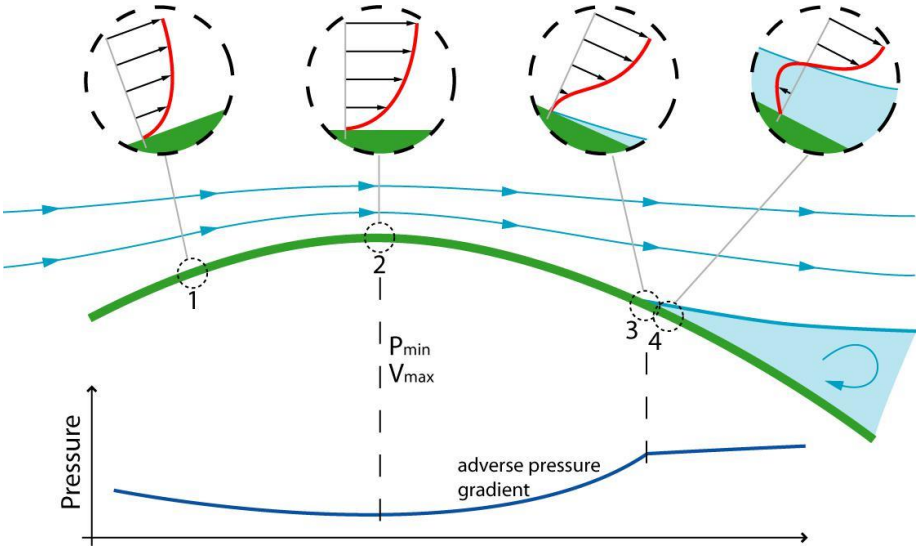


Figure 1. Consequence of an adverse pressure gradient when fluid flows Lukes (2006)

At point 1 in Figure 1 the flow in the boundary layer is relatively unaffected by the presence of the obstruction. As it progresses to point 2, the top of the obstruction, the pressure is lowest and the velocity highest. Downstream of point 2 the pressure increases and provides a resistance which slows the near-wall velocity. By point 3 the flow has been further affected by the increased pressure gradient so that it is now stationary adjacent to the wall; it is at point 3 where the flow will separate. Downstream of the separation point, at point 4, the flow is recirculating and the boundary layer flow is reversed. Aerodynamic drag has a huge dependency on the point at which the flow separates from the surface. If the flow separates early from the surface a large recirculation region will result, giving a relatively large pressure drag. This compares to a later separation giving a smaller wake and a smaller drag. Achenbach showed when the flow in the boundary layer is turbulent separation occurs later than when the flow is laminar. Comparison of the normalized laminar and turbulent boundary layer profiles explains why this is so (Figure 2). A turbulent boundary layer has a much faster average speed near to the wall than a laminar boundary layer. This is because a turbulent boundary layer, by its nature, has more mixing, therefore the higher velocity outer-section interacts with the near-wall section giving the higher speed nearer the wall. The consequence of this is the turbulent boundary layer has more momentum, and this avoids being pushed back so rapidly by the adverse pressure gradient, therefore it separate later.

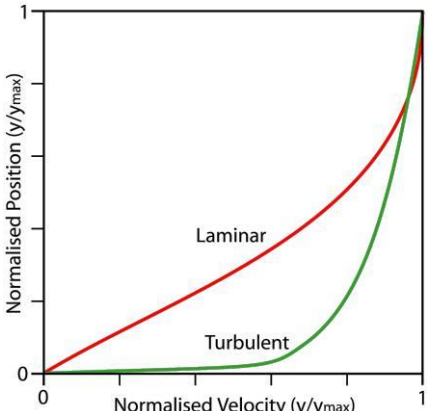


Figure 2. The normalized boundary layer of a laminar and turbulent flow (Massey, 1998)

Transition from laminar to turbulent flow occurs as *Re* increases beyond critical *Re*, Re_{crit}. The value of Re_{crit} is highly dependent on the specific geometry. It is for example around 10 in filters, 2200 in smooth circular pipes, 300,000 around smooth spheres and above 300,000 for aerofoils *(Massey, 1998)* . The overall result of the later separation of turbulent boundary layers is that for convexly curved objects drag force will reduce as *Re* increases beyond Re_{crit}.

Achenbach (1974) also showed that if a roughness were applied to the surface Re_{crit} would be reduced. The roughness excites the laminar flow in the boundary layer and encourages transition at an earlier point than if the surface were smooth. For this method to be effective, the boundary layer flow has to be laminar, and close to becoming turbulent, therefore it is highly dependent on *Re*. A golf ball fits the criteria perfectly and for that reason it has dimples to encourage boundary layer transition. The relevance to cycling shall be discussed later, when aerodynamic aspects of cycling are explored.

Filling the downstream area of a projectile reduces the drag, for example changing from a circular to an elongated profiled cross-section. Taking this step may entirely eliminate separation and greatly reduce the pressure drag. The skin friction drag will rise due to the greater shearing of the flow next to the surface, however this is small in comparison to the reduction of pressure drag and therefore an overall drag reduction will result. Distorting an object in such a way, so as to prevent separation, is often referred to as streamlining and is a proven method of drag reduction. (Figure 3)

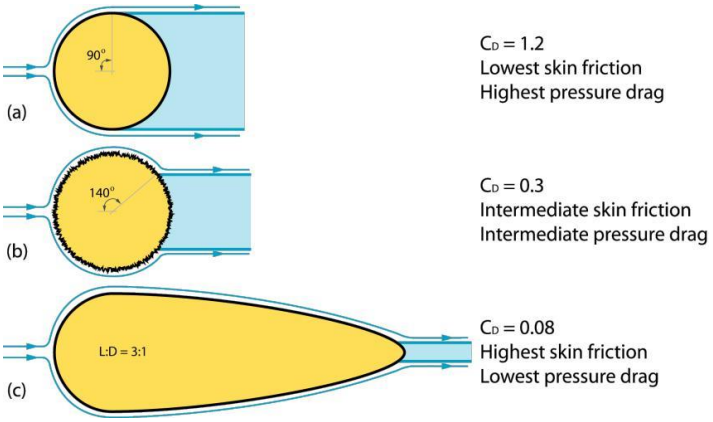


Figure 3. Methods of reducing drag at Red=200,000: (a) smooth infinite-length cylinder, (b) rough infinite-length cylinder and (c) streamlined infinite-length form (from Achenbach (1968) and White (2003))

Flow separation is very often unavoidable in practical applications. Once the flow has separated and a wake is formed it is not uncommon that vortices form and are shed into the wake. Sometimes this shedding is periodic from one side of the object then the other, which is known as a von Karman vortex street. Vincenz Strouhal derived an equation for calculating the frequency of vortex shedding, valid from Re = 250 to Re = 200,000. At higher Re vortex streets disintegrate and random turbulence dominates the unsteady nature of the flow.

The above serves as a brief introduction to the fundamental fluid dynamic concepts that will be referred to in this thesis. Additional concepts will be introduced at a later stage and when doing so an explanation will be provided.

2.3 Cycling

The disciplines in cycling can be broadly split into road, off-road and track. This thesis will focus on cycling events where aerodynamics has a dominant influence on a rider's performance. To that end, off-road events shall not be investigated beyond a few references in the literature review. Track and road cycling events shall be the focus of investigation in is these events aerodynamics can have a key role. To be more specific, the individual-start endurance events will be investigated. In these events cyclists ride a set distance either on the road or track, as an individual or in a team. These events have a very low dependence on tactics and a high dependence on aerodynamics, which contrast with very tactical mass-start events.

2.3.1 UCI regulation

All professional cycling disciplines are governed by the *Union cycliste international* (UCI). The UCI have had a significant effect on the way cycling technology has developed, and it is therefore important to have an understanding of their role.

The historical influence of the UCI, although interesting, is beyond the scope of the thesis and will not be presented here. The UCI has the power to ban a bicycle at any stage of competition, without having to formally state its reasons and with no right of appeal for the banned party. A good example of UCI intervention is given by *Melton (1990)*. The UCI redefined the equipment rules in 2000 and they remain largely the same in 2006 *(UCI, 2006)*.

The purpose of the UCI rules is to create an environment where all competitors compete on an equal footing. There are strict definitions regarding bike sizes and shapes; the weight of a bicycle cannot be less than 6.8 kg and the frame must maintain a traditional two-triangle shape. The tubes of frames have certain length and width criteria that they have to adhere to, for example time-trial and a track frame seat-tube cross-section should have a minimum 25 mm width and maximum 50 mm length. Handlebar extensions (tri-bars) are permitted for road time-trials and track pursuits, kilometre and 500 m events. The rules state that protective screens or fairings are prohibited. Profiled frame and component tubing is tolerated as long as the cross-section length to width ratio does not exceed 3.

As the seat-tube of the frame is essentially a fairing in front of the rear wheel they must not get too close. It should be possible to slide a flat rigid object, like a credit card, between the seat-tube and the rear wheel without flexing it. Freewheels, multiple gears or brakes are prohibited in track cycling.

Particularly UCI demands that Wheels of the bicycle may vary in diameter between 70 cm maximum and 55 cm minimum, including the tire. For the cyclo-cross bicycle the width of the tire shall not exceed 35 mm and it may not incorporate any form of spike or stud.

For massed start road races and cyclo-cross events only wheel designs granted prior approval by the UCI may be used. Wheels will have minimum 12 spokes; spokes can be round, flattened or oval, as far as no dimension of their sections exceeds 10 mm. In order to be granted approval wheels must have passed a crash test as prescribed by the UCI in a laboratory approved by the UCI. The test results must show that the failure characteristics obtained are compatible with those resulting from an impact sustained during normal use of the wheel. The following criteria must be fulfilled:

- On impact, no element of the wheel may become detached and be expelled outwards.
- The fracture must not present any shattered or broken off elements, or any sharp or serrated surfaces that could harm the user, other riders and/or spectators.
- The fracture characteristics must not cause the hub to become separated from the rim in such a way that the wheel becomes detached from the forks.

Without prejudice to the tests imposed by the laws, regulations or customs, standard (traditional) wheels are exempted from the rupture test referred to above. A traditional wheel is deemed to be a wheel with at least 16 metal spokes; the spokes may be round, flat or oval, provided that no dimension of their cross sections exceeds 2.4 mm; the section of the rim must not exceed 2.5 cm on each side.

The previous stated rules banned tri-spoke from road racing along with climps and other movable objects; these wheels and part remains legal for other sanctioned races, like track cycling and triathlon events (not under UCI rule book).

2.3.2 Aerodynamics

The importance of reduction of aerodynamic resistance in cycling has been recognized by engineers and cyclists for many years. There are many examples where superior aerodynamics has yielded a crucial and sometimes victorious advantage. One such example came in the 1989 Tour de France.

Before the final stage, the American Greg Lemond trailed the French leader, Laurent Fignon by 50 seconds. All that remained was a 25 km individual time trial around the streets of Paris, just half an hour of racing. Lemond left the starting house with ‘aerodynamic’ tri-bars, an ‘aerodynamic’ helmet and a rear disc wheel.

Fignon followed with no helmet, exposing his long hair to the air stream, and using the conventional ‘cow-horn’ time trial bars of the time. Lemond gained 58 seconds over Fignon on the day and won the tour by just 8 seconds, which today remains the smallest ever margin of victory in the race. We will never truly know for sure, however it seems that Fignon might have secured the race title if his choice of equipment had been more influenced by the most up-to-date engineering knowledge.

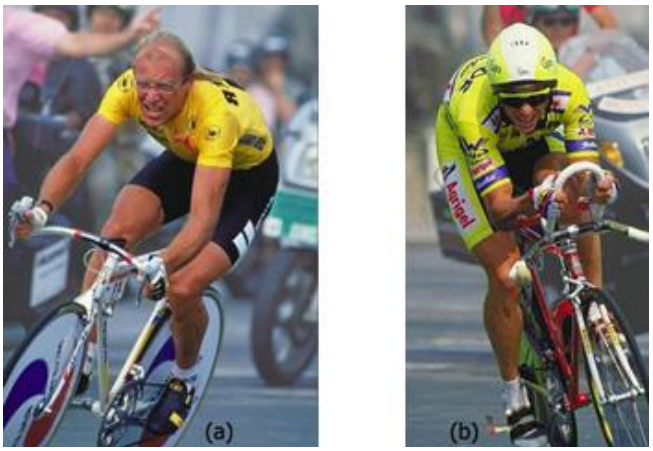


Figure 4. A prestigious race; (a) Laurent Fignon and (b) Greg Lemond at the Tour de France 1989.

Throughout this review data from a variety of studies will be compared. These studies often present their data in different ways, for example in terms of C_D, power requirement or oxygen consumption. To draw straightforward comparisons between different publications every effort has been made to present common data. The vast majority of data is presented in terms of drag coefficient area or $C_D \times A$(m^2). This value is easy to extract from most studies and where assumptions are made about experimental conditions are stated (air density for example). Comparing $C_D \times A$ gives a direct representation of the drag force on the rider, therefore the optimum configuration is immediately apparent. $C_D \times A$ comparison remove the uncertainty of the choose of the reference area, usually in aerodynamics studies the Area choosen is the projection of the frontal area of the object of the study (for example car aerodynamics) but in our case this area changes with the yaw angle complicating the problem, a possibility is to choose the wheel area determined by the area given by the circular section described by the external diameter of the wheel as *Godo (2009)* and other studies.

A brief overview of the complete bike a more depth analysis of wheel aerodynamics study will follow.

2.3.3 Basic notation and terminology

The wind speed that invests the bicycle is determined by the sum of the wind speed viewed by a stationary point and the speed of the bicycle itself.

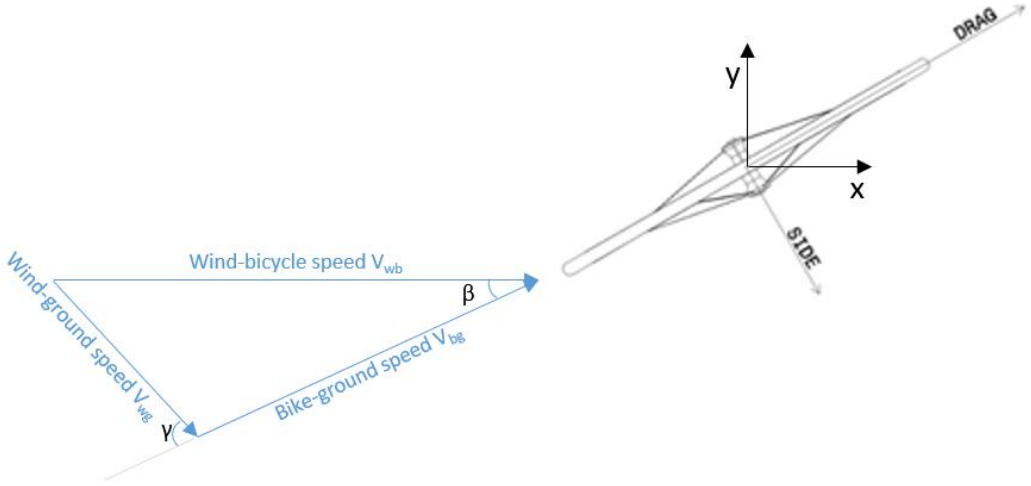


Figure 5. Wind and ground speed relative to the biker's direction

As shown in the Figure 5 above the bike-ground speed is the speed of the rider, the wind-ground speed is the wind speed relative to a stationary point, and the wind-bicycle speed is the wind experienced by the rider-bike complex.

Normally in wind tunnel tests and numerical simulation the fluid velocity at the inlet is maintained constant, as the same velocity as the bike tested, therefore V_{bg} and V_{wb} are the same, meanwhile the angle β changes, this means that the speed of the wind relative to the ground changes at every angle tested.

This kind of testing is useful because it easier to set-up, and we often cannot choose a fixed wind speed, besides some cases when it is relatively low or if we choose a mean value.

Later in this work a we tried a different approach, fixing the bicycle speed V_{bg} and the wind-ground speed V_{wg}, varying the angle γ and therefore the relative speed V_{wb} at the domain input and the angle β.

Excluding the disc wheels, often wheels are defined by the rim depth which is the distance between the inner part of the rim and the outer and the rim width as shown in Figure 6.

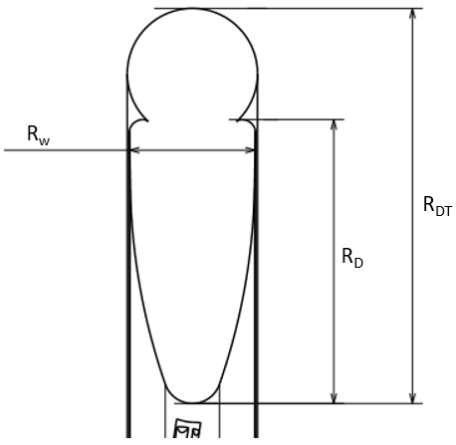


Figure 6. Wheel basic dimensions

Where:

- R_D is the rim depth.
- R_{DT} is the rim depth plus the tire.
- R_W is the rim width.

Typically a wheel with a high rim depth is called a high profile wheel meanwhile a more traditional wheel, with a reduced rim depth, is called a low profile wheel.

2.3.4 Bicycle aerodynamics

One of the very first investigators of cycling aerodynamics was *Nonweiler (1956)*. This study examined a cyclist in a racing position, i.e. holding the dropped handlebars, and in a 'touring' position. It was found that a racing cyclist has a drag area ($C_D \times A$) of 0.31 m^2 and the touring cyclist 0.37 m^2. Using an estimation of 0.33 m^2 for the racing cyclist frontal area C_D of 0.93 was reported. This value is perhaps a little high, seemingly due to an under-estimation of frontal area. This work should perhaps not be noted for its accurate C_D but instead for being a founding study of cycling aerodynamics. A later publication *Nonweiler (1958)* brought about three conclusions; measurements of drag are repeatable to within ± 3%, the subject's size and posture in the usual racing position have a relatively slight effect, and drag varies with the square of speed.

Pugh (1974) studied the oxygen consumption of six competition cyclists whilst cycling on an airfield. Expired gas was collected using a bag and vacuum cleaner tube suspended from a boom attached to an accompanying vehicle. Comparative measurements were also made on an ergometer, allowing a relationship between oxygen consumption, work rate and speed to be determined. The results were used to derive information about rolling and air resistance. It was found that the C_D of bicycle and rider was 0.79, where the frontal area was found to be 0.42 m^2 from analysing photographs. This study showed an alternative method of cycling performance analysis and an improvement from Nonweiler's previously predicted C_D. However, due to the uncertainties associated with having two experimental methods the result should not be assumed entirely accurate. *Di Prampero et al. (1979)* described an equation of motion of a cyclist. The experimental procedure involved towing two cyclists on racing bicycles in the dropped position on the Monza motor-racing circuit. Tests were carried out in calm air and the cyclists were towed at a constant speed along the flat track. The investigation ran at different speeds (5 to 16.5 ms^{-1}) and from the results a relationship between resistive force (R_T) and air velocity (v) was found, $R_T = 3.2+0.19v^2$. It was assumed that the resistance at zero wind velocity (3.2 N) can be entirely attributed to rolling resistance, and the $0.19v^2$ term is the aerodynamic resistance. Therefore, the constant 0.19 is equal to $0.5\rho C_D A$, giving $C_D \times A$ as 0.317 m^2 for standard atmospheric conditions. Using the frontal area reported by Pugh (0.42 m^2), Prampero di et al. found a C_D of 0.75. Towing a rider is however flawed in that, the bike is travelling downstream of the vehicle and an accurate dynamometer and a very stiff rope are required.

Oxygen consumption when cycling against a range of wind speeds was studied *(Davies, 1980).* Fifteen male cyclists were studied in a wind tunnel. They were required to pedal their bicycles at a set speed of 4.7 ms-1 on a motor driven treadmill against a headwind varying from 1.5 to 18.5 ms^{-1}. The subjects cycled for 8 minutes in the given conditions and the oxygen consumption was measured over the last 3 minutes of exercise. The results showed that at constant treadmill speed the oxygen consumption was proportional to the square of the wind speed. The results were used

to calculate a C_D of a bike and rider, which was found to be 0.56 based on a frontal area of 0.5 m^2 ($C_D \times A$ = 0.28). This C_D is notably lower than that reported in previous publications, and like the *Pugh (1974)* study uncertainties may stem from the combination of two experimental procedures and perhaps the inclusion of a treadmill in the wind tunnel. In possibly the most in-depth study of the time a number of ways of improving a racing bicycle were investigated *Kyle and Burke (1984)*. Over a two-year period over 600 wind tunnel tests, 400 coast-down tests and over 200 other tests were conducted. For the study three prototype track bikes were produced and one road bike. From the coast-down tests it was found that aerodynamic resistance outweighs rolling resistance at speeds above 3.6 ms^{-1}, and by 8.9 ms^{-1} the aerodynamic contribution to drag is 90%. Therefore at racing speeds the contribution of aerodynamic drag is over 90%. The drag on the bare bicycle, determined from the wind tunnel tests, was used to ascertain that the force on the bike is 31% to 39% of the total resistance. This reasoning assumes that there is no interaction between rider and bicycle. In reality it may be the case that the rider's presence slightly reduces the drag on the bike making the contribution of the bike lower than that quoted by Kyle and Burke. A simple, but important, three-tier hierarchy for reducing the cycling resistance was stated; the rider position is most important, followed by the bike geometry and finally the rolling resistance. Kyle & Burke highlighted three ways of improving aerodynamic efficiency: lowering frontal area, streamlining the geometry and lowering the surface roughness. A later study (Bassett *et al.*, 1999) examined the power requirements of elite cycling and included a section outlining the factors that influence aerodynamic drag. The frontal area of a number of cyclists was found by examining photographs taken at a distance of 50 m. This data on frontal area was then used to determine its relation to rider height (h) and mass (m_r), finding that when the cyclist uses tri-bars:

$$A = 0.2025 h^{0.725} m_r^{0.425}.$$

This relationship gives a reasonable approximation of frontal area, however not an entirely accurate one as the scatter in the data was significant. Chester Kyle presented the C_D of a standard racing bicycle as 0.8 to 0.9 (Kyle, 2003).

2.3.5 Wheel aerodynamics: experimental Study

The wheels make up a large and important proportion of the bike geometry, and are foreseeably an influential factor on the airflow around the bike. Several studies have specifically focused on wheel aerodynamics both from an analysis and optimization standpoint. Bicycle wheels generally take one of six forms, which are illustrated in Figure 7.

Figure 7. The six general wheel forms: (a) traditional, (b) deep-section rim, (c) five-spoke, (d) four-spoke, (e) tri-spoke and (f) lenticular (Lukes 2006)

Traditional wheels generally have 32 to 36 spokes and a shallow rim. Deep-section rim wheels tend to have a reduced number of spokes and a rim 0.1 to 0.15 m deep. Five-spoke, four-spoke, tri-spoke and disc wheels speak for themselves and are manufactured from composites.

Following the introduction of various new wheels in the 1980s, bicycle-component manufacturers HED carried out wind tunnel tests *Kyle (1989)* with rotating wheels. The tests revealed trends but no fine details between wheels. By the author's admission this was due to the insensitivity of the force balance. The results showed that disc-wheels and tri-spoke-wheels had the lowest drag. A conventional wheel with an 18 mm tire and a wheel with an aero rim with 18 bladed spokes were only slightly higher.

Two studies examined the effect that riding with different wheels may have on oxygen consumption *Hagberg and McCole (1990), McCole et al.(1990).* Eight sets of wheels were tested but only two stood out as being noticeably beneficial. An aerodynamic set of wheels with 16 front spokes and 18 rear spokes reduced the oxygen consumption by 7±5% and a set of disc wheels reduced the oxygen consumption by 3±4%. It is supposed that this method of analysis does not have the sensitivity to recognise the subtle difference between many wheels. Such components are possibly easier to analyse in a more isolated and controlled environment.

Kyle (1991a) presented a comprehensive analysis of 25 wheels carried out in a 0.6 by 0.9 m working section wind tunnel at the University of California in Irvine. The wheels were tested at 8.9 to 13.4 ms^{-1} on a force balance accurate to ±0.05 N with a stationary wheel or ±0.1 N if the wheel is rotating. Five different groups of wheels were investigated: (1) seven lenticular disc wheels (a disc wheel that curves outward between the rim and hub), (2) five flat disc wheels, (3) six three and four spoked composite wheels, (4) five conventional aero wheels with steel bladed spokes and shallow aluminium aero rims, (5) two newly developed aero wheels with steel bladed spokes and deep aluminium aero rims. Tests on wheels of different sizes revealed that there wasn't much difference in drag between 0.61 m, 0.66 m, and 0.69 m diameter wheels. The latter wheels are universally used in track and road cycling today; therefore the examination of results shall remain solely on these wheels. The results of $C_D \times A$ taken from this study for stationary and rotating conditions are shown in Table 1, where the operating density is assumed to be 1.2 kg/m^3.

Table 1. Summary of Kyle (1991a) study on wheels

		stationary		*rotating*	
		$C_D \times A$ [m^2]	% variation in $C_D \times A$	$C_D \times A$ [m^2]	% variation in $C_D \times A$
Conventional Aero Spoke	Min	0.0111		0.0136	
	Max	0.0118		0.0148	
	Average	0.0115	0	0.0142	0
Three and four spoked Wheel	Min	0.0081		0.0097	
	Max	0.0099		0.0133	
	Average	0.0089	-22.6	0.0114	-19.2
Disc Wheel	Min	0.0066		0.0092	
	Max	0.0090		0.0112	
	Average	0.0072	-37.2	0.0101	-28.5

The results show that in general the fewer spokes the more aerodynamic the wheel will be. The drag on the wheel increases by around 20% to 40% between the stationary and rotating conditions. It is interesting to note an overlap in drag between the wheel groups, for example some of the disc wheels produced more drag than some three- or four-spoked wheels, despite the overall average drag being significantly lower. Kyle observed that with the wind parallel to the axis of the wheel the lenticular and flat disc wheel performed similarly. However, lenticular wheels are favourable in the presence of a crosswind, as the flat disc wheel will stall before the lenticular wheel. Different tire sizes were also investigated which revealed that the rim should be no wider than the tire. If the two remain the same width then the likelihood of separation is reduced and the drag is minimized.

Zdravkovich (1992) investigated wheel aerodynamics in a wind tunnel. The ratio of wheel diameter to tyre tube cross-sectional diameter, known as the shape factor, was found to have a large influence over the body axis drag on the wheel. Experiments using a narrower racing tire, thus increasing the shape factor, increased the C_D significantly. Zdravkovich supposed that the use of thin plates located behind bluff bodies, known as splitter plates, may reduce the drag. A splitter plate of length 1.5 times the diameter of a circular cross section object will prevent the formation of a vortex street and reduce the pressure drag *(Apelt et al., 1975)*. Splitter plates of 2d and 4d length were fixed between the spokes and attached to the rim of the wheel, where d is the cross-sectional distance from the inside of the rim to the outside of the wheel (Figure 8). Zdravkovich also constructed a disc wheel made from a standard bike wheel with sheets attached to the sides. The wind tunnel experiments showed that the 2d splitter plates resulted in a 5% axial $C_D \times A$ reduction. In contrast, the 4d splitter plates and the full disc actually increased axial drag on the wheels. This finding is in clear contrast to *Kyle (1990a)*, who showed a clear and consistent axial drag reduction for stationary and rotating disc wheels. This discrepancy is not addressed in the publication, however it is believed it could be a result of a differing quality between the self-made wheel of Zdravkovich and the professionally-made wheels tested by Kyle. The side forces on the wheels were also measured, and found to increase linearly with yaw angle. The 2d splitter plates were found to increase the side force by 3.5 times that of a standard wheel, and the disc wheel 11 times the side force of a standard wheel. Zdravkovich concluded that the vortex shedding around bicycle geometry was weak and therefore the splitter plates had little effect on the reduction of drag.

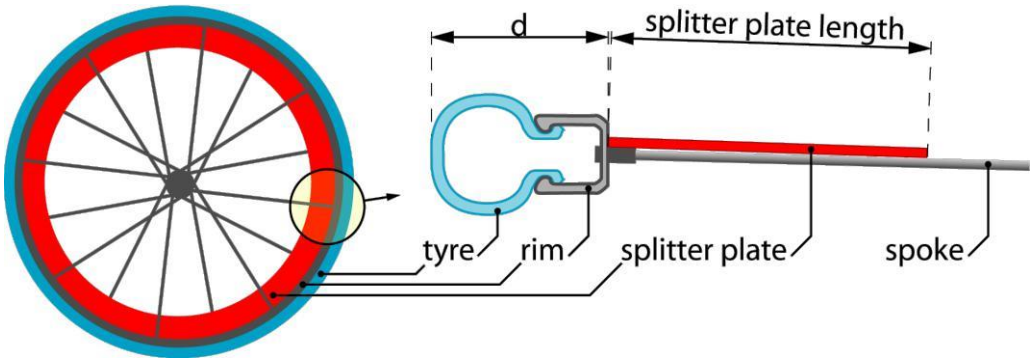


Figure 8. Zdravkovich (1992) experimental set-upl

Capelli et al. (1993) attempted to determine whether a lenticular wheel outperforms a traditional flat disc wheel under the same conditions. In this study two cyclists were towed around a velodrome on four different bike configurations; an aerodynamic frame with lenticular wheels, an aerodynamic frame with traditional, low profile, wheels, a traditional frame with lenticular wheels, and a

traditional frame with traditional wheels. The riders were towed by a bicycle at distance of 10 m. A load cell was used to measure the force on the towed cyclist, from which the traction resistance was determined. Complying with previous findings the traction resistance was found to be proportional to the square of wind velocity. The study found that in terms of mechanical energy saving, the role of lenticular wheels is negligible. From these results and with a view to previous findings, it is believed that the uncertainty associated with this experimental technique is too high to recognize the subtle differences in drag that occur when changing wheels.

Greenwell et al. (1995) studied the body axis and side forces exerted on seven bicycle wheels in a full-scale wind tunnel. The wheels tested were: one standard low profile wheel, two deep-section rim wheels, two tri-spoke wheels, and two disc wheels. The study used an experimental set-up consisting of the rear of a frame and wheel suspended from a center balance

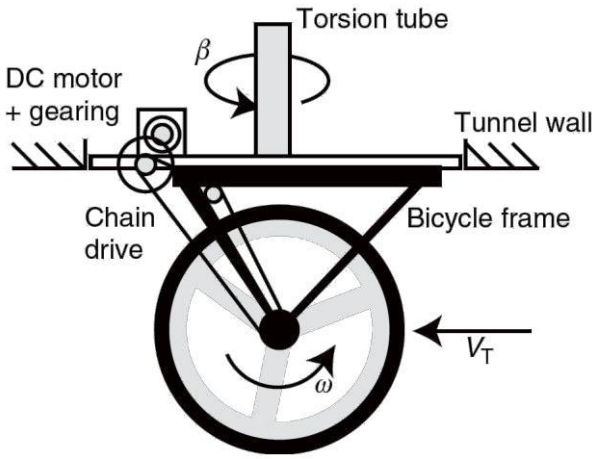


Figure 9. Greenwell (1995) experimental set-up

It was found that due to the tread of the tire no significant laminar region occurred, and the flow transitioned to turbulent almost immediately, a characteristic that would be enhanced in real situations, due to debris on and around the tire. For all but one of the wheels the results were negligibly affected by the different rotating speeds, the one exception being the disc wheel drag at very high yaw angles. Table 1 shows *Greenwell et al.* found that at a zero yaw angle the deep-section rim and tri-spoke wheels gave a 23% reduction in $C_D \times A$ and the disc wheels a 26% reduction, compared to the drag on a standard wheel.

Table 2. Greenwell $C_D \times A$ comparison data

	Spokes	$C_D \times A$ *[m²]*	*% variation in* $C_D \times A$
Conventional low profile	36	0.0189	0
Campagnolo Shamal	16	0.0145	- 23.2
HED CX	24	0.0146	- 22.8
Specialized tri-spoke	3	0.0146	- 22.8
FIR tri-spoke	3	0.0147	- 22.2
HED disc (lenticular)	-	0.0139	- 26.5
Zipp 950 disc (flat)	-	0.0140	- 25.9

The deep-section rim and tri-spoke wheels were found to have similar drag levels at a 0° yaw angle; however, their geometrical differences played more of a role as yaw angle increased. As yaw angle increased, the disc wheels outperformed all others in terms of $C_D \times A$ reduction. However, the

magnitude of the side force was found to increase linearly with the projected side area of the wheel. The side force imposed on the tri-spoke and deep-section rim wheels were found to increase gradually with yaw. The disc wheels behave almost like a wing in that a distinctive stall was witnessed at a yaw angle of approximately 25º. The side force on the disc wheels at very high yaw angles was five times that of a standard wheel, and at very low yaw angles eight to ten times.

Tew and Sayers (1999) studied the aerodynamic characteristics of six different bicycle wheels. The investigation was carried out on full-scale wheels in a wind tunnel at a range of yaw angles, between 0° and 30°. Drag and side forces were resolved using the wind tunnel force balance. The wheels were tested at different rotational speeds and different wind tunnel velocities. The wheels investigated were one standard wheel, two deep-section rim wheels, one Spinergy four-spoke wheel, one tri-spoke wheel and a disc. The results showed that of the six wheels the deep-section rim, Spinergy and tri-spoke wheels all preformed similarly, and will be collectively known as 'intermediate wheels'. The $C_D \times A$ found by Tew and Sayers for the six wheels is shown in Table 3

Table 3. Tew and Sayers $C_D \times A$

	Spokes	$C_D \times A$ *[m²]*	*% variation in* $C_D \times A$
Conventional low profile	36	0.0232	0
Campagnolo Shamal	16	0.0087	- 62.3
Mavic cosmic	12	0.0084	- 63.9
Spinergy four-spoke	4	0.0087	- 62.3
Specialized tri-spoke	3	0.0099	- 57.4
Disc (flat)	-	0.0072	- 68.9

At 0° yaw angle with the wheels rotating the intermediate and disc wheels produced 61% and 79% less $C_D \times A$ than the standard wheel, respectively. As found by *Greenwell et al. (1995)* the $C_D \times A$ of a standard wheel increased with yaw angle, due to the increased exposure of the spokes to the air stream and increasing frontal area. In contrast to the Greenwell *et al.* results, the intermediate wheels also experienced this increase in $C_D \times A$ but to a lesser extent. The $C_D \times A$ on the disc wheel was found to increase suddenly, although the angle at which this sudden increase occurs decreased with wheel speed. This finding contradicts the Greenwell *et al.* results, which showed a significant reduction in drag force when increasing the yaw angle on the disc wheel. Greenwell *et al*. tested two disc wheels in their study, one flat and one lenticular. It was the lenticular wheel that displayed a much lower drag than the flat-disc wheel as yaw angle increased. This point shows that the drag on a disc wheel can be largely affected by subtle geometrical changes, and that this could be part of the cause of discrepancy between the two publications. Tew and Sayers do not mention the manufacturer of the disc wheel, and it is not recognizable from the image in the publication. It is possible that the wheel tested by Tew and Sayers encouraged a far earlier separation, an almost immediate stall, which gave the contrasting results as yaw angle increased. The publications did both show that for a given yaw angle of the disc wheel the $C_D \times A$ reduced with increasing speed. For the standard and intermediate wheels the side force component was independent of speed. The standard wheel had the lowest side force component whereas the disc had the highest. The study showed that drag and side force were independent of the rotational speed of the wheel.

To summarize the main findings of experimental wheel studies to date the results from the three most significant studies have been condensed into Table 4.

Table 4. Literature comparison

	Kyle (1991a)		*Greenwell et al. (1995)*		*Tew and Sayers (1999)*	
	$C_D \times A$ [m^2]	% variation in $C_D \times A$	$C_D \times A$ [m^2]	% variation in $C_D \times A$	$C_D \times A$ [m^2]	% variation in $C_D \times A$
Conventional spoked	0.0142	0	0.0189	0	0.0232	0
3 to 4 spoked	0.0114	- 19.2	0.0146 - 0.0147	- 22.5	0.0099	- 57.4
Disc	0.0101	- 28.5	0.0139 - 0.0140	- 26.2	0.0072	- 68.9

The data of *Kyle and Greenwell et al.* agree in terms of drag reduction, however the magnitude of $C_D \times A$ is higher in the *Greenwell et al.* study.

This could simply be due to differences in experimental equipment and possibly a calibration error. The Tew and Sayers study reported the highest force for a conventional wheel and lowest for the 3 to 4 spoke and disc wheels, and results in Tew and Sayers reporting over double the drag reduction of the other studies. Alongside the aforementioned contrasting results for a yawed disc wheel, these results do bring into question the reliability of this study. It is therefore believed that the work *Kyle (1991a)* and *Greenwell et al. (1995)* gives the most reliable understanding of the magnitude of $C_D \times A$ reduction.

2.3.6 Wheel aerodynamics: CFD Study

These studies try to simulate the wheel aerodynamics by means of several approaches. For example, numerical studies dealing with different racing wheels were performed using a steady state RANS model, by means of relative reference frame computations to consider the motion of the wheel *Knupe (2009)*. Another numerical work *Godo (2009)* models by means of RANS approach the whole domain by dividing it into two sub-volumes, one containing the spokes, hub and inner edge of the wheel rim, and the other containing the remaining toroidal wheel surface. This partitioning technique makes easier the model setup in case of repeated changes in wheels and rims geometry. Godo also utilized unsteady approach to simulate the behavior of the wheel at 10°, a typical mean angle, as found in *Easton (2010)*, and compared the numerical results to the experimental ones obtained from Zipp.

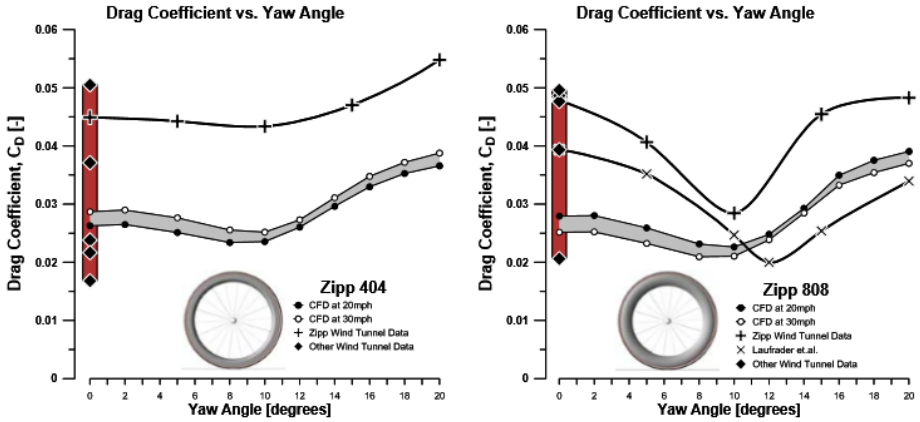
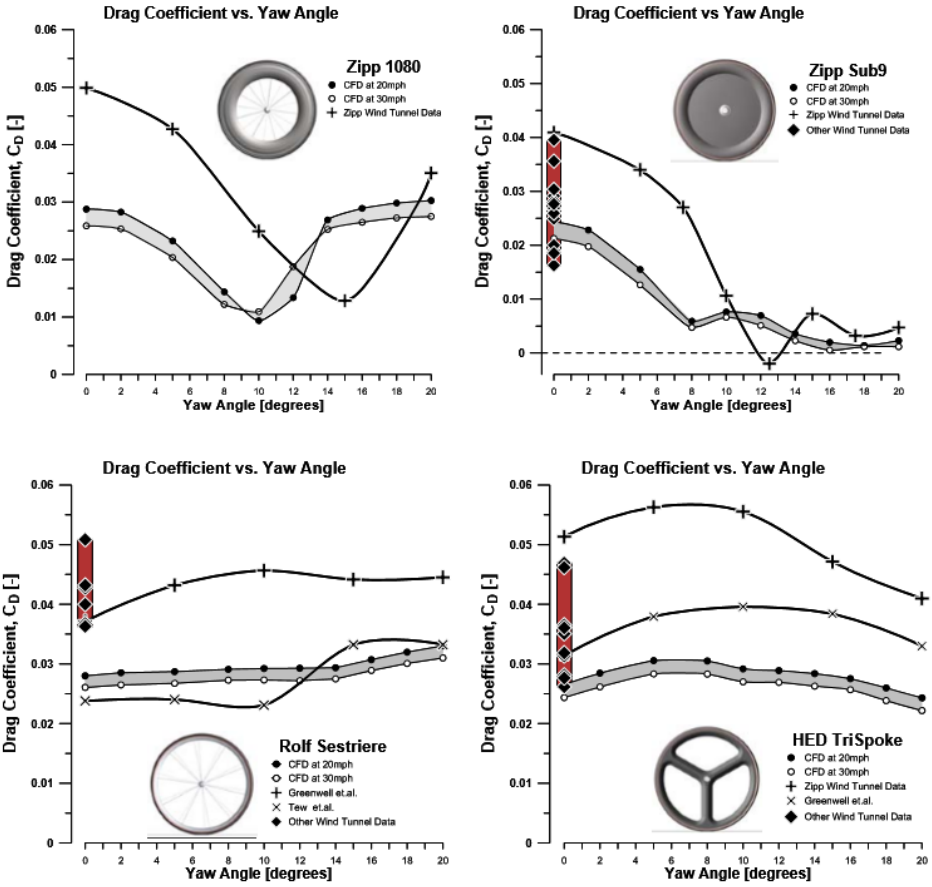


Figure 6a: Comparisons between Wind Tunnel Results and CFD for Zipp404 and 808



Graph 1 Comparison between Godo CFD results and Zipp wind tunnel results

Hanna (2002) claimed there was little clear evidence to support the use of rear disc wheels in cycling, and aimed to quantify the speculation that the wheel acts as a sail. The study was performed using computational fluid dynamics (CFD). It was demonstrated that CFD has the advantageous ability of being able to compute the skin friction and pressure drag independently, quantities which are not easy to measure independently. Simulations were carried out on a representative bike and rider travelling at 8.9 ms^{-1}, with a constant 90° crosswind imposed. Both spoked and disc rear wheel configurations were analysed, and the crosswind varied from 0 to 13.4 ms^{-1}. The results demonstrated the increasing magnitude of drag experienced in a crosswind. The simulations showed that in still air a rear disc wheel reduces the overall drag by about 2%. In an 8.9 ms^{-1} crosswind (39° yaw angle) the rear disc wheel reduced the overall drag by 17%. However the

disadvantage of the disc wheel was also outlined because when subjected to an 8.9 ms^{-1} cross wind the overall side-force was doubled. This study presents no actual data, just relative values, so without any comparison with experiment it offers little in terms of validation. However, the purpose of the study is more a demonstration of CFD and is perhaps not so concerned with achieving validated findings.

The numerical studies in wheel behaviour are still an area in development, most of the literature focus on the complete bicycle performance, excluding *Godo* and *Knupe* works and the only wide research are the one carried out by *Godo*.

There isn't a general consensus about the model that should be used, because rarely a comparison is available, the aim of this work is to develop a reliable method to test the wheel design, and a we are going to show the standard k-ε model, one of the most used in CFD, seems to give a good agreement in the steady-state simulation using a multi-reference frame.

For unsteady-state simulation with rigid motion the optimal solution seems to be more toward the k-ω *SST* in the RANS field, that seems less stable in steady state conditions and, using a more complex model, DES and LES models shows good results.

The main part of the work is carried out using a commercial software *Star-CCM+* from CD-Adapco, in the final part of the Thesis we give an overview of the open-source code *OpenFOAM.*

Being an open-source model *OpenFOAM* is certainly cheaper obtain, on another hand the operator costs can increase significantly due to the code nature and complexity, exist the possibility to make custom application to this field.

Chapter 3

Theory overview

3.1 Introduction

This chapter introduces computational fluid dynamics, or CFD as it is commonly abbreviated. CFD has been used extensively in this thesis, and it is therefore important that the reader has an appreciation of the fundamentals introduced in this chapter.

The chapter divides into three main sections. The first of the three introduces CFD and presents the fundamental procedure of a CFD simulation. The second section describes the CFD mathematical model, which is the individual foundation on which every CFD simulation is based. Within the mathematical model section the fundamental equations, turbulence modelling and near-wall modelling are introduced. The third and final section shows the numerical method which is used to provide a solution for a given mathematical model.

Throughout the study the commercial CFD code Star-CCM (from version 9-11), has been used. As a result, the *Star-CCM+ User Guide* is a source of continued reference in this chapter as well as two other sources: *Versteeg and Malalasekera (2007)* and the *Fluent User guide.*

3.2 Background

CFD is a mathematical and computational process whereby a mathematical model of fluid dynamics is resolved to simulate a fluid flow. CFD bridges a gap between fluid dynamics, mathematics and computer science. In terms of its place in fluid dynamics, it sits between pure theory and pure experiment *Anderson (1995)*.

As a research tool, CFD has many benefits and allows a trained user to obtain results that would be very expensive and challenging using experimental techniques. CFD avoids the need to make physical models and experimental apparatus, and gives results in a repeatable environment. A great benefit of CFD comes in the post-processing of data as it has the ability to analyses an extensive number of flow parameters at any point in the flow domain. Analysis on this level of detail is often unfeasible with experimental techniques.

The origins of CFD lie in meteorology. *Roache (1972)* provides an interesting overview of the background of CFD, which may be of interest to the reader. By the 1960s aerospace engineers were beginning to use CFD to carry out basic analyses of aerodynamics. The development of the then ‘high speed’ digital computers allowed numerical solutions of the governing Navier-Stokes equations to be obtained. Since this initial realization that computational methods can be used to solve the governing equations, CFD has grown significantly. It has not been until relatively recently that commercial software has been readily available with ‘user friendly’ interfaces. Now, in 2006, commercial CFD software is commonplace in many industries. Its development, mostly driven by advancing computer resources, is progressing rapidly and will continue to do so for the near future.

CFD has extremely broad applications, which span beyond the bounds of simply mechanical engineering. When CFD has been applied to external flows, it has traditionally been in the aerospace, motor-sport and marine industry. In daily life there are numerous examples of fluid flow, and with each example comes a possible application of CFD: the heat transfer from a radiator or air conditioning unit to a room, the prediction of meteorological patterns in the atmosphere, the flow

of water from bath taps, the mixing process in internal combustion engines, blood flow in veins, the flow of steam in power plants, and many more.

It was once conceived that CFD might completely replace experimental testing. This certainly has not been the case. It is true that since the advent of CFD some experimental testing has been less necessary. However, experimental results remain vital in all fluid dynamics applications, which should become apparent as the thesis progresses.

3.3 CFD Methodology

The methodology of the CFD analysis can be neatly divided into three stages; pre-processing, solving and post-processing. Further elaboration of these stages is presented once they have been introduced in this section.

3.3.1 Pre-Processing

Before any flow is simulated the problem has to be defined, which is often referred to as the pre-processing stage. The stage itself divides into three subsections; geometry acquisition, mesh generation and definition of mathematical model.

The starting point of a CFD simulation is the definition of the geometry and the flow domain; the area in which the fluid flows. Traditionally the geometry is obtained using computer aided design (CAD). In this study, as well as traditional CAD, a relatively new technique of geometry acquisition is used, that of 3D non-contact laser scanning. Once the geometry is obtained it is placed inside a flow domain, akin to a computational wind tunnel, giving the completed geometry.

The generation of the computational mesh is essentially dividing the flow domain into many cells, which are all control volumes. Of all the stages in a CFD analysis it is the mesh generation stage that tends to take the most time, and requires the most skill and understanding from the user. A good mesh will make the best use of the available resources by having an appropriate cell size and orientation, in the correct places. On the other hand, a poor mesh may give an enormously misleading simulation.

Once the mesh is completed, all that remains of the pre-processing stage is to define the mathematical model. Things that may require definition at this stage are boundary conditions, fluid properties, governing equations and additional equations to account for matters such as turbulence.

3.3.2 Solving

The stage in which the equations of fluid motion are solved comes when the problem has been entirely defined. The code in this study, and many other commercial CFD codes, uses the finite volume method. This involves the following steps:

1. Formal integration of the governing equations over all the cells in the flow domain.
2. Conversion of the integrated equations into a system of algebraic equations using a variety of approximations.
3. Solution of algebraic equations by an iterative method.

Depending on the size of the problem and the computational resources, this stage may take anything from seconds to days. During the iterative process there is little to no interaction from the user, therefore, if the option is available, this task will often be submitted to an external server, and retrieved when it is complete.

3.3.3 Post-Processing

The post-processing stage of the analysis involves the retrieval of data from the solution files. It is at this stage where CFD is often advantageous over physical testing. It is possible to retrieve a number of flow parameters at all points in the flow domain and output the results as high-resolution graphical outputs. The graphics can be manipulated to a suitable range and view and exported as either static images or animations. Such output can have a very powerful impact and have become commonplace in descriptions of fluid flow

.

3.4 Mesh Generation

Mesh generation, as has already been stated, is the stage in the simulation process that often requires the most time, skill and understanding from the user. The meshing process involves dividing the flow domain into a number of elements or cells. The size, shape and orientation of the cells can have a large effect on the flow solution. The specific mesh generation techniques used in this study will be discussed when each case is introduced. This brief section should serve as an introduction to mesh generation, which will be further developed with the later discussions.

There are some good practices for ensuring a good quality mesh. Firstly, in the majority of cases, the smaller the cells the more accurate the solution will be. However, the computational effort required increases in proportion to the number of cells. Bearing this in mind, it is therefore important to maximize the available resources by making the most of the potential number of cells in a mesh. It is most desirable to have small cells in the regions of the flow where the high gradients occur, i.e. where the flow is changing most rapidly over the smallest distance. Consider the case of the flow around a sphere. Close to the surface, and in the wake, high gradients will be present. It is therefore in these regions where the cell size should be small. In the free stream the gradients are relatively small and in turn the elements can be larger. With a situation such as this the transition between large and small elements should be as smooth and gradual as possible to avoid a poor quality mesh. Another important consideration in the creation of a computational mesh is that, where possible, the flow should travel either perpendicularly or parallel to a cell face. In other words the flow should not enter a cell at a skewed angle. Entire satisfaction of this constraint is impossible for all but the most uniform of flows. Referring again to the flow around a sphere, although the obstruction will create recirculation in the wake, the average direction of the flow will be clearly defined. It is therefore possible to create cells that, in most cases, satisfy the perpendicular and parallel constraint for the majority of the flow domain, especially in the free stream. There are further guidelines regarding mesh generation in the near wall region which will be introduced when modelling of the flow in the flow region is examined later in the chapter.

3.5 The Mathematical Model

The mathematical model is essentially the definition of the entire CFD simulation without the solution strategy. It consists of governing equations, additional equations, boundary conditions, fluid properties, geometry and physical dimensions. The first four on this list will be discussed in this section. The definition of the mathematical model is an important stage of this study, as it is this that will be modified specifically for the analysis of cycling aerodynamics.

3.5.1 The governing equations

An appropriate starting point for this discussion is the Euler equations, as this is the foundation upon which the governing equations are based. Eighteenth century mathematician Leonard Euler takes the credit for the equations which are based upon the three following principles:

1. The mass of a fluid is conserved (continuity equation)
2. The force on a fluid particle is equal to its mass multiplied by its acceleration, Newton's second law (momentum equation)
3. The energy in a fluid is conserved (energy equation)

The conservation of energy will not be employed in this investigation. The energy equation is required when heat transfer is present and maybe required when there are extreme velocity gradients in the flow, for example a rapidly converging duct. In the case of cycling aerodynamics the flow is assumed isothermal and the largest of velocity gradients would not warrant the use of the energy equation, hence no equation of state is required.

Towards the end of the eighteenth and at the start of the nineteenth century the independently combined efforts of two Frenchmen, Claude-Louis-Marie-Henri Navier and Jean-Claude Barre de Saint-Venant, and one Irishman, Sir George Gabriel Stokes, furthered the inviscid Euler equations to include the effects of friction. The result was the governing equations of a three-dimensional, unsteady and viscous flow, the Navier-Stokes equations. As with the Euler equations the Navier-Stokes equations described continuity, momentum and energy.

3.5.2 The different approaches of CFD

In developing an understanding of the mathematical model of CFD, it is important that the reader appreciates the different CFD approaches. The process of solving the numerical equations in CFD takes on different forms ranging from time-averaged approaches at one extreme to a direct resolution of the governing equations at the other. The skeleton of the foundations of CFD is shown in Figure 10.

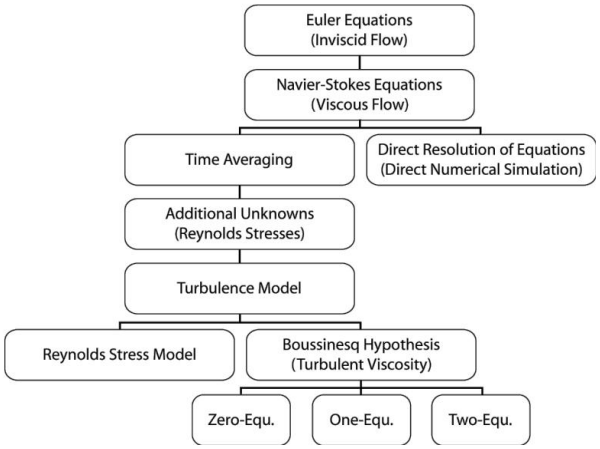


Figure 10. Schematic foundations of CFD experiments

3.5.3 Time Averaging: Reynolds-Averaged Navier-Stokes

The time averaging approach to flow problems, pioneered by Osborne Reynolds at the end of the nineteenth century, in essence avoids calculating all the fluctuations of a turbulent flow and simply computes the mean characteristics of a flow problem. This solution involves time averaging the governing Navier-Stokes equations, giving the Reynolds-Averaged Navier-Stokes (RANS) equations,

Time-averaged continuity

$$\frac{\partial U}{\partial x}+\frac{\partial V}{\partial y}+\frac{\partial W}{\partial z}=0$$

Time-averaged momentum

x-direction

$$\rho\left(U\frac{\partial U}{\partial x}+V\frac{\partial U}{\partial y}+W\frac{\partial U}{\partial z}\right)=\frac{\partial}{\partial x}\left(\mu\frac{\partial U}{\partial x}-\rho\overline{u'^2}\right)+\frac{\partial}{\partial y}\left(\mu\frac{\partial U}{\partial y}-\rho\overline{u'v'}\right)+\frac{\partial}{\partial z}\left(\mu\frac{\partial U}{\partial z}-\rho\overline{u'w'}\right)-\frac{\partial P}{\partial x}$$

y-direction

$$\rho\left(U\frac{\partial V}{\partial x}+V\frac{\partial V}{\partial y}+W\frac{\partial V}{\partial z}\right)=\frac{\partial}{\partial x}\left(\mu\frac{\partial V}{\partial x}-\rho\overline{u'v'}\right)+\frac{\partial}{\partial y}\left(\mu\frac{\partial V}{\partial y}-\rho\overline{v'^2}\right)+\frac{\partial}{\partial z}\left(\mu\frac{\partial V}{\partial z}-\rho\overline{v'w'}\right)-\frac{\partial P}{\partial y}$$

z-direction

$$\rho\left(U\frac{\partial W}{\partial x}+V\frac{\partial W}{\partial y}+W\frac{\partial W}{\partial z}\right)$$
$$=\frac{\partial}{\partial x}\left(\mu\frac{\partial W}{\partial x}-\rho\overline{u'w'}\right)+\frac{\partial}{\partial y}\left(\mu\frac{\partial W}{\partial y}-\rho\overline{w'v'}\right)+\frac{\partial}{\partial z}\left(\mu\frac{\partial U}{\partial z}-\rho\overline{w'^2}\right)-\frac{\partial P}{\partial z}$$

where, U, V and W represent the time-averaged velocity and *u*', *v*' and *w*' the instantaneous velocity in the x, y and z directions, *x*, *y* and *z* represent the displacement in the respective direction, μ is the dynamic viscosity and *P* is the time-averaged pressure. The Reynolds stress terms: $-\rho\overline{u'^2}$, $-\rho\overline{v'^2}$, $-\rho\overline{w'^2}$, $-\rho\overline{u'v'}$, $-\rho\overline{u'w'}$ and $-\rho\overline{v'w'}$ now appear in the time-averaged momentum equation. These terms are additional unknowns within the RANS equations and involve second-order moments of the fluctuating velocity field. The cost of time averaging the Navier-Stokes equations is that there are now more unknowns than equations, and thus this is no longer a closed problem. The fact that the RANS equations are not closed reflects the incompleteness of the mean flow as a full description of turbulent flow.

In an attempt to close the RANS equations by solving the Reynolds stresses one may call upon a turbulence model, which is a set of algebraic or differential equations.

3.5.4 Direct numerical simulation

In contrast to solutions based on averaged equations, with closure approximations, the Direct Numerical Simulation (DNS) method solves the three-dimensional Navier-Stokes equations numerically. Apart from the inherent spatial and temporal discretization, DNS involves no approximations, leading to immaculately accurate results. The weakness of DNS is in its immense computational time and storage requirements, which both increase rapidly with Re. Due to the immense expense of DNS it is entirely impractical as an engineering tool, and at present is limited only to the world's fastest computers.

3.5.5 Large eddy simulation (LES) and Detached-Eddy Simulation (DES)

RANS and DNS are clearly the two extremes in terms of required resources and output. A technique exists which attempts to combine the advantages of these two approaches.

Large eddy simulation (LES) is a popular technique for simulating turbulent flows. An implication of Kolmogorov's (1941) theory of self similarity is that the large eddies of the flow are dependant on the geometry while the smaller scales more universal. This feature allows one to explicitly solve for the large eddies in a calculation and implicitly account for the small eddies by using a subgrid-scale model (SGS model).

The simulation of turbulent flows by numerically solving the Navier–Stokes equations requires resolving a very wide range of time and length scales, all of which affect the flow field. Such a resolution can be achieved with direct numerical simulation (DNS), but DNS is computationally expensive, and its cost prohibits simulation of practical engineering systems with complex geometry or flow configurations, such as turbulent jets, pumps, vehicles, and landing gear.

The principal idea behind LES is to reduce the computational cost by ignoring the smallest length scales, which are the most computationally expensive to resolve, via low-pass filtering of the Navier–Stokes equations. Such a low-pass filtering, which can be viewed as a time- and spatial-averaging, effectively removes small-scale information from the numerical solution. This information is not irrelevant, however, and its effect on the flow field must be modeled, a task which is an active area of research for problems in which small-scales can play an important role, such as near-wall flows reacting flows, and multiphase flows.

Mathematically, one may think of separating the velocity field into a resolved and sub-grid part. The resolved part of the field represent the "large" eddies, while the sub-grid part of the velocity represent the "small scales" whose effect on the resolved field is included through the sub-grid-scale model. Formally, one may think of filtering as the convolution of a function with a filtering kernel G:

$$\bar{u}_i(\vec{x}) = \int G(\vec{x} - \vec{\xi}) u(\vec{\xi}) d\vec{\xi},$$

resulting in

$$u_i = \bar{u}_i + u_i',$$

where $\bar{u}_i$ is the resolvable scale part and u_i' is the subgrid-scale part. However, most practical (and commercial) implementations of LES use the grid itself as the filter (the box filter) and perform no explicit filtering.

This page is mainly focused on LES of incompressible flows. For compressible flows, see Favre averaged Navier-Stokes equations.

The filtered equations are developed from the incompressible Navier-Stokes equations of motion:

$$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\nu \frac{\partial u_i}{\partial x_j}\right).$$

Substituting in the decomposition $u_i = \bar{u}_i + u_i'$ and $p = \bar{p} + p'$ and then filtering the resulting equation gives the equations of motion for the resolved field:

$$\frac{\partial \bar{u}_i}{\partial t} + \bar{u}_j \frac{\partial \bar{u}_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \bar{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\nu \frac{\partial \bar{u}_i}{\partial x_j}\right) + \frac{1}{\rho}\frac{\partial \tau_{ij}}{\partial x_j}.$$

We have assumed that the filtering operation and the differentiation operation commute, which is not generally the case. It is thought that the errors associated with this assumption are usually small.

The extra term $\frac{\partial \tau_{ij}}{\partial x_j}$ arises from the non-linear advection terms, due to the fact that

$$\overline{u_j \frac{\partial u_i}{\partial x_j}} \neq \bar{u}_j \frac{\partial \bar{u}_i}{\partial x_j}$$

and hence

$$\tau_{ij} = \bar{u}_i \bar{u}_j - \overline{u_i u_j}$$

Similar equations can be derived for the subgrid-scale field (i.e. the residual field).

Subgrid-scale turbulence models usually employ the Boussinesq hypothesis, and seek to calculate (the deviatoric part of) the SGS stress using:

$$\tau_{ij} - \frac{1}{3}\tau_{kk}\delta_{ij} = -2\mu_t \bar{S}_{ij}$$

where $\bar{S}_{ij}$ is the rate-of-strain tensor for the resolved scale defined by

$$\bar{S}_{ij} = \frac{1}{2}\left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}\right)$$

and ν_t is the subgrid-scale turbulent viscosity. Substituting into the filtered Navier-Stokes equations, we then have

$$\frac{\partial \bar{u}_i}{\partial t} + \bar{u}_j \frac{\partial \bar{u}_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \bar{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left([\nu + \nu_t]\frac{\partial \bar{u}_i}{\partial x_j}\right),$$

where we have used the incompressibility constraint to simplify the equation and the pressure is now modified to include the trace term $\tau_{kk}\delta_{ij}/3$.

The difficulties associated with the use of the standard LES models, particularly in near-wall regions, has lead to the development of hybrid models that attempt to combine the best aspects of RANS and LES methodologies in a single solution strategy. An example of a hybrid technique is the detached-eddy simulation (DES) approach. This model attempts to treat near-wall regions in a RANS-like manner, and treat the rest of the flow in an LES-like manner. The model was originally formulated by replacing the distance function d in the Spalart-Allmaras model with a modified distance function

$$\tilde{d} = \min[d, C_{\mathrm{DES}}\Delta],$$

where C_{DES} is a constant and Δ is the largest dimension of the grid cell in question. This modification of the S-A model, while very simple in nature, changes the interpretation of the model substantially. This modified distance function causes the model to behave as a RANS model in regions close to walls, and in a Smagorinsky-like manner away from the walls. This is usually justifed with arguments that the scale-dependence of the model is made local rather than global, and that dimensional analysis backs up this claim.

The DES approach may be used with any turbulence model that has an appropriately defined turbulence length scale (distance in the S-A model) and is a sufficiently localized model.

In practice, more programming is needed than simply changing the calculation of the length scale. Many implementations of the DES approach allow for regions to be explicitly designated as RANS or LES regions, overruling the distance function calculation. Also, many implementations use different differencing in RANS regions (e.g. upwinded differences) and LES regions (e.g. central differences).

Regions near solid boundaries and where the turbulent length scale is less than the maximum grid dimension are assigned the RANS mode of solution. As the turbulent length scale exceeds the grid dimension, the regions are solved using the LES mode. Therefore, the grid resolution is not as demanding as pure LES, thereby considerably cutting down the cost of the computation.

Due to the huge requirements of LES and DES, this analysis will mainly use RANS techniques for the CFD analysis.

3.6 Moving Reference Frames

To obtain a steady-state analysis, we applied the MRF (Multiple Reference Frames) method in which a different frame can rotate (and/or translate) with respect to the laboratory reference frame. By imposing a rotating frame to the region containing the wheel we took into account the effect of the rotation on the fluid around it, without providing a rigid movement to the wheel and an unsteady simulation needing more computational resources.

3.6.1 Governing Equations in Moving Reference Frames

Moving Reference Frames (MRF) are reference frames that can rotate and translate with respect to the laboratory reference frame.

MRF models in STAR-CCM+ assume that:

- The angular velocity of the body is constant.
- The mesh is rigid.

The governing equations can be written in terms of:

- the velocity in the laboratory reference frame (absolute velocity), or
- the velocity relative to the moving reference frame (relative velocity).

This section presents the incompressible Navier-Stokes equations for the sake of simplicity. The extra terms that appear in the compressible Navier-Stokes equations receive no contribution from the motion of the reference frame.

The major source for this section is the Star-CCM+ User Guide.

In the laboratory reference frame, the incompressible Navier-Stokes equations are written as:

$$\nabla \cdot \mathbf{u}_0 = 0$$

$$\frac{D\mathbf{u}_0}{Dt} = -\nabla (p/\rho) + \nu \nabla \cdot \nabla \mathbf{u}_0$$

Expanding the substantial derivative in the momentum equation:

$$\frac{\partial \mathbf{u}_0}{\partial t} + \nabla \cdot (\mathbf{u}_0 \otimes \mathbf{u}_0) = -\nabla (p/\rho) + \nu \nabla \cdot \nabla \mathbf{u}_0$$

The following relation is used to transform the convection term in the conservative form:

$$\nabla \cdot (\mathbf{u}_0 \otimes \mathbf{u}_0) = \mathbf{u}_0 \cdot \nabla \mathbf{u}_0 + \underbrace{\left(\nabla \cdot \mathbf{u}_0\right)}_{=0} \mathbf{u}_0$$

Using the previous the momentum equation becomes:

$$\frac{\partial \mathbf{u}_0}{\partial t} + \mathbf{u}_0 \cdot \nabla \mathbf{u}_0 = -\nabla (p/\rho) + \nu \nabla \cdot \nabla \mathbf{u}_0$$

Incompressible Navier-Stokes Equations in a Moving Reference Frame with Relative Velocity

Consider a moving reference frame p rotating with a constant angular velocity $\boldsymbol{\omega}_p$ as measured in the laboratory frame 0. The velocity $\mathbf{u}_p$ relative to the moving reference frame is related to the absolute velocity $\mathbf{u}_0$:

$$\mathbf{u}_0 = \mathbf{u}_p + \boldsymbol{\omega}_p \times \mathbf{r}$$

where $\mathbf{r}$ denotes the position vector from the origin of the moving reference frame p to the center of a control volume.

the previous equation is used to obtain the Navier-Stokes equations in terms of the relative velocity. First, the left-hand side of is transformed:

$$\frac{D\mathbf{u}_0}{Dt} = \frac{D\mathbf{u}_p}{Dt} + \frac{\mathrm{d}\,\boldsymbol{\omega}_p}{\mathrm{d}\,t} \times \mathbf{r} + 2\boldsymbol{\omega}_p \times \mathbf{u}_p + \boldsymbol{\omega}_p \times \boldsymbol{\omega}_p \times \mathbf{r}$$

$$\frac{D\mathbf{u}_0}{Dt} = \frac{\partial \mathbf{u}_p}{\partial t} + \nabla \cdot \left(\mathbf{u}_p \otimes \mathbf{u}_p\right) + \frac{\mathrm{d}\,\boldsymbol{\omega}_p}{\mathrm{d}\,t} \times \mathbf{r} + 2\boldsymbol{\omega}_p \times \mathbf{u}_p + \boldsymbol{\omega}_p \times \boldsymbol{\omega}_p \times \mathbf{r}$$

The diffusive term is:

$$\begin{aligned} \nabla \cdot \nabla \mathbf{u}_0 &= \nabla \cdot \nabla \left[\mathbf{u}_p + \boldsymbol{\omega}_p \times \mathbf{r}\right] \\ &= \nabla \cdot \nabla \mathbf{u}_p + \underbrace{\nabla \cdot \nabla \left(\boldsymbol{\omega}_p \times \mathbf{r}\right)}_{=0} \\ &= \nabla \cdot \nabla \mathbf{u}_p \end{aligned}$$

Applying all transformations, the momentum equation becomes:

$$\frac{\partial \mathbf{u}_p}{\partial t} + \nabla \cdot \left(\mathbf{u}_p \otimes \mathbf{u}_p\right) + \frac{\mathrm{d}\,\boldsymbol{\omega}_p}{\mathrm{d}\,t} \times \mathbf{r} + 2\boldsymbol{\omega}_p \times \mathbf{u}_p + \boldsymbol{\omega}_p \times \boldsymbol{\omega}_p \times \mathbf{r} = -\nabla (p/\rho) + \nu \nabla \cdot \nabla \mathbf{u}_p$$

Both the convection and convected velocities are in terms of velocity relative to the rotating frame. This formulation is also known as the relative velocity formulation.

Similarly, the continuity equation is transformed using:

$$\begin{aligned} \nabla \cdot \mathbf{u}_0 &= \nabla \cdot \left[\mathbf{u}_p + \boldsymbol{\omega}_p \times \mathbf{r}\right] \\ &= \nabla \cdot \mathbf{u}_p + \underbrace{\nabla \cdot \left(\boldsymbol{\omega}_p \times \mathbf{r}\right)}_{=0} \\ &= \nabla \cdot \mathbf{u}_p \end{aligned}$$

giving

$$\nabla \cdot \mathbf{u}_p = 0$$

Incompressible Navier-Stokes Equations in a Moving Reference Frame with Absolute Velocity

In this form, the convection velocity is the velocity $\mathbf{u}_p$ relative to the moving reference frame and the convected velocity is in terms of the velocity $\mathbf{u}_0$ in the laboratory frame (that is, the absolute velocity). The convection term is of the form $\nabla \cdot \left(\mathbf{u}_p \otimes \mathbf{u}_0\right)$.

The term corresponding to the convective fluxes $\nabla \cdot \left(\mathbf{u}_p \otimes \mathbf{u}_p\right)$ is written as:

$$\begin{aligned} \nabla \cdot \left(\mathbf{u}_p \otimes \mathbf{u}_p\right) &= \nabla \cdot \left[\mathbf{u}_p \otimes (\mathbf{u}_0 - \boldsymbol{\omega}_p \times \mathbf{r})\right] \\ &= \nabla \cdot \left(\mathbf{u}_p \otimes \mathbf{u}_0\right) - \underbrace{\nabla \cdot \mathbf{u}_p}_{=0} (\boldsymbol{\omega}_p \times \mathbf{r}) - \mathbf{u}_p \cdot \nabla (\boldsymbol{\omega}_p \times \mathbf{r}) \end{aligned}$$

The term $\mathbf{u}_p \cdot \nabla(\boldsymbol{\omega}_p \times \mathbf{r})$ is expanded as follows:

$$\mathbf{u}_p \cdot \nabla(\boldsymbol{\omega}_p \times \mathbf{r}) = \left(\mathbf{u}_p \cdot \underbrace{\nabla \boldsymbol{\omega}_p}_{=0}\right) \times \mathbf{r} + \boldsymbol{\omega}_p \times \left(\mathbf{u}_p \cdot \nabla \mathbf{r}\right)$$
$$= \boldsymbol{\omega}_p \times \mathbf{u}_p \quad \text{with} \quad \nabla \mathbf{r} = \mathbf{I}$$

Therefore

$$\nabla \cdot \left(\mathbf{u}_p \otimes \mathbf{u}_p\right) = \nabla \cdot \left(\mathbf{u}_p \otimes \mathbf{u}_0\right) - \boldsymbol{\omega}_p \times \mathbf{u}_p$$

Simplifying the steady terms in the left-hand side:

$$\nabla \cdot \left(\mathbf{u}_p \otimes \mathbf{u}_p\right) + 2\boldsymbol{\omega}_p \times \mathbf{u}_p + \boldsymbol{\omega}_p \times \boldsymbol{\omega}_p \times r$$
$$= \nabla \cdot \left(\mathbf{u}_p \otimes \mathbf{u}_0\right) + \boldsymbol{\omega}_p \times \mathbf{u}_p + \boldsymbol{\omega}_p \times \boldsymbol{\omega}_p \times r$$
$$= \nabla \cdot \left(\mathbf{u}_p \otimes \mathbf{u}_0\right) + \boldsymbol{\omega}_p \times (\mathbf{u}_p + \boldsymbol{\omega}_p \times \mathbf{r})$$
$$= \nabla \cdot \left(\mathbf{u}_p \otimes \mathbf{u}_0\right) + \boldsymbol{\omega}_p \times \mathbf{u}_0$$

The unsteady term is transformed as:

$$\frac{\partial \mathbf{u}_p}{\partial t} = \frac{\partial \mathbf{u}_0}{\partial t} - \frac{\partial(\boldsymbol{\omega}_p \times \mathbf{r})}{\partial t}$$
$$= \frac{\partial \mathbf{u}_0}{\partial t} - \underbrace{\frac{\partial \boldsymbol{\omega}_p}{\partial t}}_{=0} \times \mathbf{r} - \boldsymbol{\omega}_p \times \underbrace{\frac{\partial \mathbf{r}}{\partial t}}_{=0}$$

Using all the transformed terms, the momentum equation in terms of the absolute velocity is:

$$\frac{\partial \mathbf{u}_0}{\partial t} + \frac{\mathrm{d}\,\boldsymbol{\omega}_p}{\mathrm{d}\,t} \times \mathbf{r} + \nabla \cdot \left(\mathbf{u}_p \otimes \mathbf{u}_0\right) + \boldsymbol{\omega}_p \times \mathbf{u}_0 = -\nabla(p/\rho) + \nu \nabla \cdot \nabla \mathbf{u}_0$$

This is the momentum equation in the moving reference frame with the absolute velocity formulation. In this formulation, $\nabla \cdot \mathbf{u}_0 = 0$ is the continuity equation. This formulation is solved in STAR-CCM+.

3.7 Turbulence Modelling

It has been established that the RANS approach requires a turbulence model. There are two variations of turbulent modelling, one follows the Reynolds Stress Model (RSM) and the other the Boussinesq Hypothesis.

3.7.1 Reynolds Stress Model

The RSM provides a direct solution by calculating the individual Reynolds stresses using differential transport equations. When using the RSM, seven additional transport equations must be solved for three-dimensional flow. The RSM is more computationally demanding than the other RANS approaches, requiring 50-60% more time than two-equation Boussinesq Hypothesis models. The RSM has been found to be superior for situations in which the anisotropy of turbulence has a dominant effect on the mean flow. When the flow is isotropic the additional computational effort of using the RSM cannot be justified as it offers little to no improvement over the Boussinesq alternatives. The external aerodynamic flow around a bike and a rider is highly unidirectional and can therefore be assumed to be isotropic. It is for this reason that the greater computational requirements of the RSM are not warranted, and therefore it will not be used in this analysis.

3.7.2 Boussinesq hypothesis

The traditional approach to closing the RANS equations involves, firstly, a hypothesis proposed by Valentin Joseph Boussinesq in 1877, which hypothesizes Reynolds stress, τ, as shown,

$$\tau = -\rho\overline{u'v'} = \mu_t \frac{dU}{dy}$$

Where, μ_t is the eddy (turbulent) viscosity. This hypothesis provides the foundation upon which engineering turbulence models are built. In 1925 Ludwig Prandtl used dimensional reasoning to derive the Prandtl Eddy Viscosity Formula, given by,

$$\mu_t \propto \rho l_s V_S$$

where, l_s represents the length scale and V_s the velocity scale. Since the framework of eddy viscosity modelling was put in place by Boussinesq and Prandtl, three methods have been derived for calculating l_s and V_s, called zero equation, one equation and two equation models.

The zero-equation model employs the Prandtl eddy viscosity formula, allowing for independent choice of l_s and V_s.
As a result, no additional differential equation to the equations of motion is required. Such an analysis is accurate for flow in local equilibrium, for example in the boundary layer, however it is not possible to prescribe this analysis to more complex flows where the flow is recirculating or rapidly evolving. Zero-equation models include the Prandtl/Taylor Mixing Length Model, the Smagorinsky Model and the Baldwin and Lomax Model.

One-equation models also employ the Prandtl eddy viscosity formula. The length scale is prescribed in a similar way to the zero-equation model, however V_s is calculated using an additional formula. The velocity scale is dependent on a variable, *k*, which is the time-averaged turbulence kinetic energy. Using this method, one partial differential equation in addition to those in the equations of motion is solved. One-equation models include the Kolmogorov Prandtl Model and the Spalart-Allmaras Model.

Two-equation models again employ the Prandtl eddy viscosity formula. In this formulation, two equations are produced for calculating l_s and V_s. Both equations relate either l_s or V_s to the time-averaged turbulence kinetic energy, *k*. A number of proposals for length scale have been produced, resulting in different turbulence models. Two-equation models include the *k-ε* and *k-ω* models.

The Boussinesq approach is based on assuming that the turbulent viscosity is isotropic, and when this not acceptable the RSM is employed. The remainder of the section will examine the turbulence models used throughout this analysis. Some of the governing formulae behind the turbulence models will be presented, although not all. If the reader seeks a complete description of the numerical models, attention should be paid to the cited publications and the Star-CCM+ manual.

3.7.3 Spalart-Allmaras

The Spalart-Allmaras model is a one-equation model used for low Reynolds number (Re) wall bounded flows. It is the least computationally expensive turbulence model available in Star-CCM. It will be used in this study because it is known to give good results for boundary layers subject to adverse pressure gradients. However, as it is a one-equation model it may not give a suitable modelling of the turbulent flow, especially when the geometry becomes more complex. This model cannot be relied upon to predict the decay of homogeneous, isotropic turbulence. The transported variable in the Spalart-Allmaras model, $\tilde{v}$, is determined by,

$$\rho\frac{\tilde{v}}{Dt} = G_v + \frac{1}{\sigma_v}\left(\frac{\partial}{\partial x}\left((\mu + \rho\tilde{v})\frac{\partial\tilde{v}}{\partial x}\right) + C_{b2}\rho\left(\frac{\partial\tilde{v}}{\partial x}\right)^2\right) - Y_v$$

where, $\tilde{v}$ is the transported variable, G_v is the production of turbulent viscosity, Y_v is the destruction of turbulent viscosity that occurs in the near-wall region, and σ_v and C_{b2} are model constants. The turbulent viscosity is calculated as,

$$\mu_t = \rho\tilde{v}f_{v1}$$

where, f_{v1} is the viscous damping function.

3.7.4 Standard *k-ε*

This is the simplest and most popular of the two-equation models. Many authors have contributed to its formulation, with the work of *Jones and Launder (1972)* being at the forefront of the development. The model introduces a new variable, ε, which is the rate of dissipation of turbulence kinetic energy, k. ε is used alongside k to calculate the length scale. This model allows for the production and destruction of turbulence. The standard *k-ε* model provides robust and economic solutions, although its accuracy can sometimes be brought into question. The robust nature of the model makes it immediately suitable for this study, as a number of complex geometries and meshes will be used. The two transport equations for the standard *k-ε* model are,

$$\rho\frac{dk}{dt} = \frac{\partial}{\partial x}\left(\left(\mu + \frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x}\right) + G_k + G_b - \rho\varepsilon - Y_M$$

$$\frac{d\varepsilon}{dt} = \frac{\partial}{\partial x}\left(\left(\mu + \frac{\mu_t}{\sigma_\varepsilon}\right)\frac{\partial\varepsilon}{\partial x}\right) + C_{1\varepsilon}\frac{\varepsilon}{k}(G_k + C_{3\varepsilon}G_b) - \frac{C_{2\varepsilon}\rho\varepsilon^2}{k}$$

where, G_k and G_b account for the generation of turbulent kinetic energy due to mean velocity gradients and buoyancy, respectively, Y_M represents the contribution of the fluctuating dilation in compressible turbulence to the overall dissipation rate, $C_{1\varepsilon}$, $C_{2\varepsilon}$ and $C_{3\varepsilon}$ are constants and σ_k and σ_ε are turbulent Prandtl numbers for k and ε, respectively, The turbulent viscosity is determined by:

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

where, the term C_μ is another constant.

3.7.5 RNG *k-ε*

The RNG k-ε model is derived using a statistical technique called renormalization group theory, hence the name. This model offers improvement for rapidly strained flows and swirling flows, which may not be relevant to cycling aerodynamics. It includes an additional term in the epsilon equation,

$$\rho\frac{dk}{dt} = \frac{\partial}{\partial x}\left(\alpha_k\mu_{eff}\frac{\partial k}{\partial x}\right) + G_k + G_b - \rho\varepsilon - Y_M$$

$$\frac{d\varepsilon}{dt} = \frac{\partial}{\partial x}\left(\alpha_\varepsilon\mu_{eff}\frac{\partial\varepsilon}{\partial x}\right) + C_{1\varepsilon}\frac{\varepsilon}{k}(G_k + C_{3\varepsilon}G_b) - \frac{C_{2\varepsilon}\rho\varepsilon^2}{k} - R$$

where, the terms α_k and α_ε denote the inverse effective Prandtl numbers for k and ε, respectively. The model calculates Prandtl number instead of using a constant value. It requires 10-15% more computational time than the standard model. The turbulent viscosity is still calculated in the same manner. Unlike the standard model, the RNG model operates for low Re flows, although its accuracy at doing so is highly dependent upon adequate representation of the near-wall region. The model is susceptible to instability in steady-state solutions and is subsequently more likely to recognize unsteady phenomena, such as vortex shedding.

3.7.6 Realizable *k-ε*

This model has only a slight increase in computational demands compared to the standard *k-ε* model. It uses an alternative transport equation for dissipation and has a modified formulation for turbulent viscosity. The term *realizable* means that the model satisfies certain mathematical constraints on the Reynolds stresses which are consistent with the physics of turbulent flows. A benefit of the realizable model is that it provides superior performance for flows involving rotation, boundary layers under strong adverse pressure gradients, separation and recirculation. The flow around a cyclist will be subject to separation and recirculation, therefore this model may well provide the most accurate solution of all the *k-ε* models used. The transport equations for the realizable model are,

$$\rho\frac{dk}{dt} = \frac{\partial}{\partial x}\left(\left(\mu + \frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x}\right) + G_k + G_b - \rho\varepsilon - Y_M$$

$$\frac{d\varepsilon}{dt} = \frac{\partial}{\partial x}\left(\left(\mu + \frac{\mu_t}{\sigma_\varepsilon}\right)\frac{\partial \varepsilon}{\partial x}\right) + \rho C_{1\varepsilon} S\varepsilon - C_{2\varepsilon}\rho\frac{\varepsilon^2}{k + \sqrt{\nu\varepsilon}} + C_{i\varepsilon}$$

Evidently, the formulation of the realizable model is very similar to the previous two models. The turbulent kinetic energy remains as it is the standard model, however C_μ is no longer constant, it is now computed from an additional formulation.

3.7.7 Standard k-ω

The k-ω model was first proposed by *Kolmogorov (1942)*. Further authors such as *Saffman (1970)* and *Wilcox (2000)* have contributed to the model formulation. Accurate modifications for low Re effects would be highly beneficial to this study, so this model is used in anticipation that it may offer improvement over the k-ε models. As this is also a two-equation model, the computational effort required is comparable to the k-ε models. As with k-ε, k represents the turbulence kinetic energy. The additional ω is the specific dissipation rate and can be regarded as the ratio of ε to k. The transport equations for the standard k-ω model are,

$$\frac{\partial}{\partial t}(\rho k) + \frac{\partial}{\partial x}(\rho k u) = \frac{\partial}{\partial x}\left(\Gamma_k \frac{\partial k}{\partial x}\right) + G_k - Y_k + S_k$$

$$\frac{\partial}{\partial t}(\rho \omega) + \frac{\partial}{\partial x}(\rho \omega u) = \frac{\partial}{\partial x}\left(\Gamma_\omega \frac{\partial \omega}{\partial x}\right) + G_\omega - Y_\omega + S_\omega$$

where, G_ω is the generation of ω, Γ_k and Γ_ω represent the effective diffusivity of k and ω, respectively, Y_k and Y_ω represent the dissipation of k and ω due to turbulence, and S_k and S_ω are user defined source terms. The turbulent viscosity is calculated using,

$$\mu_t = \alpha^* \frac{\rho k}{\omega}$$

where, α^* is a coefficient which damps the turbulent viscosity causing low *Re* correction. The turbulent viscosity relates to the transport equation via the Γ_k and Γ_ω terms using,

$$\Gamma_k = \mu + \frac{\mu_t}{\sigma_k}$$

$$\Gamma_\omega = \mu + \frac{\mu_t}{\sigma_\omega}$$

Where σ_ω is the Prandtl number for ω.

The wall boundary conditions for the k-ω model are the same as the wall boundary conditions for the k-ε model when using the enhanced wall treatment. This means that a wall function approach

will be used for coarser meshes, whereas meshes with adequate refinement will revert to a two-layer approach, as explained in the following section.

3.7.8 k-ω SST

The shear-stress transport (SST) k-ω model uses an approach combining the merits of the standard k-ω model in near-wall region and the transformed k-ε model in the free stream. Both models, in theory, have aspects suited to cycling analysis, which if combined suitably may be ideal for this study. The SST k-ω model combines the other models by employing a blending function. The blending function is designed to be one in the near-wall region, activating the standard k-ω model, and zero away from the surface, activating the transformed k-ε model. The SST model incorporates a damped cross-diffusion derivative term in the ω equation. Also the definition of μ_t is modified to account for the transport of the turbulent shear stress. The additional modifications in the SST k-ω model make it more accurate and reliable for a wider class of flows. The transport equations for the SST k-ω model are,

$$\frac{\partial}{\partial t}(\rho k) + \frac{\partial}{\partial x}(\rho k u) = \frac{\partial}{\partial x}\left(\Gamma_k \frac{\partial k}{\partial x}\right) + G_k - Y_k + S_k$$

$$\frac{\partial}{\partial t}(\rho \omega) + \frac{\partial}{\partial x}(\rho \omega u) = \frac{\partial}{\partial x}\left(\Gamma_\omega \frac{\partial \omega}{\partial x}\right) + G_\omega - Y_\omega + D_\omega + S_\omega$$

Where D_ω represents a cross-diffusion term. The turbulent viscosity is determined by,

$$\mu_t = \frac{\rho k}{\omega} \frac{1}{\max\left(\frac{1}{\alpha^*}, \frac{SF_2}{a_1 \omega}\right)}$$

Where S is the strain rate magnitude, F_2 is a blending function and a_1 is a model constant.

3.8 Near wall modelling

Accurate modelling in the near wall region, where the boundary layer develops from stationary fluid at the wall to continuously moving fluid in the free-stream, is crucial to the modelling of the entire flow. It is in the near-wall region where the solution variables are changing most rapidly. The boundary layer, although small can have a dominant effect on the overall flow, which was shown at the beginning of Chapter 2. As external aerodynamics is greatly influenced by the presence of walls it is crucial for this investigation that the near-wall region is represented as realistically as possible.

Turbulence models provide solutions in the free-stream, a certain distance from a wall. The technique however is incomplete as it fails to represent the flow in the near-wall region. It is therefore necessary to call upon additional formulae to describe the flow close to the wall, with the exception of the Spalart-Allmaras model. The Spalart-Allmaras model can be applied throughout the boundary layer provided the mesh has a sufficient resolution.

It is important here to reiterate the distinction between a laminar and turbulent boundary layer. Due to the nature of turbulent flow there is more energy being transferred within the fluid. As a result the energy from the outer region of the boundary layer is being readily transferred to the region very close to the wall giving the fluid a higher normalized velocity, and a much steeper velocity gradient close to the wall.

The structure of the turbulent near-wall region can be divided into three sub-layers. The layer closest to the wall is termed the viscous sub-layer; here viscosity plays a dominant role in momentum and heat or mass transfer, thus the flow is similar to a mean laminar flow. The outer layer, furthest from the wall, is dominated by turbulence and is subsequently called the fully

turbulent layer. The intermediate region between the two is where the effects of molecular viscosity and turbulence are equally important. The three regions within the turbulent boundary layer can be demonstrated by examining a plot of u^+ against y^+ (Figure 11), where u+ and y+ are defined as,

$$u^+ = \frac{u}{u^*}$$

$$y^+ = \frac{u^* x_w \rho}{\mu}$$

Where: x_w is the distance from the wall and U_f is the frictional velocity, given by:

$$u^* = \sqrt{\frac{\tau_w}{\rho}}$$

Where τ_w is the shear stress at the wall

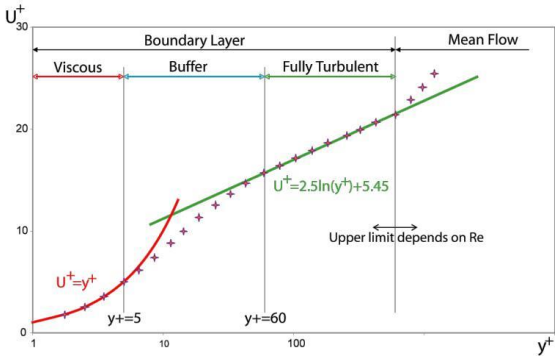


Figure 11. The three divisions of the near-wall region (Fluent manual)

The figure above demonstrates that when y^+ is less than around 5 (the viscous sub-layer) it has a linear relationship with U^+. When y^+ is above around 60 (the fully turbulent layer) it has an inverse logarithmic relationship with U^+. When y^+ is between these values (the buffer layer) the relationship blends from one to the other.

There are two main approaches to simulating the flow in the near-wall region; wall functions and near-wall modelling. These are shown schematically in Figure 12.

Wall functions are the less computationally demanding and do not resolve the viscous effected region. Instead, the area from the wall to the fully turbulent region is computed using semi-empirical formulas called wall functions. The size of this region is determined by the size of the mesh adjacent to the wall.

Near-wall modelling requires the turbulence models to be modified and the mesh to be sufficiently course to analyse the flow all the way to the wall, including the viscous sub-layer. This method requires a far greater mesh resolution, which often results in it not being feasible, particularly for high *Re* flows.

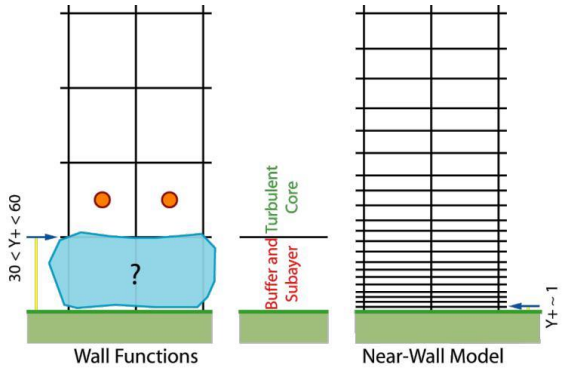


Figure 12. A representation of the computational mesh for the two near-wall modelling approaches (Fluent Manual)

This figure 10 highlights the difference in mesh size between the two approaches. Both approaches have to satisfy constraints on the y+ value at the first cell in the mesh. For a wall functions approach y^+ should ideally be between 30 and 60, and never below 11.225. On the other hand a y^+ of less around 1 is necessary when using the near-wall model. Therefore if meshes are generated correctly the distance from the first node in the mesh to the wall for the near-wall model will be around 1/30th that of the wall function model. This demonstrates the massively different computational requirements of the techniques.

3.8.1 Using Wall Treatment Models

This section explains key concepts that are related to wall treatment, including wall functions, the Wall Distance solvers, and wall laws.

A wall treatment in STAR-CCM+ is the set of near-wall modelling assumptions for each turbulence model.

STAR-CCM+ provides three types of wall treatment:

- The high-y^+ wall treatment assumes that the near-wall cell lies within the logarithmic region of the boundary layer and uses a wall function. A wall function is the set of mathematical relations that are used to obtain the boundary conditions for the continuum equations. These calculations are based on the following assumptions:

  - The turbulence model is valid only outside the viscous-dominated region of the boundary layer, and the viscous-affected region of the boundary layer is not resolved; and

  - The centroid of the near-wall cell lies within the logarithmic region of the boundary layer.

  - Velocity, turbulence, and other scalar quantities are distributed consistently.

  The main advantage of using wall functions is the significant savings in terms of near-wall mesh resolution.

- The low- y^+ wall treatment is suitable only for low-Reynolds number turbulence models: this treatment assumes that the viscous sublayer is properly resolved.

  Low-Reynolds number turbulence models are valid throughout the boundary layer, including the viscous sublayer. To resolve the viscous sublayer, these models require a sufficiently fine mesh. There is little or no need for modelling to obtain the wall boundary conditions. The computational expense that is associated with this approach can be significant, particularly for large Reynolds numbers.

- The all- y^+ wall treatment is a hybrid treatment that attempts to emulate the high- wall treatment for coarse meshes, and the low- wall treatment for fine meshes. It is also

formulated with the desirable characteristic of producing reasonable answers for meshes of intermediate resolution (that is, when the wall-cell centroid falls within the buffer region of the boundary layer).

The wall treatments have been specialized according to each turbulence model, since assumptions specific to that model must be made for the wall boundary conditions for the turbulence quantities. Both the high- y^+ and all- y^+ wall treatments share a common need to specify profiles of the mean flow quantities in the near-wall region of turbulent boundary layers. These profiles are termed wall laws.

3.8.2 Wall laws

A wall law is a mathematical description of mean flow quantities, such as velocity, temperature and species concentration, in turbulent boundary layers. These formulations include non-dimensional quantities, asymptotic limits, and standard and blended wall laws. The most basic wall laws apply only to equilibrium situations, although there have been some attempts to extend wall laws to include non-equilibrium effects.

Two types of wall laws are used in STAR-CCM+:

- standard wall laws, which are slope-discontinuous between the laminar and turbulent profiles;
- blended wall laws, which include a buffer region that smoothly blends the laminar and turbulent profiles together.

You cannot select wall laws; they are chosen automatically based on the behavior of the turbulence model.

Turbulence models that contain damping functions (or behave as if they do) typically use blended wall laws in their all- y^+ wall treatment. The standard wall law is used with high- y^+ wall treatments.

The wall laws differ only in their treatment in the buffer region; the viscous-sublayer and log-layer behaviours are identical.

Temperature equations are not showed in the following formulation.

In the viscous sublayer velocity distribution is modelled as:

$$u^+_{lam} = y^+$$

In the log-layer velocity distribution is modelled as:

$$u^+_{trb} = \frac{1}{k}\ln(E' y^+)$$

Where

$$E' = \frac{E}{f}$$

in addition, the default values of the coefficients are the Von Karman constant k= 0.42 and the empirical constant E = 9.0. The value of the f is unity for smooth walls.

Standard Wall Laws

The standard wall laws have slope discontinuities between the laminar and logarithmic regions.

They are given by:

$$u^+ = \begin{cases} u^+_{lamb} & for\ y^+ \le y^+_m \\ u^+_{trb} & for\ y^+ > y^+_m \end{cases}$$

For the momentum wall law, a Newton iteration determines the intersection of the viscous and fully turbulent regions y_m^+.

The advantage of the standard wall functions comes in its robust nature when applied to a broad range of applications. However, when the flow becomes affected by large pressure gradients and is in strong non-equilibrium, the accuracy of the solutions may decrease. If this is the case then alternative solutions may be sought.

blended wall laws

The blended wall laws are intended to represent the buffer layer by appropriately blending the viscous sublayer and logarithmic regions.

For momentum, *Reichardt's law (1951)* is used:

$$u^+ = \frac{1}{k}\ln(1+ky^+) + C\left[1 - \exp\left(-\frac{y^+}{D}\right) - \frac{y^+}{D}\exp(-by^+)\right]$$

Where

$D = y_m^+$

$C = \frac{1}{k}\ln\left(\frac{E'}{k}\right)$

$b = \frac{1}{2}\left(\frac{D_k}{C} + \frac{1}{D}\right)$

Reichardt's law is also used to compute the near-wall reference velocity u* if the appropriate option is selected;

The Wall laws used varies slightly between the model used, in this case we refer to the STAR-CCM+ manual.

3.9 Boundary Conditions

An important step in the definition of a CFD simulation is an assignment of boundary conditions. The boundary conditions that have been used throughout this research will now be introduced.

3.9.1 Inlet

It is at the inlet that the characteristics of the fluid behavior will be defined. In this case the boundary condition is *velocity inlet* which requires an input of velocity and turbulence specification. The velocity direction is consistently perpendicular to the inlet. The specification of turbulence for the simulation should be as accurate and consistent as possible, especially when comparing the results to experimental results. Turbulence can be specified in terms of k and ε or ω. However, the alternative method of using the software to determine k and ε or ω from definitions of turbulence intensity, or in our case using the Turbulence viscosity ratio

3.9.2 Outlet condition

For the outlet there is a choice of two conditions for a problem such as this. Firstly *outflow* requires no specification of pressure or velocity, but does require that the boundary should be at a sufficient distance downstream so that the flow is close to fully developed. The second choice is *pressure-outlet* which requires a specification of static pressure and will result in better convergence characteristics if the fully developed condition is not satisfied.

3.9.3 Wall

At the surfaces of the geometry a *wall* boundary condition will be applied. As the flow is viscous a zero slip condition will be satisfied, therefore at the wall the velocity will be zero. It is possible to impose a translational or rotational velocity on a wall boundary. For example, the road under a bicycle would have an imposed velocity equal to the velocity of the bicycle, in our case; we impose the velocity of the road and of the outer wheel, using a different approach for the wheel inner part.

3.9.4 Symmetry

The outer wall of the flow domain connecting the inlet to the outlet that is not a wall can be modelled as *symmetry*. This condition assumes that the fluid behavior on the meshed side of the boundary is mirrored on the adjacent side. The application of symmetry to the outer wall of the domain can be used if the flow is uniform and the boundary is a sufficient distance away from the examined geometry, otherwise we should use a wall surface.

3.10 Numerical method

In the previous section the mathematical model behind the CFD solution was introduced. This section explains how the mathematical model is then manipulated to provide a solution.

3.10.1 The finite volume method

The solver provides a method of computing the governing equations to give the solution. This solver, like the majority of other commercial CFD solvers, uses the finite volume method. The finite volume method divides into the following stages;

1. Generation of non-overlapping control volumes in the flow domain, i.e. the computational mesh.
2. Integration of the governing equations over all the volumes in the flow domain.
3. Approximations representing flow processes (diffusion, convection, etc.) are substituted into the integrated equation. This provides an algebraic equation for each control volume, which is only valid over that control volume.
4. An iterative method is used to solve the algebraic equations.

The majority of finite volume methods are node centered techniques, To maintain consistency between notations with other sources of CFD literature the letter P describes the cell center where the information is stored, uppercase W and E represent the upwind and downwind cell centers (west and east) and the lowercase w and e are the upwind and downwind faces of the control volume.

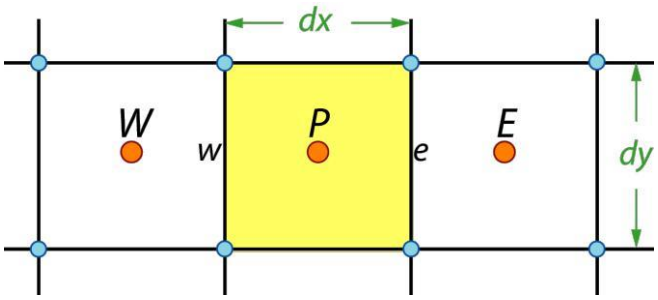


Figure 13. A basic two-dimensional mesh used to demonstrate the finite volume technique (Fluent manual)

To understand the following explanation of the discretization procedure it is necessary to introduce the steady state diffusion of a scalar quantity, ϕ,

$$\frac{d}{dx}\left(\Gamma_\phi \frac{d\phi}{dx}\right) + S_\phi = 0$$

Where: Γ_ϕ is the diffusion coefficient of ϕ, S_ϕ is the source term of ϕ. Once the mesh has been generated, the first stage of the solution procedure is the integration of the governing equations over the control volumes. The integration of Equation over the control volume shown in Figure 13 gives the following,

$$\int_{\Delta V} \frac{d}{dx}\left(\Gamma_\phi \frac{d\phi}{dx}\right) dV + \int_{\Delta V} S_\phi = \left(\Gamma_\phi A_{cv} \frac{d\phi}{dx}\right)_e - \left(\Gamma_\phi A_{cv} \frac{d\phi}{dx}\right)_w + \overline{S_\phi}\Delta V = 0$$

Where, A_{cv} is the cross sectional area of the control volume face, ΔV is the volume and $\overline{S_\phi}$ is the average of S_ϕ over the control volume. The equation satisfies a balance over the control volume in that it states that the diffusive flux of ϕ leaving e minus the diffusive flux of ϕ entering w is equal to the generation of ϕ. To make use of this equation, flow properties must be calculated at the cell faces as well as their centres. To do this, a discretization procedure interpolates the values stored at the cell centers to the cell faces. There are a number of discretization schemes available, which will now be discussed.

3.10.2 Discretization

The discretization schemes follow an up winding scheme procedure. Unwinding schemes derive ϕ at the face from the conditions in the upstream cell, hence 'up winding'. There are various up winding schemes available, the essence of which is captured in Figure 14, followed by a discussion below.

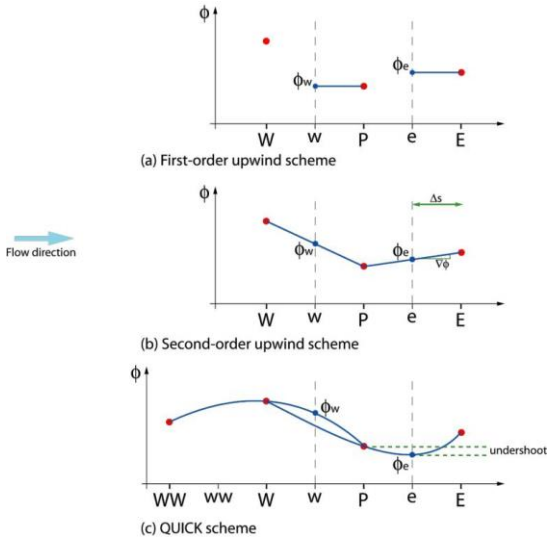


Figure 14. A schematic of how the discretization schemes calculate ϕ at cell faces (ϕ_w and ϕ_e) using: (a) first-order upwind, (b) second-order upwind and QUICK scheme (c)

First-order upwind scheme

The first-order scheme assumes that the value of ϕ at the cell center represents the value on the upwind faces. This approach is valid for relatively simple flows aligned with the mesh. However, when it is not aligned with the mesh or large gradients develop the numerical discretization error, or numerical diffusion, increases.

Power-law scheme

The power-law discretization scheme (Patankar, 1980) reverts to the first-order scheme, a linear scheme or uses the integral of a one-dimensional convection-diffusion equation. It makes the decision of which scheme to choose based on a calculated value of Peclet number. The power-law scheme generally provides the same accuracy as the first-order upwind scheme.

Second-order upwind scheme

Second-order accuracy is achieved at cell faces through a Taylor series expansion of ϕ, about the cell-centroid,

$$\phi_e = \phi + \nabla\phi\Delta s$$

where, Δs is the displacement vector from the upstream cell centroid to the face centroid. The gradient in each cell, $\nabla\phi$, needs determining for each cell. This is achieved by using an additional equation, the divergence theorem. The use of the second-order upwind scheme reduces the likelihood of numerical diffusion, especially for unstructured meshes and non-uniform flows. There are two downsides to this scheme. First, an increased computational effort is required. Secondly, it is often subject to instability when used at the beginning of simulation, however this complication can be avoided by running it after the initial iterations have been performed using the first-order scheme.

QUICK scheme

The QUICK scheme provides a solution for unique shaped quadrilateral or hexahedral cells using higher-order calculations. This scheme is a variation of the second-order upwind scheme, which is used even when the QUICK scheme is selected on unstructured cells in hybrid or partitioned meshes. The scheme uses 3 cell-center ϕ values to determine ϕ at the face (Figure 14). The scheme can be subject to 'undershoot' or 'overshoot' whereby ϕ is under- or over-predicted. The highest accuracy is achieved with the QUICK scheme when a structured mesh is used with a correctly aligned flow. This scheme is known to be advantageous for rotating or swirling flows.

Numerical diffusion

Misleading solutions can be obtained if a large amount of 'false' or 'numerical' diffusion occurs. A finite amount of numerical diffusion is necessary for each of the discretization schemes, however too much is undesirable and should be avoided. There are ways to limit the likelihood of numerical diffusion occurring. Using the second-order discretization scheme will decrease the possibility of too much numerical diffusion occurring. The chance of numerical diffusion occurring is largely dictated by the mesh size and alignment. A fine mesh will be less likely to experience numerical diffusion than a coarse mesh. In addition, if the cells in the mesh are aligned with the flow then there is less chance of numerical diffusion occurring. For these reasons a structured mesh is highly preferable to an unstructured mesh, as it is possible to achieve a mesh aligned to the mean flow with fewer cells.

3.11 The Solver

Following the discretization of the integrated governing equations, a solver will carry out the final processes to obtain a solution. There are two types of solver available in Star-CCM+ segregated and coupled.

3.11.1 Segregated solver

When the segregated solver is used the equations are solved sequentially, shown schematically in Figure 15. As the equations are coupled and non-linear it is necessary to perform several iterations.

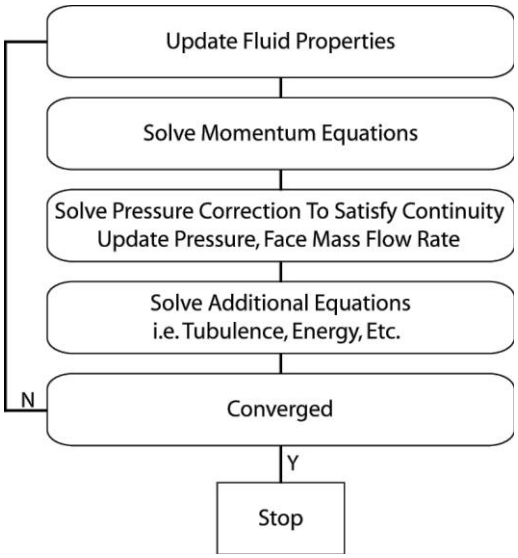


Figure 15. The segregated solver

With the segregated solver the governing equations are linearized implicitly. The linearization is a system of linear equations, with one equation for each cell in the domain. The resulting system of linear equations is solved for the dependent variable in each cell. The segregated approach therefore solves for a single variable field by considering all the cells at the same time, then continues to solve another variable.

3.11.2 Pressure-velocity coupling

It is necessary for the pressure to be coupled to velocity, which is achieved by deriving an equation for pressure from the discrete continuity equation. Fluent™ has the following algorithms to carry out this task.

- SIMPLE uses a relationship between velocity and pressure corrections to enforce mass conservation and to obtain the pressure field.
- SIMPLEC is a modification on the previous algorithm and has been shown to offer accurate convergence when pressure-velocity coupling is the main deterrent to a solution (Vandoormaal and Raithby, 1984).
- PISO is again part of the same group of algorithms, only this is based on the higher degree of the approximate relation between the corrections for pressure and velocity. The PISO algorithm takes a little more CPU time per iteration, but at the same time it can decrease the number of iterations required for convergence.

In general SIMPLE and SIMPLEC are the recommended solvers for steady-state simulation, and PISO for transient. PISO may be advantageous for all simulations with highly skewed meshes. SIMPLEC offers convergence rate advantages over SIMPLE for simple flows such as laminar.

3.11.3 Coupled solver

The coupled solver solves the equations for momentum, continuity and, when necessary, energy and species transport simultaneously, hence 'coupled'. Any additional scalar equations are solved sequentially, as with the segregated approach. Again, it is often necessary to perform many iterations.

With the coupled solver the user has the choice of selecting either implicit or explicit analysis for the linearization of the coupled set of equations. The equations solved sequentially are linearized implicitly. The coupled implicit approach solves for all variables in all cells at the same time. This compares to the coupled explicit approach, which solves for all variables cell by cell.

In this analysis the coupled solver will not be used because of the incompressible flow assumption, justified by the low Mach numbers in the simulations.

3.12 Convergence

The solver uses an iterative method to provide a solution, and as a result it is necessary to monitor this process and define a point when the solution is converged. An error in the judgement of convergence can lead to inaccurate and misleading results and therefore this stage is as important as any of the preceding ones.

For every governing equation there is an imbalance, otherwise known as a residual. If the iterative process proceeds as it should this residual level will reduce over a given number of iterations. When the residuals have fallen below a certain value the solution can be deemed converged. However, the often confusing nature of this convergence procedure is that there are no strict definitions about what levels of residuals dictate a converged solution. The default stopping convergence criterion in Star-CCM+ is 10^{-3}, the code show the residuals as a ratio between the initial residual and the current one, it is the experience of the author that this value is often too high and a more realistic criterion will be 10^{-4} or 10^{-5}, depending on the model and on the Force and Velocity diagram.

It is good practice also to examine a flow variable monitor at point of interest within the flow domain. For example, a monitor of velocity magnitude in a region downstream of the wheel, where the flow may be detached and recirculating, will provide the velocity at that point for every iteration. If this monitor is stable for a significant number of iterations then the indication is that the solution is converged.

Another monitor that helps ascertain whether a solution is converged is a force monitor. As with a variable monitor, if a force monitor reports that the force imposed on a body is no longer changing with further iterations then the suggestion is that the solution is converged.

Chapter 4

Development of simulation methods

4.1 Introduction

Utilizing previous studies as a base, we developed an initial model. This model was tested with a small fluid domain, this was based on a real wind tunnel that we were supposed to use, to compare the results of the Wind tunnel with the CFD data.

The initial model was based on a disc wheel inspired to the Zipp Sub9 (from *Godo 2009*) we had the profile of the rim but not the complete geometry)

We initially compared the small domain results with experimental results taken from literature, but the small domain showed some limitations.

To overcome this problems, and because the small wind-tunnel wasn't available, we developed another method with a larger domain, using the symmetry condition for the side and top walls, a tyre-floor contact and a floor local velocity compatible with the bicycle/wheel speed.

The model was based on a larger tunnel that was used later in the work for the study of some models.

4.2 Early modelling

4.2.1 Model geometric definition

The code used was Star-CCM+ 9.02.007. Initially, the numerical modeling was oriented to reproducing the results presented in a literature study by *Godo et Al. (2009)* reported earlier due to the availability of the profiles of different racing wheels:

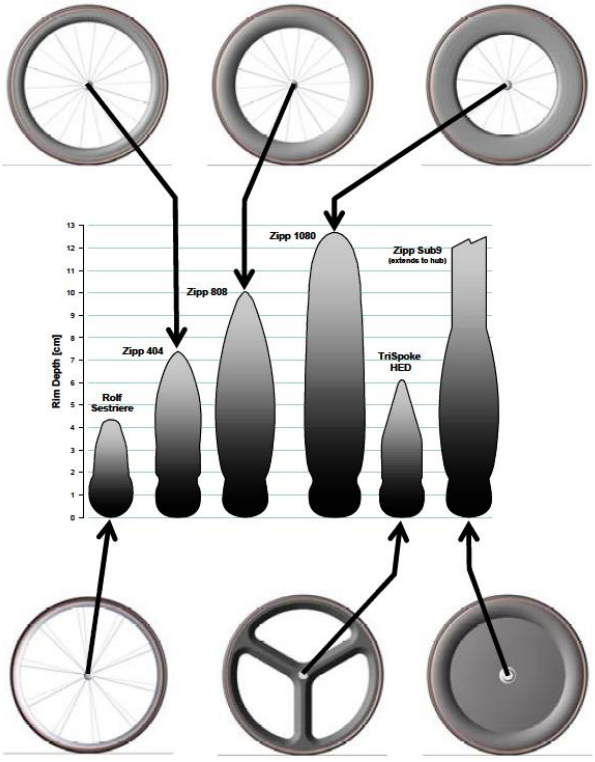


Figure 16. Wheel profiles from Godo (2009)

Using the image as guide and the profile of the disc wheel Zipp Sub9 we created a model of the wheel, using the scale on the left as a guide.

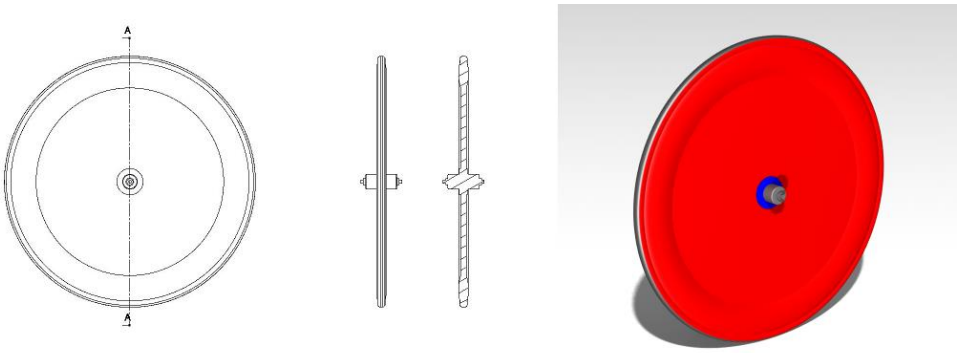


Figure 17. Wheel profile reconstructed using Dassault Systemes CATIA Software

We choose this wheel due its lack of geometrical features that can increase the cell number, then the simulation complexity, in the first part of this study.

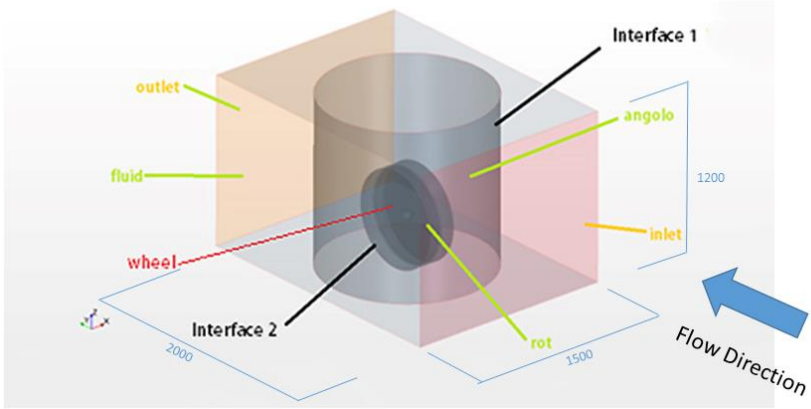


Figure 18. Domain and region definition

To obtain a steady-state analysis, we applied the MRF (Multiple Reference Frames) method in which a different frame can rotate (and/or translate) with respect to the laboratory reference frame. By imposing a rotating frame to the region containing the wheel we took into account the effect of the rotation on the fluid around it, without providing a rigid movement to the wheel and an unsteady simulation needing more computational resources.

We also introduced a second cylindrical region to change the angle of the wheel without the need of re-meshing at each incidence angle simulation. Two interfaces were therefore created between the rotating region, the incidence region and the overall fluid region to guarantee the continuity.

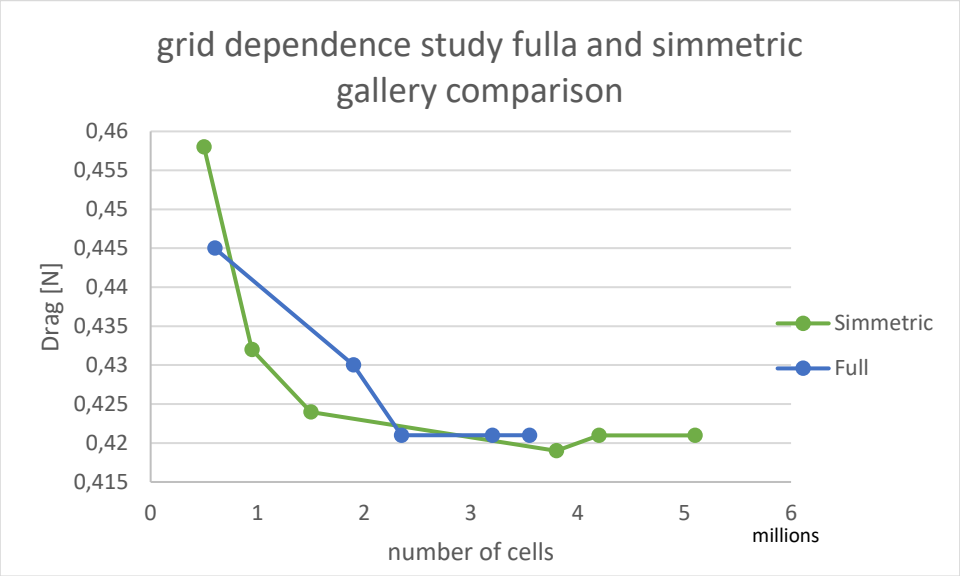
The fluid domain was based on the characteristics of the wind tunnel that we were planning to use to test experimentally the wheels: the wind tunnel test section is 2m long, 1,5m wide and 1,2m high.

The model was build using the wheel only, initially without the fork or any support: the mesh was defined using a prism layer mesher near the domain walls and a polyhedral mesher for the fluid domain with a prism layer near the wall region. The mesh was refined by imposing a minimum and mean surface size in the wheel and in both the interfaces, and then by setting a low surface growth rate and increasing the density of the mesh in the two inner regions. The simulations ran on steady

state model using a constant density gas (because of the low speed) with a standard K-ε turbulence model with two-layer *y+* treatment. The wind speed was 8,94 m/s matched by the wheel rotating speed.

4.1.2 Method Testing

After defining the geometry, the field mesh was created: after a number of preliminary tests, comprehensive of a grid sensitivity study, showed in graph 1, we chose the geometry dimensions.



Graph 2. Grid sensitivity study

The sensitivity of CFD results from grid size showed in graph 1 was also obtained by means of calculations performed on a halved domain that uses the symmetry condition to halve the number of the cells, comparing the results with the full domain simulation. The wheel used in this test was a disc wheel.

The computational grid was slightly modified to perform the simulations of the spoke wheels. In fact, due to computational resource limits, it was not possible to use the same setting used for the disc wheels. Therefore, the final grid counts about 4.2 million elements (instead of 16 millions). The sensitivity of the result from the grid size was checked by comparing these results with those obtained by using the symmetrical mesh presented before (the actual spokes disposal is not perfectly symmetrical but the influence is negligible) and showed that results are comparable (with differences of about 0,01 N in drag).

The prism layer mesh has been fixed to 15 prism cells with 1,2 growing rate. These parameters allow for proper values of the y+, keeping it around 1 on the wheel surfaces, as show in the example on fig.17.

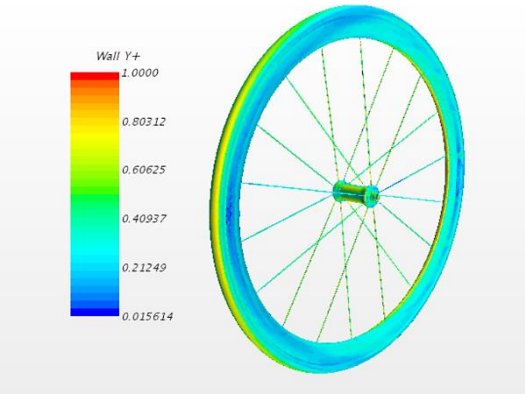


Figure 20. y+ value for a spoked wheel

The most used turbulence models in literature were the Spalart-Allmaras model, the k-ω and the k-ε (standard and realizable version): *Lukes (2006)* used the standard k-ε and compared these models obtaining similar results, after a model comparison on a basic disc wheel we decide to use the Standard k-ε model initially.

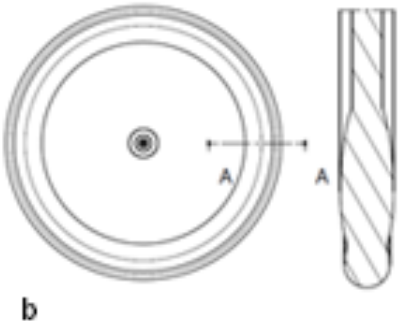
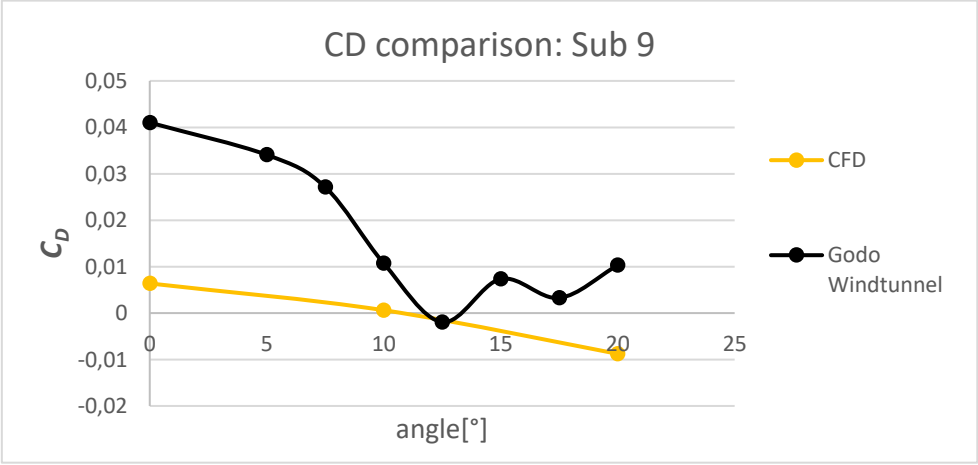


Figure 19. Wheel design used for the simulation, based on Zipp Sub9 design taken from the literature

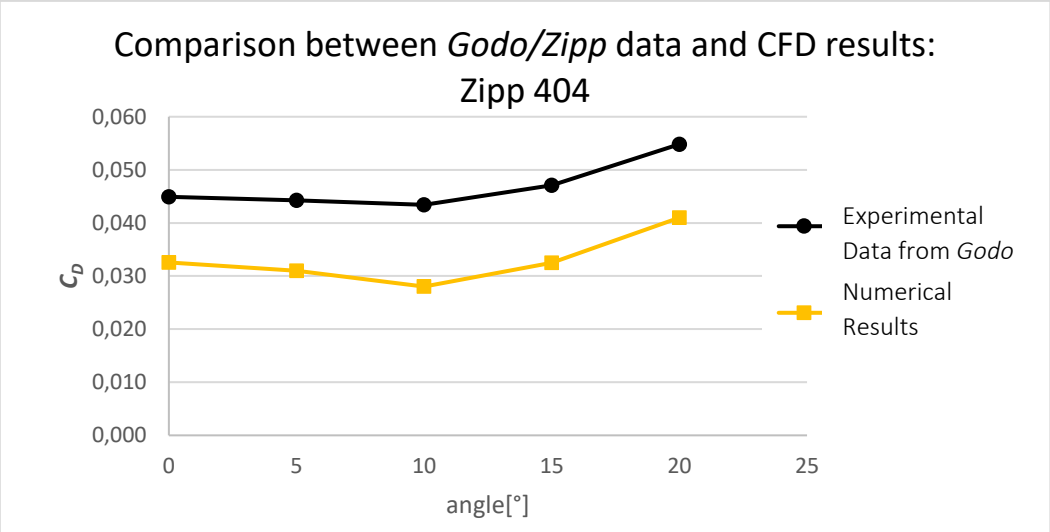


Graph 3. Small Fluid Domain Comparison, standard k-ε

It can been see that the forces are under predicted by a large margin, partially because we initially underestimate the wheel width at the hub, making our model more close to a plane wheel with a

profiled part of the rim than to the original Sub9 which we discover later, had an inclined side. There is a small error caused by the use of the more pratical but less precise MRF steady model. The unknown wind tunnel setup, also there is no contact between the wheel and the floor.

Being the results of the disc wheel non-satisfactory enough we modelled another whell from the same paper, the Zipp 404, being a spoked wheel, the difference between the tested designs were less influent. We partially based the hub of the wheel on an early Campagnolo Bora design and compared the wind tunnel results with the standard k- ε model.



Graph 4. Comparison between the calculated C_D and the experimental C_D of the S2 wheel

As before the drag coefficient is calculated using the diameter as the reference area.

It can be seen in graph 4 that the predicted C_D are still lower than the one showed by the real data, but the general trend is capture. The drag differences may be due in part to differences between the boundary conditions applied for the CFD simulations and those present in the Zipp wind tunnel, considering the unknown size and wheel support used in this case.

This test however show that the standard k- ε seems to capture the wheel behaviour on a good way

4.2 Model 2

4.2.1 Model modification

The model built for the previous testing could give a good representation of the wheel behavior but had severe limitation, the wheel doesn't touch the ground because it is contained entirely in the MRF volume, the fluid domain is small. It was common to have reverse fluid velocity at the outlet mining the model convergence.

So a new fluid model, named Model 2, was build based on the one showed before, the domain principal dimensions can be seen in figure 21.

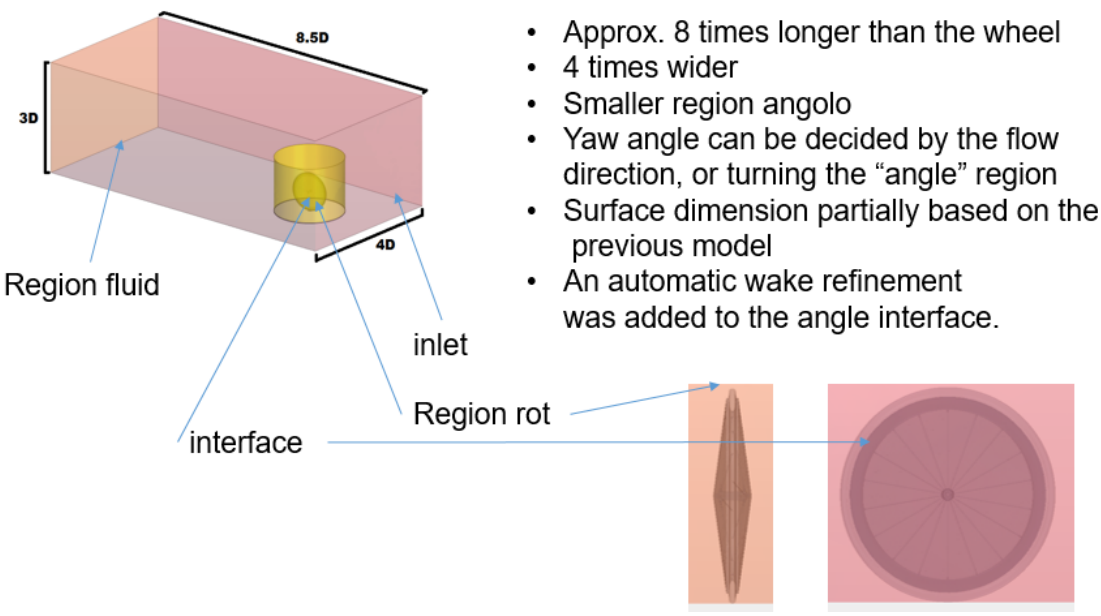


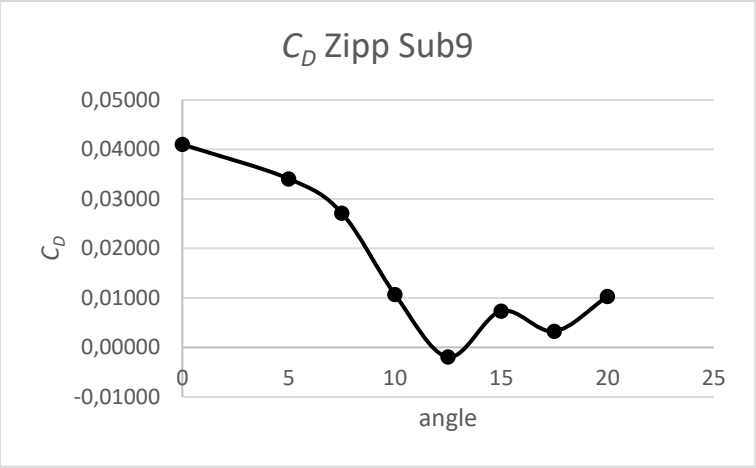
Figure 21. Modification to the original model

The MRF region changed in shape containing only the inner part of the wheel, the spokes and the hub; on the outer part of the wheel (containing the tire), a simple wall local rotation approach is used. The wheel itself penetrate the ground for 2mm; the tire deformation was not taken in account, all the surfaces were considered smooth surface beside the tire that had a fictional wall roughness applied. The Central cylindrical region was maintain to perform a better refinement; the mesh refinement is obtained via a control volume in the two inner region and a wake refinement to the angle central interface. It could be useful to use a refinement based on a field function but considering that the angle and the wheel will change during the simulation this seems to be the better approach; between the regions, an imprint operation was performed. The Imprint operation imprints close surfaces within a user-defined tolerance. Part contacts are created between the surfaces that are coincident and you can choose to create a conformal or non-conformal surface mesh between parts. We choose a conformal mesh between the interfaces, obliviously the angle region interface is conformal only at 0°.

The mesh was refined on the wheel surfaces, fixing the target and the minimum surface using a surface control; the mesh was also refined on the bottom part of the fluid domain; the surface grown limit was set at 1.2 and the volume grown at 1.1 to obtain a smooth mesh grown from the boundary.

The grid sensitivity was again tested with a spoke wheel (25mm tire; 50mm wheel rim depth) the grid size was progressively reduced.

With the wind tunnel data reported in the graph below (*Godo et al)* we made a comparison using the disc wheel showed before to speed up the processing.

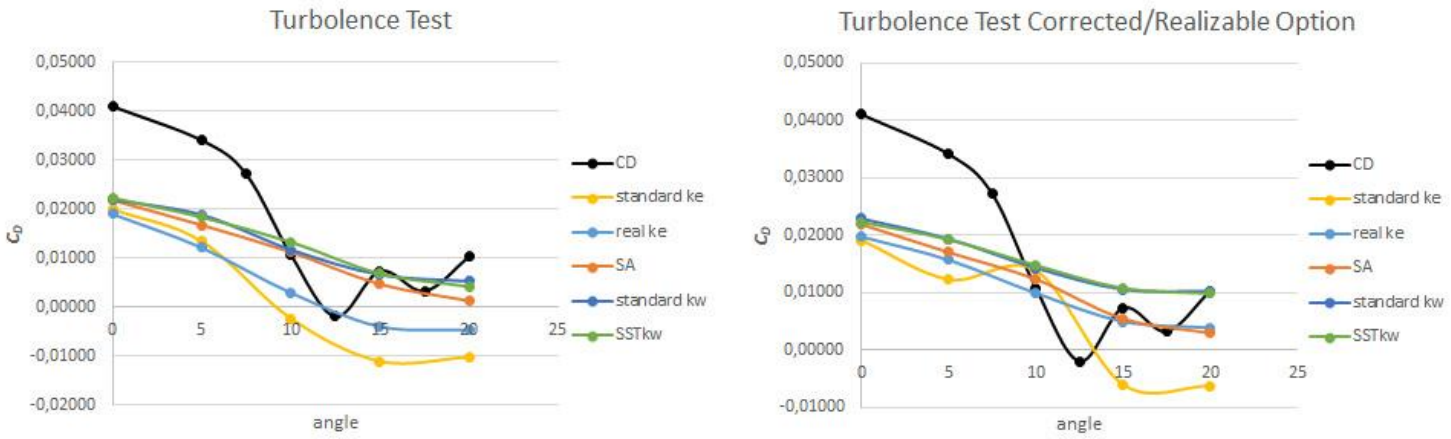


Graph 5. Zipp sub9 drag coefficient, experimental data from literature

The drag forces are reported by mean of the C_D coefficient based on the wheel diameter, including the tire, at the speed of 30 mph, this test cannot be really accurate because we don't know exactly the wheel geometry, or the support used.

The simulation were carried out using the standard k-ε, the realizable k-ε, the k-ω SST model and their correction model avaivable of Star-CCM+.

The results can be found in the two graph below.



Graph 6. CFD early results compared to the given data

It can been see that the forces are under predicted, partially caused by the same reason stated before. However if we adjust the forces with a constant offset value it seems that the Standard k-ε model came closer to the wind tunnel data reported by the paper, but it shows a strange behaviour after 10°, with a negative drag force, caused probably by the large side force measured, being the drag force in the same direction of the rider with a perpendicular side force.

The realizable k-ε models and the SA model also provided stable and consistent solutions, whereas the k-ω SST seems to be unsuitable to a steady-state solution, being less stable and often providing higher residual values.

4.2.2 Surface Roughness

We decide to introduce a mean surface roughness on the model boundary. The wheels, ground plane and tunnel sides, top and bottom were set as walls with surface roughness and roughness constants as shown in Table below

Table 5. Surface roughness

Component	*Surface Roughness (m)*	*Roughness constant*
Wheels	$2x10^{-6}$	0.5
Tires	0.0001	0.5
Ground plane	$2x10^{-6}$	0.5
Tunnel sides and top	$8x10^{-6}$	0.5

4.2.3 Spoke effects

We decided to do a test on the spokes. Modelling itself is very numerical expensive, and can increase the total number of elements by more than 50%. We want to check the spokes influence on the model. Therefore we made a comparison between no spokes, aero spokes and round spokes (2mm) as shown in graph 6.

The “Aero” spokes shape are shown in Figure 22.

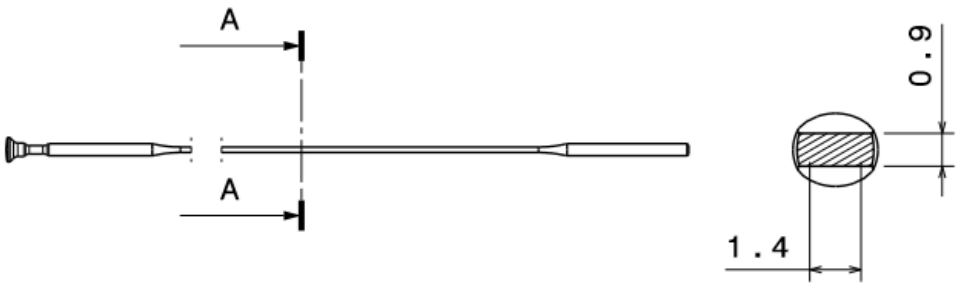


Figure 22. Aero Spokes

Graph 7. Comparison between different spokes shape and no spokes normalized on the “aero” spoke results

4.2.4 Test methods

Our tests were conducted at different wheel yaw angles, typically 0°,5°,10°,15° and 20°, higher relative angles won't be useful since they will be unusual in normal road condition, *EASTON Cycling (2010).*

The drag is considered in the same direction of the wheel with the side forces in a normal direction as shown in figure 23.

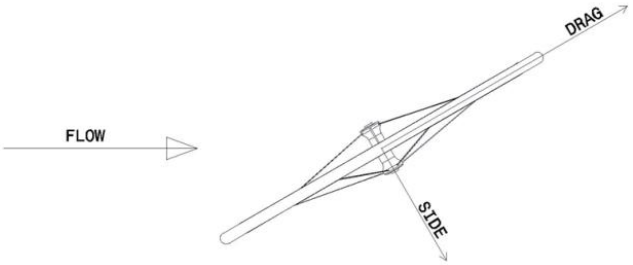


Figure 23. Forces direction

A uniform velocity profile (13m/s) was applied to the upstream boundary of the computational domain. This velocity was chosen to match the forward speed of the bicycle. A constant eddy viscosity inlet condition was specified to be 0.001.

The ground plane was modelled as a no-slip surface, with a constant translational velocity matching the specified upstream speed. The ground plane velocity components were parallel to the forward motion of the bicycle.

In the next chapter we will present the results of tests regarding different wheel geometry, starting with the small domain tunnel and then the larger one, two tests regarding the wheel principal dimensions, we will introduce two performance index and an unsteady state test of a high-profile wheel.

Chapter 5

Results

5.1 Introduction

In this chapter we illustrate the aerodynamic force computed on different models with different wheels. We firstly present the results of the Small Early fluid domain based on a real wind tunnel dimensions, and then of the Larger Fluid domain. The wheel tested are subdivided in 2 categories: the disc wheels and the spoked wheels.

We used the model initially on two different disc wheels named D1 and D2 showed in fig 24.

Note that the two wheels feature the same hub geometry.

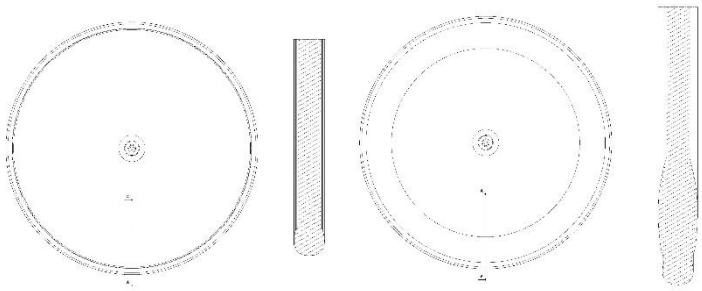


Figure 24. Disc wheels a) D1 b) D2

The D2 wheel is the wheel set up to test the model, and the profile is inspired by the Zipp Sub9 profile reported by *Godo.*

The manufacturer directly made the geometry of S1, S3, S4, S5 and S6 wheels available and the tire profile was obtained using a coordinate measuring machine. Wheels were fitted, when possible, with the same size tire, the profile of the S2 wheel was taken from literature.

The profile and the main characteristics of the wheels can be view in the following image and table.

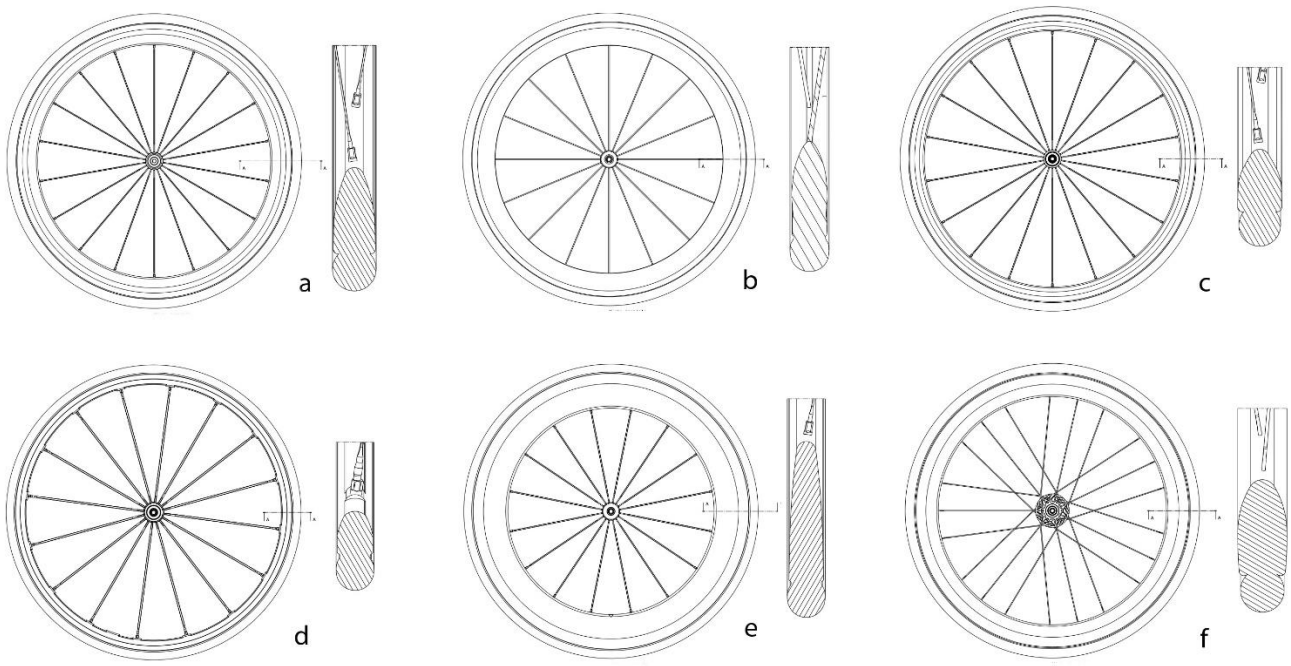


Figure 25. S1 (a), S2 (b), S3 (c), S4 (d), S5 (e) and S6 (f)

Table 6. Wheel details.

Commercial Name	*Wheel label*	*Rim Material*	*Type*	*Wheel depth with tire (R_{WT})*
Bora 50	S1	Carbon	18 Spokes	70mm
Zipp 404	S2	Carbon	18 Spokes	75mm
Bora 35	S3	Carbon	16 Spokes	50mm
Shamal	S4	Aluminium	18 Spokes	55mm
Bora 80	S5	Carbon	16 Spokes	100mm
Prototype	S6	Carbon	24 Spokes	70mm

The other spoke model, named S2, was built using the same hub and spokes and a profile taken from literature *Godo et Al. (2010)*.

The S3 wheel is low profile carbon fibre high-end wheel, the S4 is an aluminium low profile wheel, the S5 is a high profile carbon fibre wheel and the S6 is prototype carbon fibre wheel. A closed surface geometric model of each wheel was built, eliminating small details that will not interfere with the aero performance but can increase the geometric complexity of the wheel.

All wheels were fitted with flat “areo” spokes, depending on the model.

A table summarizing the simulation is provided below.

Table 7. Resume of the wheel tested and relative domain/method

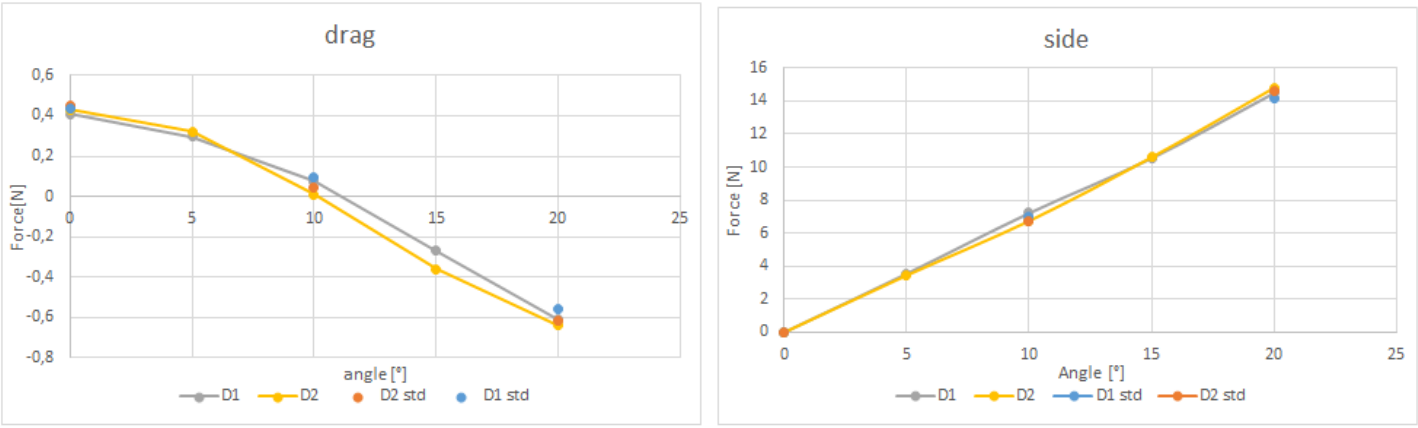
	Disc				*Spoke*			
Wheel	D1	D2	S1	S2	S3	S4	S5	S6
Small Domain	✓	✓	✓	✓	✓	✓	✓	-
Experimental Ref.	-	*Godo*	*WT Camp.*	*Godo*	*WT Camp.*	*WT Camp.*	*WT Camp.*	*WT Camp.*
Wide Domain	-	✓	✓	✓	-	-	✓	✓
Experimental Ref.	-	*Godo*	*WT Camp.*	*WT Camp.*	-	-	*WT Camp.*	*WT Camp.*

5.2 Small fluid domain

We use the model initially on two different disc wheels named D1 and D2 as showed.

Table 8. Comparison of Drag and side forces for the disc wheels, calculated by using the standard and the realizable k-ε models.

	Realizable k-ε				*Std k-ε*			
	D1		D2		D2 STD		D1 STD	
Angle	Drag	Side	Drag	Side	Drag	Side	Drag	Side
[°]	[N]	[N]	[N]	[N]	[N]	[N]	[N]	[N]
0	0,41	0,00	0,43	0,00	0,45	0,02	0,44	0,00
5	0,30	3,55	0,32	3,45	-	-	-	-
10	0,08	7,22	0,01	6,75	0,04	6,74	0,10	7,03
15	-0,27	10,54	-0,36	10,60	-	-	-	-
20	-0,61	14,49	-0,64	14,81	-0,62	14,64	-0,56	14,15



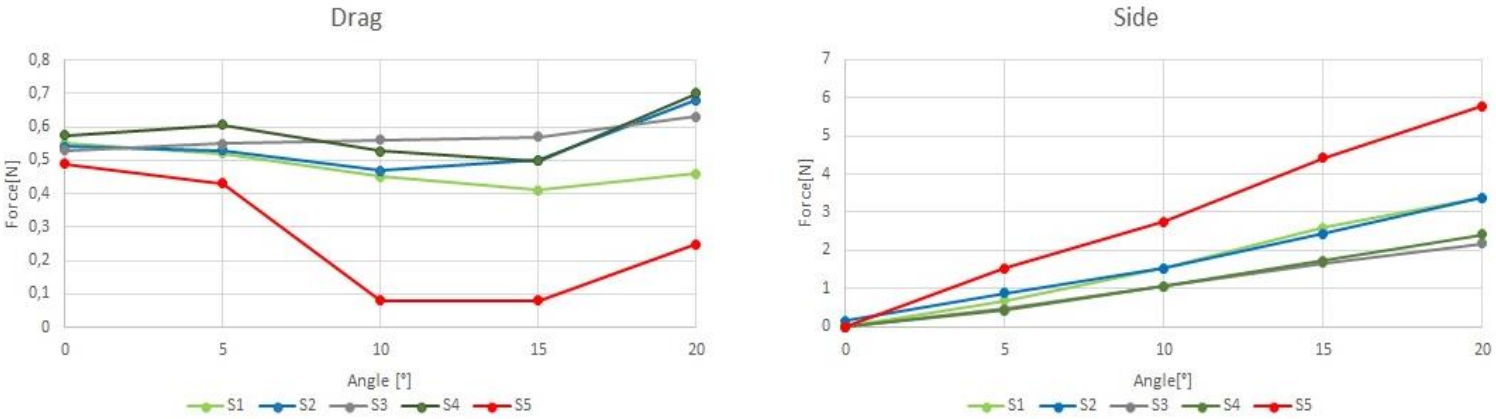
Graph 8. Drag and Side forces realizable and standard k-ε

The D2 disc drag at 0° is higher than that of D1 disk whereas both the D2 drag and side forces are lower than the corresponding parameters related to D1 disk when incidence angles reach values around 10°. It is worth noting that incidence angle different form 0° is the more common real world scenario.

We performed the full range of simulations on different spoke wheels, here is shown an example of a comparison between five wheels (named from “S1” to “S5”).

Table 9. Drag and side forces results for the spoke wheel

	S1		*S2*		*S3*		*S4*		*S5*	
Angle	Drag	Side	Drag	Side	Drag	Side	Drag	Side	Drag	Side
[°]	[N]	[N]	[N]	[N]	[N]	[N]	[N]	[N]	[N]	[N]
0	0,55	0,02	0,54	0,16	0,53	0,00	0,64	0,00	0,49	0,00
5	0,52	0,69	0,53	0,88	0,55	0,48	0,60	0,44	0,43	1,53
10	0,45	1,53	0,47	1,53	0,56	1,07	0,52	1,06	0,08	2,75
15	0,41	2,59	0,50	2,43	0,57	1,68	0,61	1,73	0,08	4,43
20	0,46	3,39	0,68	3,39	0,63	2,19	0,65	2,42	0,25	5,78



Graph 9. CFD Drag and side forces graphs for Spoke wheels

In general, a considerable variation is observed between wheels. As the depth of the rim increases, the extent of the total drag minimum is also seen to increase. This drag minimum is seen to occur for the S2, S4, and S5 at between 10° and 15° yaw, which is close to the real world conditions; In fact this range of yaw is considered to be predominantly experienced by cyclists, and is regarded as a performance target by wheel manufacturers.

The higher the wheel depth the higher will be the side forces, generally side force were observed to increase in a nearly linear fashion with respect to increasing yaw, as stated in previous works.

Considering the fact that the wheel tested are various and extremely different by the shape of their profile we can still compare the closest ones.

The S1 and the S2 models present a similar rim depth, We expected S2 wheel to have a lower drag at 0° due to the smaller frontal area; this hypothesis was confirmed by the simulation. The S1 anyway wheel present better performance for each angle tested over 10°, the side forces are similar, probably due to a similar wheel depth.

The S3 model is the closest to a typical road bike wheel’s profile, this wheel does not present the typical decay of the drag force between 10° and 15° degrees as the other wheels do, meanwhile the performance at 0° it had a similar performance of the low profile wheel and this wheel show less

variation in the drag forces at the variation of the wind angle than the other wheel tested, the side forces were the lowest found among these wheels.

The S4 wheel is a low profile “aero” wheel, it present a higher drag force than expected, especially comparing it to the S3 wheel, the S4 presents better performance only for angle greater than 10°, meanwhile the side force is comparable with the behavior found in the S3 wheel

the high profile S5 wheel presents a high decay at 10° too, it shows better drag performance for every angle tested, it also present a high side forces (that is less than an half of the force experienced by the full disc wheel).

5.3 Wide fluid domain – Model 2

With the limit of the model explained before, we run a different set of simulation using the second model and comparing the results with the experimental data obtained by Campagnolo at **GST Windkanal** wind tunnel in Immestaad.

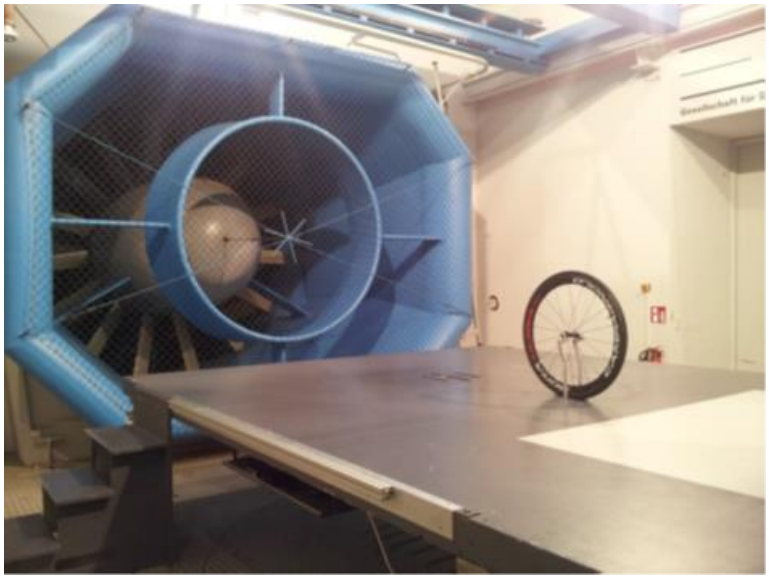
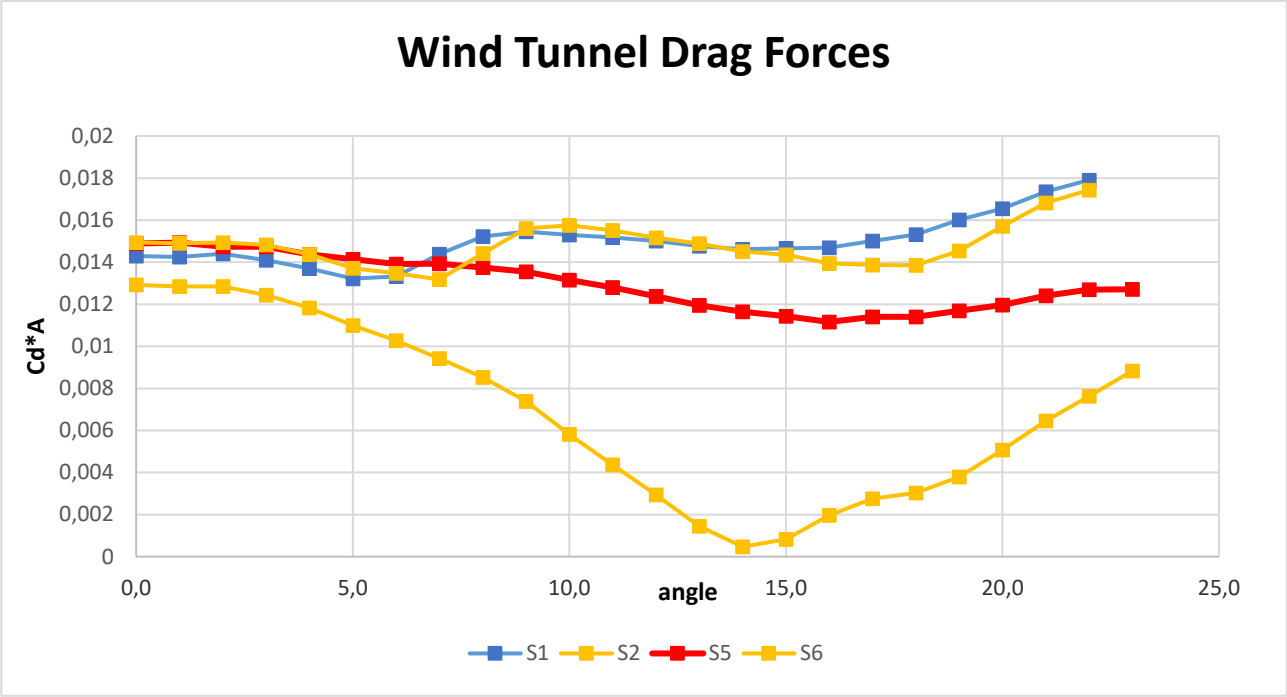


Figure 26. GST Windkanal wind tunnel

The turbulence model was revised, since the standard model seems to under predict the forces at higher angle.



Graph 10. Drag experimental measurements for S1, S2, S5 and S6 wheel

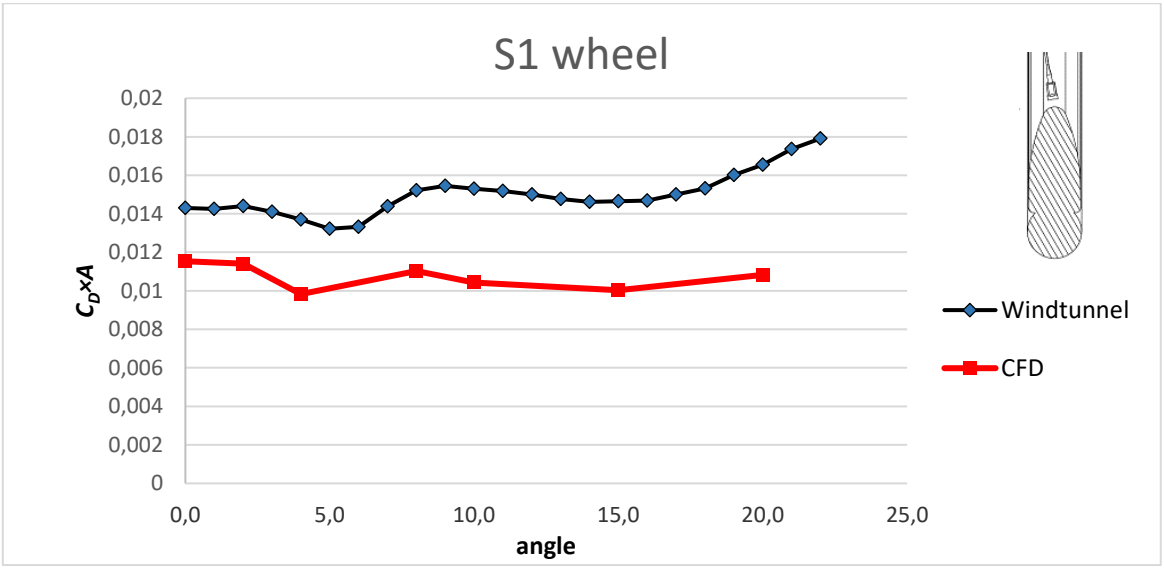
Results are showed in $C_D \times A$ normalized drag.

$$C_D \times A = \frac{D}{\frac{1}{2}\rho V^2}$$

5.3.1 Simulation Results

Wheel S1

As in the previous cases the model usually underestimates the overall forces;

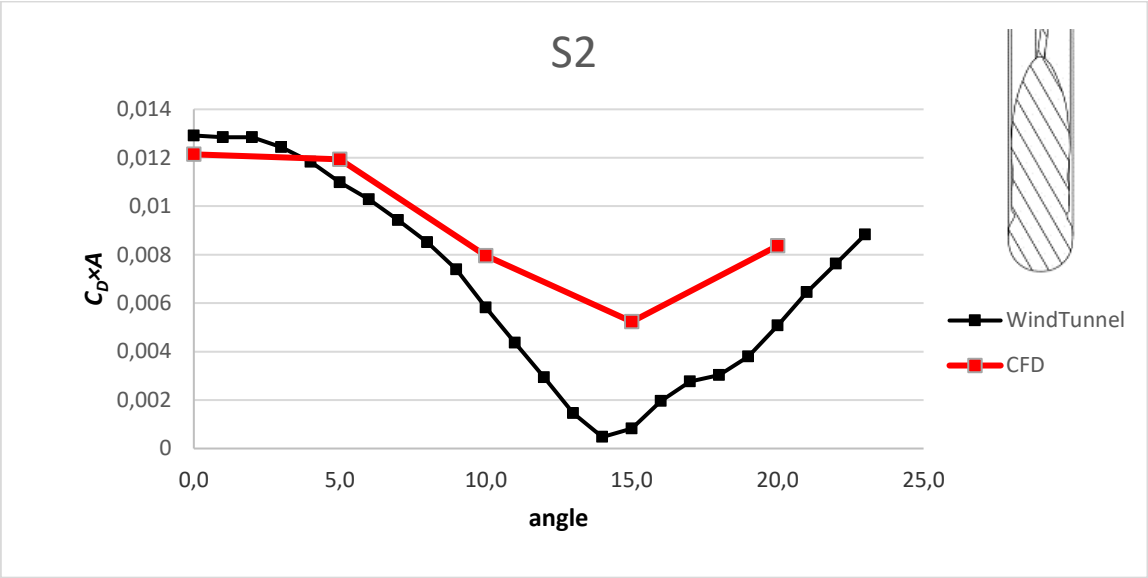


Graph 11. . Comparison between CFD results and S1 wheel (Bora 50)

The model of the S1 wheel, even if it predicts lower forces, was able to capture the general behavior of the wheel even if the drag at higher angle was underestimated.

Generally flow detachment seems to happen a little earlier on the model, anyway the wheel behavior (when uninfluenced by other factors like tire pattern) seems to be captured by the model, making it a good instrument to estimate the wheel performances.

Wheel S2



Graph 12. Comparison between cfd results and S2 wheel (Zipp 404)

The model over predict the drag for the S2 wheel; this may be caused by the factor that our S1 wheel is based on an older design, model was based on *Godo*'s work therefore the wheel is a m.y. 2009 or earlier, meanwhile the wind-tunnel data's refer to a m.y. 2015, in fact the behavior of our model is more close to the one given by the article.

The new design may be different and it had a special wheel surface treatment, having a dimpled surface, as showed in the figure below.

Figure 27. Zipp 404 wheel surface from zipp.com

Preliminary numerical testing on a rim, without the spokes or the hub (showed below) with similar dimpled surface actually shown a slightly higher drag, since the appreciable difference we suspect that this was caused by the grid.

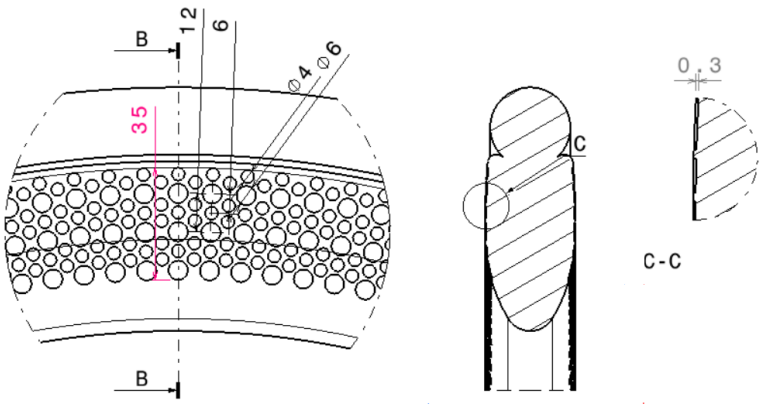
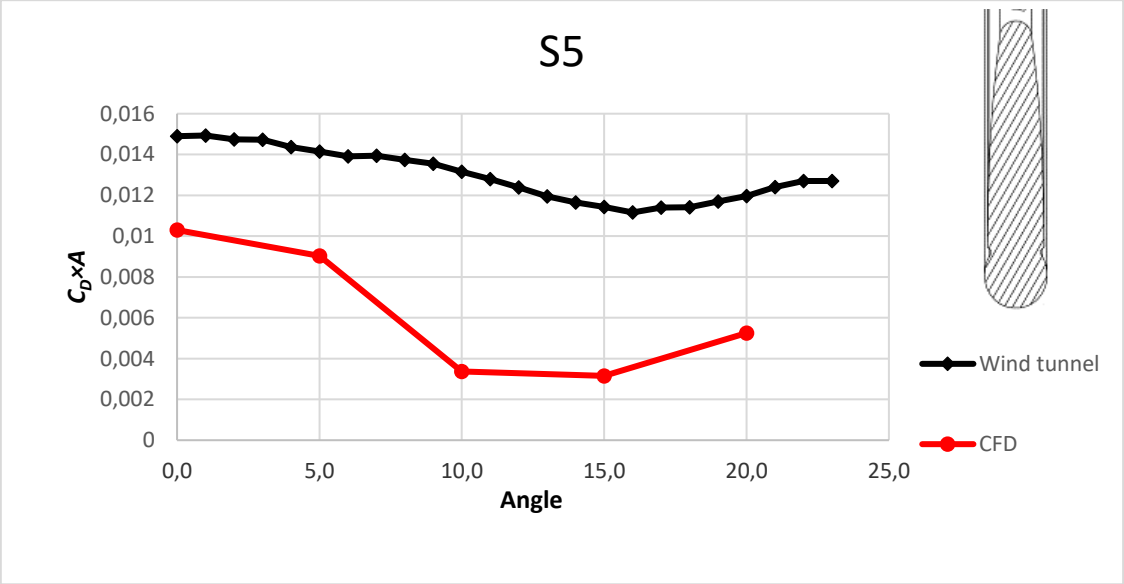


Figure 28. Rim used to test the dimpled surface (not the actually zipp surface)

The grid containing the dimpled rim had more than 5 million cells, meanwhile the one containing the smooth wheel had less than a quarter of that, and the result were non satisfactory, probably due to this difference in mesh size. Nevertheless, we think that this is an interesting field that should be study in deep.

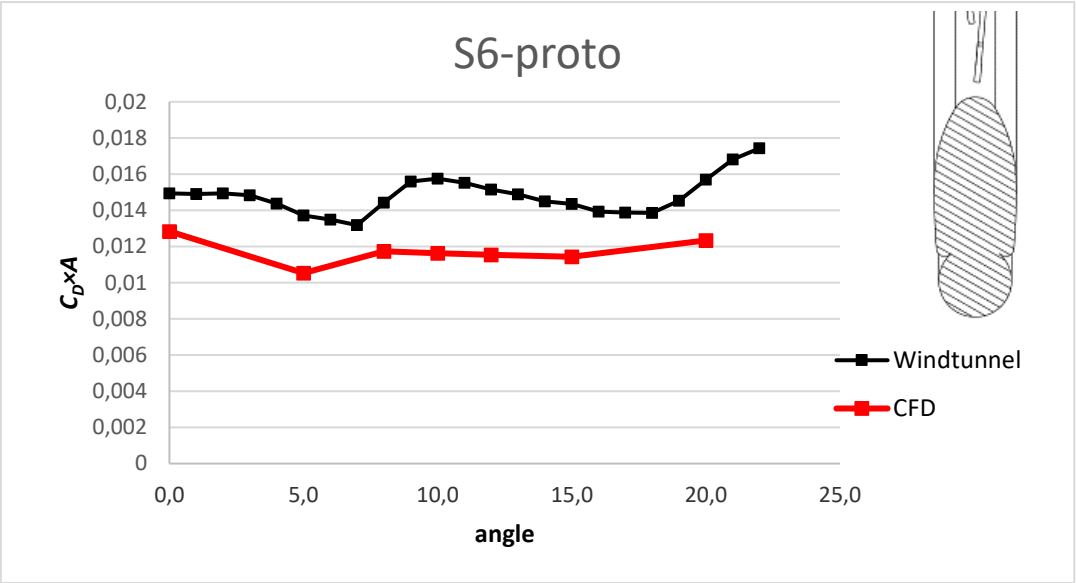
Wheel S5



Graph 13. S5 high profile wheel drag

Meanwhile with the high profile S5 tire the behavior was similar as stated before, the model under predicts the total Drag forces and the stall of the profile had come earlier on the model and with higher effects.

Wheel S6



Graph 14. S6 Prototype drag

The S6 wheel is an evolution of the S1 wheel, it shares the same wheel depth but some change in the profile has been made.

The wheel is also a disc brake wheel, therefore the spokes configuration and the hub geometry are completely different.

The results are similar to the one obtained for the S1 wheel, the Forces are around the same order of magnitude and the wheel behavior seems to be captured by the model.

5.4 Wheel design comparison

5.4.1 Introduction

To understand the influence of the wheel rim design we compare different rims with some light variations.

Even if it is not accurate, as we showed before in chapter 4, we choose to model only the rim because in the last part of this chapter we try to simulate a dimpled wheel. This means that the grid has to be sufficiently fine to capture the dimple depth that design as low as 2e-2 mm, this highly increase the cell number to more than 6 million cells for the wheel rim alone, making impossible for our solver to compute the solution.

5.4.2 Rim inner radius

We based the design on the Bora50, the inner radius of the wheel was variated from the original 8mm to a smaller 6mm and a larger 10mm as shown in the figure below.

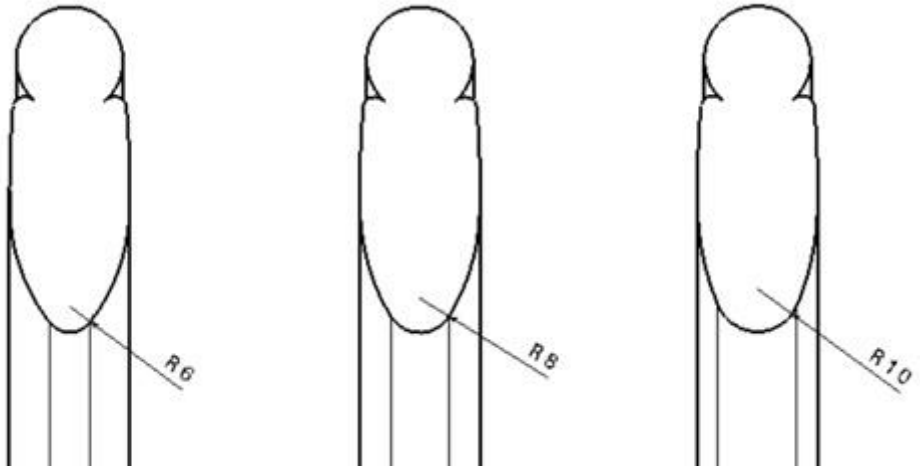


Figure 29. Wheel inner radius the original radius R8 compared to R6 and R10

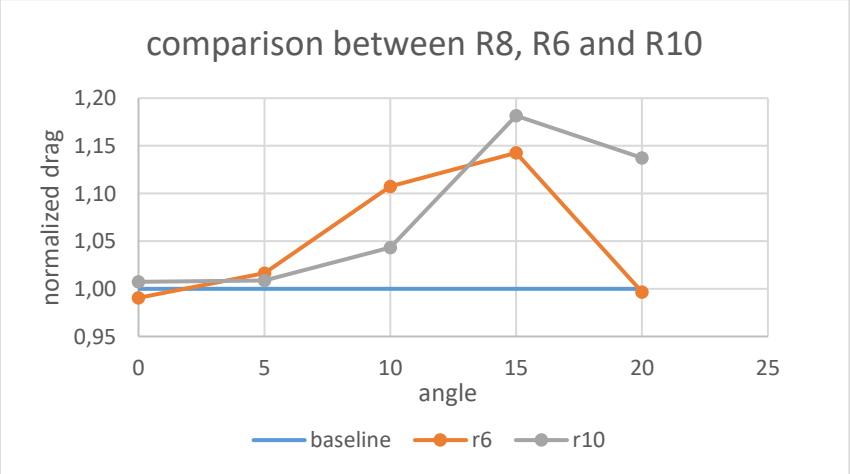
The wheel design is based on a set of circumferences arches for mould design reason, this is mainly made to facilitate the work of the company CAM machines, so we vary the radius using the base design and imposing a tangential constrain with the lateral surface of the rim.

We run the simulation for 0°,5°, 10°, 15° and 20°

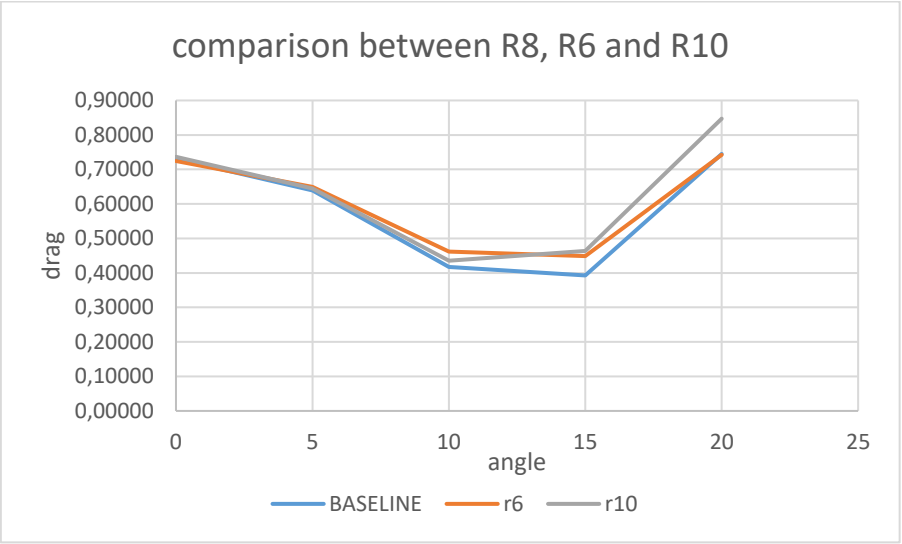
The results are normalized on the R8 baseline rim, because we want to make a clear representation of the advantage or disadvantage of the other two solution.

Table 10. Calculated normalized drag

angle	*baseline*	*R6*	*R10*
[°]			
0	1	0,99	1,01
5	1	1,02	1,01
10	1	1,11	1,04
15	1	1,14	1,18
20	1	1,00	1,14



Graph 15. Calculated normalized drag R8,R6, R10



Graph 16. Comparison between R8, R6 and R10

5.4.3 Rim width R_w

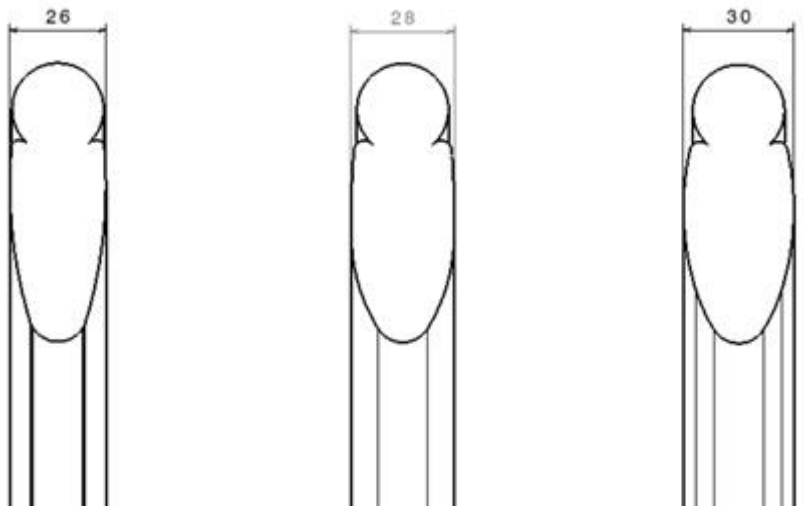


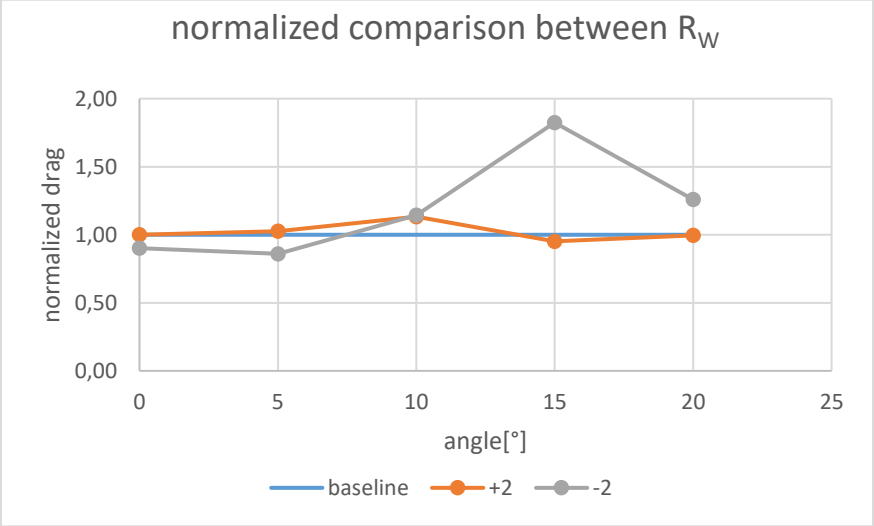
Figure 30. Wheel width comparison the original wheel width 28 compared to 26 and 30mm

As stated before wheel design is based on a set of circumferences arches, but in this case a "large" portion of the wheel was redesign, we choose to maintain part of the lower wheel and build a new arc of circumference and imposing the tangential condition on the brake zone and on the inner part of the rim, imposing the arc to be tangential on a vertical lines that is positioned at the desired width. We run the simulation for 0°,5°, 10°, 15° and 20°.

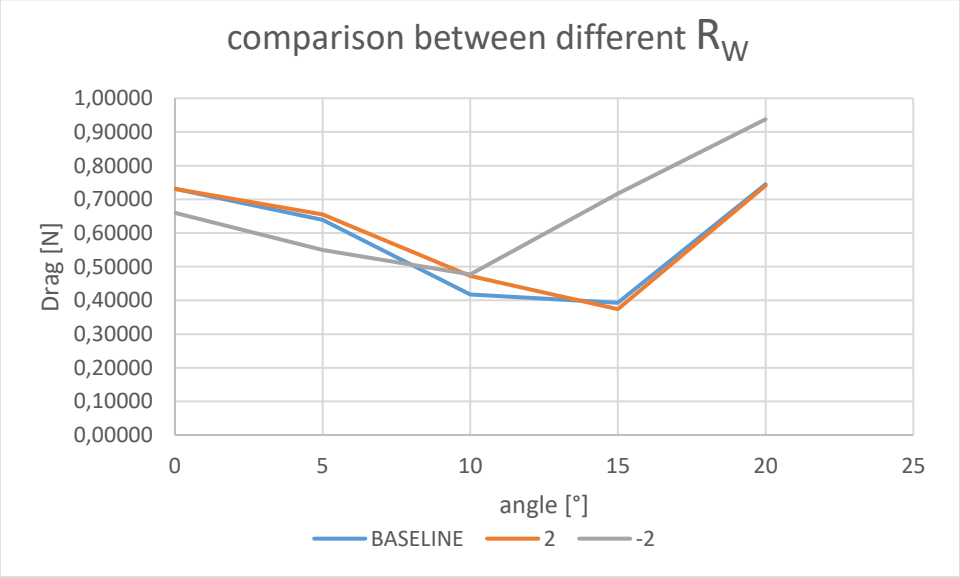
The results are normalized on the R8 baseline rim, because we want to make a clear representation of the advantage or disadvantage of the other two solution.

Table 11. Calculated normalized drag

angle	*baseline*	*+2mm*	*-2mm*
[°]			
0	1,00	1,00	0,90
5	1,00	1,03	0,86
10	1,00	1,13	1,14
15	1,00	0,95	1,82
20	1,00	0,99	1,26



Graph 17. Calculated normalized drag "normal" rim +2mm and -2mm



Graph 18. Calculated normalized drag "normal" rim +2mm and -2mm

This simple comparison between different rim inner radius and different wheel width shows that small geometric differences can change the wheel behaviour, but the difference in performances comes at different angles.

If we consider track-cycling naturally the best wheels is the one that shows a better performance at low angles, but for the road cycling things are different.

As stated before the most common condition encountered on the road are angle around 10°, making the slimmer wheel the worst in our case, even if it shows better performances until around 7°.

We also expect that the inner rim radius effect will be lower if we take in account the spokes effect.

There also different consideration to take in account in wheel design, meanwhile a certain profile can be the most favourable, there are some commercial limitations, certain measures can be copyrighted by a different industry, therefore the work of the wheel designer is to get the best results with the limitation imposed, with this in mind we think that CFD simulation is a powerful tool to help to overcome those limitations, an automated optimization tool can give few basic design for the experimental testing reducing the total costs.

5.5 Performance index

5.5.1 Introduction

We decided to introduce two performance index: those indexes are useful for a quick comparison of the performance of the wheel, giving an otherwise complicated information with a single pair of number relative to the drag and the side forces measured on the wheel for a variety of angle.

5.5.2 Mean performance index

The first and simplest one consists in an integral of the drag and side curves subdivided by the angle tested.

$$C_d = \frac{F_D}{\frac{1}{2}\rho V^2 A} \quad C_{dm} = \frac{\int_0^{\phi} C_d(\phi) d\phi}{\phi}$$

Where:

C_D - Drag coefficient.

V - Undisturbed Speed.

A - Reference area (we choose the area of circle based on the wheel diameter).

F_D - Drag Forces.

Accordingly the side forces are.

$$C_s = \frac{F_S}{\frac{1}{2}\rho V^2 A} \quad C_{sm} = \frac{\int_0^{\phi} C_s(\phi) d\phi}{\phi}$$

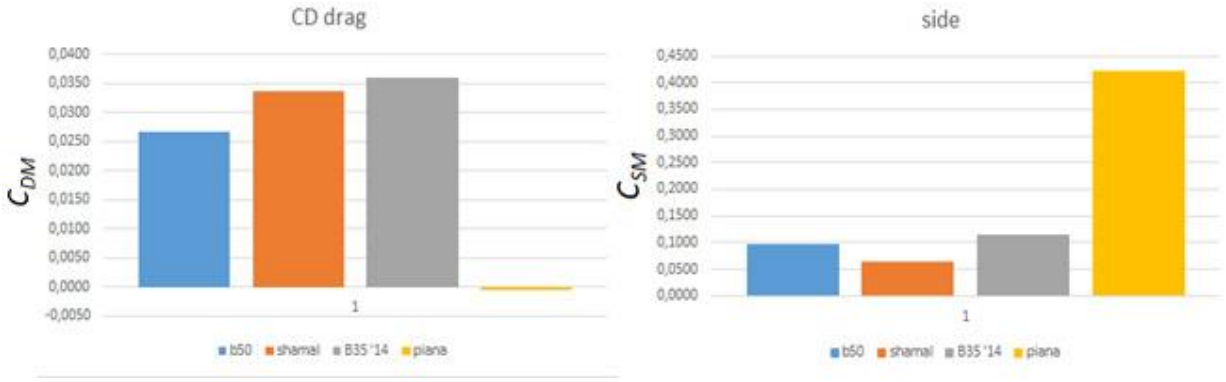
Where:

C_D -Side force coefficient.

V - Undisturbed Speed

A - Reference area (we choose the area of circle based on the wheel diameter)

F_S - Side Forces



Graph 19. Mean of the drag based on the angle tested

As the graph show, 4 wheel were tested: a disc wheel, and three spoked wheel with a wheel depth varying from 55mm 100m

5.5.3 Rotating wind performance index

A different model based on a rotation of the relative wind around the biker with the objective of obtain the drag was studied, the speed of the bike is fixed, as well as the modulus of the wind speed, the angle of the wind relative to the biker changes from 0° to 180° in favour of the biker.

Wind speed (V_{wg}) 3m/s, Bike speed (V_{wb}) 12,5m/s (45km/h), the vector of the wind rotate around the bike, as shown in the example:

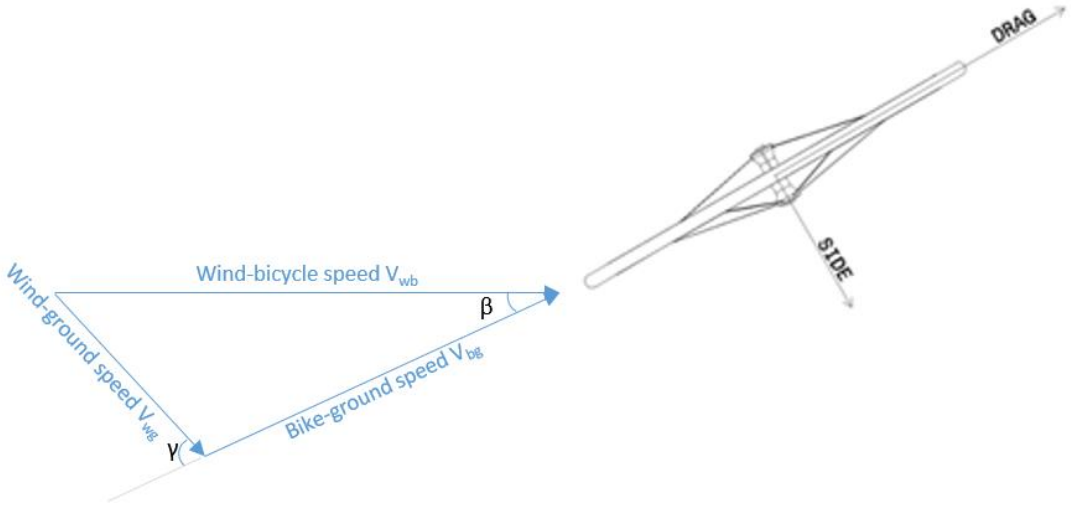


Figure 31. Relative speed and angles

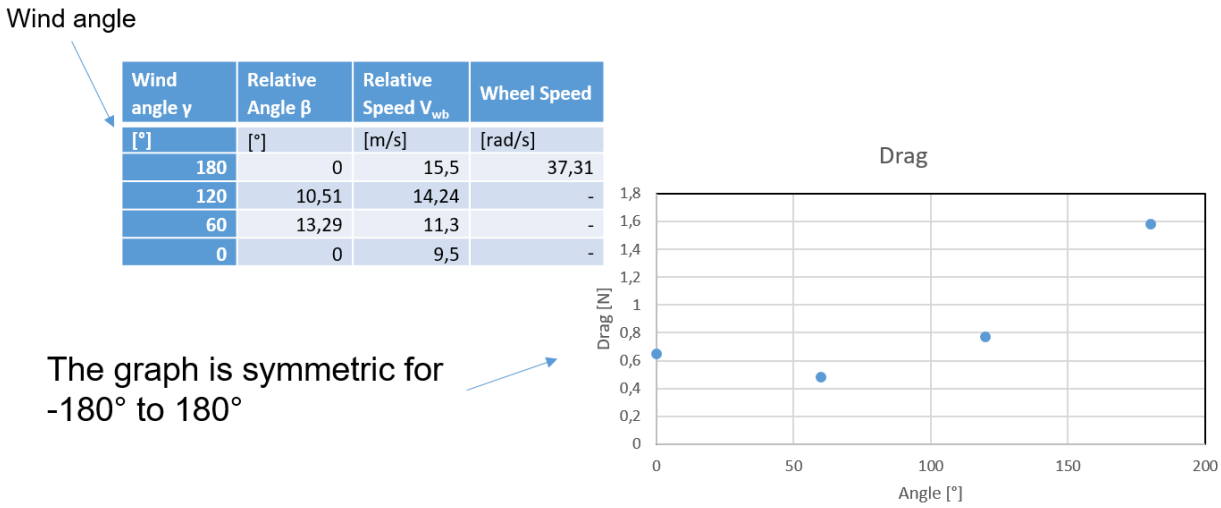
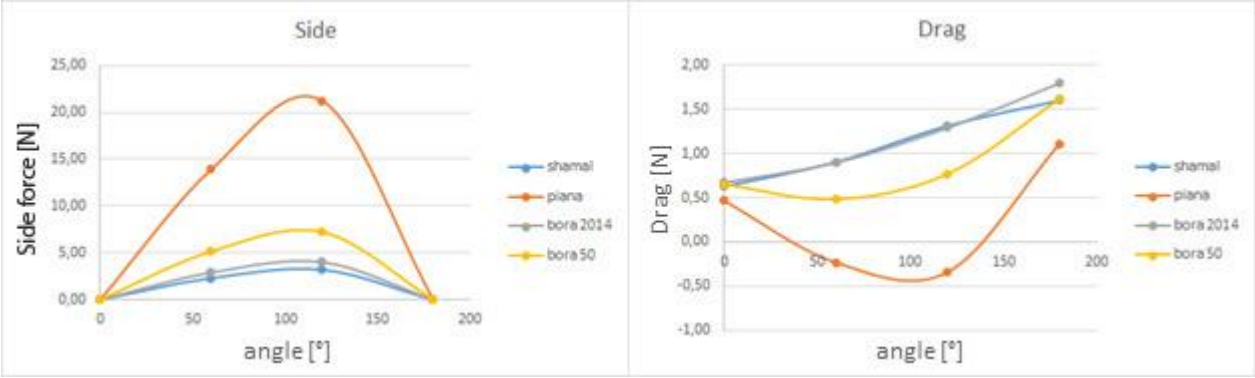


Figure 32. Testing procedure for a S1 wheel

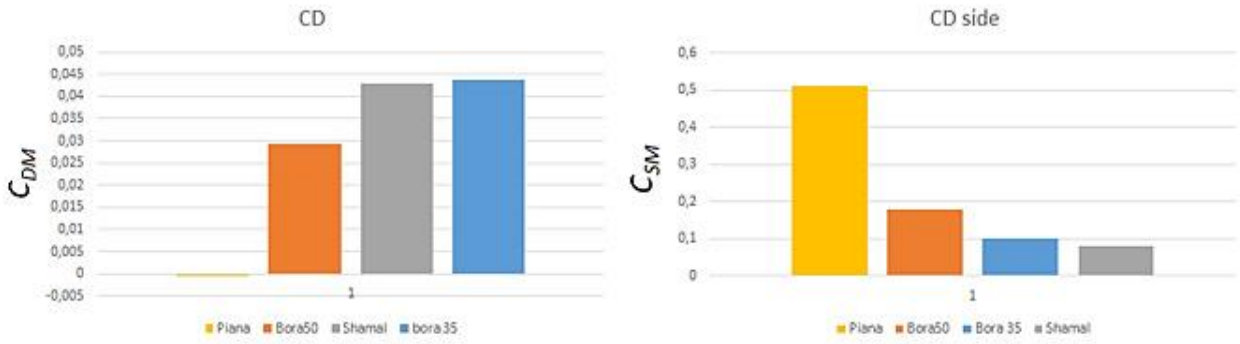
As it can be seen, the relative angle of the wind (in this case) is relatively small, less than 15°

We run this model for four wheels.

Measured Forces:



Graph 20. Mean of the drag based on the angle tested



Graph 21. Performance index, calculated between 0° and 180°

This approach is original and seems to need a study for every couple of wind speed and bike speed; however, it seems to catch the behavior of a bike in the real world, this method anyway should be tested and developed more, maybe with the inclusion of a probability coefficient relative to the wind angle.

5.6 Unsteady state model

5.6.1 Introduction

As the steady-state model can be a powerful tool to comprehend the behaviour of the wheel forces, it is unavoidably an approximation of the reality and the better results can be obtained using an unsteady-state model.

Unluckily the unsteady state model introduce a necessity for a greater number of iteration since a complete simulation is actually a sum of different simulation carried a different time-steps.

5.6.2 The Model

hose time steps should be choose wisely because if they are too small inevitably they will prolong the time used to obtain a converged solution, on the other side if we choose a larger time step the possibilities to not capture all of the phenome that characterize the fluid regions, and may cause instability of the model itself.

Based on this the time step is a function of the used model and mesh size, for example it is advisable to keep a low Courant number with every model to ensure stability to the code, in the explicit-scheme it is necessary to keep it below 1.

The Courant–Friedrichs–Lewy (CFL) condition is a necessary condition for convergence while solving certain partial differential equations (usually hyperbolic PDEs) numerically by the method of finite differences. It arises in the numerical analysis of explicit time integration schemes, when these are used for the numerical solution. As a consequence, the time step must be less than a certain time in many explicit time-marching computer simulations, otherwise the simulation will produce incorrect results.

For example for one-dimensional case, the CFL has the following form:

$$C = \frac{u\Delta t}{\Delta x} \leq C_{max}$$

Were C is the Courant number, u is the fluid speed, Δt the time step and Δx the length interval determined by the mesh, the formula actually give a minimum time-step to impose that between steps the fluid particle would scan the entire mesh domain, specially using an explicit solver this number has to be lower than 1.

In our case, we have a high number of cells, with some very small ones in the prism layer near the wheel, so we decided to use an implicit solver; Implicit (matrix) solvers are usually less sensitive to numerical instability and so larger values, and a time step value of 0,001 s.

We decide to test a Bora 80 wheel at 13 m/s and to limit the test at an angle of 10° which is a common angle in the real world scenario *Easton cycling (2010) Godo et Al(2009).*

The floor of the fluid region move in the same direction of the wheel at the same speed, we impose the symmetry condition on the top and on the side of the fluid domain.

The moving region of the wheel as before is separated from the rest using an interface, the movement imposed to the inner part of the wheel is a rigid body motion, on the outer part of the wheel we impose a rotational speed to the wheel's walls (the tire and the outer part of the rim).

We test the previous turbulence model in an unsteady state model, adding two higher model the DES and the LES model.

5.6.3 Results

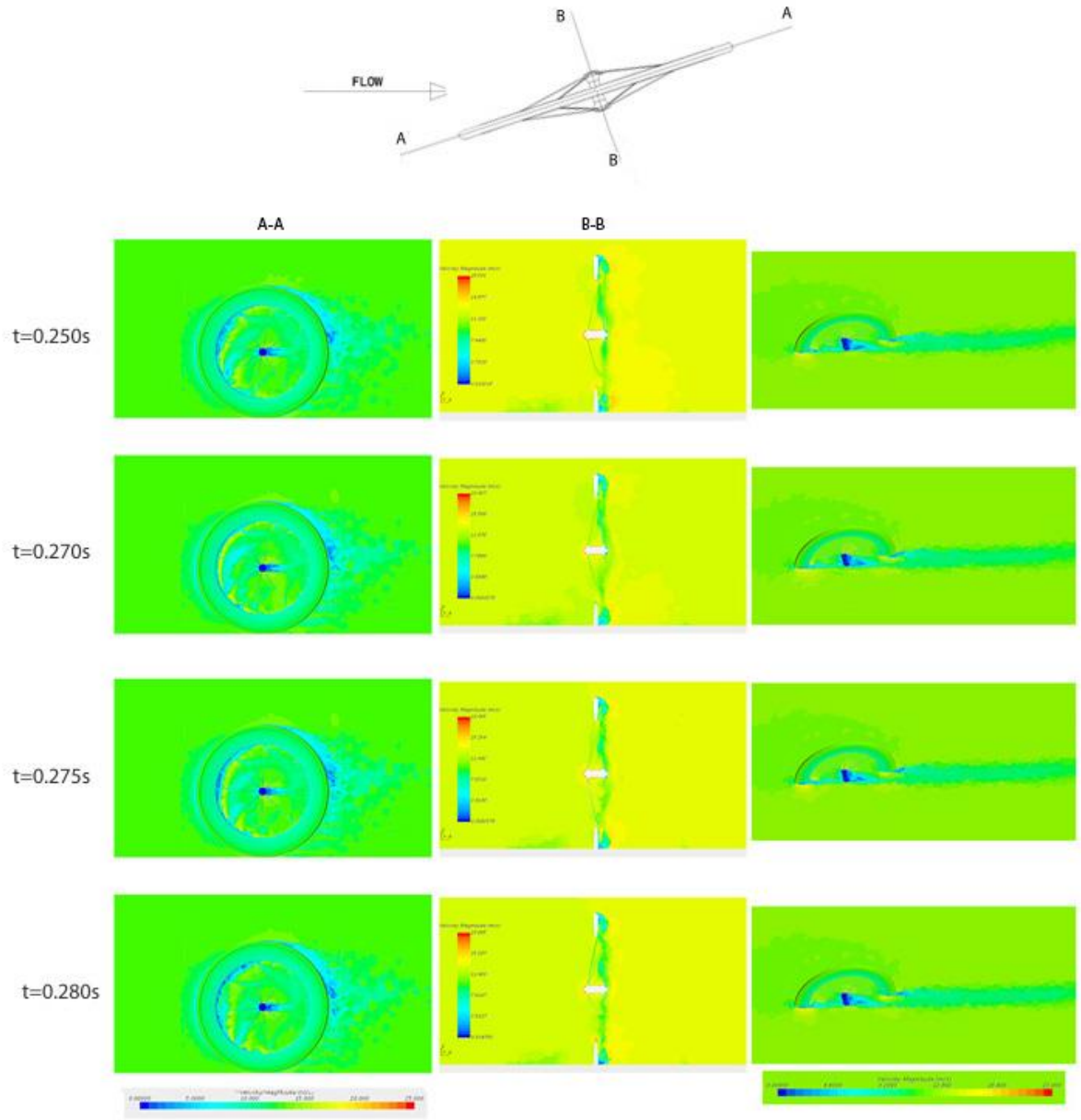
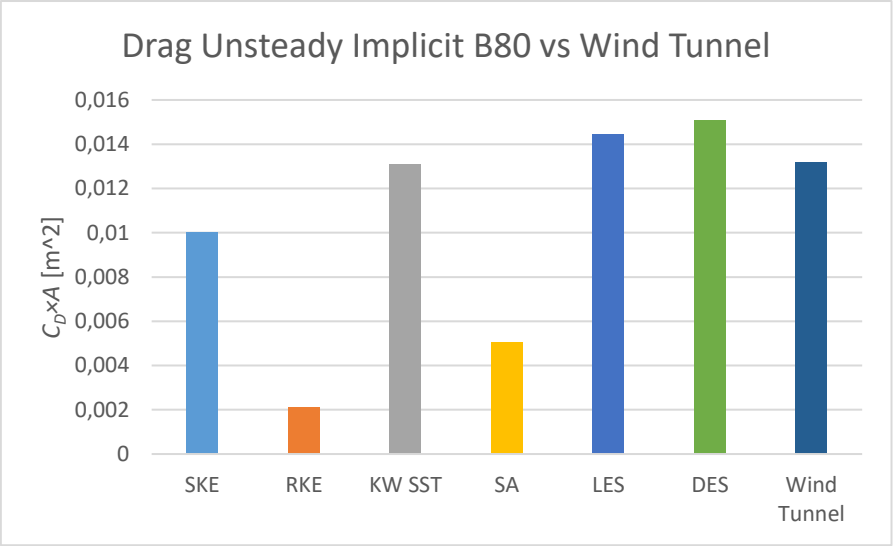
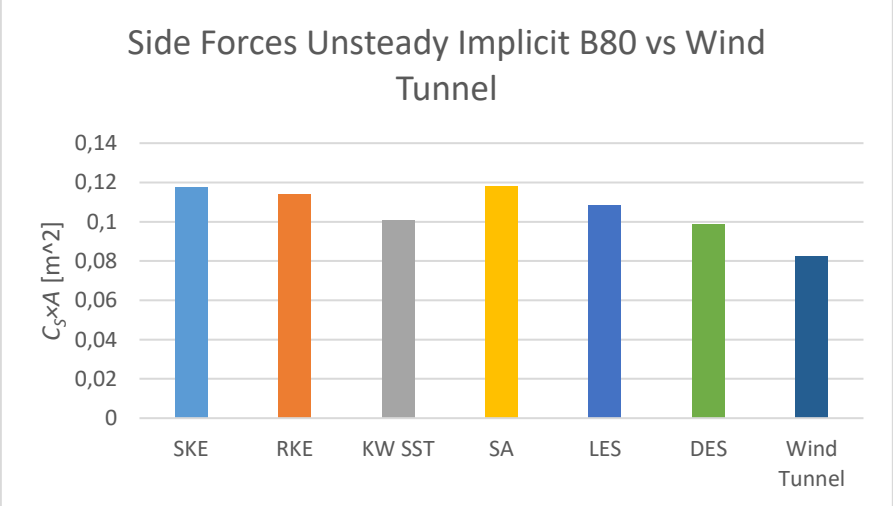


Figure 33. Speed field at 12m/s at 10° for a Bora 80 the second picture is taken 20ms after the first, the third after 25ms from the first and the last at 30ms from the first

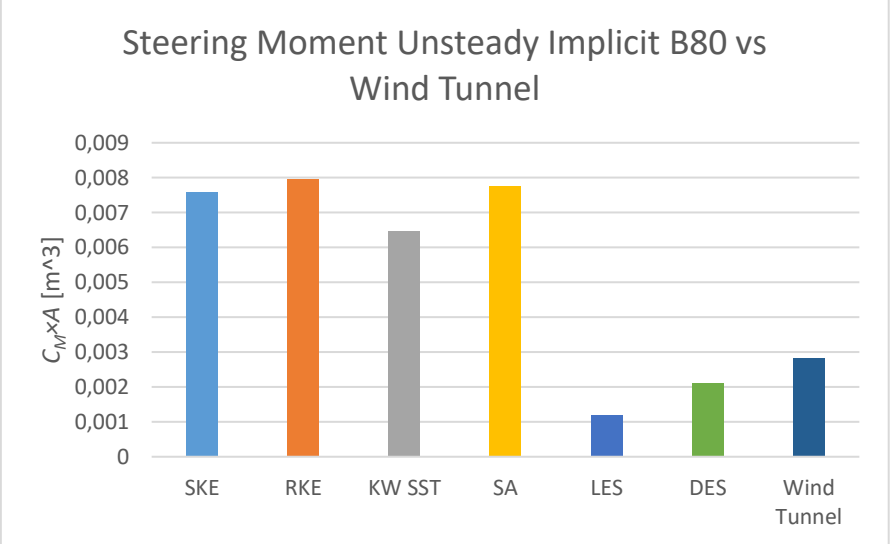
In the imagine above it can be noted a that there recirculating zone behind the inner part of the wheel, with vortex generating in the trailing edge of the wheel, it can be noted also the slower zone behind and on top of the wheel, caused a deceleration of the fluid determined by the wall movement opposed to the flow direction.



Graph 22. Unsteady model, drag results Cd x A [m^2] compared to the wind tunnel



Graph 23. Unsteady model, side results Cs x A [m^2] compared to the wind tunnel



Graph 24. Unsteady model, side results Cm x A [m^3] compared to the wind tunnel

These results are obtained for only a wheel may suggest that the closest one is unavoidably the best model, this is not the case, and the model can behave in a good way in this particular condition and work in a worse way in other models.

Still we can draw some conclusions, the “typical” model seems to under-predict the forces by a quite large margin in drag measurements the over predict the forces in side and steering moment (measured with an axis normal to the road)

The $K-\omega\ SST$ model, meanwhile in the steady state was not able to converge with a satisfactory low residuals, behave much better in this condition, the Standard k-ε model is still the faster and the more stable but it under predicts the total drag forces.

This model has probably a mesh too coarse to use the LES model, and the time step should be lower using less inner iteration, but beside some difficulties to converge (resolved lowering the under-relaxation factor) the results are in the same region of the wind tunnel results, over estimating slightly the drag forces. The DES was faster and converge earlier showing a stable behavior during the simulation, it is still more resource demanding than the RANS turbulence model but less than the LES model.

Even if we need more unsteady state test we can draw some conclusion. Depending on the computational resource at your disposal, then steady-state model works better with a $k-\varepsilon$ model, the unsteady state RANS model works good with $K-\omega\ SST$ and $k-\varepsilon$ models, with an advantage for the $K-\omega\ SST$ but *DES* seems to give the best results without the computational resource used by the *LES.*

Chapter 6

Evaluation of the Method on OpenFOAM code

6.1 Introduction

The major part of the simulation carried out in this thesis has been done using a commercial code (Star-CCM+); the cost for the maintenance, for the annual license and for the use of this program can be relatively high so we developed a base model for the open-source program OpenFOAM 4.0, the geometry of the wheel was defined with the open-source program SALOME in stl file, and the result was viewed with ParaView.

OpenFOAM (for "Open source Field Operation and Manipulation") is a C++ toolbox for the development of customized numerical solvers, and pre-/post-processing utilities for the solution of continuum mechanics problems, including computational fluid dynamics (CFD). The code is released as free and open source software under the GNU General Public License.

This model was used with Ubuntu OS, in a laptop with 16GB ram and 2.0 GHz i7 processor.

The main source for this brief introduction is the OpenFoam users guide and the online openfoamwiki, and the guide made by *J. Casacuberta Puig (2014),* the tutorials by *T. Holzmann "holzmann-cfd.de"* was also useful to make this model.

6.2 About OpenFOAM

OpenFOAM is primarily a C++ library, used primarily to create executables, known as applications. The applications fall into two categories: solvers, that are each designed to solve a specific problem in continuum mechanics; and utilities, that are designed to perform tasks that involve data manipulation. The OpenFOAM distribution contains numerous solvers and utilities covering a wide range of problems. One of the strengths of OpenFOAM is that new solvers and utilities can be created by its users with some pre-requisite knowledge of the underlying method, physics and programming techniques involved. OpenFOAM is supplied with pre- and post-processing environments. The interface to the pre- and post- processing are themselves OpenFOAM utilities, thereby ensuring consistent data handling across all environments. The overall structure of OpenFOAM is shown in the figure below (openfoam.com user guide)

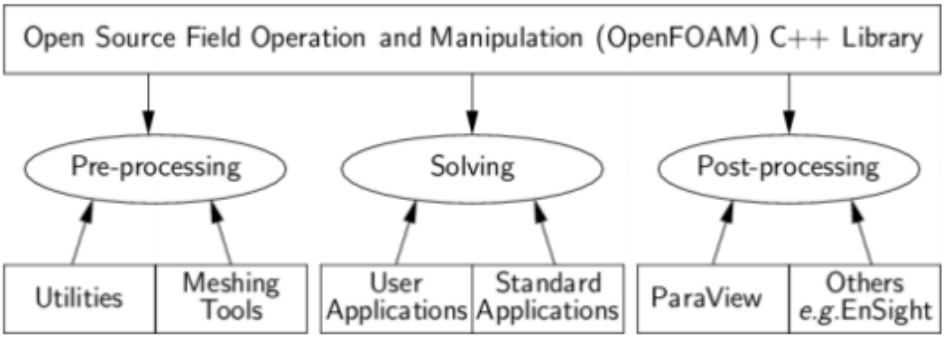


Figure 34. Overview of OpenFOAM structure, extracted from official guide

6.3 OpenFOAM Data Structure

The OpenFOAM lower-level libraries outline generic interfaces for data input and output to ensure consistency throughout the simulation environment and provide re-usable components as building blocks for higher-level libraries. For example, the lower-level libraries define discretization and parallelization details which are then implicitly used in higher-level code. Standalone executables are generated for utilities and solvers; generally, standalone solvers are written to manage one specific physics regime, such as buoyancy simulation or combustion. Multi-physics simulation requires combining existing solvers to create new customized solvers. This component-based approach gives the user a great amount of flexibility, which may not be available in commercial CFD codes, and the availability of all source code eliminates the 'black box' factor in commercial codes in which the details of model implementation are hidden. To discretize the problem domain, OpenFOAM uses an internal unstructured mesh format which supports tetrahedral, hexahedral, and polyhedral cell types. All meshes are explicitly three-dimensional; one- or two-dimensional calculations are performed by using special boundary conditions on any plane(s) normal to the coordinate direction(s) of interest which switch off the solution algorithm in the unnecessary direction(s). OpenFOAM requires the following mesh quality criteria to be met, as stated in the official User Guide, for a mesh to be computationally valid:

- Contiguous – cells must cover the full computational domain and not overlap one another.
- Convex – every cell must be convex such that the cell center must be contained within the cell.
- Closed – each cell should be closed, such that the sum of all face area vectors oriented outwards for each cell should equal zero, and each edge is used by exactly two faces of the given cell.
- Orthogonality – for each internal face, the angle between the face area vector and the center-to-center vector must be less than 90º.

Figure 35 and Figure 36 detail a polyhedral cell illustration and its corresponding OpenFOAM mesh representation. As shown in Figure 36, the polyhedral mesh in OpenFOAM is represented with four linked lists: a list of point coordinates, a list of cell faces with references to the point coordinates, a list of owner cells for each face, and a list of neighbour cells for each face. This well-defined approach is optimal for interfacing with the unstructured mesh during the solution process.

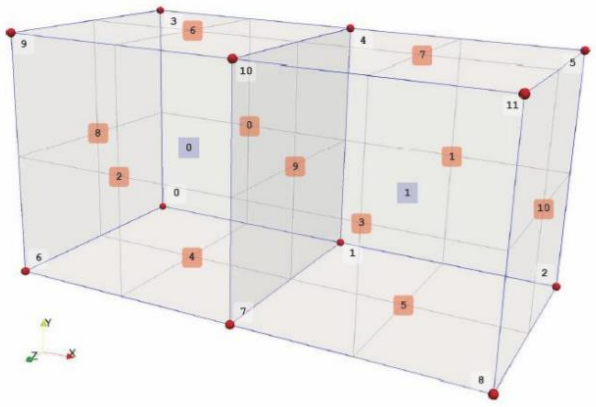


Figure 35. Two cell in polyhedral mesh, J. Casacuberta Puig (2014)

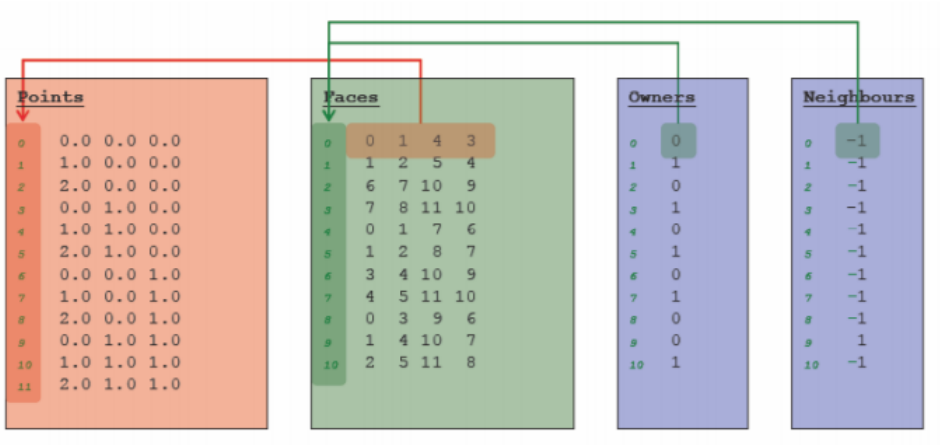


Figure 36. OpenFOAM linked list mesh representation of two cell mesh

6.4 OpenFOAM model structure

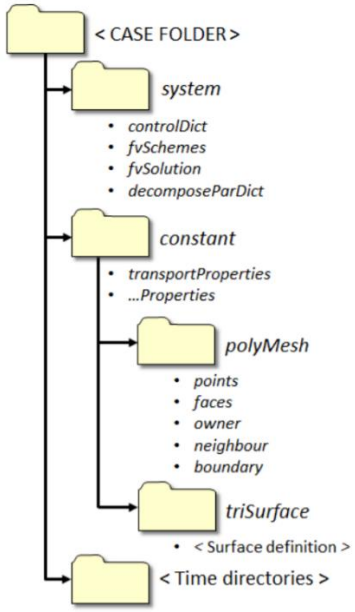


Figure 37. OpenFOAM case directory structure

6.4.1 System Directory

This directory contains discretization and solution procedure information. General solver and utility settings in OpenFOAM are controlled by dictionary files that have the -Dict suffix. These files are read dynamically during solver execution and allow parameters to be varied during runtime. The following files are located within the system directory:

- controlDict – this dictionary contains run control parameters, such as start/end time, time step controls, and data output settings. Additional data output can be defined here using pseudo-code user functions described in Section 6.5.5.
- fvSchemes – this file defines the discretization schemes used for each algebraic tensor operation and each high-level mathematical expression.
- fvSolution – the iterative equation solver settings, tolerances, and specific algorithm control parameters are defined within this file.

- decomposeParDict – this file defines domain decomposition parameters for use with the decomposePar tool for parallel simulation.

6.4.2 Constant Directory

The constant directory holds the mesh description and any files with the -Properties suffix describing required physical properties, such as viscosity. Additionally, turbulence model parameter files are located here if necessary for the simulation. The polyMesh directory contains the following mesh representation files:

- points – the points file is an ordered list of vectors describing the three-dimensional location of each point of the mesh in sequential order. Each row contains the global x, y, and z coordinates of the point in dimensions of meters.
- faces – the faces file contains the faces defined by the vertices in the points file. Each row of the list corresponds to a face in sequential 25 order and is comprised of a counter indicating the number of vertices making up the face, followed by a vector with the corresponding vertex labels. Because OpenFOAM supports polyhedral meshes, faces can have more than three (tetrahedral mesh) or four (hexahedral mesh) points.
- owner – each face in OpenFOAM is associated with an 'owner' cell and a 'neighbour' cell. The owner file is a sequential list of cell numbers which directly corresponds to the list in the faces file.
- neighbour – the neighbour file is a sequential list of cell numbers, similar to the owner file, except the listed cell numbers correspond to the neighbour cells for each face. Boundary faces are given a -1 index within this file as each boundary face belongs to an owner cell only.
- boundary – this file contains a list of patches (geometrical definitions of the boundaries) for the simulation. The number of corresponding patches is required at the beginning of the list. Each patch is given a name and described by a dictionary containing three entries. The geometrical definition of the patch is the type entry, and takes values of patch (used to apply most boundary conditions), wall (denotes boundaries for distance to wall calculations), and empty (defines patches to be ignored for one- or two-dimensional calculations). The faces comprising the patch are defined by a beginning face index with the startFace entry and the number of subsequent faces in the nFaces entry.

The polyMesh directory also contains the blockMeshDict file, which is used for input to the blockMesh mesh generation utility. The other directory in the constant directory, but not required at simulation runtime, is the triSurface directory. Any necessary surface definitions, such as Stereolithography (STL) files, are located here.

6.4.3 Time Directories

The time directories are named using the specified numerical format for the time step in the controlDict file in time-accurate simulations, or the iteration number in steady-state simulations. All data necessary for post-processing the simulation and for restarting at a specified time step or iteration are contained within the time directories. These directories are created dynamically by the running solver as the solution is generated; if a directory already exists, the solution files contained within the existing directory will be overwritten. For each simulation, initial conditions are defined within the 0 directory. Every initialized simulation field is required to have a corresponding file in the 0 directory; for example, the velocity and pressure fields are defined by U and p files, respectively. Each file follows a specific formatting scheme. Dimensions of each field are specified

by a vector, which corresponds to a list of SI units of measure. The initial conditions of the field are specified with an internalField entry. Boundary conditions on each patch for each field are set with a boundaryField entry which lists the patches defined in the boundary file in the polyMesh directory. Each boundary condition requires various inputs, which are discussed in further detail in the official User Guide. Each subsequent time directory after the 0 directory contains all outputted simulation field results in a single file for each field for post-processing or for restarting the simulation. The outputted fields follow the same file format described above; boundary condition information for each field is carried in a boundaryField entry as in the initial condition files. In each field file, simulation data for all cells are recorded following the internalField entry in a sequential list based on cell number. This enables the same files to be used for both simulation restart as well as post-processing results from the simulation.

6.5 OpenFOAM Utilities

An assortment of utilities that provide various levels of functionality is included with the standard OpenFOAM package. Each utility is a separate executable, which is called within the case directory to perform certain operations. Sample dictionary files are incorporated with most utilities that allow utility settings to be specified. Pre-processing utilities cover field mapping between meshes, setting initial conditions, and other various initialization functions. Meshing utilities are included for mesh generation, mesh conversion, and mesh manipulation. Post-processing utilities cover data conversion to various thirdparty packages, basic field calculations, and data sampling. Some of the common utilities are discussed in this section; a complete list of utilities is published in the official User Guide.

6.5.1 Pre-processing

Most of the pre-processing utilities perform initialization of fields for various purposes. The setFields utility allows the user to set non-uniform initial values of fields based on bounding shapes or other parameters defined in the setFieldsDict file. The createTurbulenceFields tool creates and initializes all of the fields necessary for turbulence modelling. Another important preprocessing tool is the mapFields utility, which allows fields to be mapped between meshes sharing similar geometry, or from a coarser mesh to a finer mesh.

6.5.2 Meshing

Basic mesh generation is provided by the blockMesh and snappyHexMesh utilities. The blockMesh utility allows the user to generate multi-block meshes and reads a geometry definition comprised of points, blocks, and patches from the blockMeshDict file in the polyMesh directory. Blocks are specified by listing the points comprising each block according to the right-hand rule and defining grid spacing requirements. For a typical user it is relatively difficult to generate complex meshes using blockMesh because the tool is based fully on text file input and has no graphical interface to visualize the designed mesh. However, the tool performs adequately for simple meshes or for an initial mesh as a basis for trim meshing with snappyHexMesh. The snappyHexMesh utility provides an alternative method for generation of three-dimensional body-fitted hexahedral grids. The tool is inherently three-dimensional and requires an initial mesh describing the computational domain and a surface representation of the complex body in STL format. From the initial base mesh,

snappyHexMesh automatically generates a mesh conforming to the surface with optional snapping of cells to the surface and optional extrusion layer generation. The utility also allows for parallel mesh generation with a load balancing step during each iteration. If the mesh has already been generated using third-party meshing software, OpenFOAM provides many utilities to convert the mesh to the OpenFOAM mesh format. Conversion utilities exist for meshes from the primary commercial CFD packages of Fluent, STAR-CD, STAR-CCM+, and CFX. Additional conversion is possible from meshes generated by standalone meshing suites such as Gambit, Pointwise and IcemCFD. After converting a mesh into OpenFOAM format, many utilities exist for modifying, fixing, or otherwise preparing a mesh for a simulation. There are utilities to refine, renumber, or rotate the mesh, as well as create baffles or additional patches. OpenFOAM also allows the definition of cellZones and faceZones, special sets of cells or faces that are automatically decomposed for parallel processing.

6.5.3 Domain Decomposition

Parallel calculations require domain decomposition, which is the process of splitting the computational mesh for allocation among a number of processors. Communication between processors is accomplished with a Message Passing Interface (MPI), which is a communications protocol for high-performance parallel computing. OpenFOAM is built with the open-source OpenMPI protocol by default; however, other MPI protocols can be supported with some minor changes in the package compilation process. The process of splitting the domain for parallel computation is performed in OpenFOAM by the decomposePar utility. Several mesh splitting algorithms are available. 'Simple' decomposition allows the user to define split regions by equivalent sectioning of the domain based on the Cartesian directions. 'Metis' and 'Scotch' decomposition are standardized partitioning algorithms based on the minimization of the number of interface boundaries for each processor. The decomposePar utility run in the case directory and generates local processor directories for every processor. Each directory contains separate time directories that hold the field values corresponding to cells allocated to that processor. During the solution process, each processor writes out the necessary fields into its local directory. After the simulation is complete, executing the reconstructPar utility in the case directory recombines all of the solution files into common time directories in the case directory.

6.5.4 Post-processing

The standard OpenFOAM installation is packaged with a post-processing utility called Paraview, an open-source data visualization package produced by Kitware and based on the Visualization Toolkit (VTK) format for data rendering and post-processing. Paraview from version 3.10 includes a built-in data reader plugin for OpenFOAM data that provides an option for working directly with decomposed data, eliminating the reconstructPar. For large cases a foamToVTK utility is available to convert OpenFOAM data directly to the VTK format for faster post-processing if necessary. If other post-processing tools are available OpenFOAM includes utilities to convert data to popular third-party post-processing utilities such as EnSight, Fieldview, Tecplot and Fluent. In addition to visual post-processing, OpenFOAM also includes utilities for numerical post-processing of simulation field files. Utilities such as Mach and streamFunction calculate quantities from the stored velocity field in each time directory and output additional field files for the calculated values. Tensor operations such as divergence, magnitude, and splitting of a field into components are available to operate on stored field files using the foamCalc utility. Lastly, solution sampling can be conducted

on stored field files at defined survey locations in the domain using the sample and probeLocations utilities.

6.5.5 Run-time Processing

In addition to post-processing, OpenFOAM allows users to conduct runtime processing using function objects. Function objects are pre-compiled libraries that can be added to the controlDict file to perform various processing tasks during run-time. The range of available functionality is extensive and includes sampling of data at point locations and along survey lines, calculation of force coefficients on bodies, and generation of elements for visual postprocessing, such as section planes and iso-surfaces.
Furthermore, writing only the data of interest requires a much smaller memory footprint than the full dataset, reducing the time required to post-process a solution as well as the overall disk space requirements for a time-accurate solution.

6.5.6 Solution Monitoring

Monitoring solution progress is important to ensuring computational accuracy and determining solution convergence. During execution of a general simulation, OpenFOAM outputs convergence residual statistics as well as local and global Courant number information for each iterative solution process. This output can be sent to a log file using the foamJob script file to initiate the solver as a background process; the log file can then be monitored in real-time with the tail UNIX command. The foamLog script file processes the log file and generates files corresponding to each monitored residual for graphical comparison.

6.6 Hypotheses of the OpenFoam TriSpoke model

The simulation uses an incompressible turbulent flow, we use a RAS turbulence model (a standard k-ε model) with wall functions, and the simulation is unsteady with the time step dependent on the Courant number.

We use a fictional three spoke wheel design just to test the program and the model, this because we want to limit the number of elements for this model, partially because OpenFOAM seem more demanding in term of computational resources and because the snappyHexMesh tetrahedral mesh utility creates a high number of elements.

Figure 38. Wheel used

The wheel was created with CATIA then imported in Salome, we use Salome to set the different patches of the wheel and then to export the stl file.

```
solid movingWall
 facet normal -2.025661e-02 -9.596524e-01 -2.804583e-01
   outer loop
     vertex -6.933471e-02 -1.142617e-02 -3.009786e-01
     vertex -6.382949e-02 -8.797965e-03 -3.103692e-01
     vertex -6.008799e-02 -1.186628e-02 -3.001405e-01
   endloop
 endfacet
 facet normal -1.152849e-02 -9.611567e-01 -2.757625e-01
   outer loop
     vertex -6.933471e-02 -1.142617e-02 -3.009786e-01
     vertex -7.233058e-02 -1.078980e-02 -3.030714e-01
     vertex -6.382949e-02 -8.797965e-03 -3.103692e-01
   endloop
 endfacet
 facet normal -2.784060e-02 -9.930150e-01 -1.146565e-01
   outer loop
     vertex -7.641355e-02 -1.142617e-02 -2.992597e-01
     vertex -6.933471e-02 -1.142617e-02 -3.009786e-01
     vertex -7.434396e-02 -1.242004e-02 -2.911545e-01
   endloop
 endfacet
 facet normal -1.230336e-01 -9.490141e-01 -2.902325e-01
   outer loop
```

Figure 39. Example of the stl file converted using the open-source software Salome

The velocity of the wheel is defined by: on the outer part of the wheel a wall rotational speed is imposed, the inner part of the wheel is contained in a rotating region dividend by an Cyclic Arbitrary Mesh Interface (AMI). The AMI interface is used to coupling condition between a pair of patches that share the same outer bounds, but whose inner construction may be dissimilar.

Other models were initially created to improve the user confidence with the program (tutorials, simple disc wheel etc.) as example we report the simulation of a three spoke simple wheel.

6.7 Meshing

The background mesh was created using the blockMesh utility (attachment A1); the fluid domain is 2 meter wide 6 meters long and 2 meter high.

The fluid boundary is set in the boundary section of the using the blockMeshDictfile (A1), we choose to use a wall boundary for the side, the top and the floor of the boundary, the floor is moving at the same speed of the rider, this speed is set in the initial condition U file.

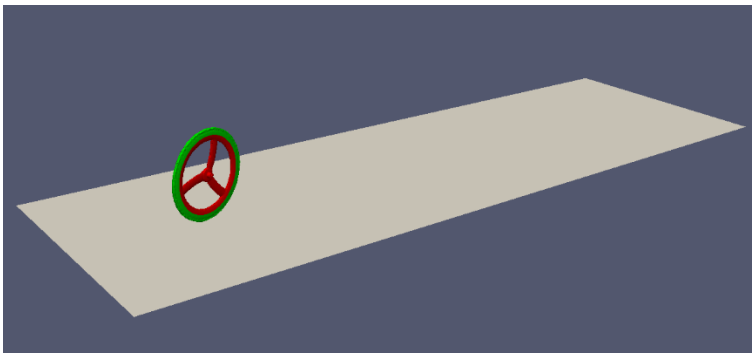


Figure 40. Floor and wheel model shown in ParaVIEW

The wheel was created with CATIA then imported in Salome; the wheel was divided in two parts, based on the wheel geometry:

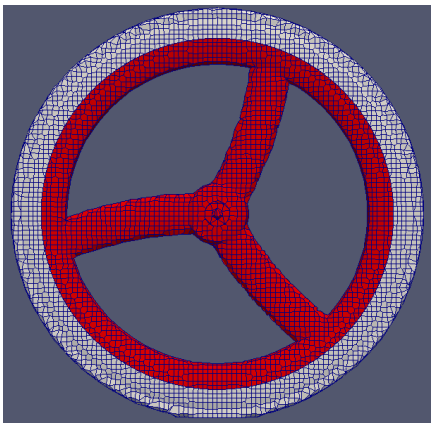


Figure 41. Wheel meshed, in white the solidWall patch in red the movingWall patch shown in ParaVIEW

- the outer solidWall: the outer part of the wheel were the movement can be imposed with a rotational wall velocity
- the inner movingWall: the central part of the wheel containing the spokes, this part is contained in fluid region where the mesh movement is applied

The wheel mesh was created using the snappyHexMesh (attachment A2) utility; the snappyHexMesh utility generates 3-dimensional meshes containing hexahedra (hex) and split-hexahedra (split-hex) automatically from triangulated surface geometries, or tri-surfaces, in Stereolithography (STL) or Wavefront Object (OBJ) format. The mesh approximately conforms to the surface by iteratively refining a starting mesh and morphing the resulting split-hex mesh to the surface. An optional phase will shrink back the resulting mesh and insert cell layers. The specification of mesh refinement level is very flexible and the surface handling is robust with a pre-specified final mesh quality. It runs in parallel with a load balancing step every iteration.

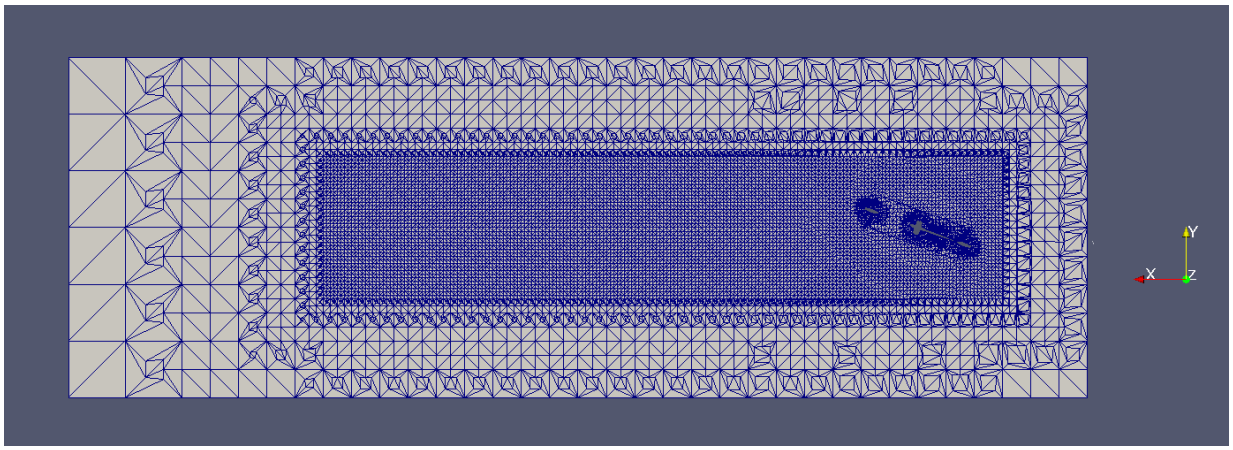


Figure 42. snappyHexMesh result in ParaVIEW

Using the createPatchDict (attachment A3), we define the interface patch necessary to create the rotating region

6.8 Initial conditions

The initial conditions are set in the 0 folder, they regard the p (initial pressure), U (the initial fluid velocity) k and nut, the txt files are shown in the Attachment A4, the speed of the fluid was set at 10m/s and the rotational speed was according set at 30 rad/s (attachment A5 dynamichMeshDict)

The fluid is air and the model used is the k-epsilon model.

6.9 Controls

The controls for the simulation was set in the system folder, since the relatively low number of the cells the simulation was compute in a sufficiently low time to impose a maximum Courant number lower or equal to 1, to ensure the stability of the simulation and the time step was calculated on this base (attachment A6). We use the PimpleDymFoam utility, the pimpleDymFoam is an implementation of the pimpleFoam solver that allows for dynamic meshes. Like pimpleFoam, the solver is transient, allowing for relatively large time steps thanks to the hybrid PISO-SIMPLE (PIMPLE) algorithm.

The discretization schemes used for each algebraic tensor operation and each high-level mathematical expression was set in the fvSchemes file and the iterative equation solver settings, tolerances, and specific algorithm control parameters are defined within this fvSolution (A7)

The domain was decomposed in four sub-domains to permit the parallel run of the simulation using the decomposeParDict utility.

6.10 Results

The main objective was to create a simple model able to run the results in term of the velocity field are shown below, if needed this model can be simply modified for future cases if the geometric definition of the wheel is respected.

In the figures below we can see some example of the obtained result, using the ParaVIEW open source software with the OpenFOAM results, the picture show from different time steps from 0.1 to 0.85 seconds (about 4 complete rotation of the wheel), the Flow Direction is the same as the x direction.

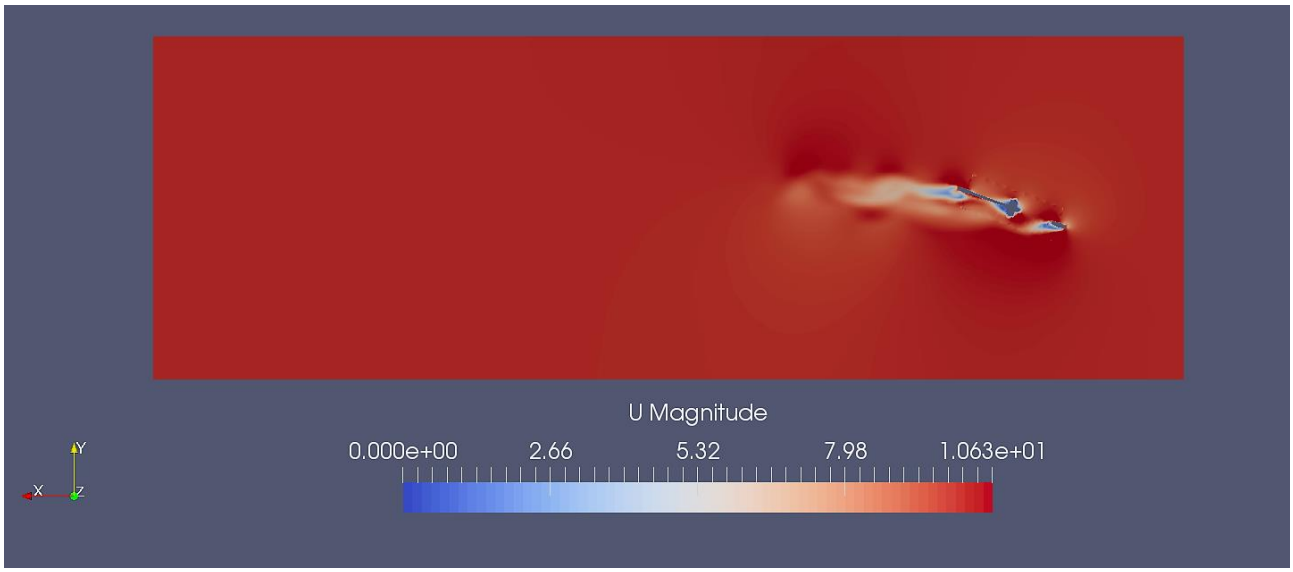


Figure 43. Velocity magnitude at 0,1 seconds

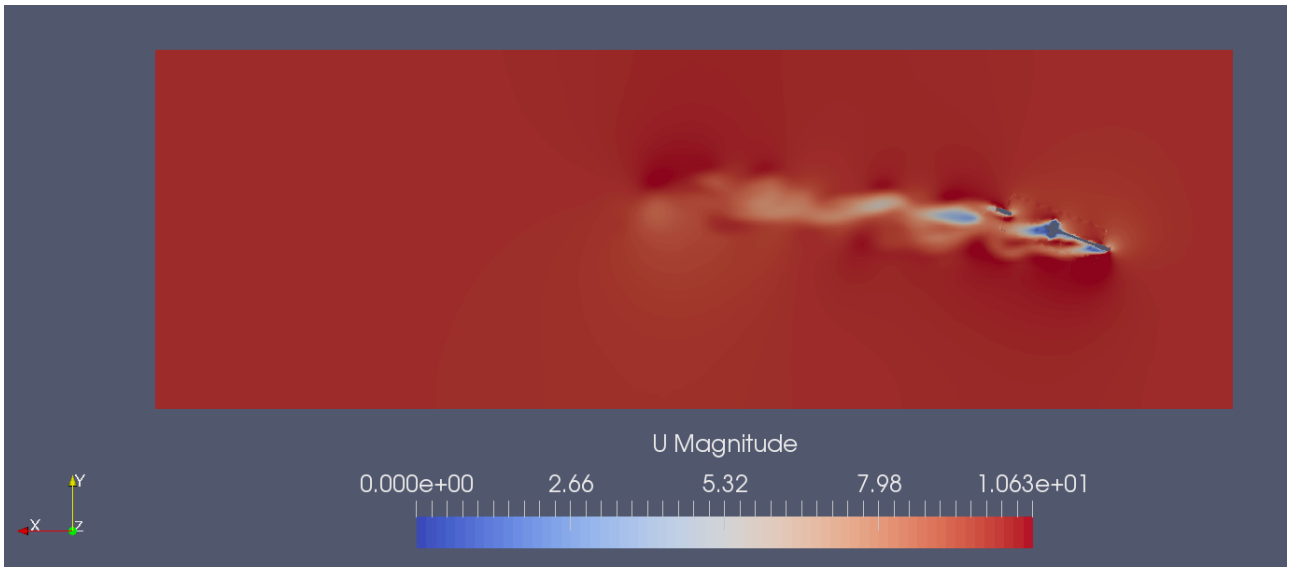


Figure 44. Velocity magnitude at 0,2 seconds

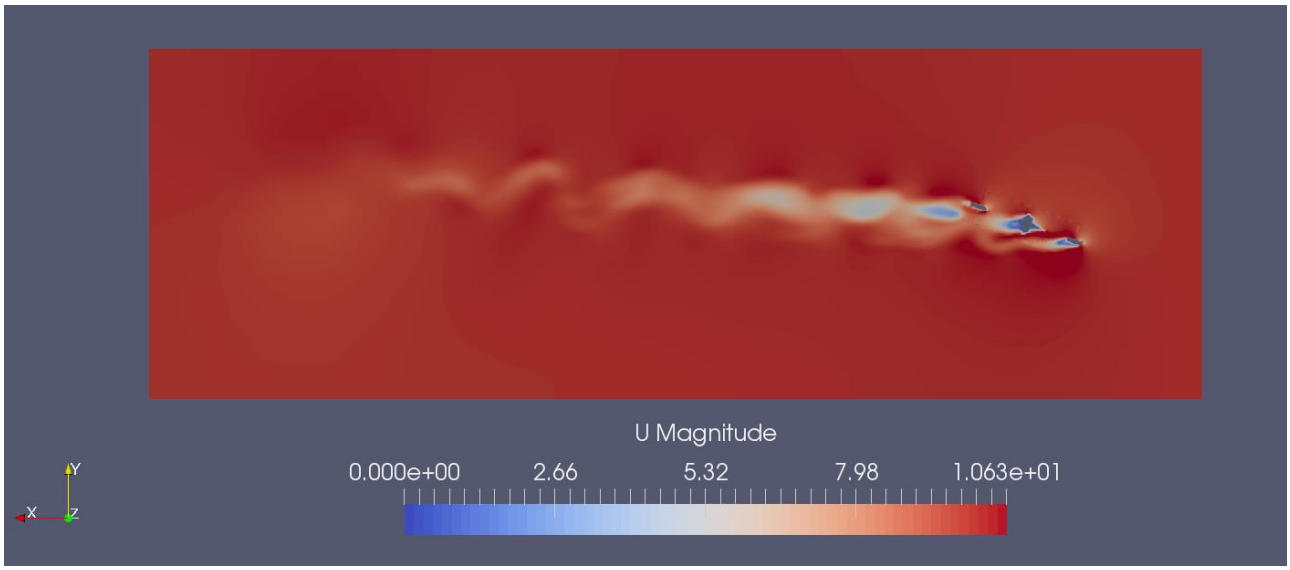


Figure 45. Velocity magnitude at 0,4 seconds

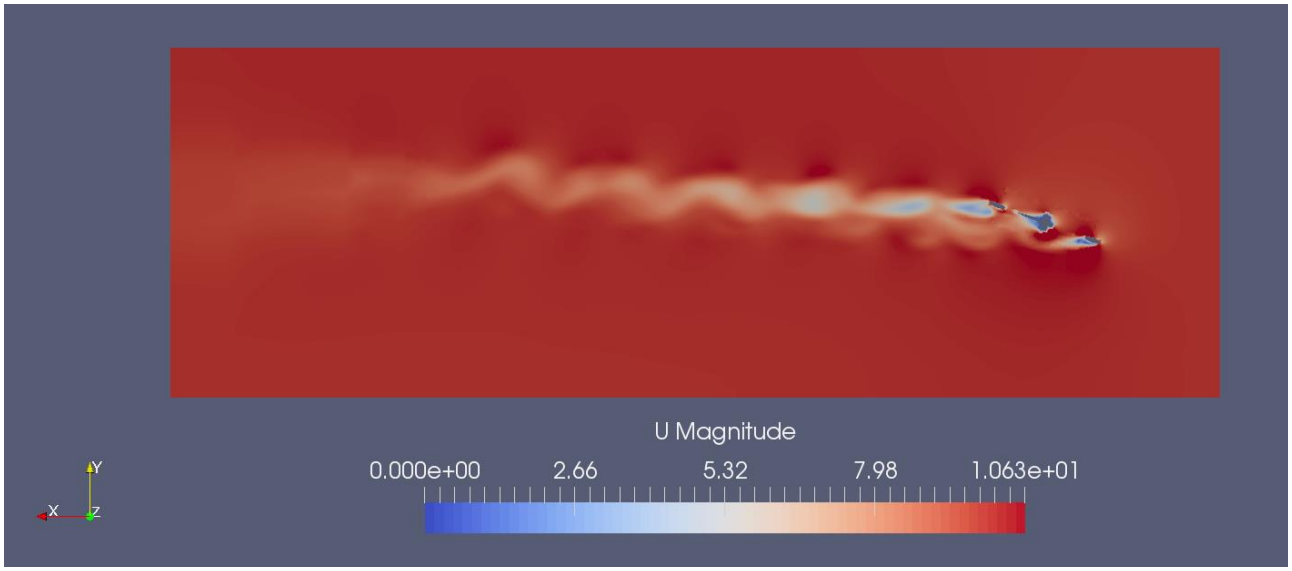


Figure 46. Velocity magnitude at 0,65 seconds

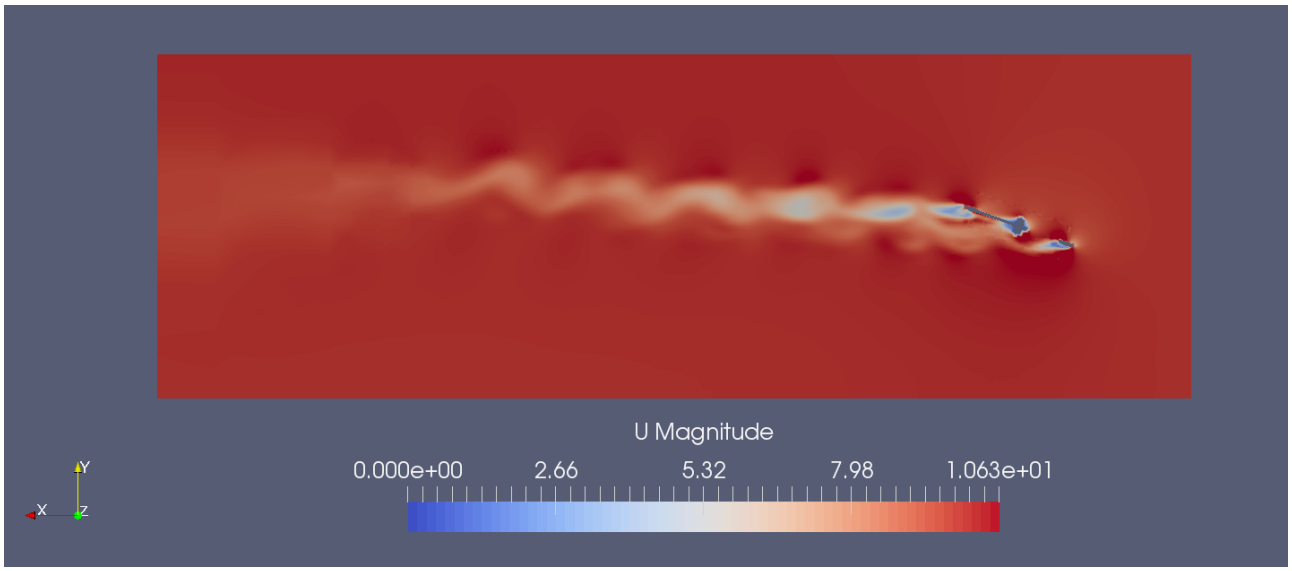


Figure 47. Velocity magnitude at 0,85 seconds

In the above figures we can see the velocity magnitude for 5 time steps 0,1 0,2 0,4 0,65 and 0,85 seconds. The flow seem to be fully developed between 0.6 and 0.85 seconds, and effect of the wake of the three rims are highlighted by the choose of the plane, but they seems a little overestimated when compared with a similar Star-CCM+ model (this model showed below uses a different polyhedral mesh, which is more dispersive), this can be also given by the Paraview displayer settings. The maximum fluid speed is also a little underestimated but I think it's caused by mesh settings.

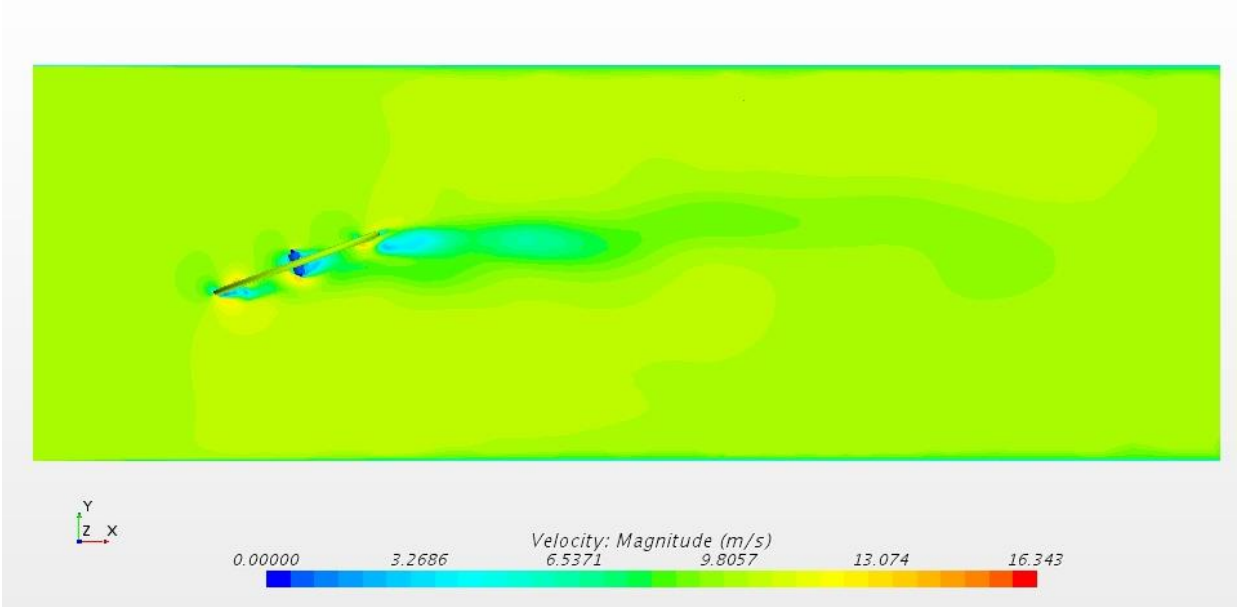


Figure 48. Velocity magnitude on a similar application on Star-CCM+ v11.04.12; t=0.2s

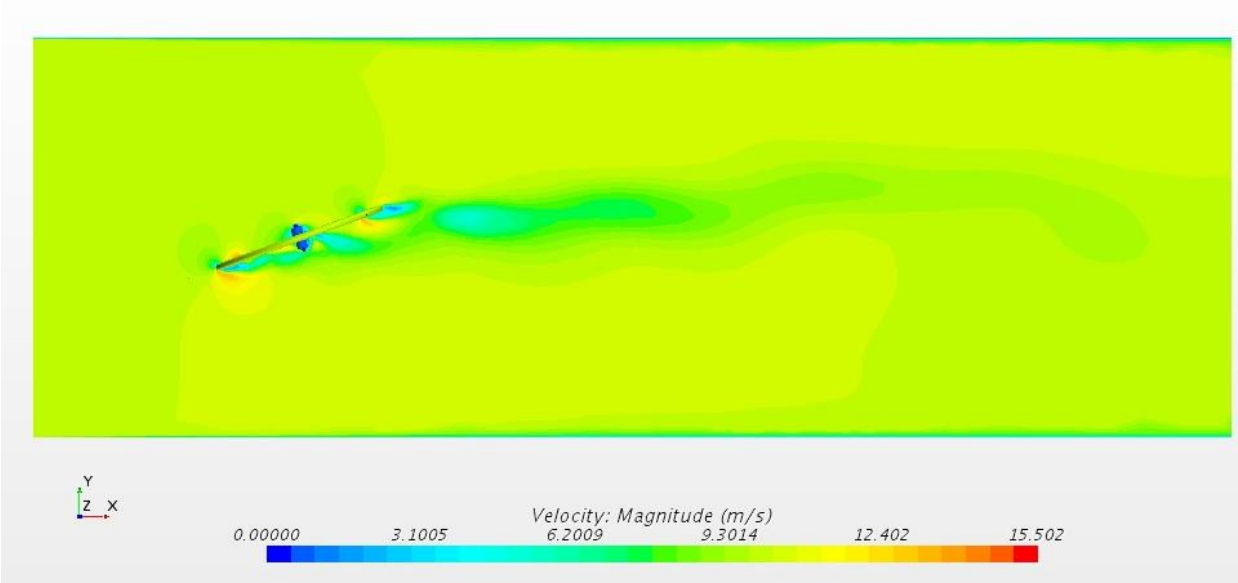


Figure 49. Velocity magnitude on a similar application on Star-CCM+ v11.04.12; t=0,4s

In some pictures (for example 0,1 and 0,2 seconds) the AMI interface can be noticed, even if the velocity of the flow before and after the interface seems not to be affected, the slower zone is caused probably by a non-perfectly coincident mesh, the mesh for this example, in fact, is a little too coarse.

This is a problem of the snappyHexMesh, the hexahedral mesh need a higher number of elements to define the geometry and OpenFoam uses a more computational resources than a commercial code.

This is only a simply model build to show the capability of the open source code, the main highlights of this model are:

- The code is open source, meaning that there is no costs for the annual license and for the uses of the software, (commercial codes charge for usage hours and sometimes for core used).
  The code is harder to implement, the changes made to the mesh and or the setting are more difficult and the lacks of a GUI make harder to check the model geometry at every step (paraFoam need to be launched every time), and the debugging of the model is difficult too, especially for a beginner.
  This means that the model setup need a lot of time if compared to a commercial code, recovering a part of the costs saved on licenses with operational labour.
- The code seems to need more resources than the star code, this can be caused by the different mesh used, the problem is noticeable when I run the program on my personal computer, but in reality for an industry this is a minor problem, considering that often they already had a computational “division” and anyway for simpler model the money saved on license in one year can easily cover the initial investment on a server of a cloud computing solution
- The code is very flexible it is a solver code, it is mainly used in cfd applications but it can be used in many other fields, structural mechanics, electromagnetics (this is true for the Star code too), finance being an open source C++ based code applications can be written by the user

- Other software can be used with OpenFoam, commercial and non-commercial GUI are available; we decide to do not use them to create a basic model for this application.
  Meshing can be imported in OpenFoam cases from different commercial codes like star or fluent using utilities like StarToFoam or Fluent3DMeshToFoam, open-source codes can be used like Salome, we decided to use Salome only for stl converting and to defining patch names on the different wheel surfaces, we use Foam internal utilities for the meshing because it was easier to define the different region domain.

Chapter 7

Conclusions

7.1 Conclusions

In this work, CFD was used to explore the complex and unsteady nature of airflow around a selection of different bicycle wheels.

The objective of this work was to establish a working CFD methodology to study the performance of an isolated, rotating bicycle wheel of commercial design, in contact with the ground. Good agreement with experimental wind tunnel studies suggests that the approach we outline holds considerable promise. Owing to the flexibility of this methodology, it is now possible to use CFD to provide more definitive answers on some of the open questions within the competitive cycling and triathlon communities.

The standard k-ε seems to give better performance on the steady-state case, both in results and stability of the model.

A novel methodology using a sliding mesh with a multi-reference frame was introduced to resolve the problem of accurately modelling the movement of the spokes. Realistic ground plane and upstream boundary conditions were used to study a selection of existing commercially produced bicycle wheel.

Agreement with the previously published wind tunnel results of *Greenwell* and *Tews* for resolved axial drag and side force or lift components was seen to be good for both the steady and unsteady calculations.

The CFD predictions of drag forces showed significant differences between wheels. For yaw angles in the range of 5 to 15 degrees, (those most commonly experienced by cyclists in practice), the deeper rim wheels offered a clear advantage on drag force, balanced by a higher side force.

It is also noted that a larger rim seems to be preferable when crosswinds are present, it presents a lower drag at higher angle zero, caused by a delay flow detach, this is not the case for track cycling.

A bigger fillet radius seems to increase Drag at low angles but generally had a lower drag at higher angle.

The steady-state models give good results using less resources and time, showing to be a good tool to compare wheel performances and design in a reasonable manner.

The unsteady state models give more accurate results and seems to work better using k-ω SST model and the more sophisticated DES model, but it need a more extended study.

7.2 Raccomandations for future works

Finally, we feel that introducing an automatic optimization for certain wheel dimensions is a straightforward extension of this work. For example for a fixed total rim depth, we can use the wheel thickness and the fillet radius of the inner part of the wheel, bearing in mind the limitation imposed by the tooling used to make the rims, that often used combination of arcs of circumference , and the current regulation and patents from other manufacturers.

Introducing a finer mesh or an unstructured one in OpenFOAM should give the open-source model complete operability.

Continuing the work with unsteady state models with a higher mesh resolution, and lower time steps to better comprehend the flow structures.

The inclusion and optimization of the front fork and frame and brake callipers in conjunction with a specific wheel can be useful, and that such optimization studies could lead to performance improvements, or change the design of the wheel to reduce the combine drag optimizing the compatibility of a wide selection of the wheels with typical forks.

Attachment 1

OpenFoam Files

A1 blockMeshDict

```
/*--------------------------------*- C++ -*----------------------------------
-*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.x                                   |
      |   \\  /    A nd          | Web:     www.OpenFOAM.org                  |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------
-*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//

convertToMeters 1;

vertices
(
    (-1 -1.0 -0.330)
    (5 -1.0 -0.330)
    (5  1.0 -0.330)
    (-1  1.0 -0.330)
    (-1 -1.0 1.66)
    (5 -1.0 1.66)
    (5  1.0 1.66)
    (-1  1.0 1.66)
);

blocks
(
    hex (0 1 2 3 4 5 6 7) (12 4 4) simpleGrading (1 1 1)
);

edges
(
);

boundary
(
    wall
    {
        type patch;
        faces
```

```
        (
            (3 7 6 2)
            (1 5 4 0)
            (4 5 6 7)
        );
    }
    inlet
    {
        type patch;
        faces
        (
            (0 4 7 3)
        );
    }
    outlet
    {
        type patch;
        faces
        (
            (2 6 5 1)
        );
    }
    floor
    {
        type wall;
        faces
        (
            (0 3 2 1)
        );
    }

);

//  ************************************************************************
//
```

A2 snappyHexMesh

```
/*--------------------------------*- C++ -*----------------------------------
-*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------
-*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      snappyHexMeshDict;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//
```

```
// Which of the steps to run
castellatedMesh true;
snap            true;
addLayers       true;


// Geometry. Definition of all surfaces. All surfaces are of class
// searchableSurface.
// Surfaces are used
// - to specify refinement for any mesh cell intersecting it
// - to specify refinement for any mesh cell inside/outside/near
// - to 'snap' the mesh boundary to the surface
geometry
{
    regionSTL.stl
    {
        type triSurfaceMesh;
        regions
        {
            staticWall      { name staticWalls; }
            movingWall      { name movingWalls; }

        }
    }

    AMI.stl
    {
        type triSurfaceMesh;
        name AMI;
    }
    refinementBox
    {
        type searchableBox;
        min (-0.5 -0.3 -0.33);
        max ( 3.5  0.3 0.85);
    }
};



// Settings for the castellatedMesh generation.
castellatedMeshControls
{

    // Refinement parameters
    // ~~~~~~~~~~~~~~~~~~~~

    // If local number of cells is >= maxLocalCells on any processor
    // switches from from refinement followed by balancing
    // (current method) to (weighted) balancing before refinement.
    maxLocalCells 2000000;

    // Overall cell limit (approximately). Refinement will stop immediately
    // upon reaching this number so a refinement level might not complete.
    // Note that this is the number of cells before removing the part which
    // is not 'visible' from the keepPoint. The final number of cells might
    // actually be a lot less.
    maxGlobalCells 4000000;
```

```
    // The surface refinement loop might spend lots of iterations refining
just a
    //  few  cells.  This  setting  will  cause  refinement  to  stop  if  <=
minimumRefine
    // are selected for refinement. Note: it will at least do one iteration
    // (unless the number of cells to refine is 0)
    minRefinementCells 0;

    // Number of buffer layers between different levels.
    // 1 means normal 2:1 refinement restriction, larger means slower
    // refinement.
    nCellsBetweenLevels 3;



    // Explicit feature edge refinement
    // ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    // Specifies a level for any cell intersected by its edges.
    // This is a featureEdgeMesh, read from constant/triSurface for now.
    features
    (
    );



    // Surface based refinement
    // ~~~~~~~~~~~~~~~~~~~~~~~~

    // Specifies two levels for every surface. The first is the minimum level,
    // every cell intersecting a surface gets refined up to the minimum level.
    // The second level is the maximum level. Cells that 'see' multiple
    // intersections where the intersections make an
    // angle > resolveFeatureAngle get refined up to the maximum level.

    refinementSurfaces
    {
        regionSTL.stl
        {
            // Surface-wise min and max refinement level
            level (0 0);
            regions
            {
                staticWall      { level (4 6); patchInfo { type wall; }}
                movingWall      { level (4 6); patchInfo { type wall; }}
            }
        }

        AMI
        {
            level (2 3);

            faceType boundary;
            cellZone innerAMI;
            faceZone innerAMI;
            cellZoneInside inside;
        }
    }
```

```
    resolveFeatureAngle 30;


    // Region-wise refinement
    // ~~~~~~~~~~~~~~~~~~~~~~

    // Specifies refinement level for cells in relation to a surface. One of
    // three modes
    // - distance. 'levels' specifies per distance to the surface the
    //   wanted refinement level. The distances need to be specified in
    //   descending order.
    // - inside. 'levels' is only one entry and only the level is used. All
    //   cells inside the surface get refined up to the level. The surface
    //   needs to be closed for this to be possible.
    // - outside. Same but cells outside.

    refinementRegions
    {
      refinementBox
        {
            mode inside;
            levels ((1E15 3));
        }
    }


    // Mesh selection
    // ~~~~~~~~~~~~~~

    // After refinement patches get added for all refinementSurfaces and
    // all cells intersecting the surfaces get put into these patches. The
    // section reachable from the locationInMesh is kept.
    // NOTE: This point should never be on a face, always inside a cell, even
    // after refinement.
    // This is an outside point locationInMesh (-0.033 -0.033 0.0033);
    locationInMesh (0.095 0 0.01); // Inside point

    // Whether any faceZones (as specified in the refinementSurfaces)
    // are only on the boundary of corresponding cellZones or also allow
    // free-standing zone faces. Not used if there are no faceZones.
    allowFreeStandingZoneFaces false;
}



// Settings for the snapping.
snapControls
{
    //- Number of patch smoothing iterations before finding correspondence
    //  to surface
    nSmoothPatch 3;

    //- Relative distance for points to be attracted by surface feature point
    //  or edge. True distance is this factor times local
    //  maximum edge length.
    tolerance 1.0;

    //- Number of mesh displacement relaxation iterations.
    nSolveIter 30;
```

```
    //- Maximum number of snapping relaxation iterations. Should stop
    //  before upon reaching a correct mesh.
    nRelaxIter 5;

    // Feature snapping

        //- Number of feature edge snapping iterations.
        //  Leave out altogether to disable.
        nFeatureSnapIter 10;

        //- Detect (geometric) features by sampling the surface
        implicitFeatureSnap true;

        //- Use castellatedMeshControls::features
        explicitFeatureSnap false;

        //- Detect features between multiple surfaces
        //  (only for explicitFeatureSnap, default = false)
        multiRegionFeatureSnap false;
}



// Settings for the layer addition.
addLayersControls
{
    // Are the thickness parameters below relative to the undistorted
    // size of the refined cell outside layer (true) or absolute sizes (false).
    relativeSizes true;

    // Per final patch (so not geometry!) the layer information
    layers
    {
        "(floor|movingWalls|staticWalls).*"
        {
            nSurfaceLayers 1;
        }
    }

    // Expansion factor for layer mesh
    expansionRatio 1.0;

    // Wanted thickness of final added cell layer. If multiple layers
    // is the thickness of the layer furthest away from the wall.
    // Relative to undistorted size of cell outside layer.
    // See relativeSizes parameter.
    finalLayerThickness 0.3;

    // Minimum thickness of cell layer. If for any reason layer
    // cannot be above minThickness do not add layer.
    // Relative to undistorted size of cell outside layer.
    minThickness 0.1;

    // If points get not extruded do nGrow layers of connected faces that are
    // also not grown. This helps convergence of the layer addition process
    // close to features.
    // Note: changed(corrected) w.r.t 17x! (didn't do anything in 17x)
```

```
    nGrow 0;

    // Advanced settings

    // When not to extrude surface. 0 is flat surface, 90 is when two faces
    // are perpendicular
    featureAngle 60;

    // At non-patched sides allow mesh to slip if extrusion direction makes
    // angle larger than slipFeatureAngle.
    slipFeatureAngle 30;

    // Maximum number of snapping relaxation iterations. Should stop
    // before upon reaching a correct mesh.
    nRelaxIter 3;

    // Number of smoothing iterations of surface normals
    nSmoothSurfaceNormals 1;

    // Number of smoothing iterations of interior mesh movement direction
    nSmoothNormals 3;

    // Smooth layer thickness over surface patches
    nSmoothThickness 10;

    // Stop layer growth on highly warped cells
    maxFaceThicknessRatio 0.5;

    // Reduce layer growth where ratio thickness to medial
    // distance is large
    maxThicknessToMedialRatio 0.3;

    // Angle used to pick up medial axis points
    // Note: changed(corrected) w.r.t 17x! 90 degrees corresponds to 130 in
17x.
    minMedianAxisAngle 90;


    // Create buffer region for new layer terminations
    nBufferCellsNoExtrude 0;


    // Overall max number of layer addition iterations. The mesher will exit
    // if it reaches this number of iterations; possibly with an illegal
    // mesh.
    nLayerIter 50;
}




// Generic mesh quality settings. At any undoable phase these determine
// where to undo.
meshQualityControls
{
    //- Maximum non-orthogonality allowed. Set to 180 to disable.
    maxNonOrtho 65;

    //- Max skewness allowed. Set to <0 to disable.
```

```
    maxBoundarySkewness 20;
    maxInternalSkewness 4;

    //- Max concaveness allowed. Is angle (in degrees) below which concavity
    //  is allowed. 0 is straight face, <0 would be convex face.
    //  Set to 180 to disable.
    maxConcave 80;

    //- Minimum pyramid volume. Is absolute volume of cell pyramid.
    //  Set to a sensible fraction of the smallest cell volume expected.
    //  Set to very negative number (e.g. -1E30) to disable.
    minVol 1e-13;

    //- Minimum quality of the tet formed by the face-centre
    //  and variable base point minimum decomposition triangles and
    //  the cell centre. This has to be a positive number for tracking
    //  to work. Set to very negative number (e.g. -1E30) to
    //  disable.
    //     <0 = inside out tet,
    //      0 = flat tet
    //      1 = regular tet
    minTetQuality -1; // 1e-30;

    //- Minimum face area. Set to <0 to disable.
    minArea -1;

    //- Minimum face twist. Set to <-1 to disable. dot product of face normal
    //  and face centre triangles normal
    minTwist 0.01;

    //- Minimum normalised cell determinant
    //  1 = hex, <= 0 = folded or flattened illegal cell
    minDeterminant 0.001;

    //- minFaceWeight (0 -> 0.5)
    minFaceWeight 0.05;

    //- minVolRatio (0 -> 1)
    minVolRatio 0.01;

    //must be >0 for Fluent compatibility
    minTriangleTwist -1;


    // Advanced

    //- Number of error distribution iterations
    nSmoothScale 4;
    //- Amount to scale back displacement at error points
    errorReduction 0.75;

    // Optional : some meshing phases allow usage of relaxed rules.
    // See e.g. addLayersControls::nRelaxedIter.
    relaxed
    {
        //- Maximum non-orthogonality allowed. Set to 180 to disable.
        maxNonOrtho 75;
    }
}
```

```
// Merge tolerance. Is fraction of overall bounding box of initial mesh.
// Note: the write tolerance needs to be higher than this.
mergeTolerance 1e-6;


//  *********************************************************************
//
```

A3 createPatchDict

```
/*--------------------------------*- C++ -*----------------------------------
-*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------
-*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      createPatchDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//

// Do a synchronisation of coupled points after creation of any patches.
// Note: this does not work with points that are on multiple coupled patches
//       with transformations (i.e. cyclics).
pointSync false;

// Patches to create.
patches
(
    {
        //- Master side patch
        name            AMI1;
        patchInfo
        {
            type            cyclicAMI;
            matchTolerance  0.0001;
            neighbourPatch  AMI2;
            transform       noOrdering;
        }
        constructFrom patches;
        patches (AMI);
    }

    {
        //- Slave side patch
        name            AMI2;
        patchInfo
```

```
        {
            type            cyclicAMI;
            matchTolerance  0.0001;
            neighbourPatch  AMI1;
            transform       noOrdering;
        }
        constructFrom patches;
        patches (AMI_slave);
    }
);

//  ************************************************************************
//
```

A4.1 initial conditions P

```
/*--------------------------------*- C++ -*----------------------------------
-*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.1                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------
-*/
FoamFile
{
    version     2.0;
    format      binary;
    class       volScalarField;
    location    "0";
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            zeroGradient;
    }
    movingWalls
    {
        type            zeroGradient;
    }
    outlet
    {
        type            fixedValue;
        value           uniform 0;
    }
    staticWalls
    {
        type            zeroGradient;
    }
```

```
    wall
    {
        type            zeroGradient;
    }
    floor
    {
        type            zeroGradient;
    }
    AMI1
    {
        type            cyclicAMI;
        value           uniform 0;
    }
    AMI2
    {
        type            cyclicAMI;
        value           uniform 0;
    }
}


// ************************************************************************
//
```

A4.2 initial conditions U

```
/*--------------------------------*- C++ -*----------------------------------
-*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.1                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------
-*/
FoamFile
{
    version     2.0;
    format      binary;
    class       volVectorField;
    location    "0";
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//

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

internalField   uniform (10 0 0);

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform (10 0 0);
    }
    movingWalls
    {
```

```
        type            movingWallVelocity;
        value           uniform (0 0 0);
    }
    outlet
    {
        type            inletOutlet;
        inletValue      uniform (0 0 0);
        value           $internalField;
    }
    staticWalls
    {
          type            rotatingWallVelocity;
          origin          ( 0 0 0 );
          axis            ( 0 1 0 );
          omega            -30.303;
    }
    wall
    {
        type            slip;
    }
    floor
    {
        type            fixedValue;
        value           uniform (10 0 0);
    }
    AMI1
    {
        type            cyclicAMI;
        value           uniform (0 0 0);
    }
    AMI2
    {
        type            cyclicAMI;
        value           uniform (0 0 0);
    }
}


// ************************************************************************
//
```

A4.3 initial conditions k

```
/*--------------------------------*- C++ -*----------------------------------
-*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.x                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------
-*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      k;
```

```
}
// * * *
```

```
    }
}


// ************************************************************************
//
```

A4.4 initial conditions nut

```
/*--------------------------------*- C++ -*----------------------------------
-*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.x                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------
-*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      nut;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//

dimensions      [0 2 -1 0 0 0 0];

internalField   uniform 0;

boundaryField
{


    inlet
    {
        type            calculated;
        value           uniform 0;
    }

    outlet
    {
        type            calculated;
        value           uniform 0;
    }

    wall
    {
        type            nutkWallFunction;
        value           uniform 0;
    }

    movingWalls
    {
        type            nutkWallFunction;
        value           uniform 0;
```

```
    }

    staticWalls
    {
        type            nutkWallFunction;
        value           uniform 0;
    }

    floor
    {
        type            nutkWallFunction;
        value           uniform 0;
    }
    AMI1
    {
        type            cyclicAMI;
        value           uniform 0;
    }
    AMI2
    {
        type            cyclicAMI;
        value           uniform 0;
    }
}


// ************************************************************************
//
```

A4.5 initial conditions Epsilon

```
/*--------------------------------*- C++ -*----------------------------------
-*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.x                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------
-*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      epsilon;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//

dimensions      [0 2 -3 0 0 0 0];

internalField   uniform 0.0495;

boundaryField
{
    inlet
    {
```

```
        type            fixedValue;
        value           $internalField;
    }

    outlet
    {
        type            inletOutlet;
        inletValue      $internalField;
        value           $internalField;
    }

    wall
    {
        type            epsilonWallFunction;
        value           $internalField;
    }
    movingWalls
    {
        type            epsilonWallFunction;
        value           $internalField;
    }
    staticWalls
    {
        type            epsilonWallFunction;
        value           $internalField;
    }
    floor
    {
        type            epsilonWallFunction;
        value           $internalField;
    }
    AMI1
    {
        type            cyclicAMI;
        value           $internalField;
    }
    AMI2
    {
        type            cyclicAMI;
        value           $internalField;
    }

}


// ************************************************************************
//
```

A5 dynamicMeshDict

```
/*--------------------------------*- C++ -*----------------------------------
-*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.x                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
```

```
\*---------------------------------------------------------------------------
-*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      dynamicMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//

dynamicFvMesh   solidBodyMotionFvMesh;

motionSolverLibs ( "libfvMotionSolvers.so" );

solidBodyMotionFvMeshCoeffs
{
    cellZone        innerAMI;

    solidBodyMotionFunction  rotatingMotion;
    rotatingMotionCoeffs
    {
        origin      (0 0 0);
        axis        (0 1 0);
        omega       -30.303; // rad/s
    }
}


// ************************************************************************
//
```

A6 transportProprierties

```
/*--------------------------------*- C++ -*----------------------
----------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  2.3.0                                 |
|     \\   /      A nd               | Web:         www.OpenFOAM.org
|
|    \\/    M anipulation  |                                                 |
\*---------------------------------------------------------------------
-----------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
```

```
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * //

transportModel  Newtonian;

nu              nu [ 0 2 -1 0 0 0 0 ] 1.46e-5;

//
************************************************************************
******* //
```

A7 turbolenceProprierties

```
/*--------------------------------*- C++ -*----------------------------
----------*\
| =========                 |                                         |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  2.3.0                         |
|    \\  /      A nd         | Web:         www.OpenFOAM.org
|
|   \\/    M anipulation  |                                         |
\*------------------------------------------------------------------
-----------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * //

simulationType RAS;

RAS
{
    RASModel            kEpsilon;

    turbulence          on;

    printCoeffs         on;
}
```

```
//
*********************************************************************
******* //
```

A8 controlDict

```
/*--------------------------------*- C++ -*----------------------------------
-*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.x                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------
-*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//

application     pimpleDyMFoam;

startFrom       latestTime;

startTime       0;

stopAt          endTime;

endTime         1;

deltaT 0.0001;
writeControl    adjustableRunTime;
writeInterval 0.01;

purgeWrite      0;

writeFormat     binary;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

adjustTimeStep  yes;

maxCo           1;
```

```
functions
{
    forces
    {
        type            forceCoeffs;
        functionObjectLibs ( "libforces.so" );
        outputControl   timeStep;
        outputInterval  1;

        patches
        (
            movingWalls
          staticWalls

        );

          rho rhoInf;
      p p;
      U U;

        log         true;
        rhoInf      1.18;
        CofR        ( 0 0 0 );
        liftDir     ( 0 1 0 );
        dragDir     ( 1 0 0 );
        pitchAxis   ( 0 0 0 );
        magUInf     10;
        lRef        1;
        Aref        1;
    }
}

//  ************************************************************************
//
```

A9 fvSolution

```
/*--------------------------------*- C++ -*----------------------------------
-*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------
-*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//

solvers
{
    pcorr
```

```
    {
        solver          GAMG;
        tolerance       1e-2;
        relTol          0;
        smoother        DICGaussSeidel;
        cacheAgglomeration no;
        nCellsInCoarsestLevel 10;
        agglomerator    faceAreaPair;
        mergeLevels     1;
        maxIter         50;
    }

    p
    {
        $pcorr;
        tolerance       1e-5;
        relTol          0.01;
    }

    pFinal
    {
        $p;
        tolerance       1e-6;
        relTol          0;
    }

    "(U|k|epsilon)"
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-6;
        relTol          0.1;
    }

    "(U|k|epsilon)Final"
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-6;
        relTol          0;
    }
}

PIMPLE
{
    correctPhi          no;
    nOuterCorrectors    2;
    nCorrectors         1;
    nNonOrthogonalCorrectors 0;
}

relaxationFactors
{
    "(U|k|epsilon).*"   1;
}

cache
{
    grad(U);
```

```
}

//  ************************************************************************
//
```

Attachment 2

Aerodynamic Balance Force preliminary study

A2.1 Introduction

One of the initial possibility of the PhD was the wind tunnel testing in a non-prepared structured build mainly for garments test, this mean that a force balance for the wheel was needed, we start a balance project that in the end wasn't used

Wind tunnel tests aims to determine the force that act on the wheel considering different variations:

- Rim shape
- Tire shape
- Spoke shape and number
- Wheel and wind speed
- Wheel angle

In order to make this possible we need this instrumentation

- A force balance
- A speed measurement system, typically an anemometer
- A wheel rotating system
- A wheel angle movement system

A2.2 General Layout

we decide to put the force balance under the wheel, so the system has to be able to support the weight of the wheel and of itself, the forces used for this first design were the forces obtain from the initial cfd simulations

The major force to take in account is the side forces from the disc wheel that can be over 30N, depending on the angle of incidence, obviously the complete bike forces are higher.

We consider giving a preloading to the wheel of 400N, the forces that the single wheel balance should be between 0 and at least 5 N for the Drag force.

The initial idea was to use a simple lever balance to obtain the result

The main advantage were

- It's "easy" to decouple the drag and the side forces
- We can amplify the forces
- Using different load cells it is possible to make the balance more versatile
- We can use commercial load cells
- Relatively simply design

The disadvantage are

- We can also amplify the errors
- Mechanical joints introduce parasites forces and possible errors.

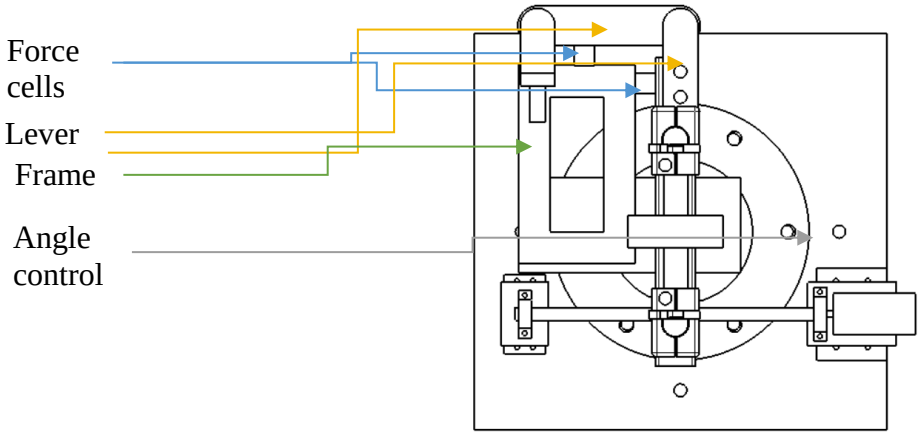


Figure 50. Lever design

Considering the low forces and the disadvantages, we decide to discard this solution.

We consider other design, like ones based on spherical joints but finally we decided that the simplest way to achieve our objective is to use a force balance based on flexible elements coupled with extensimeter as shown in the figure below.

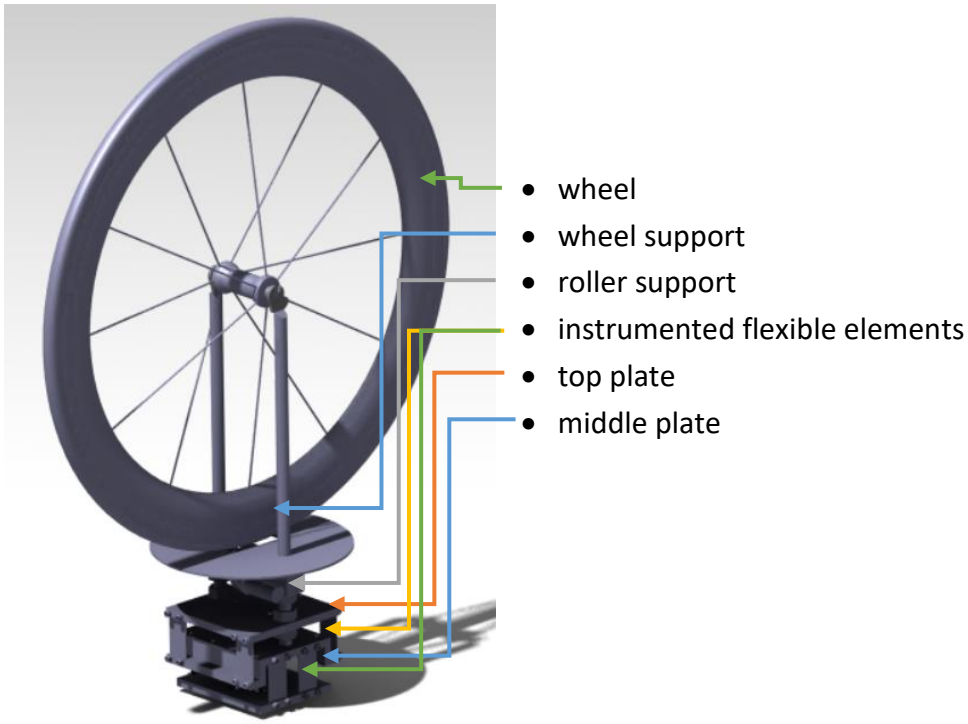


Figure 51. Later design

The balance is divided in two main planes connected by four instrumented flexible elements for the drag and for the side forces, this system permit to decouple the forces without introducing joints losses.

The wheel is supported by vertical support with fork end, the bottom of the support is made with a threaded tube were the roller support is secured. The wheel support includes two bearings for wheel roller, the wheel roller rotation is given by an electric motor with an encoder by gear system, four M16 nuts are used to preload the wheel and support the roller system.

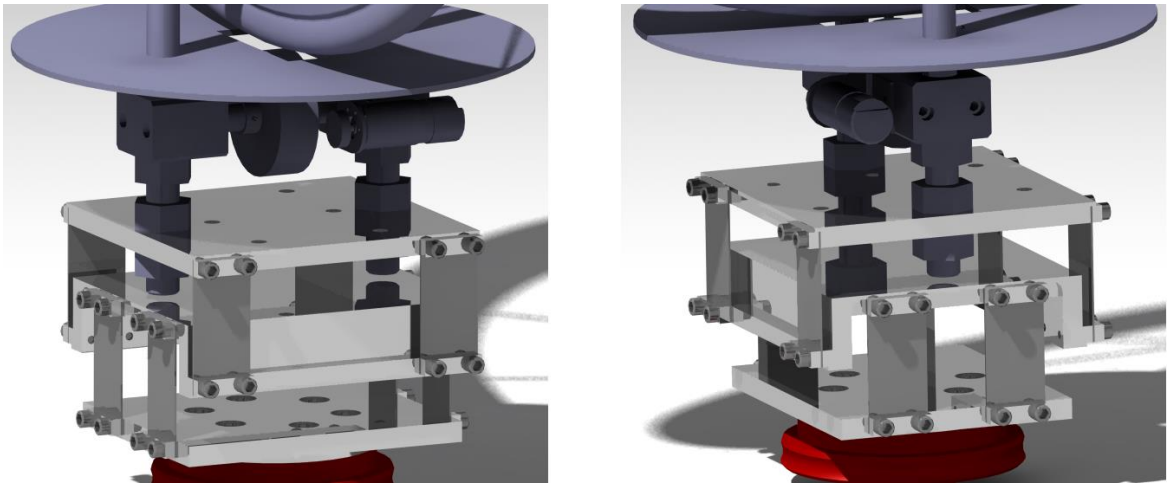


Figure 52. Force balance particular

The wheel-roller-motor-complex is secured to the top plate with 4 M16 nuts. The higher plate is aluminium made and had four housings for the flexible elements, those elements are fixed with two M4 screw for every plate. The top plate also have housing for the complete bike system.

The middle plate is a three element aluminium plate made with a C-shaped element with the housing for the lower flexible element and two spacer to permit the assemble the higher flexible element.

Finally the lower plate is a simple plate with a cylindrical base where a worm wheel is mounted, this plate is mounted on a tapered roller bearing.

The two motor was chosen as long as the dimensions of the sensible elements and of the four bearing.

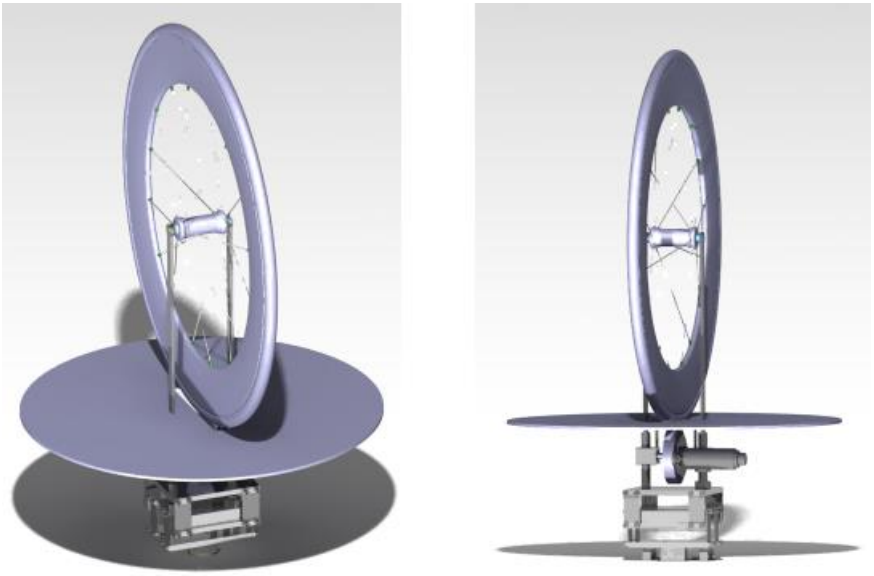


Figure 53. Force balance

Figure 54. Proposed bike support

Bibliography

- Achenbach, E. (1968). Distribution of local pressure and skin friction around a circular cylinder in cross-flow up to Re= 5e6. Journal of Fluid Mechanics, 34(4), 625639.
- Achenbach, E. (1974). The effects of surface roughness and tunnel blockage on flow past spheres. Journal of Fluid Mechanics, 65(1), 113-125.
- Barry et al. (2012). Effect of crosswinds and wheel selection on the aerodynamic behavior of a cyclist. Monash University, Melbourne, Aus, 9° Conference of the international sports Association.
- Capelli, C., Rosa, G., Butti, F., Ferretti, G., Veicsteinas, A. and di Prampero, P. (1993). Energy cost and efficiency of riding aerodynamic bicycles. European Journal of Applied Physiology, 67(1), 144-149.
- Casacuberta J. Puig (2014). OpenFOAM GUIDE FOR BEGINNERS. the-foam-house5.webnode.es/
- Cd-ADAPCO Star-CCM+ User Manual
- CFD Direct(2011-2017). OpenFoam user manual. OpenFOAM Foundation.
- Chowdhury et.al, (2011). A full scale bicycle testing methodology. 5th APCST conference, p.94-99.
- Di Prampero, P. E., Cortili, G., Mognoni, P. and Saibene, F. (1979). Equation of motion of a cyclist. Journal of applied physiology, 47(1), 201-206.
- EASTON Cycling (2010). Defining, Measuring and Improving AERODYNAMIC TESTING IN THE BICYCLE INDUSTRY.
- Fluent-Incorporated (2005). Fluent 6.2 User's Guide. Lebanon, USA.
- Forrestery, A.I.J. (2008). On the Aerodynamics of Bicycle Wheels. Computational Engineering and Design Group School of Engineering Sciences, University of Southampton, UK.
- Godo M. N. et.al. (2009). A comparative aerodynamic study of commercial bicycle wheels using cfd. Intelligent Light, American institute of aeronautics and astronautics, USA 2010.
- Godo M. N. et.al. (2009). An Aerodynamic Study of Bicycle Wheel Performance using CFD. Hyperworks, American Institute of Aeronautics and Astronautics.
- Greenwell et al (1995). Aerodynamic characteristics of low-drag bicycle wheels. Aeronautical J., 99 (983), pp. 109-120.Aeronautical Journal.
- Hagberg, J. M. and McCole, S. D. (1990). The effect of drafting and aerodynamic equipment on the energy expenditure during cycling. Cycling Science, 2(3), 1922.
- Holzmann T. “holzmann-cfd.de” OpenFoam tutorials
- Kyle, C. R. (1979). Reduction of Wind Resistance and Power Output of Racing Cyclists and Runners Travelling in Groups. Ergonomics, 22(4), 387-397.
- Kyle, C. R. (1989). The aerodynamics of handlebars and helmets. Cycling Science, 1,2225.
- Kyle, C. R. (1990). Chain friction, windy hills and other quick calculations. Cycling Science, 2(3), 23-26.
- Kyle, C. R. (1991a). New aero wheel tests. Cycling Science, 3(1), 27-30.
- Kyle, C. R. (1991b). Wind tunnel tests of aero bicycles. Cycling Science, 3(1), 57-61.
- Kyle, C. R. (2003). Selecting Cycling Equipment. In High-Tech Cycling: The science of riding faster(Ed, Burke, E. R.) Human Kinetics, Colorado, pp. 1-48.

- Kyle, C. R. and Bassett, D. R. J. (2003). Aerodynamics and the UCI. In High-tech cycling: The science of riding faster(Ed, Burke, E. R.) Human Kinetics, pp. 181183.
- Kyle, C. R. and Brownlie, L. (2002). http://www.nike.com/nikecycling/index_flash.html.
- Kyle, C. R., Brownlie, L. W., Harber, E., MacDonald, R. and Nordstrom, M. (2004). The Nike Swift Spin cycling project: Reducing the aerodynamic drag of bicycle racing clothing by using zoned fabric. In The 5th International Conference on the Engineering of Sport (Ed. Hubbard, M., Mehta, R. D. and Pallis, J. M.) UC Davis, USA, pp. 118-124.
- Kyle, C. R. and Burke, E. R. (1984). Improving the racing bicycle. Mechanical Engineering, 106(9), 34-35.
- Kyle, C. R. and Weaver, M. D. (2004). Aerodynamics of human-powered vehicles. Proceedings of the Institute of Mechanical Engineers, V218, Part A. Journal of Power and Energy, 141-154.
- Knupe J., Farmer D. (2009). Aerodynamics of High Performance Race Bicycle Wheels. Wing-light-de, Germany.
- Kolmogorov (1942). Equations of turbulent motion of an incompressible fluid. Izvestia Academy of Sciences, USSR; Physics, 6(1 and 2), 56-58.
- Lukes R. A.(2006). Improving track cycling performance using computational fluid dynamics. University of Sheffield.
- Massey, B. (1998). Mechanics of Fluids (Seventh Ed.), Stanley Thornes Ltd., Cheltenham, UK.
- McCole, S. D., Claney, K., Conte, J. C., Anderson, R. and Hagberg, J. M. (1990). Energy expenditure during bicycling. Journal of applied physiology, 68(2), 748753.
- ParaView user guide.
- Pogni M., Petrone N., Antonello M., Gobbato P. (2015) Comparison of the aerodynamic performance of four racing bicycle wheels by means of CFD calculation. 7th Asia-Pacific Congress on Sports Tecnology, APCST 2015.
- Pogni M., Petrone N. (2016). Comparison of the aerodynamic performance of five racing bicycle wheels by means of CFD calculations. PROCEDIA ENGINEERING, vol. 147, p. 74-80, ISSN: 1877-7058, doi: 10.1016/j.proeng.2016.06.192.
- Prandtl, L. (1904). Ueber Flussigkeitsbewegung bei sehr kleiner Reibung. In 3rd International Mathematical Congress (Ed. Leipzig) Heidelberg.
- Pugh, L. G. C. E. (1974). The relation of oxygen intake and speed in competition cycling and comparative observations on the bicycle ergometer. Journal of Physiology, 241(1), 795-808.
- Saffman, P. G. (1970). A model for inhomogeneous turbulent flow. Proceedings of the Royal Society, London, A317, 417-433.
- Salome User Guide and Tutorials.
- Tew G.S., Sayers A.T., (1997). Aerodynamics of yawned racing cycle wheels. Journal of wind engineering and industrial aerodynamics. University of Cape Town , South Africa.
- UCI (2006). UCI cycling regulations: Part 1 general organization of cycling as a sport.
- Versteeg, H. K. and Malalasekera, W. (2007) “An Introduction to Computational Fluid Dynamics: second edition” Peason Education Ltd., London.
- Vandoormaal, J. P. and Raithby, G. D. (1984). Enhancements of the SIMPLE method for predicting incompressible fluid flows. Numerical Heat Transfer, 7, 147-163.

- White, F. M. (2003). Fluid Mechanics (Fifth Ed.), McGraw-Hill, New York.
- Wilcox, D. C. (2000). Turbulence Modelling for CFD (2nd), DCW Industries, Inc.
- Zdravkovich, M. M. (1992). Aerodynamics of bicycle wheel and frame. Journal of Wind Engineering and Industrial Aerodynamics, 40(1), 55-70.